A Study of Roll Call Automation using Machine Learning and Facial Recognition

Presented by **Guan-Zhong Wang**

Supervised by **Prof. Chun-Ming Tsai**

臺北市立大學
UNIVERSITY OF TAIPEI

A thesis submitted to University of Taipei
in partial fulfillment of the requirements for the degree of
Bachelor of Science with a major in Computer Science

Feb 2020

Contents

1 Introduction **2**

1.1 Introduction to Facial Recognition Technology (FRT) . . . 3

1.1.1 Histogram of Oriented Gradients (HOG) . . . 4

1.1.2 Face Landmark Estimation . . . 4

1.1.3 Deep Metric Learning . . . 5

1.1.4 k-Nearest Neighbor (k-NN) Classification . . . 5

1.2 Challenges of Automated Roll Calls . . . 6

1.3 Thesis Organization . . . 6

2 Related Works **7**

2.1 Facial Recognition Technology . . . 7

2.2 Barcode Scanning . . . 8

2.3 RFID Chip Scanning . . . 8

2.4 GPS Location Tracking . . . 8

3 Roll Call System using Facial Recognition Technology: PyRollCall **10**

3.1 System Overview . . . 10

3.2 PyRollCall Libraries . . . 13

3.2.1 OpenCV . . . 13

3.2.2 dlib . . . 13

3.2.3 face_recognition . . . 13

3.2.4 PyGTK . . . 13

3.3 The Design of PyRollCall . . . 14

3.3.1 Use Cases . . . 14

3.3.2 System Architecture . . . 15

3.3.3 Implementation 17

4 Experiments and Results **19**

4.1 Experimental Settings 19

4.2 Experimental Results 20

5 Conclusion and Future Work **21**

5.1 Conclusion 21

5.2 Future Work 22

Bibliography **23**

Abstract

With the rapid advancements in technologies over the past few decades, facial recognition software has become ubiquitous nowadays. Despite being a relatively new technology, facial recognition has been extensively employed in many products we use day-to-day. Large corporations, such as Google, Apple, Microsoft, have implemented biometric authentication in their products, enabling their users to incorporate such features into their daily lives. Intelligence agencies as well as police agencies have been using such technology to track down criminals and certain targets. Some governments have been using it to automate border control and even on some trivial tasks such as identifying the text on car plates, reducing a huge amount of tedious works that would have otherwise been carried out by human manually. Educational organizations could also benefit from facial recognition by utilizing it to automate roll calls, allowing professors and teachers to have more time to focus on teaching and allowing students to have more time to spend on learning during classes. Traditionally, a teacher needs to call each student's name one by one in order to check who is present at a class, and if each traditional roll call takes 5 minutes to perform and every semester consists of 18 weeks, then it could take 90 minutes in total to perform roll calls. In this thesis, we combined facial recognition as well as deep metric learning and proposed a functional roll call automation system which can be practically employed in educational organizations, aiding professors to record students' attendance. We will also evaluate the practicality of our method by comparing similar approaches with our methods such as signing into classes by scanning RFID chips, and signing in to classes via mobile devices using GPS location tracking. The result of this research is expected to reduce the chance of fraudulent class sign-ins that could arise in other similar methods, while ensuring the roll calls could be smoothly automated by our system.

Chapter 1

Introduction

Facial Recognition Technology (FRT)[1] has been extensively employed in modern software and products. Nowadays, various factions such as intelligence agencies, governments and corporations have already been using it to carry out certain tasks. For instance, Google has implemented biometric authentication in Google Pay, a digital online payment system which allows its users to make payments using their mobile devices. Apple has introduced *Face ID* to iOS, allowing its users to unlock their devices with facial recognition technology. Microsoft has developed *Windows Hello* in Windows 10, enabling Windows 10 users to login via their faces. Intelligence and police agencies have been using facial recognition to identify person of interest.

This technology can also be applied in educational organizations. Since lectures are limited in time, we can try to come up with a solution to automate roll calls, giving both teachers and students more time to spend on teaching and learning. If each traditional roll call takes approximately 5 minutes and every semester consists of 18 weeks, then a total of 90 minutes will be wasted as the semester ends. Therefore, not all professors would perform a roll call every time since it can be time consuming, and some students would take advantage of it by skipping classes while staying uncaught, leaving the grading results unjustified.

[1]We'll use the terms Facial Recognition Technology and FRT interchangeably in the following context.

Machine Learning and Facial Recognition technologies have come a long way and they are widely applied in various fields nowadays. In this thesis, we propose the architecture of a roll call automation system which utilizes *Deep Metric Learning* as well as *Facial Recognition* to automate the traditional roll call procedures, and we evaluate the practicality of this system by comparing alternative approaches with it.

1.1 Introduction to Facial Recognition Technology (FRT)

Facial recognition is a technology which is capable of identifying or verifying the individuals in a digital image or video. Generally, it consists of four steps:

1. Face detection
2. Face normalization
3. Facial feature extraction
4. Classification

The first step of facial recognition is to identify all the faces in it. The biggest challenge of face detection is that faces can be tilted and turned to any direction, and the program must be able to detect them regardless of these situations. There are also some other factors that can prevent the faces from being correctly detected: lighting, expressions, noises, face occlusions, etc. Secondly, the faces have to be normalized before performing facial feature extraction. This step can include geometric normalization, lighting normalization, angle normalization.

Now we are ready to extract the unique features from a face, but questions arise: which features should we collect, and how should we quantify a face? It can be the width between the eyes, the length of the nose, or even the width-height ratio of the face. It turns out that the best approach to extract facial features is by using deep learning. This is usually done by training a deep Convolutional Neural Network (CNN) which can generate a face encoding for each face in the image. The last step is comparing this face encoding with other face encodings that already exist in the database. If a match is found, then we've successfully identified the face of an individual. The above process is actually similar to how we identify people in real life:

1. Find all the faces within our vision.
2. Identify the faces regardless of their angle, direction and lighting.
3. Look for the unique features of the faces.
4. Compare these unique features with all the people you know.

1.1.1 Histogram of Oriented Gradients (HOG)

Face Detection went popular in 2001 when Paul Viola and Michael Jones published the *Viola–Jones object detection framework*, enabling objects on a camera to be detected in real time. Later on, more robust approaches are proposed. We will be using Histogram of Oriented Gradients (HOG) in our system for this task.

Histogram of Oriented Gradients (HOG) is a feature description technique used in image processing with the aim of object detection. The main idea is to split an image into several portions of the same size (since doing this for every pixel will give us too much information more than we actually need), and then find out the major direction which brightness changes for each portion. The result will be a representation of the fundamental structures of a face. Now we can look for the regions which are similar to a known facial pattern generated from a huge amount of facial images. This way we can detect all the faces in an image.

1.1.2 Face Landmark Estimation

Face Alignment is an important step in facial recognition. Before performing facial feature extraction, geometrically normalizing faces can greatly improve the accuracy of facial recognition. We will attempt to locate 68 facial landmarks by estimating their positions with an algorithm proposed by Vahid Kazemi and Josephine Sullivan in 2014 that is packaged with *dlib*, and then use affine transformation to align the faces.

1.1.3 Deep Metric Learning

Previous research has shown that deep learning does a better job than human in knowing which facial features should be measured. Deep Metric Learning, a machine learning model, plays a vital role in modern facial recognition technology. It works by training a deep Convolutional Neural Network (CNN) to generate 128-d measurements for each face. The training process is typically carried out using a *Triplet Network* which requires three images as input:

1. an image of person A
2. another image of person A
3. an image of person B

In each step of training, the algorithm will tweak the neural network, making the measurements of photo 1 and 2 slightly closer to each other while making the measurements of photo 2 and 3 slightly different from each other. The training process only have to be done once and it has already been done by Davis King, the original author of *dlib*. In addition, the trained neural network is publicly available so that we don't have to train the network ourselves. We can directly use the pre-trained network to generate measurements for any face.

1.1.4 k-Nearest Neighbor (k-NN) Classification

The last step in facial recognition is running a classification algorithm to find out which category the object in the image belongs to. The k-NN algorithm achieves this by using the distances of feature vectors to determine k "nearest" data points. In these k "nearest" data points, there could be several different categories, and the category with the most data points would be chosen as the result of classifictaion. In our system, we use k-Nearest Neighbor (k-NN) classifier to identify whom a face belongs to.

1.2 Challenges of Automated Roll Calls

Human beings can correctly identify familiar faces even under certain circumstances such as poor illumination, facial expressions, partial face occlusions, aging and facial appearance changes. However, this might not always be the case for a computer system. Partial face occlusions and facial appearance changes can pose two of the most challenging problems in an FRT roll call system.

There are two typical ways to use the roll call automation system, either (1) take photos of all students individually, or (2) take photos of all students at once. The former approach can take much longer time than the latter one, but it is obviously easier to request students to remove the accessories that occlude some parts of their faces. The latter method requires only one photo taken but it could be harder to achieve, since not all students would simply cooperate.

1.3 Thesis Organization

This thesis is divided into five chapters. In Chapter 2, we review the related works done in the field. In Chapter 3, we deal with the design and implementation of our roll call automation system. In Chapter 4, we evaluate the practicality and performance of the system. In Chapter 5, we summarize our conclusions and discuss the opportunities for future research.

Chapter 2

Related Works

In this chapter, we review the related works on roll call systems using: (1) FRT, (2) barcode scanning, (3) RFID chip scanning and (4) GPS location tracking. Although various methodologies has been proposed in previous studies, roll call systems using FRT are less frequently studied comparing to other ideas. Currently, there is no "the best way" to perform a roll call, since each approach will usually have its own vulnerabilities and can be exploited by the students. Hence, we will also discuss the pros and cons of each methodology.

2.1 Facial Recognition Technology

William E. Miller (2018) proposed a solution [1] to recording classroom attendance using Facial Recognition Technology (FRT). The author provides three Python scripts in its appendicies which can assist school faculties in performing automated roll calls using FRT, but it only supports Windows and it seemingly lacks a user-friendly Graphical User Interface (GUI). Consequently, we perform extensive research in this area based on his previous works, proposing the architecture of a cross-platform and easy-to-use roll call automation system. By making this system cross-platform, we open up many possibilities in this area such as enabling the installation of our system on a portable embedded system like Raspberry Pi.

2.2 Barcode Scanning

Several cram schools in Taiwan, such as Chen-Li Educational Group, are known for using barcode scanning to record student attendance. When students arrive at the cram school, they are required to show their student ID card to the staff before they can enter the classroom. Their student ID card has a barcode printed on it and the staff will use a barcode scanner to record the attendance of each student. Although this approach has been implemented for a while, it can be inefficient if the size of classroom is too large, since students usually have to wait in a very long line before entering the classroom.

2.3 RFID Chip Scanning

Ching Hisang Chang (2012) published a paper [2] regarding an RFID roll call system. By installing an RFID reader near the entrace of a school or classroom, students' attendance can be recorded with ease as they walk pass the RFID reader. In comparison with the FRT approach, this method offers the best compromise between convenience and security since no personal information have to be collected at all, while FRT roll call system requires students to provide their personal photos in advance. However, this approach can lead to fraudulent sign-in activites because one can simply carry the student cards of their other fellow students and exploit the mechanism of the RFID roll call system.

2.4 GPS Location Tracking

Zuvio IRS [3] is a digital education tool which has been widely used in many Taiwanese universities including National Taiwan University, National Central University, National Taiwan Normal University, etc. It allows teachers and students to perform online quizzes, discussions and roll calls. The most important feature we will be discussing here is *Online Roll Calls using GPS Location Tracking.* When a teacher starts a roll call session, the students are required to use the Zuvio application on their mobile devices to sign in to a class. As a student clicks the ”sign in” button, the system will check their location using GPS, and the students are expected to be near to the classroom. However, this system can be exploited with a GPS location spoofing program which can fake users' locations, leading to possible fraudulent sign-in activities.

Chapter 3

Roll Call System using Facial Recognition Technology: PyRollCall

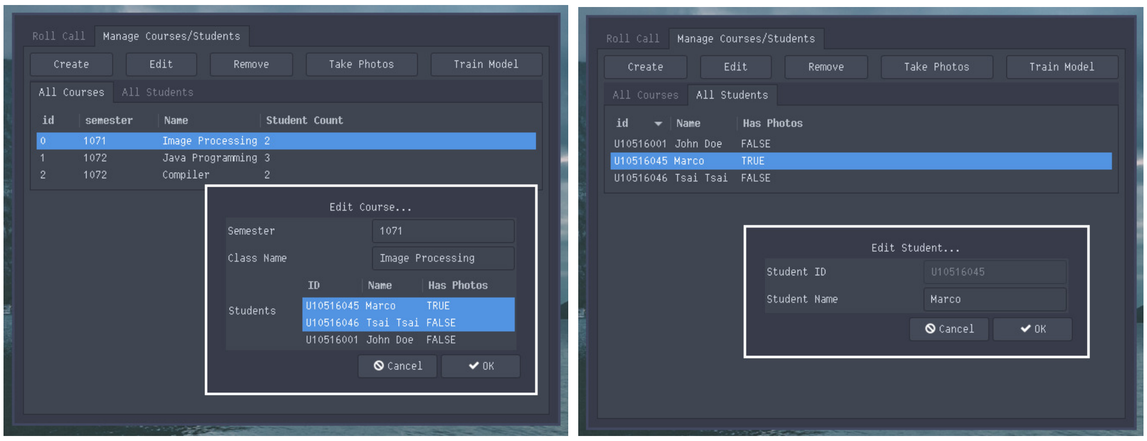
In this chapter, we firstly discuss the usage of PyRollCall, our roll call system using FRT. Next, we review the design of PyRollCall's system architecture, and get into the implementation details of our system.

3.1 System Overview

Our system's primary goal is to help school faculties perform roll calls with ease. Traditional roll calls require teachers to call the names of students individually to record students' attendance, while PyRollCall let a teacher simply take a photo of the entire class and record students' attendance using Facial Recognition, thus saving more time for both teachers and students in classes.

Before we can actually use the system to recognize the faces of students, teachers should collect at least 10 to 15 photos for each student, and let the system compute the measurement (or embedding) of each face. These pre-computed facial embeddings will be saved in the database for later use. When a face is caputured in roll calls, the system will use k-NN classification and votes to determine whom the face belongs to.

The user (usually a teacher) should enter the names of the courses into the system, as well as all students' names and IDs, and for each course select which students are taking it. After entering these data into the system, PyRollCall will be ready to perform roll call for a specific course using facial recognition. To summarize, users should (1) collect approximately 10 to 15 photos for each student and let PyRollCall pre-compute their facial embeddings, and (2) set up the courses' and students' data. Figure 3.1 demonstrates the GUI used to manage the data of courses and students.



id	semester	Name	Student Count
0	1071	Image Processing	2
1	1072	Java Programming	3
2	1072	Compiler	2

Edit Course...

Semester: 1071

Class Name: Image Processing

Students:

ID	Name	Has Photos
U10516045	Marco	TRUE
U10516046	Tsai Tsai	FALSE
U10516001	John Doe	FALSE

(a) Managing courses.

id	Name	Has Photos
U10516001	John Doe	FALSE
U10516045	Marco	TRUE
U10516046	Tsai Tsai	FALSE

Edit Student...

Student ID: U10516045

Student Name: Marco

(b) Managing students.

Figure 3.1: PyRollCall running on a Linux machine with X11[1] installed.

When a roll call ends, the system automatically exports a statistical report to a file. These reports will be categorized nicely into different directories by date, with the filename being the course name. Listing 3.1 demonstrates an example file containing the result of a roll call.

[1]X11, also known as X Window System, or simply X, is a windowing system that provides the basic framework for building GUI environments.

Listing 3.1: Layout of PyRollCall's root directory.

```
$ cat logs/2019/01/01/Java.txt
Course Name: Java
Export Time: 2019/01/0120:37
Arrived: 8 / Late: 2 /
Total: 10
Late Students: Tsai,John
```

Now that we have understood the prerequisites of the facial recognition features in PyRollCall, it is time that we get to the installation details of this system. PyRollCall is able to run on Windows, macOS and unix-like systems as long as the system supports Python 3 and GUI applications. Furthermore, PyRollCall is open-source and managed with *git*[2], and all of its external dependencies are already packaged in the project via Python's virtualenv[3], thus the users do not have to worry about what and which versions of libraries they need to install. To checkout and run the project:

Listing 3.2: Shell commands to checkout and run PyRollCall

```
$ git clone https://github.com/aesophor/pyrollcall.git
$ cd pyrollcall
$ source venv/bin/activate
$ ./pyrollcall.py
```

The entry point to the system is *pyrollcall.py*, a script located under the project's root directory. Before running it, source the shell script *venv/bin/activate* to activate the project's virtual environment.

[2]git is a distributed version control system used for tracking file content changes in source code during software development

[3] virtualenv is a tool to create isolated Python environments by maintaining a set of libraries that a specific application depends on.

3.2 PyRollCall Libraries

3.2.1 OpenCV

OpenCV (Open Source Computer Vision) [4] is a cross-platform computer vision and machine learning library which was originally developed by Intel Corporation. It is used to develop real-time image processing, computer vision and pattern recognition programs. The library is primarily written in C++, thus most interface of OpenCV is for C++ and C. It also has bindings for other programming languages such as Python, Java and MATLAB. Our project uses OpenCV to capture images from cameras.

3.2.2 dlib

Dlib [5] is a C++ toolkit containing machine learning algorithms and tools to solve complex real world problems. It is widely used in robotics, embedded devices, mobile phones, and large high performance computing environments. Despite being written in C++, dlib also has bindings for Python. The use of *dlib* in PyRollCall is employing deep metric learning algorithms and dlib models to solve facial recognition tasks.

3.2.3 face_recognition

face_recognition [6] is an easy-to-use face recognition library for Python which is built upon dlib's state-of-the-art deep metric learning model with an accuracy of 99.38%. We use face_recognition to detect faces in an image, estimate facial landmarks, compute a 128-d measurement for each face and perform facial classification.

3.2.4 PyGTK

PyGTK [7] is a set of Python wrappers for the GTK+, a free, open-source and cross-platform widget toolkit for developing GUI applications. It was originally developed in C by GNOME developer James Henstridge. PyGTK will be phased out and replaced with PyGObject which uses GObject introspection to dynamically generate bindings for Python and other languages, eliminating the delay between GTK updates and binding updates for other languages.

3.3 The Design of PyRollCall

In this section, we present the use cases and the design of PyRollCall's architecture. In the following context, a "user" refers to a "teacher" who wishes to perform roll calls using PyRollCall, since the end user of this system is typically a teacher. In other words, students are not the direct users of this system. Figure 3.2 shows the use case diagram of PyRollCall, whereas Figure 3.3 shows the system architecture of PyRollCall.

3.3.1 Use Cases

- Users can maintain the data of the courses and students they teach.
- Users can collect students' photos and generate their facial measurements.
- Users can start a roll call, recording students' attendace via FRT.
- Users can export the results of roll calls to files.
- Users should be able to keep their data the next time they use the system.

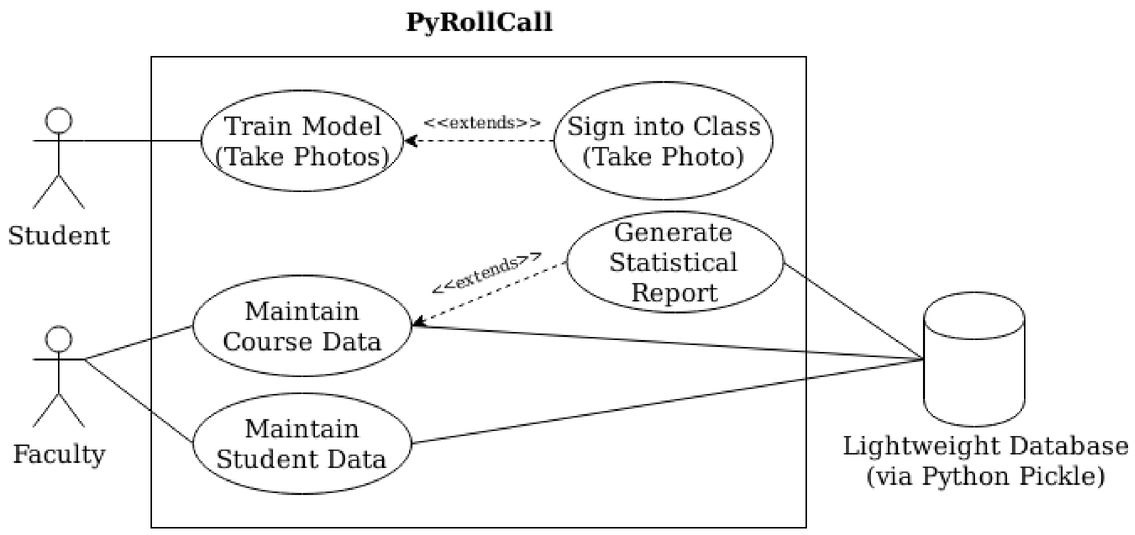

PyRollCall
Student
Faculty
Train Model (Take Photos)
<<extends>>
Sign into Class (Take Photo)
Generate Statistical Report
<<extends>>
Maintain Course Data
Maintain Student Data
Lightweight Database (via Python Pickle)

Figure 3.2: Use Case Diagram

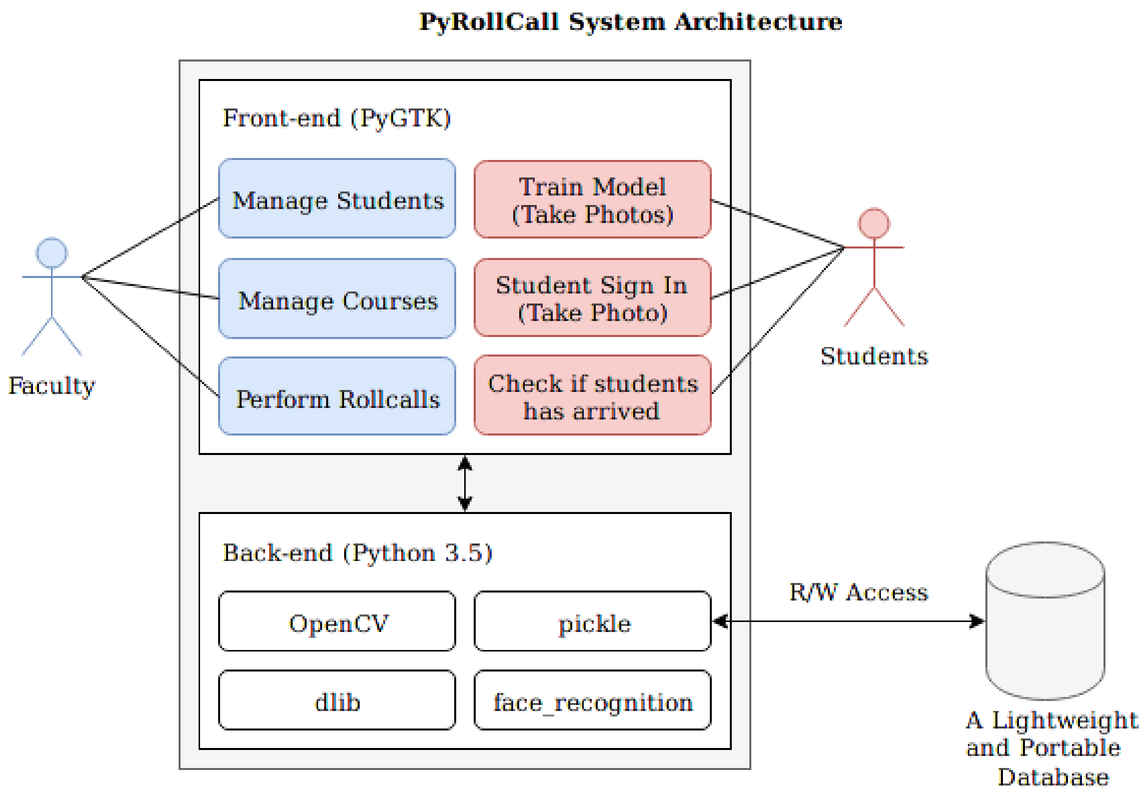

PyRollCall System Architecture
Front-end (PyGTK)
Manage Students
Train Model (Take Photos)
Manage Courses
Student Sign In (Take Photo)
Perform Rollcalls
Check if students has arrived
Faculty
Students
Back-end (Python 3.5)
OpenCV
pickle
dlib
face_recognition
R/W Access
A Lightweight and Portable Database

Figure 3.3: System Architecture

3.3.2 System Architecture

Figure 3.3 shows the system architecture of PyRollCall. The GUI is built with *PyGTK*, and it is already included in the project's *virtualenv*, making PyRollCall a highly portable system. The user can manage the data of courses and students, take photos for students, compute their facial embeddings and perform roll calls via GUI. Underneath the GUI, we use *OpenCV* to capture images via cameras, *face_recognition* and *dlib* to recognize faces of students, and *pickle*[4] to serialize and deserialize facial measurements (or facial embeddings).

The layout of project's root directory is presented in listing 3.3. Note that trivial entries are omitted here for obvious reasons. The entry point to the entire system is *pyrollcoll.py*, whereas all modules are placed inside the *pyrollcall* directory. The

[4]pickle is a Python module which implements binary protocols to serialize and deserialize Python objects.

user's data is stored in *rollcall.db*, a lightweight database implemented with Python's pickle, inclusive of the courses he or she teaches, the course-student mappings and all the pre-computed facial embeddings of students.

Listing 3.3: Layout of PyRollCall's root directory.

```
$ ls -la
total 48K
drwxr-xr-x 3 aesophor aesophor 4.0K Dec  1 14:04 faces
drwxr-xr-x 2 aesophor aesophor 4.0K Jan 25 11:10 pyrollcall
drwxr-xr-x 5 aesophor aesophor 4.0K Nov 13 09:13 venv
drwxr-xr-x 7 aesophor aesophor 4.0K Jan 25 12:12 .git
-rw-r--r-- 1 aesophor aesophor  258 Jan 23 12:40 rollcall.db
-rwxr-xr-x 1 aesophor aesophor  197 Sep 26 20:33 pyrollcall.py
```

The *pyrollcall* directory contains PyRollCall's modules and class definitions, as shown in listing 3.4. Note that trivial entries are also omitted here. Readers have probably noticed that there's also another *pyrollcall.py* in this directory, this module contains code that will initialize the user's data from *rollcall.db* and start the GUI, while the top-level *pyrollcall.py* merely serves as the entry point to the system.

Listing 3.4: PyRollCall's modules and classes.

```
$ ls -la pyrollcall
total 64K
-rw-r--r-- 1 aesophor aesophor 1.2K Sep 26 20:33 course.py
-rw-r--r-- 1 aesophor aesophor 3.1K Sep 26 20:33 database.py
-rw-r--r-- 1 aesophor aesophor 5.8K Jan 18 14:22 face.py
-rw-r--r-- 1 aesophor aesophor  493 Sep 26 20:33 __init__.py
-rw-r--r-- 1 aesophor aesophor  18K Sep 26 20:33 mainwindow.py
-rw-r--r-- 1 aesophor aesophor  399 Sep 26 20:33 pyrollcall.py
-rw-r--r-- 1 aesophor aesophor 2.0K Sep 26 20:33 session.py
-rw-r--r-- 1 aesophor aesophor  827 Sep 26 20:33 student.py
-rw-r--r-- 1 aesophor aesophor  399 Sep 26 20:33 utils.py
-rw-r--r-- 1 aesophor aesophor 4.2K Sep 26 20:33 widget.py
```

Among all modules, the most important one is *face.py*, a module which contains code to capture images with a camera, compute facial embeddings and recognize faces, the details of which will be covered in the next subsection. The photos of students should be of png format with their filename being *name_number.png*. Furthermore, photos of a specific student should be grouped into a single directory with the directory name being *studentID_name*. All directories containing students' photos should be put in the *faces* directory. A minimal example is provided in listing 3.5; the student's name is "Marco" and the student's ID is "U10516045".

Listing 3.5: Example hierarchy of the *faces* directory.

```
$ tree faces
faces
|____ U10516045_Marco
    |__ Marco_0.png
    |__ Marco_1.png

    1 directory, 2 files
```

To use PyRollCall completely without a camera connected to the computer, simply follow the rules mentioned above and place the files in the corresponding directories with correct filenames.

3.3.3 Implementation

The heart of PyRollCall is *encode_faces()* and *recognize_faces()* in *face.py*. The first function, encode_faces(), enumerates all the images in the *faces* directory and compute the embeddings of each faces in the images. Each image in this directory should contain exactly one face in order to prevent erroneous results during k-NN classification.

encode_faces() takes two arguments: an instance of Database and a string specifying the absolute path to the *faces* directory. It extracts the student's ID from the directory name, get the bounding box of each face, compute the embeddings for faces inside the bounding boxes, and save all the embeddings to the database. The idea is shown in Algorithm 1.

Algorithm 1 Pre-compute facial embeddings (encodings) of all faces in the *faces* directory.

```
procedure ENCODE_FACES(database, faces_dir)
    for each image in faces_dir do
        $box \leftarrow get\_bounding\_box(image)$
        $e \leftarrow compute\_encoding(image, box)$
        $id \leftarrow get\_student\_id(image)$
        database.face_encodings.insert(FaceEncoding(e, id))
    end for
end procedure
```

recognize_faces() takes two arguments: an instance of Database and an image object. Firstly, we detect all bounding boxes of the faces in the given image, and then compute all facial encodings in this image according to where these bounding boxes are located. Secondly, for each facial encoding generated from this image, we compare it with all known encodings in the database. Each comparison should give us an array of boolean values, we can then count the votes, find out which student has the highest votes, and mark that student as arrived. The pseudocode is shown in Algorithm 2. Please note that the pseudocode shown in this subsection do differ from the actual implementation, but the general idea is exactly the same.

Algorithm 2 Recognize faces in an image and mark corresponding students as arrived.

```
procedure RECOGNIZE_FACES(database, image)
    $boxes \leftarrow get\_bounding\_boxes(image)$
    $encodings \leftarrow compute\_encodings(image, boxes)$

    for each e $\in$ encodings do
        $array\_of\_votes \leftarrow compare\_faces(database.face\_encodings, e)$
        $student \leftarrow get\_student\_with\_most\_votes(array\_of\_votes)$
        student.set_arrived(True)
    end for
end procedure
```

Chapter 4

Experiments and Results

4.1 Experimental Settings

In this section, we assess the practicality of PyRollCall. Two of the most important functionalities of a facial recognition roll call system are: (1) face detection and (2) facial recognition. If these two features work accurately, then the system is considered to be working as intended. Firstly, we evaluate the performance of face detection. Figure 4.1 shows that PyRollCall is able to detect all faces within the image regardless of head tilt, rotation and even facial expressions.

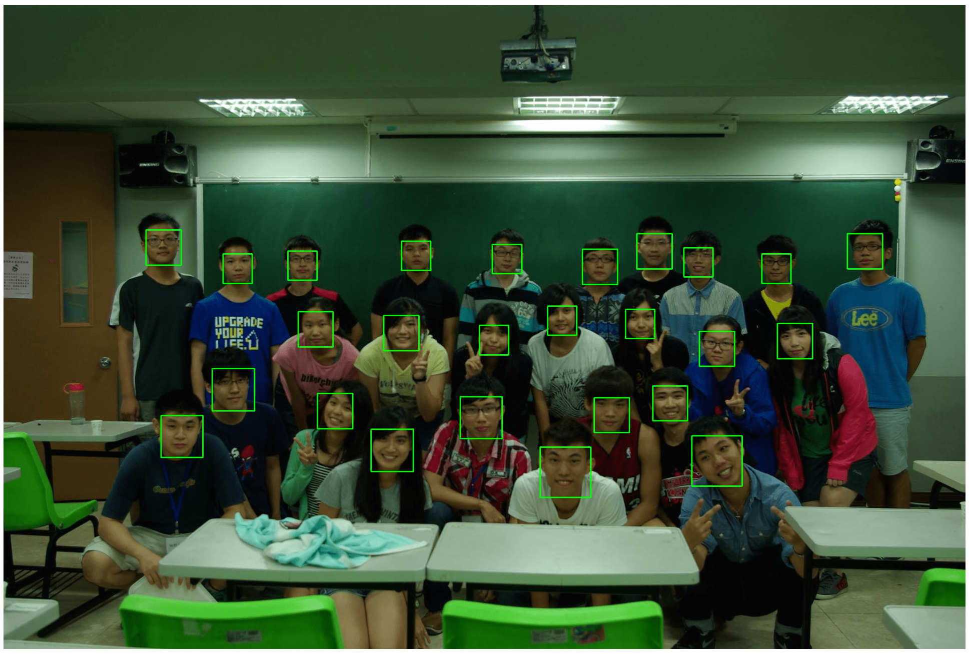


Figure 4.1: Face Detection

In our experiments, all faces are facing upfront to the camera, and thus the successful rate of face detection is rather high. In real-world scenarios, the faces of students may be turned into various angles and the results of face detection will vary. Hence, when taking photos from the students, the teacher will have to ask for the attention from the students for a few seconds.

Secondly, we evaluate the effectiveness of facial recognition. Figure 4.2 demonstrates a successful facial recognition performed by PyRollCall. In order to make the facial recognition feature works correctly, the teacher should collect 10 to 15 photos (or ideally 15 to 25 photos) from each student and let PyRollCall pre-compute all of the facial embeddings from each photo.

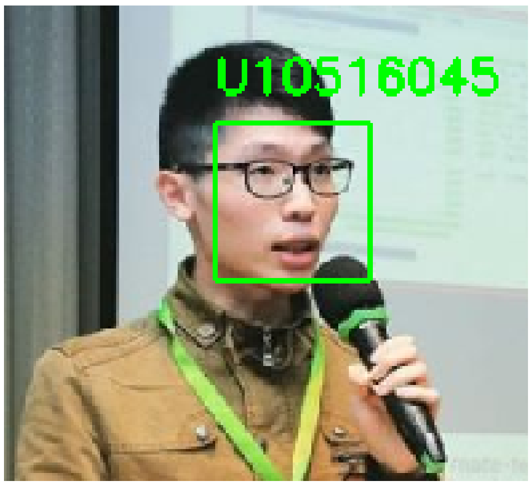


Figure 4.2: Facial Recognition

4.2 Experimental Results

With sufficient photos of students provided and their facial embeddings pre-computed, the system will be able to detect and recognize faces correctly. However, to save time in classes, teachers and students will have to spend equivilently extra amount of time before classes on the tasks such as collecting photos and pre-computing facial embeddings. This proves that there is a trade off between convenience and efficiency.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this thesis, we have proposed a roll call system using Deep Metric Learning and k-NN classification. The system enables educational organizations to record students' attendance simply by taking a photo of the entire classroom, thus saving a huge amount of time for both teachers and students. We have also evaluated the practicality of our system by examining the accuracy of facial recognition.

Implemented in Python 3.5 and with the help of *virtualenv*, PyRollCall is cross-platform and highly portable, since all of its dependencies are already packaged with the project. It is implemented with *OpenCV* for capturing images from a camera, with *face_recognition* which is a wrapper library of *dlib* for performing facial recognition with Deep Metric Learning, and with *PyGTK* for designing a user-friendly GUI.

There are two prerequisites which should be fulfilled before using PyRollCall:

1. The user should enter the data of all students and courses he or she teaches into the system; for each course, select the students that are currently taking it.
2. Collect approximately 10 to 15 photos from each student and let the system pre-compute their facial embeddings (or encodings).

5.2 Future Work

Finally, we discuss the opportunities for future work. Currently, the system is decentralized and is designed for faculty's personal use, but future work can be done to expand the size of this project.

- The system architecture can be adjusted, allowing multiple users to use the system simultaneously.
- RESTful technologies can be employed to integrate the roll call system into student information systems.
- Advanced features such as notifying the parents of a student via SMS can be developed.

Bibliography

[1] William E. Miller (2018). Utilizing Facial Recognition Software to Record Classroom Attendance.

[2] Ching Hisang Chang (2012). Smart Classroom Roll Caller System with IOT Architecture.

[3] Zuvio IRS, An Interactive Educational Tool for Universities.
`https://www.zuvio.com.tw/student`

[4] OpenCV (Open Source Computer Vision Library), A cross-platform Computer Vision and Machine Learning Library.
`https://opencv.org/`

[5] Davis King. Dlib, A C++ Toolkit Containing Machine Learning Algorithms and Tools to Solve Complex Real World Problems.
`http://dlib.net/`

[6] Adam Geitgey. face_recognition, The World's Simplest Facial Recognition API for Python and the Command Line.
`https://github.com/ageitgey/face_recognition`

[7] PyGTK, A Convenient Wrapper for GTK+ Library for Use in Python Programs.
`https://wiki.gnome.org/Projects/PyGTK`

[8] Adrian Rosebrock (2018). Face Recognition with OpenCV, Python, and Deep Learning.
`https://www.pyimagesearch.com/2018/06/18/face-recognition-with-opencv-python-and-deep-learning/`

[9] Himanshu Mallik (2015). An Automated Student Attendance Registering System using Face Recognition.

[10] Abidi, M.A. and Gonzalez, R. C. (1992). Data Fusion in Robotics and Machine Intelligence. Academic Press, New York.

[11] Abramson, N. (1963). Information Theory and Coding. McGraw-Hill, New York.

[12] Md. Shafiqul Islam, Asif Mahmud, Azmina Akter Papeya, Irin Sultana Onny (2017). Real Time Classroom Attendance Management System.