Introduction

My project is based around a facial recognition door security system using a Raspberry Pi 3B. This is a relatively inexpensive ARM based computer that is around the same size as a credit card and the same weight as a light mobile phone. This makes the Pi easy to mount in a case anywhere the security system may want to be used by the client. The security system works based on entering faces into a Python list-like data structure. The program will look for faces in a viewport from a Pi Camera, and using a file of trained faces, return the person that it predicts is in the viewport. I will be using a library such as the OpenCV library in python as a wraparound for the facial recognition portion of my project, along with many other libraries to make the project complete. The data for the faces can be entered at any time using a person stood physically at a camera and a dataset then being captured then and there. I will later add the functionality of a web interface to allow people to upload multiple photos of people from a mobile or desktop device, in addition to the option of them standing physically at the camera. This would be beneficial as it is not always possible to have a user on site at short notice, this allows the user to enter their facial data from anywhere. Once entered, the camera can be left in a constant ‘monitor mode’. ‘Monitor mode’ will scan the camera viewport for anything that resembles a face – if one is found , it will then be cross referenced to the data structure for faces that have been trained into the system. An unknown face will be picked up by the system if it was not a match to any in the database, returning an ID of 0 and a nametag of ‘Unknown’. In this case, a SMS message will be sent to a target mobile device using the Twilio API to inform the client of an unknown person at the camera. In ‘monitor mode’, the camera will also record to disk a snapshot of the viewing area of the stream at any time any unknown face is recognised, with overlays to display where the face is in the image – this is to keep records of intruders into the property for later use. As well as this, the user can choose to take a snapshot at any other time other than automatically if they wish. In order to comply with data protection laws surrounding the use of biometric data, the user will have to agree to a use-policy and privacy notice by using the device. For example, it is forbidden for a user to add biometric data for someone who has not given their express consent and it is the responsibility of the user to ensure they do not do this. That is the reason the physical facial enrolment is fundamentally a lot more compliant with data protection laws than the web upload option – it would be masses more difficult to get someone to enrol their face without their permission if they were not going to be stood at the door facing the camera. Unknown faces are not trained by the system, and the Haar Cascade will only detect them as a face without any other analysis on the data and treat them with complete anonymity. The web interface will be reasonably straightforward and not too advanced; a viewport for watching a live stream, buttons to perform actions such as training models or capturing data and downloading photos.

Solving this project by using computational thinking – what features allow this?

Dans work did not have these two parts set in clear like this, im going to leave the headers here and come back to them later. I looked at the textbook but I still did not understand these areas

Solving this project by using computational thinking – why is it amenable to a computational approach?

Stakeholders

My project is suited towards anyone over the age of 16 and/or a homeowner with the interest of their home security being close to their heart. It is probable that the user interface will require a small amount of technological literacy and it is true that it will not be fool proof. However, the GUI will make the program much easier to use its CLI counterpart – which daunts most novice users and would be described by many to be ‘not user friendly’. Most of the processing is done behind the scenes and the only operation is adding photos online, enrolling faces physically and downloading snapshots of the Viewport from the Pi. Although I am using the Twilio API myself, the end-user will NOT have to change any settings regarding the API, as it is my job as the account holder to ensure the code handles adding their own number to the system. The GUI and Web-Interface should be a fairly ‘plug and play’ experience, that is, to work from the get-go with limited hassle and a low level of learning needed. Despite this, any errors may still occur and there will be help on the web interface to assist users of any skill level with troubleshooting.

Similar problems and solutions that are relevant

A similar problem to mine would be the Ring™ doorbell system which connects to a Wi-Fi hotspot within a home and allows the homeowner to stream video from the doorbell at any time. The Ring™ doorbell also alerts the homeowner when someone is at the door. Whilst both these key ideas are vitally like my own, the Ring™ doorbell does not allow users to add people to a whitelist and it does not allow users to store footage locally – it **must** be kept on Amazon servers. As well as this, the relationship between Amazon and Law Enforcement services is not crystal clear so it is a grey area on how your data might be used – especially with (at least some of) the UK leaving the European Union which could allow for data protection and privacy laws to be overhauled to a large degree. Whilst I fundamentally disagree with the use of facial recognition be it by the government or a large company, I would rather a small home user have the power to use it for their own good rather than the good of large multi-national corporations, the farming of data is now widespread across the whole globe Even if you never ever use a service such as Facebook, in this day and age it is probable that they know who you are or have data on you at the very least. With that, my project will differ as it allows for centralised storage of recorded and cached footage that can be modified at any time by the user, kept on a device within the LAN of the router and firewalled to prevent outside contact. My project will also have to comply with data protection [laws](https://ico.org.uk/for-organisations/guide-to-data-protection/guide-to-the-general-data-protection-regulation-gdpr/special-category-data/what-is-special-category-data/) surrounding biometric data in order to keep the user(s) of the product safe just as any system using biometric data must comply with data protection laws.

Another product with a similar theme to mine is the Nest Hello doorbell offered by google. This is again like the Ring™ doorbell with live streaming of a camera and the offer of continuous recording. Another similarity is the storage on the cloud as opposed to local storage – leaving the end-user with less control over their data. It is especially with the biometric and other sensitive data types that I would like complete control over how my data is used and how the data of other people who consent to using the product is used. Unlike the Ring™ though, the Nest Hello doorbell must be a wired connection – a feature that could make installation harder.

Finally a product that is very similar to mine can be seen in [this](https://www.techradar.com/uk/how-to/build-a-raspberry-pi-cctv-camera-network/2) article by techradar. This uses MotioneyeOS in order to stream video from the Raspberry Pi camera to a webpage accessible by devices on the network via the browser. There is a reasonably simple control panel for controlling settings and configuring devices. Rather handily, there is an option to upload recorded files straight to many popular cloud storage services such as Google Drive and Dropbox, something I may consider adding in the future as it is very convenient for the end user.

Many doorbell-based security products require a subscription service of some sort in order to access all the features of the security system. This means that the price you pay for the product is not the final price and many people likely will not know this until they have purchased the product.

I hope to take the best from these existing similar projects and improve on places where they have perhaps fallen short. Some things, such as battery connectivity, may be a little too far-fetched for my limited budget though – these are not affected by code however, and could easily be added at any point of the project. Unlike both the Ring™ and the Nest Hello doorbells, I will ensure all files such as captured images stay local and free for the end-user. There will be no cloud subscription plans and the price paid for the product is the final price, all the features available will be usable as soon as it is installed, **no** extra fees.

Essential Features of a Computational solution

Firstly, my program will have to allow the input and output of a video stream from a Raspberry Pi NoIR camera in order to be analysed with the library of some sorts, likely OpenCV. This is like the other products shown above which all handle an incoming video stream from a built-in hardware camera. The main data coming through the system will be the data from the video stream. This is handled by the python software layer and interpreted as needed. There will also be some user input over the web interface using a keyboard and mouse – allowing the end user to control the system from their own device. This data will be passed over the internet and straight to the Flask webserver on the Pi, where it is then interpreted to perform actions as my code requires.

Limitations of the Solution

The main limitation of my solution will be my inexperience using facial recognition and AI / Machine Learning as a whole. Despite researching this extensively beforehand, I have limited practical experience and I strongly suspect there to be many errors along the way during the coding and practical building of my project. I will combat this by writing each important section of the code as separate files (which will form a library of sorts) to ensure that basic building blocks of my code can be completed and left intact without any errors from changing other sections of the code. These can then be put back together in a final runtime file to be used 24/7. Strong and vigorous testing and debugging procedures at extremely frequent intervals will ensure I do not venture too far from the last stable state of my project and if anything becomes too ambitious, I may have to find an alternative solution or move on from the idea altogether. GitHub will allow me to see past versions of my project very easily – different branches can be made for different features and issues can be made and attached to features, closed by a pull request. This workflow will allow me to go through the project with an absolute stability and ensure that any errors are never permanent. I believe that using GitHub will allow me to experience more of a real-world workflow, despite me being the only contributor. Version control and bug management are key things that coders must control whether by themselves or in a team. GitHub is great for this as it uses the push/pull commit system which is great for version control – if I ever find that I have stepped too far with my new code I can always easily reset to an earlier version, without having hundreds of differently named files of different versions (namely things such as ‘asdfghjkl.py’ which I find myself to hoard often)

The next limitation is time. There are certain things that are beyond my control such as waiting for programs to compile and other such things. Installing and compiling programs might end up taking a significant portion of time out of writing code and testing my project to ensure it is of a high standard. The only way I can think to combat this is to try and multitask at an extremely efficient level – moving on to other parts whilst waiting for others to complete or finish a task. As well as this, I will have to be extensively reading documentation and debugging instead of moving forward with new features and general progress. I have less than a year to complete a project that is on a totally different level to what I have done previously. I do not wish to rush any code and ensure that it is both completed in the timeframe given, and to a very high standard. If my project does not meet all the success criteria outlined later, I will personally consider it a complete failure.

The project requires some sort of graphical front-end which will be in the form of a web-interface in my project. I currently have 0 experience in web development and linking this to my python code that will be executed at runtime. Despite researching beforehand, this somewhat ‘side’ portion of my project could potentially be the hardest part for me personally despite it being somewhat simple for others. I hope that reading documentation and debugging errors will be enough for me to move forward and complete the web-facing portion of my project in a reasonable amount of time whilst still maintaining a reasonable amount of quality. Even if the base code works and can be executed through a CLI on the Raspberry Pi directly, without a GUI, my project might be void. It will therefore be difficult to complete the web-facing side of the project whilst still focusing enough on the bulk that goes on behind the scenes processing and performing operations in order to create a working facial recognition security system.

The camera in my project is the Pi Noir Camera that operates in low light / dark conditions using infrared. If there is no IR source in the room or outside area however, the image returned will be either extremely broken and grainy or just plain black – a failure of the key feature of my system. Furthermore, the way in which faces are recognised is going to be with ‘Haar Cascades’ which I will discuss later in this coursework document. Whilst being a very popular way to recognise objects such as faces; it is also known to give false readings in the forms of both false positives and false negatives. It is unlikely that the end-user will have a vast library of photos of each user to be added to the system. And it is unlikely that every user will want to enrol their face in different weather conditions and different times of day in order to get more confident readings (regardless of if they are positive or negative). For this reason, the user may receive too many annoying messages of false positives and move away from my product. It will be difficult to find the correct balance between speed and accuracy.

Finally, the use of a camera and the raw computational power required to record video, stream video, analyse each frame for a human candidate and train faces means that CPU and RAM / DISK usage will be very high. Even at idle load, my Raspberry Pi runs at an abnormally high temperature possibly due to a manufacturing defect. It is possible that my Pi may throttle and have significantly reduced performance, it might even break altogether. Both situations would render my project to be broken and unsuitable for the end-user (and therefore a fail altogether). Not only this but the Pi will need a quality power supply, and the placement of the camera may be tethered by the length and fidelity of the power supply – resulting in less than convenient positions available for the camera to operate in. This is why a battery may be a much better choice – however my budget does not allow for that luxury and the power supply will need to be hard-wired into the wall or a cable ran to an extension lead near the placement of the product.

Minimum Required Hardware and Software configuration

Storage - 32GB or greater, the reason for this size is to allow both the installation of the operating system and the ability to save lots of snapshots onto the disk.

CPU - 2 Core 1GHz CPU with graphics support, the reason for this is that OpenCV is CPU heavy and without a decent CPU, the processing may grind to a halt or even hang.

RAM - 1GB DDR2 RAM, the processing is also RAM heavy, without enough RAM (or a swap file) the system may crash.

Network - Ethernet connectivity or 2.4GHz wireless to allow for the webserver to work (and for me to connect to the Pi to code headlessly).

I have owned a Pi 3 for a while and this is what I will be using for the project, despite a Pi 4 being a better candidate. The Pi Zero offers a small form factor which may be better for a discrete camera, but it lacks the processing power for OpenCV. I will be using PuTTY to SSH into the Pi for the initial setup and then moving onto a VNC server to allow me to access the Pi from my standard desktop PC. I think that this will be easier than hot swapping keyboards and mice and changing inputs on my monitor 24/7.

Success Criteria for the device setup

1. Ensure that the Pi is enabled for SSH connection from my desktop computer.
2. Set up the VNC server on the Pi and allow connection from my desktop.
3. Set up the NoIR camera with the Raspberry Pi using raspi-config.
4. Change the swap size to allow for a greater memory capacity.
5. Ensure that the latest version of Python 3 is installed (that is compatible with OpenCV) along with needed modules such as pip.
6. Acquire the latest versions of libraries with the pip command in the terminal.

Success Criteria to ensure the enforcement of chosen programming constructs

1. Prefix variable names with important information to make the code more readable.

*Taking the time to prefix information such as datatype or what the use of a variable is makes the code easier to read. Not only this, but if the code needs to be debugged then it is easier to understand what the purpose of each variable is as it is already there in the name. Whilst not the primary function, a helpful side effect is that the auto-completion offered by the IDE may work more efficiently; when you know you are going to be using a variable that is a number, typing ‘int\_’ will return the auto-complete list that contains all integer variables – the coder will not have to search through the code for the correct variable name. It is likely that I will follow the naming convention as below:*

*“{Datatype}\_variableNameHere”*

*This is a combination between Hungarian notation and switch (camel) case, two very popular programming standards. The camel case makes the stem part of the variable name easier to read by marking different words by using a capital letter.*

1. Add detailed comments to code

*Comments allow other people to read your code and still understand what is going on, even if they would code a solution to the problem in a different way – they explain how the code works which is especially useful for when the purpose of the code isn’t amazingly clear. For me, this will allow me to return to previous code and understand what my thought process was whilst making that piece of code, even if I had forgot. Yet again, this will also make debugging easier as the comments can allow the programmer (me) to see if there are any logical incompatibilities in the code at a quicker glance than working through the actual code line by line to see what happens with the variables.*

1. Use GitHub for version control

*GitHub as a platform allows me to commit edits to code and ensure that they are well documented. Each feature can have its own branch that is merged back to the main branch once the feature is fully complete. This ensures broken code is not added back to the production level source code. Each commit has bug tracking and I can choose to fallback to different versions of code if I have overstepped and broken the code to an unrecoverable level. I think it is realistic to be using version control such as GitHub as it is very common to find this system being used in program production houses.*

1. Ensure code is modular

Success Criteria for the solution

1. Gather an incoming camera stream from the Pi Camera in python.
   1. Is the view of the camera clear and is it orientated correctly? It should be the standard way up and look clear.

*This is important as it will ensure the feed will be working properly for later sections of the project, if it works now and fails later then we know it is something recent that broke during that stage of coding as opposed to this crucial early step. Setting this up correctly now ensures that the code later during the rest of the project will be able to rely on this fundamental code. A bad way to code would be to rush straight in and try to code the whole thing at once, making it much harder to debug and identify and solve as there is so many more potential failures.*

* 1. Is there any lag, how much? Are there framerate drops? Does it adversely affect performance?

*This is important as it is preferred to have a smooth video to easily see what is going on. Choppy video negates the positives of a camera monitoring an area, as we may not even be able to see the actions of people through the camera very clearly. Additionally, input lag would make the experience for anyone watching the online stream less enjoyable – in extreme cases, the stream may not at all accurately represent what is going on in front of the camera. This would make the promised ‘live streaming’ aspect a lie and make the solution void.*

1. Use a machine learning library to identify objects that look like faces and draw a bounding box (or multiple) around the identified face(s).
   1. Is the box clear to see? It should be at least 2 to 3 pixels thick at source.

*This is important as the feed will later be able to be viewed on a smartphone with a smaller screen. It is essential that the box is clear so it can still be seen around any objects in a frame – even on a mobile device. It should be at least 2 to 3 pixels thick on the source frame, before any post-resizing on other devices. I would say that the bounding boxes that identify faces are a fundamental part of this project – they show the end user that the solution is, indeed, identifying faces in a frame.*

* 1. Does it identify faces well? Are there any false positives or negatives?

*Both false positives and false negatives will impact the accuracy of my solution and make it a less viable product in general. I hope to especially reduce the rate of false positives of unknown people (which will send annoying texts to the user through Twilio on the condition of repeated false positives). There are various parameters that can be tweaked in order to change how many machine learning libraries identify faces, and the accuracy at which this happens. I hope to perform tests to find which sets of parameters yield the most accurate result.*

1. Add code that can take photos of the Pi Camera viewport to be used in the collection of a dataset. Crop from facial bounding box.
   1. Do the photos crop correctly to the bounding box?

*It is important that the photos saved are only of the faces and not of other objects that are present in the frame, this is so the trainer has the correct data to train the recogniser.*

* 1. Are the saved photos the correct object (the face in the viewport)?

*This is important especially for the trainer step later. The dataset needs to be coherent for the trainer to coherently and accurately know what unique properties ton look for in an object to match to a person. The more people added to a dataset that they should not be in, the more chance that the trained recogniser will recognise too many people as the same person – a failure. It is important that each unique face has a different ID so that the trainer separates different faces.*

* 1. Does the code save enough photos?

*This is something that can and will be down to me to decide. The trainer needs enough photos with enough variation in order to accurately know what to look out for in different faces. Despite this, there are going to be space limitations – especially if many people are added to the system. I will have to find a good balance between having enough photos for accuracy whilst keeping the size on disk low.*

* 1. Are the files named correctly so that they can later be read and parsed by the training code?

*The naming scheme of the files need to follow a specific format for the trainer to match different people’s photos together with the same ID, and form collections of the same person. It is also required that each file has a unique name. This will likely be in the format:*

*This is simple enough to be read by the trainer function whilst keeping the functionality necessary for the function to work – a good compromise. It is likely that there could be 100s of different photos with many different people.*

1. Add code that trains the dataset gathered into a file which can be read by the machine learning library in order to then detect registered faces in the viewport. This will use an ID system obtained from the images.
   1. Can the code read the files in the dataset and use a trainer function correctly?

*The code needs to be able to recognise the naming scheme of the dataset files in order to then train the photos to be recognised by a recogniser function. It is important that all files are read properly so that the trained file is the most accurate that it can be. The trainer function should output a single file which can be used by a recogniser – a function that scans the viewport for trained faces.*

* 1. Does the trainer file save successfully?

*If the file does not save, no recognition can take place as the recogniser will not have anything to work with.*

1. Add code that uses the trainer file to then recognise faces in the viewport
   1. Can the recogniser read the trainer file?

*The recogniser needs to be configured to read the trainer file, and then scan the viewport for faces that match any faces in the trained file. This forms the basis of the identification part of the recognition system.*

* 1. Is the accuracy of the recogniser great enough to be pushed to production?

*It is important that the recogniser can detect whose face belongs to who with reasonable accuracy, failure to do so will result in an incoherent security system that could potentially not function correctly at all – allowing intruders to gain access without much notice.*

* 1. Can it recognise more than one face?

*There will often be situations in which there is more than one face in the viewport. It is therefore required that the recogniser can scan a viewport that contains more than one face and work out the ID of each face present accordingly, this is to prevent people from possibly fooling the system by placing a printed known face and placing Infront of the camera whilst moving past undetected themselves.*

* 1. Can it return the correct IDs even in extreme circumstance?

*Can the recogniser return correct IDs on people that look extremely similar? Probably not and I do not expect this part of the success criteria to be passed in my opinion. I do not think the Raspberry Pi has the computing power necessary in order to run at a quality high enough to detect such small differences in appearance.*

* 1. Does it allow for returning ‘unknown’ for an object that resembles a face but is not registered?

*This is important especially for the stage that will incorporate Twilio API into my python code. The recogniser must account for any faces that are seen that are not trained to be recognised. These should be marked as ‘unknown’. This will allow a Twilio subroutine to then send a text to the user and inform them of the date, the number of unknown faces and hopefully also send a photo of the frame at that specific moment.*

1. Add code that starts a webserver in python (This will almost certainly be done with the Flask library)
   1. Does the webserver run locally, and can it be accessed on the house intranet using a different device?

*The webserver needs to be accessed by devices on the same network before thinking about expanding to the whole internet using port forwarding. The webserver is what will allow the end-user to access the video stream and control the software so without this working correctly, the project fails.*

* 1. Can the webserver display HTML pages with CSS?

*In order to add buttons and other various functions, I need to be able to display HTML pages with CSS. The CSS will be simple but will allow me to make the buttons look just a little bit nicer than before and ensure that the user interface is of a standard that even a novice user could understand. In the event that I finish my solution earlier than expected, I should also add a toggle to enable advanced settings for very experienced users.*

1. Transfer the previous camera-based code over to the webserver code.
   1. Can both sets of code still be called?

*I need to make sure that I did not create any conflicts by merging these wo sets of code together – even if it doesn’t look like it at the surface it may impede performance and usability later down the line if I do not thoroughly check the code. I should also check for any forms of spaghetti code and remove these as required, keeping the source code as optimised and easy to read as possible.*

1. Add a page to the website that livestreams the video feed from the camera after machine analysis using the selected library.
   1. Is the quality of video good?

*The quality of video needs to be good enough to be used to identify details of any unknown people caught by the software in a still image. If the image is broken, blurry or has any other artifacts, the project will have no benefit at all in comparison to a normal, cheap CCTV camera from a cheap electronics store. The Pi NoIR camera has a decent resolution so any reduced quality will likely be from my process of handling the image data.*

* 1. Can this be viewed on other devices?

*I do not know what device the end user will choose to view the web-interface on – so it is important that I create the website so that it can be viewed on many types of device with many different display configurations. There are many such factors similar to this that a programmer must consider when writing a solution. This is probably one of the more basic considerations when making a website but any further important considerations that I find can be added when I perform testing later on in the development stage.*

* 1. Can it be viewed on more than one device at once?

*In the situation that there is more than one person living at a house (which will be common). It is important that more than one device can read the video stream on the website at once in case multiple members of the household would like to view the stream. This may sound simple, but I expect to run into threading and file locking issues due to the nature of how I am going to serve the frames to the webserver. The code should allow the stream to be viewed by more than one device with ease.*

* 1. Can the frame caught still be used by other parts of the code whilst it is being streamed?

*This relates to the threading and locking issues I described above. I aim to have a universal frame that can be used by many sections of the code such as image saving and the web-interface. This should be updated at least at 10 frames per second to avoid the video seeming too jittery. However, all functions that require the frame must finish their job before the frame changes, this will add a small amount of lag but most importantly will require the use of a thread and lock system which can be done with a number of libraries on Python. The threading and locking system also ensure that only one thread is accessing the camera at once.*

1. Add code that allows the user to take screenshots of the current frame and save them to the device.
   1. Does the file have an appropriate name?

*This is important as the file needs to have information both in the name and the metadata to accurately describe the context of the photo. Whilst metadata can usually do the job of containing this information, many users (especially novice users) may not be able to access this important data. Due to this, most of the important context will be held in the filename such as the date and time that the photo was taken.*

* 1. Add code that allows users to upload their own photos to add to the dataset, without having to stand in front of the camera.
  2. Does the webserver allow the user to upload multiple images at once?
  3. Does the solution handle cropping the photos and adding to the dataset correctly?