2022-2023 Computer Science A
Project: Facial Recognition

Project Outline

In this project you will create your own version of Face ID using the Kazemi Landmarks Algorithm which has been provided. You will use 3 photos to create a 'base face' which you will compare other faces with. The goal is to create an algorithm that can uniquely identify your face while rejecting the faces of your classmates and other people. You will have a lot of freedom in how you implement your version of Face ID, but you will need to use facial vectors combined with facial proportions as you lack a depth camera.

Camera Based Face Unlock vs Depth Projection Face Unlock

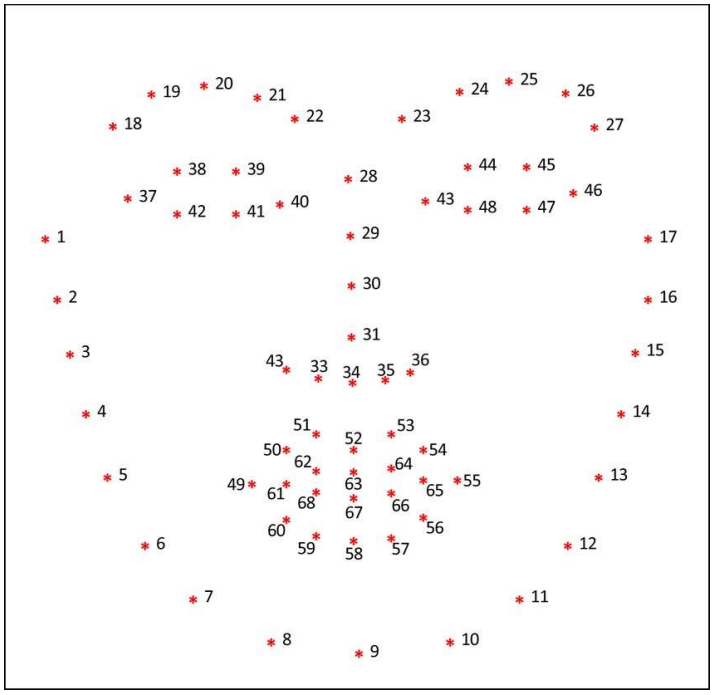
Camera based face unlock uses just a camera to uniquely identify a person, while a depth projection system uses an IR (Infrared) dot projector to create a depth map of a person's face to determine whose face it is. This project is similar to how some phones implement Face unlock such as the Google Pixel 7, which just use a camera to identify the user. Google uses advanced machine learning models for it's facial recognition but even then it is generally not considered very secure as camera based facial recognition systems can easily be fooled and make errors – as you'll see.

The most common type of machine learning algorithm used for facial recognition is a deep learning Convolutional Neural Network or CNN. CNN's are a type of artificial neural network that are well suited to image classification tasks. CNN's learn to extract features from images and can identify complex facial features such as the shape of a chin or eyes.

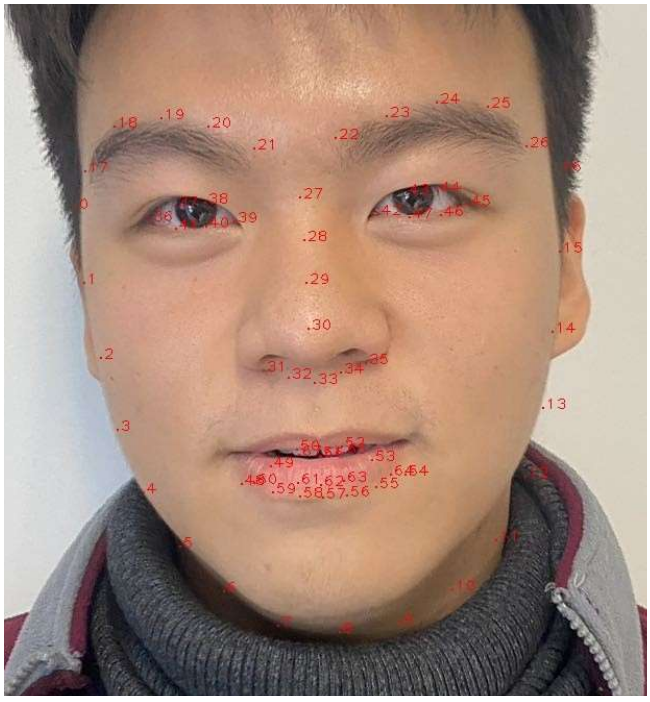
To do what Apple does with Face ID you would need an IR projector which projects a grid of infrared dots on a person's face. By building a depth map of a person's face you accurately measure distances between a person's eyes and mouth and the width of their head, with a far smaller chance of an error. According to Apple there's a 1 in a million chance that someone else could unlock your iPhone. As we lack a depth sensing camera system and for simplicity's sake we'll be doing the more error prone and simpler camera based system.

Kazemi Landmarks Algorithm

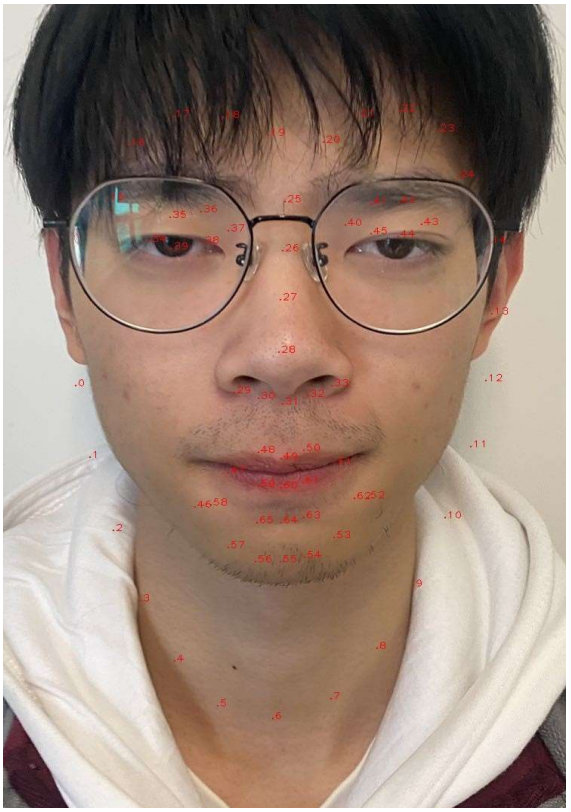
The Kazemi landmarks algorithm is an algorithm from a university in Sweden that can quickly identify facial landmarks (below) and you will be using it in your Face ID project. The class `Facemark.java` contains the method `getFacemarks(String PhotoNameAndLocation, String CreateMarkers)`, which when given a photo this will return a 2D array of type int that contains the X and Y coordinates of the face marks. The returned 2D array has a length of 68, representing each of the 68 facial landmarks in the image below. Each of the 68 arrays only contains an X and Y coordinate, and so each array only has a length of 2. In the returned 2D array, the array at position [0] will contain the [X,Y] coordinates of point 1 (*and at position [1] you'll find the [X,Y] coordinates of point 2 and so on*) in the picture below:

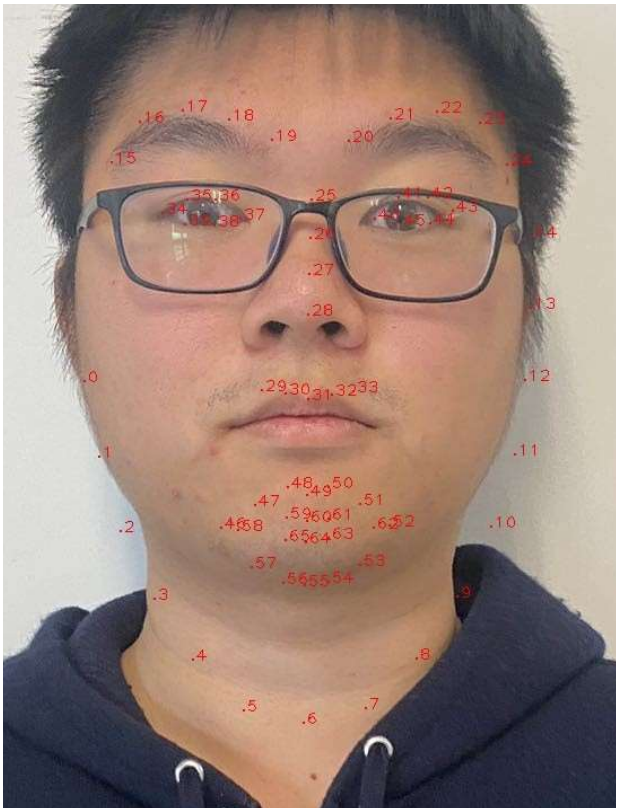


When you run `getFacemarks()` and save the photo you should get an output similar to this:



However the Kazemi Facial Landmarks algorithm isn't perfect and you can get strange results which puts the facial landmarks in the wrong places like so:





Often the 'long face' issue arises when the landmark algorithm cannot clearly identify where the chin is, when creating your algorithm I would suggest putting more emphasis on the eyes and eyebrows proportions rather than the mouth and nose positions. Another issue I've found is the algorithm can put different markings on the exact same photo, at the time of writing I am still unsure why this happens, but be aware of it.

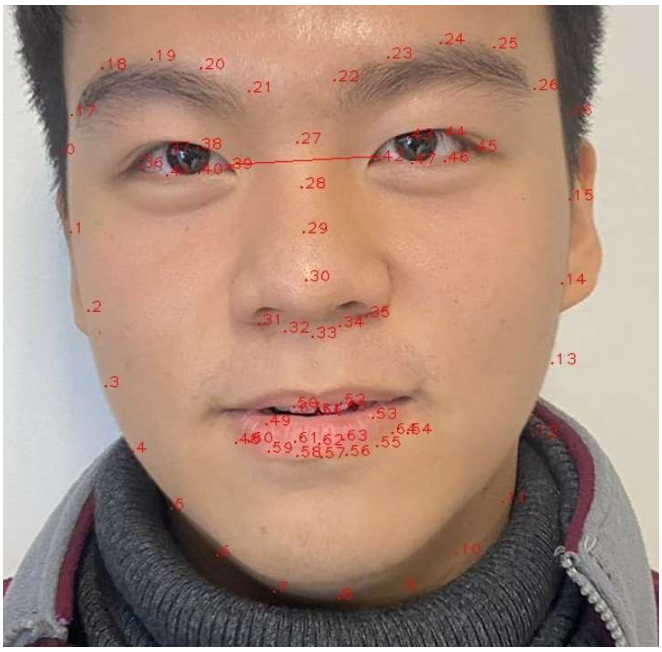
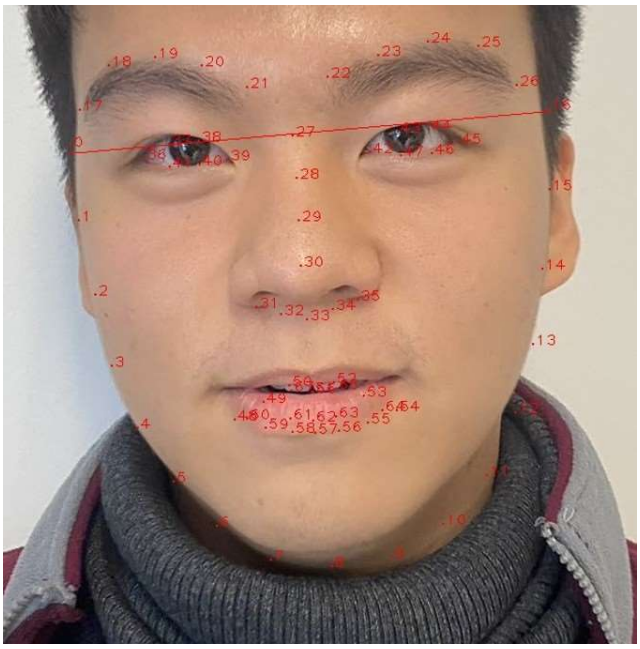
Creating your own version of Face ID

As stated above when using the `getFacemarks()` method you'll get a 2D array of coordinates, and to use this to uniquely identify a face you'll need to calculate the distance between 2 points which can be done with the following formula:

$$\sqrt{[(x_2 - x_1)^2 + (y_2 - y_1)^2]}$$

You should put this formula into a method as you will need to repeatedly use it. Note however the distance given is in pixels which in itself isn't particularly useful as we don't know the distance between the camera and the face nor the width of the face. You'll want to use the distance in comparison with other distances, for example the width of the face in comparison to the width of the eyes or the length of the nose compared to the length of the face.

For example, you could calculate the distance between points 1 → 16 which represents the width of the head, and then calculate the distance between points 39 → 42 which represents the shortest distance between the eyes. Using both of these distances you can then calculate the % distance. You then need to do this for several measurements such as the length of the eyebrows or length of the nose, you can choose whatever you think is best. On the next page I've given an example using our good friend Aaron.

In the image above the X and Y location of point 0 is 346, 667 and the X and Y location of point 16 is 813, 626 which gives a distance using the formula above of 468.796. Remember you get the X and Y coordinates from the `getFacemarks()` method, in this case the array at position [0] contains [346,667] and the array at position [16] contains [468,796]. You can see the calculation below:

Distance:	$\sqrt{(813 - 346)^2 + (626 - 667)^2}$
	$\sqrt{(467)^2 + (-41)^2}$
	$\sqrt{219770}$
	468.79633104366

You don't need to be a math genius to do this, but if your struggling to implement the formula in Java don't worry there's plenty of examples on Google or Baidu. Carrying on, the X and Y coordinates of point 39 are 497, 670 and for point 42 the X and Y coordinates are 638, 662. Again using our wonderful 2D distance formula, the distance is 141.226, which you can see below:

Distance:	$\sqrt{(638 - 497)^2 + (662 - 670)^2}$
	$\sqrt{(141)^2 + (-8)^2}$
	$\sqrt{19945}$
	141.22676800097

Now we can easily calculate the % distance that points 39 → 42 make from points 1 → 16, 141.226/468.796 = 0.3012. So, the eye bridge distance makes up approx 30% of the width of Aaron's face. Now if you were to calculate the distance for everyone in the class you'll find that for some people it will be greater than 30% while for others it will be smaller. When creating your version of Face ID you will want to accept some room for error (e.g. accepting + or - 5%), while rejecting faces whose proportions are very different (e.g. + or – 10%). You will need to do this with several parts of the face so you can be sure that you can choosing the correct one. Note that to create the images above you'll need to change `getFacemarks()` so that it will return and image instead of a 2D int array and to draw a line on the photo you need to use the method `AddLineOnPhoto()`.

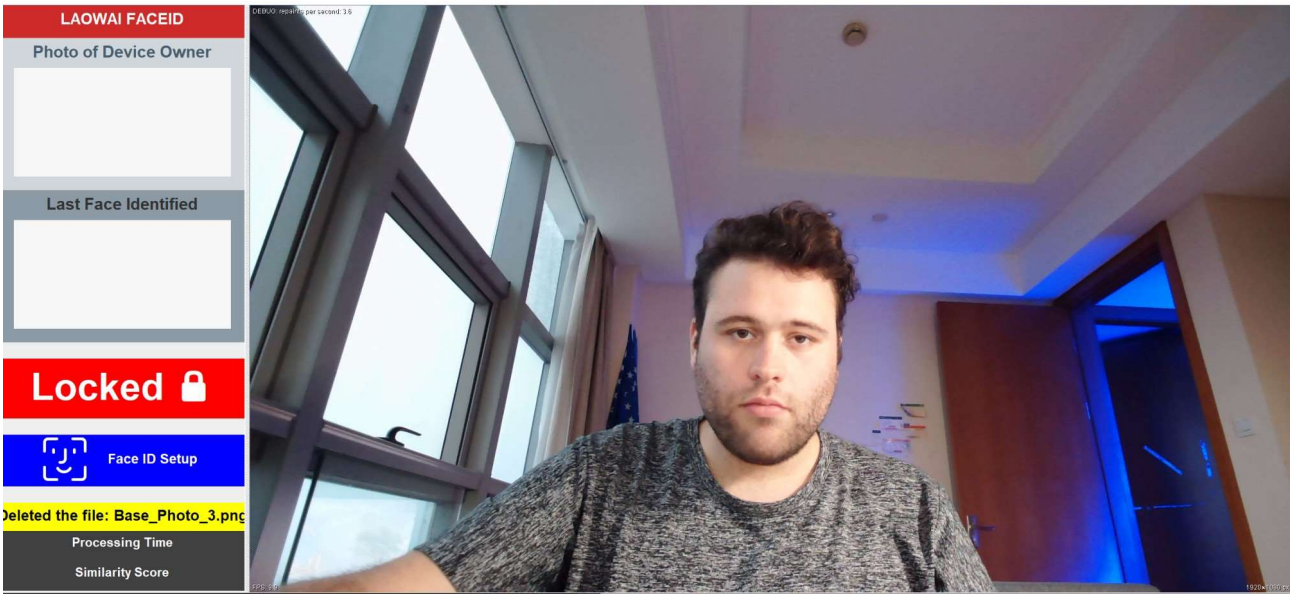
Project Requirements

I will give you a lot of freedom in how you carryout this project so that you can demonstrate your Java skills. I have provided you 2 classes, `Facemark` and `Camera_GUI_System`, furthermore I've given you a new version of OpenCV that works on Macs (yeah!). There are 2 ways your project will be assessed, 1 through a presentation of your working program to the class along with a PPT explaining your algorithm and the other being a test similar to that of the ANPR project, where you need to sort a folder of images of faces, creating a .csv file showing which photos passed your algorithm and which did not. You can do this project in any way you like, the only requirements are that you don't use APIs, plagiarize, and you must include class inheritance in at least one part of your program.

1. GUI Presentation

You will need to give the class a demonstration of your project using the GUI provided in `Camera_GUI_System`. The GUI is similar to the one seen the ANPR project and you should be familiar with it, the biggest change being the implementation of a 'Setup Face ID' button which you must use as well as a 'locked' and 'unlocked' panel which changes based on whether your algorithm has successfully identified your face.

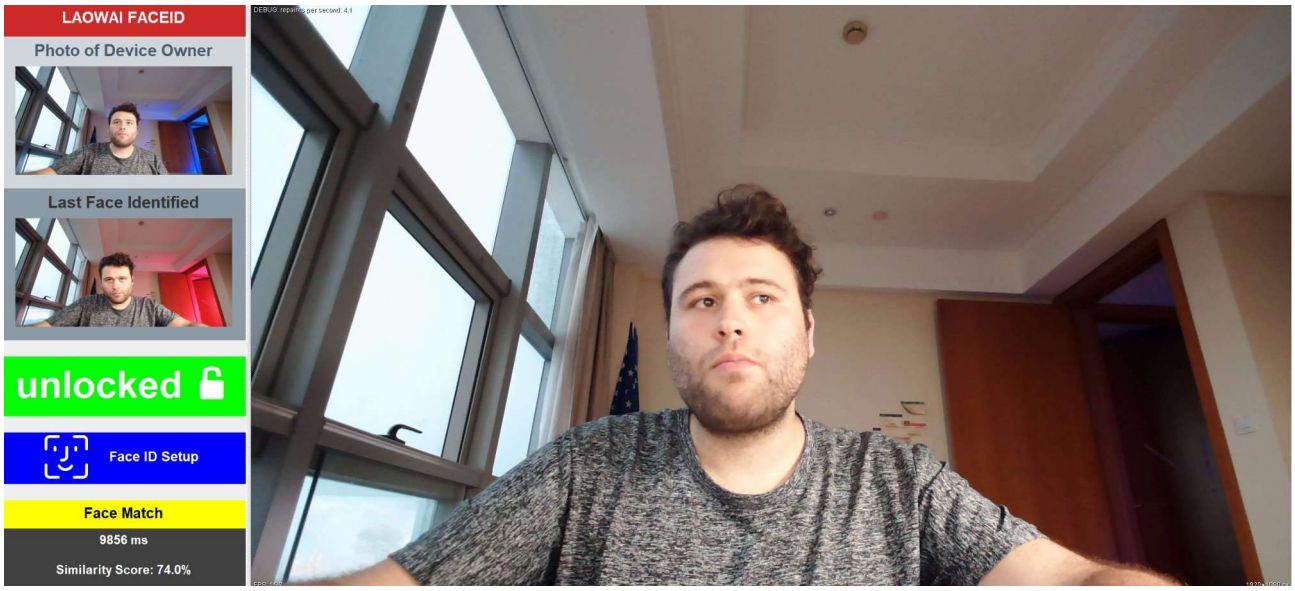
When clicked the 'Face ID Setup' button it will delete 3 photos in the folder \Photos\Base Photos called `Base_Photo_1.png, Base_Photo_2.png and Base_Photo_3.png`. Then it will take 3 photos using the webcam called `Base_Photo_1.png, Base_Photo_2.png and Base_Photo_3.png`. You need to use these photos to compare other faces to, with your algorithm 'passing' faces that are a match to those 3 photos. You will need to create your own classes, objects and methods to do this, for example you could take the average proportional distances from the 3 photos to create your 'Base Face' which you compare other faces to. If a new face is similar to the 'Base Face' then you pass it, otherwise you reject it.



Step by step guide to how the GUI should work

Steps	Action
1	Start the program, and the GUI loads
2	The user presses the 'Setup Face ID' button and the program deletes 3 photos, then takes 3 photos, called `Base_Photo_1.png, Base_Photo_2.png and Base_Photo_3.png`. You will need to use a while loop to wait for these photos to be taken, I have included the variable `BaseFaceSet;` in the class `Camera_GUI_System`. Use a while loop to wait until this variable is set to true.
3	When step 2 is completed, you need to use these 3 photos in your algorithm so that other photos of faces can be compared to it. Send each photo to the method `getFacemarks()` to get the Face mark data and then for example, you could use save the average proportions of the faces or the maximum and the minimum proportions. You could create a class that generates all the facial proportional data for each photo and then create a 'base face' object that stores it – but again it's down to you!
4	Then set up a for loop like the one you saw in the ANPR project that makes the GUI keep taking photos for a certain length of time. For each photo you want to check if there is a face in it, if there is and your algorithm determines that the face in the photo is similar enough the base face photos then set the 'lock/unlock' panel to 'Unlocked' using the method `updateLockedStatus (Boolean Status)` in the `Camera_GUI_System` class. Otherwise set the panel to 'locked'.

You need to set it up so that the program will automatically just keep taking photos (use a do-while loop), and for each photo that is taken it is processed using your algorithm and the method `getFacemarks()`. After each photo is taken using the `Camera_GUI_System` class you need to send the photo to `getFacemarks()`, if null is returned set the panel to 'Locked', otherwise you need to use your algorithm to check whether the photo taken is similar enough to the 3 base face photos taken (`Base_Photo_1.png etc`) when someone presses the 'Face ID Setup' button. If your algorithm rejects the face as not similar enough to the base face photos then set the updates panel to 'Locked' otherwise you need to set the updates panel to 'Unlocked' like this:



You will need to give a live demonstration version to the class showing your algorithm working, although don't worry about mistakes as Facial recognition is not a perfect science and is error prone. Even Apple's Face ID isn't perfect! Your GUI presentation and PPT will be marked by your fellow students and not by me.

2. Processing a Folder of Images

This will be very similar to the ANPR project where I will give you a folder of photos and your algorithm must decide which photos include a photo of you, and which photos do not. You must use the base face photos previously mentioned to setup up your facial recognition algorithm, so you are only allowed a maximum of 3 photos, for which all other faces are compared against.

When running your algorithm you need to generate a .CSV file which will be used to mark your work. For each photo you must include the photo number, a similarity score, a passing mark, whether the photo is a match or not and the processing time. The processing time is the time it takes for your algorithm to decide whether the face is a match or not – but does not include the time to take a photo or anything else. You can generate the .CSV file by using the method `addFaceIDData` in the class `Camera_GUI_System`, just like in the ANPR project. I've not provided you with the code this time, so you will need to make the method to run this test however I will check your code so that I know you are doing it honestly – if there is any intention to cheat I will give you 0 for the whole project. The output should be similar to what you see below:

	A	B	C	D	E	F
1	Photo Name	Photo Number	Similarity Score	Similarity Passing Mark	Lock or Unlock	Processing Time
2	Aaron1.jpg	1	6	45	No Match	2756
3	Aaron2.jpg	1	36	45	No Match	2161
4	Aaron3.jpg	1	0	45	No Match	2338
5	Charlie1.jpg	1	36	45	No Match	1772
6	Charlie2.jpg	1	0	45	No Match	1867
7	Charlie3.jpg	1	27	45	No Match	2261
8	Charlie4.jpg	1	28	45	No Match	2001
9	Edwin1.jpg	1	0	45	No Match	2163
10	Edwin2.jpg	1	0	45	No Match	2210
11	Edwin3.jpg	1	12	45	No Match	2646
12	Edwin4.jpg	1	0	45	No Match	3083
13	Edwin5.jpg	1	3	45	No Match	2896
14	Evelyn2.jpg	1	39	45	No Match	2503
15	Evelyn3.jpg	1	36	45	No Match	2126
16	Evelyn4.jpg	1	55	45	Face Match	3164
17	HappyTim1.jpg	1	11	45	No Match	2761
18	HappyTim2.jpg	1	44	45	No Match	2749
19	HappyTim3.jpg	1	31.5	45	No Match	3047
20	Jony1.jpg	1	39.5	45	No Match	2497
21	Jony2.jpg	1	42	45	No Match	2363
22	Jony3.jpg	1	13	45	No Match	1930
23	Laoban1.jpg	1	0	45	No Match	2111
24	Laoban2.jpg	1	12	45	No Match	2609
25	Laoban3.jpg	1	10	45	No Match	2278
26	Leo1.jpg	1	3	45	No Match	2047
27	Leo2.jpg	1	11	45	No Match	2537
28	Leo3.jpg	1	24.5	45	No Match	2235
29	Leo4.jpg	1	5	45	No Match	1992
30	Zarc1.jpg	1	17	45	No Match	2014
31	Zarc2.jpg	1	13.5	45	No Match	2038
32	Zarc3.jpg	1	20.5	45	No Match	2215
33	Zarc4.jpg	1	44	45	No Match	2350
34	Zarc5.jpg	1	12	45	No Match	2267
35	Zarc6.jpg	1	39.5	45	No Match	2240
36	Zarc7.jpg	1	5	45	No Match	2112
37						

As you can see my algorithm rejects all of your faces with the exception of Evelyn, and your goal is to make the algorithm that can reject as many faces as possible BUT doesn't reject pictures of yourself. When it comes to making your project you will be penalized if your algorithm rejects photos of yourself, you want to allow false positives. I will also test your algorithm using other people as the base face, such as celebrities or your class mates so you want to make an algorithm that doesn't just work for yourself but can be applied universally.

To help you I've provided around 10,000 images of faces which you can use to test your algorithm, Use a portion of these that you think would be useful to test your algorithm, 100-200 would be a good number and then create a method that can calculate the % of photos that your Face ID algorithm passes and rejects. It will take a lot of time to find the correct algorithm that can accurately select your face out a bunch of photos.

How will this project be graded?

1. GUI Presentation

You will need to give the class a demonstration of your project, showing your algorithm unlocking to your face but not unlocking to your classmates (Don't worry about errors!). You will also need to give a PPT, explaining the facial features you have chosen to set up your version of Face ID, focusing on your experience testing your algorithm. Your presentation will be marked by your classmates and the average will become your grade for this section. This will make up 35% of your final grade.

2. Percentage Correct (Processing a folder of Images)

You will be given a folder of photos that you have not seen before and these will be used to test your algorithm, with it being tested several times using different base faces. You will receive a grade for this section based on the % accuracy, with penalties if your program rejects a photo of the base face person. Again here you want to have false positives, as in reality you don't want to make it impossible for a person to unlock a Face ID system. The .csv file generated in the class `Camera_GUI_System` will be used to grade your work, and so you must correctly implement it. This will be 25% of your project grade.

3. Processing Time (Processing a folder of Images)

Similar to number 2, except here you will be graded on the time it takes for your program to process a photo and to decide whether it is a match to the base face or not. You will get time penalties for mistakes, with greater penalties for when you program rejects photos of the face it should accept. You will need to implement this testing system. Your processing times will be compared with your classmates in a ranked list, with the fastest person getting 100%. This will be 25% of your project grade.

4. Quality of Code

When the project is completed you will be required to print out your code and this will be marked for quality, for example are you commenting your code and does it include unnecessary functions or strange variable names. With a large project you should appreciate the importance of good quality code as otherwise things can get confusing very quickly. This should be an easy 15%.

	% of Final Project Grade	Xiaobao Individually Graded
GUI Presentation	35%	YES
% Correct	25%	YES
Processing Time	25%	YES
Quality of Code	15%	YES
Whole Project	100%	YES

Remember: If you don't understand something, don't worry!!!
Please ask questions, I'm here to help you!!!