**CLASS ATTENDANCE & FACE MASK COMPLIANCE DETECTION SOFTWARE**

**Technical Specification**

**Authors**

David Weir (19433086)

Cian Mullarkey (19763555)

**Supervisor**

Hossein Javidnia

**Date Finished**

04/03/2022

**Contents**

1. **Introduction**

**1.1 Abstract**

The general area covered by this project is that of machine vision, in particular facial recognition. The aim of the project is to achieve a functioning classroom attendance log using a camera and a facial recognition algorithm to identify and then log and record students who have attended the particular class. The project is also capable of noting whether a student is wearing a face mask and making note of this along with their attendance.

**1.2 Glossary**

**Machine Vision** - Machine learning is the study of computer algorithms that can improve automatically through experience and by the use of data.

**Facial Recognition** - Facial recognition system is a technology capable of matching a human face from a digital image or a video frame against a database of faces by pinpointing and measuring facial features from a given image.

**Dataset:** A folder containing a number of subfolders storing the images of students and named after the relevant student.

**GUI:** A graphical user interface, a menu that allows the user to control the system and its functions.

**Feature Extraction:** The process of extracting face component features (e.g. eyes, nose, mouth) from the image of a human face.

**System Architecture** - A system architecture is the conceptual model that defines the structure, behaviour, and more views of a system.

**High Level Design (HDL)** - Explains the architecture that would be used in the development of a system. Provides an overview of the system identifying the main components of the system

**1.2 Overview/motivation**

Our motivation to undertake this particular project came about through our research of potential project ideas earlier in the academic year. While neither of us had a particular project or technology that we had our heart set on, after some research we both agreed that machine vision and facial recognition were technologies that appealed to both of us as it was something we both found interesting, we felt it would be a challenge to us, as well as the fact that it’s a relevant topic in the current technical landscape.

We found that this project had practical applications that could be used outside of academia which made the project all the more interesting to us. Many institutions and organisations can always make use of facial recognition software to take attendance in classes or meetings etc. Additionally, with the prevalent threat of COVID19 the use of machine vision could assist organisations with ensuring compliance to the face mask rules set by the government.

1. **High-level design**
   1. **High Level Design (HLD) Description**

* Lecturer runs the application GUI.
* The students first and last name are entered and dataset generator function is run, capturing images of their face which are entered in a database.
* Lecturer runs feature extraction function which creates a recognition model for use during attendance taking. It only needs to be run once after all students have database entries and anytime a student has been added or removed from the database.
* Lecturer runs attendance function to begin a live webcam of the class that is sent to the facial recognition system.
* The system takes the frames from the live webcam and using the faces from each frame matches the face to images from the student image database.
* The system updates the student’s attendance based on finding a match or not.
* The mask detection system then determines if the student is wearing a mask.
* The student’s mask compliance is updated on the class list database by the system.
  1. **High Level Design Context Diagram**

Initial HLD Context Diagram Version 1.0

Diagram

Description automatically generated

Current HLD Context Diagram Version 2.0

Diagram

Description automatically generated

* 1. **Data-Flow Diagram**

Initial Data-Flow Diagram Version 1.0

Diagram

Description automatically generated

Current Data-Flow Diagram Version 2.0

Diagram

Description automatically generated

1. **System Architecture**

Initial System Architecture Diagram Version 1.0

Diagram

Description automatically generated

Current System Architecture Diagram Version 2.0

Diagram

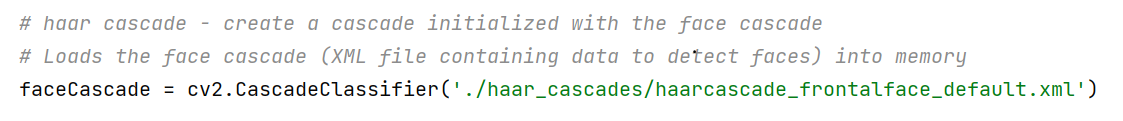
Description automatically generated

1. **Implementation**

**‘dataset\_generator.py’**

Our implementation begins with the ‘dataset\_generator.py’ file. This is the opening stage of the implementation as this file is the generates the images of the individual which are later used to build the facial recognition model.

After creating the dataset folder using the user inputted individuals first and last name, a haar cascade is loaded in. This is an XML file containing data which is used to detect data.



Following this we have three different functions, front, right and left. The purpose of these functions is very similar, firstly to take 30 images of the front of the user’s face, followed by 5 of the left of their face and 5 of the right. Whilst we originally implemented just 1 function which took 30 photos of the user’s face, we implemented these extra 2 functions based on supervisor feedback in order to increase the applications recognition ability. The functions run in order and are separated by a ‘sleep(5)’ call to give the user time to move their head.

Graphical user interface, text, application

Description automatically generated

Each function sets the default webcam as the video source then oops the program until it captures the required number of images, whilst also drawing an identifying rectangle around the users face.

Text

Description automatically generated

The images taken are stored in the previously created datasets

**‘feature\_extract.py’**

The next file in our implementation is ‘feature\_extract.py’. This file trains the recognition model by extracting facial features from the images in each dataset sub folder generated by the previous file.

This file begins by retrieving the path of each sub-data folder in the datasets folder. 

It then loops through the images and extracts each user’s name for recognition from the path. The images are converted from OpenCV ordering BGR to dlib ordering RGB in order to allow the program to locate the faces using the facial recognition library.

Graphical user interface, text, application

Description automatically generated

Facial embeddings of the face are then computed and saved, along with the names, in a data dictionary saved to the ‘face\_enc’ file.

**‘webcam\_recogntion.py’**

**‘gui.py’**

The final part of our implementation was to create a user interface for our application. While we initially planned to utilise a simple command line interpreter based interface, however as the project developed we felt it would benefit from a proper window application UI.

To create this we utilised the python GUI framework Tkinter to create this simple GUI. This involved standard set up of the window size, initial title, labels and setting of an icon and background image.

Following this we implemented 4 different buttons, the first of which would run the user manual pdf. Graphical user interface, text

Description automatically generated

The next button was for the dataset generator and involved the creation of entries with sufficient error handling to allow the user to enter the desired name for the dataset. Text

Description automatically generatedThe following button would simply run the feature extract program described above, and this was the same again for the webcam recognition program when it came to the last button. Text

Description automatically generated

1. **Problems solved**

Whilst we faced many individual problems over the course of this project, the major problem we faced was implementing the mask detection itself.

1. **Future work**

If we are to continue to work on this project in the future, we would definitely focus our immediate attention upon improving the recognition model itself. Whilst we are quite satisfied with the levels of recognition that it can currently achieve as we feel it is satisfactory for this project, there are certainly areas of it that could be improved upon.

Obstacles such as glasses, long hair as well as lighting are areas with which our application can have trouble with, and if continuing to work on this project into the future it would be these obstacles that we would try and initially tackle and improve upon.

We also feel that there is a lot of potential for a project such as this to branch out to even further uses than we have designed it for, that being classroom attendance and mask detection and compliance in classrooms. Mask detection in particular could be an interesting area to further explore outside of just the classroom environment, though we are aware of the potential ethical concerns that may prove an expansion like this challenging.