PPM Project Report - Group Kii

Magnus Frater System

Callum Axon (N0727303), Callum Carney (N0741707), Jordan Brightmore (N0732961), Finlay McKinnon (N0743587), Vital Harachka (N0731739), Wing Chiang (T0086366)

Abstract

Magnus Frater (or Big Brother) has been created to help tackle the ongoing issue of security within large open campuses and premises, these sorts of locations inherently have an increased potential for intrusion through unmonitored sections of land. The group analysed the recent spree of attacks on schools and offices (for example the shooting that occurred at the YouTube headquarters in 2018) and found that in a large amount of these attacks there were open doors and spaces that allowed the attacker to enter with ease. As a consequence to this, the idea of creating a facial recognition system to analyse and report known and unknown people within a campus/large open setting was conceived.

As mentioned, the main purpose of the project was to create a system that would accurately detect and report people walking around an area to the associated security team, this data would differentiate between employees or authorised users and unknown people by linking into the companies employee/student database. Not only would this allow a security team to monitor who is within a set area at any one time, but it would also allow administrative users to track any persons movements and activities within a set time frame, through tracking of the targets face across multiple cameras. Another advantage to this project is that administrative users can view analytics in relation to the usage of campus properties, an example use case for this would be within a University. Admins could check what buildings within the campus are being utilised most by students.

After the main purpose behind the project was defined, the group decided on how to proceed in regards to the requirements for the project, most importantly how we should proceed with splitting up the individual hardware and software components so that the system could functions within any scenario or environment. It was decided that there will be 4 different modules, these being:

1. A Raspberry Pi that would be responsible for processing any facial data that is captured by the camera
2. A Camera module that would connect directly to the Raspberry Pi and provide images to the Raspberry Pi
3. A website created for administrators and security personnel to administer and manage hits/rejections.
4. An API (Application Programming Interface) used within the website and the Raspberry Pi for collation and provision of data.

These modules will work together to create the Cameras that report facial data and the web interface that is used to manage the data received by the camera, the connection between these modules was outlined in the design documentation (for example the Data Flow Diagram and Entity Relationship Diagram).

Once the components and requirements were completed, the group began to consider which programming languages and setups would be best suited for the type of project this is (Facial Recognition with

Web Related components). It was clear that Python should be used for the facial recognition section of the project due to its strong existing libraries. NodeJS would be used for the Web Frontend, PHP would be used to power the backend API that links all of the components together and the API would be using a MySQL database to hold all of the data. The system would work in the following way:

1. The Camera feeds data to the Raspberry Pi
2. The Python application on the Raspberry Pi calculates if a face is present
3. Any potential face found is sent to the API where corresponding facial data is requested from the database
4. If no corresponding data is found, then the face is unknown, otherwise the image will be linked to the person the face associates with.
5. The Website will update using data from the API to show new detections, known or unknown.

Once the product had been developed, testing took place to ensure that the facial recognition software worked from a variety of different distances and in unfavourable circumstances (heavy rain, fog, etc). While some of the tests passed, others failed to detect faces when they were present, however this only occurred in extreme circumstances. We made small enhancements to the facial detection algorithm to improve its effectiveness during these scenarios.

Due to the nature of this system, there are a lot of potential legal and ethical issues, people may not consent to the recording of their faces, people may not wish to have their faces processed and stored by this system. Therefore it was important for us to implement a blacklist system that would stop the system from performing facial data processing, however, this is a complex system because we first need to process a persons face to understand what to blacklist, which could cause further legal or ethical issues.

Table of Contents

- Abstract
- Table of Contents
- Introduction
  - Aims
  - Objectives
- Requirements
  - Functional Requirements
- References

Introduction

Aims

The main aim of the project is to provide organisations with open campus settings a way to effectively track and monitor who is on campus and where they are located at any time of day. This is to help reduce or prevent intrusions and attacks that occur on these types of locations.

Objectives

To ensure clear and appropriate objectives have been created for the project, the SMART (specific, measurable, achievable/appropriate, realistic, time-constrained) goals were used. SMART allows us to create objectives that provide the project with lots of functionality, that will be meaningful to the objects, and still stay within the projects deadlines. For the project to be successful the following objectives should be met:

Staff Members should be able to: * Add new faces or people to the system through a simple yet effective web interface; * Monitor the movements of people across buildings and campuses, whether they are registered as people or not; * Manage alerts of unknown people entering the campus; * Provide temporary passes to unknown people to authorize them for a set amount of time;

As a requirement to this, the camera and associated Raspberry Pi module should be able to provide the following: * A stream of video that can be analysed by the algorithm on the Pi in order to find faces; * A constant stream of face detections to the central server that manages all hits;

In general the following objective should be met: * The camera and web interface should be able to talk to each other through an API (Application Programming Interface)

To meet the objectives set out for Staff Members, the group will be creating a web interface using NodeJS, this interface will have the functionality set out above and will interface with the API to get and set data. It was mentioned that the interface should be simple yet effective, we could easily bombard the user with a lot of metadata from the cameras, however, the web interface will only show the required information and actions to ensure that a staff member can quickly and easily identify if there is an intruder currently on campus. All of the outlines objectives are achievable and can be implemented in a timely manner.

To meet the requirements set out for the Raspberry Pi and API, we will have to ensure substantial testing of the facial detection algorithm takes place, the group wants to avoid experiencing a scenario in which multiple people are not identified. However, the algorithm cannot be 100% effective, there will always be scenarios in which the algorithm misses a person, or mis-identifies them, it would be unrealistic and a waste of development time to be chasing after a 100% success rate. We will also have

to ensure that the API is tested thoroughly, not only for functionality, but for security purposes, if an attacker gained access to another users facial data then this would be a breach of GDPR, therefore we will be implementing multiple security procedures to ensure that the API is secure, including the use of security based unit testing and manual testing.

Requirements

Functional Requirements

FR#	Function	Goal	Actor	Justification	Importance Rating (out of 5)
1	Face Scanning	A stationary camera is able to detect a face and scan certain data points for analysis	Stationary Camera	In order to provide a product that tracks people on a large campus, we must have an effective face scanning algorithm to track people across cameras	5 - This functionality is required for the system to work
2	Position Reports can be filed	Once a person has been identified all of the associated metadata is compiled and submitted as a report to the API	Camera - Rasp-berry Pi	In order to provide person tracking functionality the API must recieve compiled position reports to query at a later date, without these the application would loose a large portion of functionality.	5 - This functionality is required for the system to work properly

FR#	Function	Goal	Actor	Justification	Importance Rating (out of 5)
3	New facial data can be added to the system	An administrative user must be able to upload new facial data to be detected at a later point in time	Administrat User	In order to match new faces to current people, an original image of a persons face must be uploaded to the system so that the two images can be compared at a later date	5 - FR4 requires this function to exist
4	New facial data is processed when uploaded to the web interface	Once an image of a person has been uploaded the associated facial data points are created and stored	API	In order to compare two faces, the system needs to generate data points from the two images and then compare the data points to calculate who has been detected	5 - Without this functionality the system would not be able to discover people
5	A person can be discovered when they have a valid position report	If a member of the security team is looking for a person, they can search and find the related position reports	Security Personnel	A person must have position reports associated with them to allow the security team to search for them and discover their past or present location	3 - The application will still function without this, however a large piece of functionality would be missing

FR#	Function	Goal	Actor	Justification	Importance Rating (out of 5)
6	A person can be located within a Campus/Location	A person must be able to be located within a campus setting.	Security Personnel	In order to allow security personnel to find people within a certain location, there must be functionality to discover a person.	3 - The application will still function without this, however a large piece of functionality would be missing
7	A temporary pass can be assigned to a person	In order to be able to allow unknown users to walk around a campus without causing alerts a temporary pass can be assigned	Security Personnel	In order to lower the amount of False Negatives within a system, administrators can assign temporary passes that will allow unknown people to walk around the campus without causing alerts	4 - The application will still function without this, however a very important feature would be missing
8	List Buildings	Display a list of buildings	Web Interface	In order to display required information to users of the system, there must be functionality to display added buildings	4 - The application will still function without this, however a very important feature would be missing

FR#	Function	Goal	Actor	Justification	Importance Rating (out of 5)
9	Add Buildings	Add a building	Web Interface	In order to manage cameras, buildings must be added so that cameras can then be associated with them	4 - The application will still function without this, however a very important feature would be missing
10	Add Cameras	Add a camera	Raspberry Pi	In order to link person discovered with cameras a camera must first be enrolled onto the system, this occurs within the Python applications code	5 - This functionality is required for the system to work properly

References

[] [@infotech_2017] [@wayner_2019]