Ecole Publique d'Ingénieurs en 3 ans

Internship report

FACE RECOGNITION AND LIVENESS DETECTION

7 august 2022

Tony Pinsel-Lampécinado,
Fabien Le Bec

School supervisor: **Régis Clouard**
Company supervisor: **Guoqiang Li**

www.ensicaen.fr

ACKNOWLEDGMENT

We would like to thank Guoqiang Li, a member of Mobai company and our supervisor at NTNU, for his trust and the knowledge he shared with us. We also thank him for his availability and the quality of his supervision during the internship.

We would like to take this opportunity to express our deepest thanks to the management and staff of NTNU for their hospitality.

TABLE OF CONTENTS

LIST OF FIGURES

INTRODUCTION

1. Context

Today, more and more systems are based on face recognition. These methods are present in almost every fields. For instance, it is becoming normal to unlock your phone or to consult your bank account by authenticating yourself using this method. This method has the potential to become the most used method of authentification thanks to its ease of use and its good user-experience. However, it must therefore reach a very high level of confidence to secure the user's data as much as possible. This method is already available on all types of smartphones.

2. Subject

In this context, we decided to develop our face recognition solution. At the beginning of our internship, we had two separate topics with the same supervisor. Both topics are related to liveness detection. For Fabien, it was about head micromovements detection; for Tony, it was eye gaze detection. Our supervisor gives us some codes in Python to discover and do our first experiments with face recognition.

However, given our subjects' proximity, we quickly decided to collaborate on developing a face recognition mobile application. This application will have to resist as much as possible to several attacks that we will explain later in this report. To answer this, we can study our initial topics.

3. Task specification

In this project, we manipulated several notions used during our projects at ENSICAEN. First, we have to deal with project and team management like every project we did at school. Then, we create and manage a local database. Finally, we try to use a maximum of software engineering methods as design patterns and Java principles.

Thus, the main goal of this internship is to develop knowledge of face recognition. Indeed, we have never worked on this topic during our school projects and courses. Also, it will be very interesting to work on this project.

4. Organization

4.1. Group background

The group is composed of Fabien Le Bec and Tony Pinsel-Lampecinado. We are two students from the French engineering school ENSICAEN. We have already done other projects together like a 3D visualization of a real part of a territory in Python, a Takuzu game in Qt/C++, a Turtle AI game in Python, or a pandemic simulation program in C. We are therefore used to

working together. We are supervised by Guoqiang Li, a researcher from Mobai/NTNU Gjovik in face recognition.

4.2. Meetings

We quickly established certain habits, such as weekly meetings and exchanges via Teams. Each of these meetings was organized with the same structure. First, we did a progress report on our work. Then, we talked about the new problems or barriers that we found. Finally, our supervisor gave us some advice for future steps.

4.3. Work schedule

At the beginning of the internship, we read articles and codes from our supervisor's company. This work taught us many things about face recognition and its problems. Moreover, we started to manipulate OpenCV, one of the best libraries for face recognition.

After one month of work and discussions, we decided to develop our face recognition app. This app will be developed in Java on Android Studio. Our supervisor then showed us ML Kit, which will become the main library we will use. We will introduce this later in this report.

4.4. Development model

We have chosen to develop this application with an incremental model. It means that we add new functionality during all the development phases. Indeed, we have decided to make it like that because, during our meetings, our supervisor gave us functionalities to develop for the next one.

Thus, at the end of the first iteration, we have only made the database creation and management. After that, we added front-end activities. Then, the collection of face features and so on to arrive at the current version of the application.

PRESENTATION OF THE COMPANY

5. NTNU Gjovik

The Norwegian University of Science and Technology NTNU was founded in 1996 in Trondheim. However, in 2016 the university was expanded to two other campuses in Gjovik and Alesund. The Gjovik campus is specialized in health, economics, design, and technology. We work for the Department of Information Security and Communication Technology at the Faculty of Information Technology and Electrical Engineering.

6. Mobai

Mobai is a Norwegian company from the Norwegian University of Science and Technology NTNU. This company developed its face recognition system that works on any camera device.

Our supervisor works both for this company and for the university. His name is Guoqiang Li. He obtained his PhD degree from the Norwegian Information Security Laboratory at NTNU in 2016. Today, his research relates to fingerprint recognition, biometric template protection, and behavioral biometrics. Since 2012, he took part in more than twenty projects and reports.

STATE-OF-THE-ART

7. Face recognition concept

This concept looks very usual to everybody. First, the user presents his face in front of the camera, the system detects his face. Then, features are extracted from his face and the face authentification system compares it to a feature database to determine if he is accepted or rejected.

7.1. Trade between user-experience and security

We must choose a threshold to compare features and the database's features. If the comparison score is higher than this threshold, the face is accepted; if not, it is rejected. Also, it can raise some problems. If this threshold is too high, the system will have a high level of security, but it can create false rejections and a bad user-experience. On the contrary, if it is too low, the authentification system will have a good user-experience, but it can create false acceptances and also a low-security level.

The threshold concept is present in several functionalities of our app. Also, we have to do tests with a dataset to choose the better threshold in these cases.

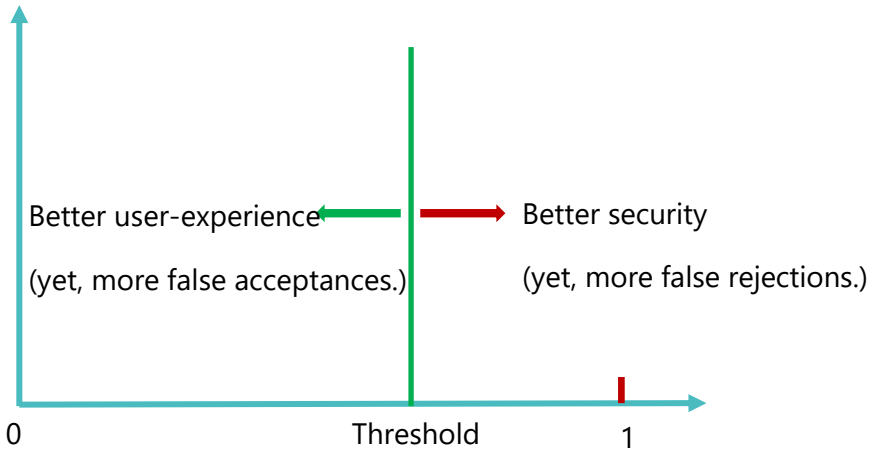

Figure 1 - Threshold dilemma

7.2. Face detection

Face detection is the concept of finding the location and size of faces in an image. It shouldn't be confused with face recognition, which uses detection.

One of the most known algorithms, due to its quickness, is Haar cascades (also used in object detection). Other, much more efficient (but more computationally expensive) algorithms based on deep learning exist such as Histogram of Oriented Gradients (HOG) combined with linear Support vector machine, Single Shot MultiBox Detection (SSD), You Only Look Once (YOLO) or Multi-task Cascaded Convolutional Neural Networks (MTCNN) to name a few.

7.3. Feature extraction

There are two main types of methods to extract features from a face. The first is named CNN-based feature extraction which is based on deep learning. And the other one is the conventional feature extraction approach.

7.3.1. CNN-based

Deep Convolutional Neural Networks methods have gained in popularity in recent years as they can learn from large training datasets thus, improving performance. Among the most popular are the ones developed by Google, FaceNet, Facebook's DeepFace, VGG Face developed by researchers at Oxford University, or the open source solution OpenFace. There is also a wide variety of hybrid approaches combining conventional methods and deep learning.

7.3.2. Conventional

Before using deep learning, a lot of conventional methods for feature extraction have been developed.

One of the most used is principal component analysis (PCA) or eigenface which is based on linear algebra techniques. Other variants include 2D PCA and Kernel PCA (KPCA). Another well-known method is the Support vector machine (SVM), it's based on the concept of decision planes that define optimal boundaries. There is also the texture classification method called Local Binary Pattern

7.4. Verification system

The goal of the verification system is to decide if the face in front of the screen is a specific person. For example, if a user with ID2 is already registered in the database the verification system can evaluate if someone in front of the camera is the user with ID2 or not.

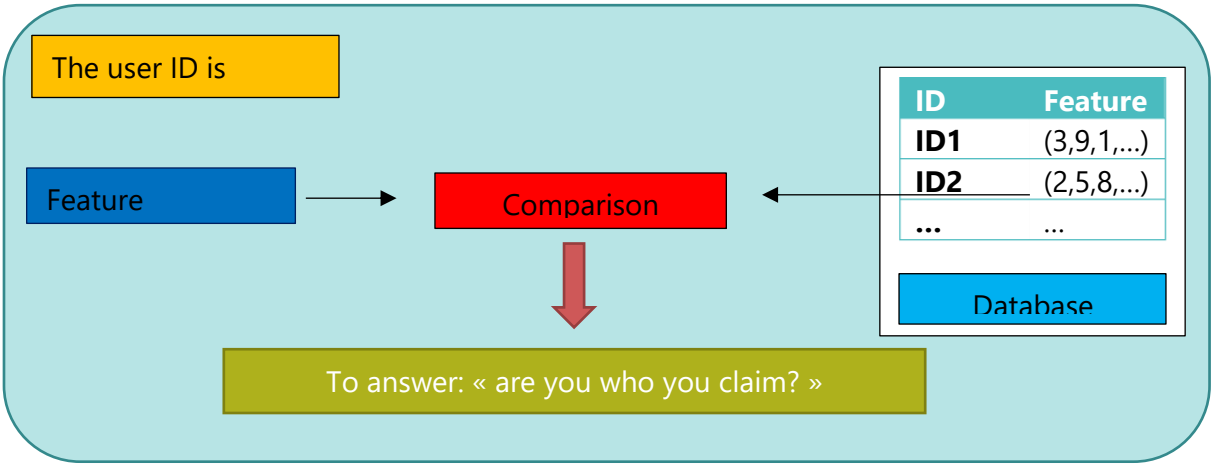

Figure 2 - Verification system concept

7.5. Identification system

The identification system is a little bit harder to develop. This system must identify a person in front of the camera. To do this, it must compare his features with all the database features and find the profile with a better comparison score. If the database is huge, it might impact the performance of the system.

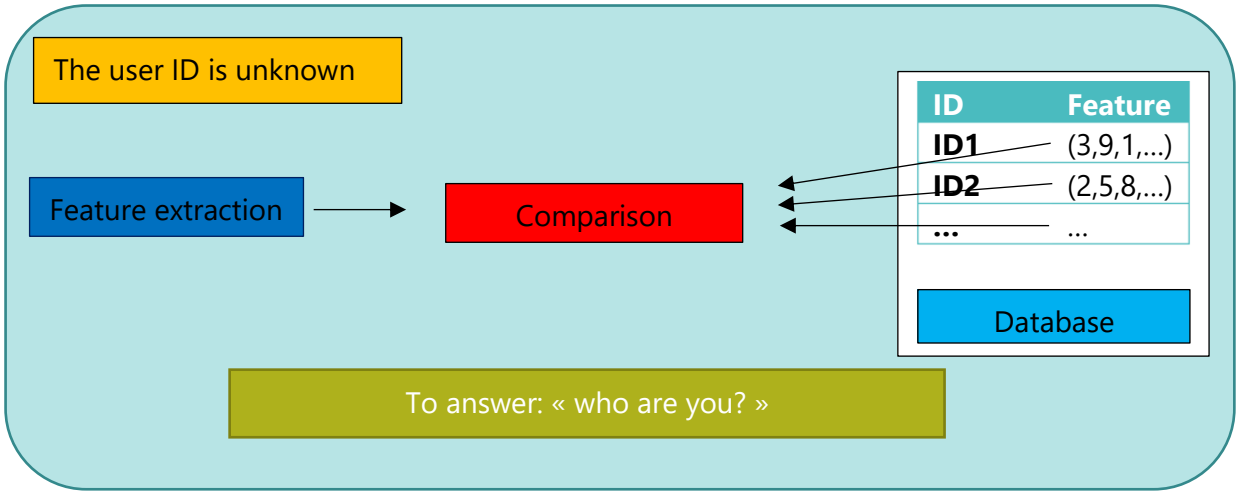

Figure 3 - Identification system concept

8. Presentation attack

According to the ISO/IEC 30107-1: 2016 the presentation attack is the presentation to the biometric capture subsystem to interfere with the operation of the biometric system. We can classify these attacks in three levels:

- level A: paper printout of the face image, mobile phone display of face photo;
- level B: level A attacks, paper masks, video display of face (with movement and blinking);
- level C: level A and B attacks, silicon masks, theatrical masks, makeup attacks.

9. Presentation attack detection (PAD)

There are two different means to detect these attacks :

- active PAD requires interaction with the user like blinking eyes, opening mouth, turning left, turning right, reading out some numbers, etc;
- passive PAD does not require user interaction and gives a better user-experience.

10. Liveness detection

Face recognition methods are not able to distinguish between bonafide (live) face and imposter face i.e presentation attack. It's an easy way to spoof face recognition systems. To protect against such spoofing, a secure system needs liveness detection which is the core of

the PAD principle. Although liveness detection methods can be applied in active detection, most research focuses on passive PAD.

The concept of face liveness detection can be separated into three categories: motion analysis, texture analysis, and life sign detection. The pros and cons[1] of these methods are summarized in fig.4.

Liveness categorie	Pros	Cons
Motion analysis	simple implementation, good results in known scenarios, possible decision from one frame, no user-collaboration needed	needs data that covers all possible attacks, problem with low textural attacks
Texture analysis	very hard to spoof by 2D face image, independent of texture, no user-collaboration needed	needs video sequences, can be spoofed by 3D sculptures, needs high quality image, challenged by videos with low motion activity
Life sign	very hard to spoof by 2D face images or 3D sculpture, independent of texture	may need collaboration from user, depends on landmark detection in the face, needs video sequences

Figure 4 - Overview of liveness detection approaches

10.1. Motion analysis

Motion analysis is based on the clue that planar objects like printed photo (2D) move differently from real faces (3D). Most of the motion analysis methods are based on optical flow. One of the assumption used is that the motion of planar objects is almost constant regardless of the area of the face. Bao et al.[2] has shown that the most discriminative movement type (illustrated in fig x) to distinguish between bonafide and impostor is swing.

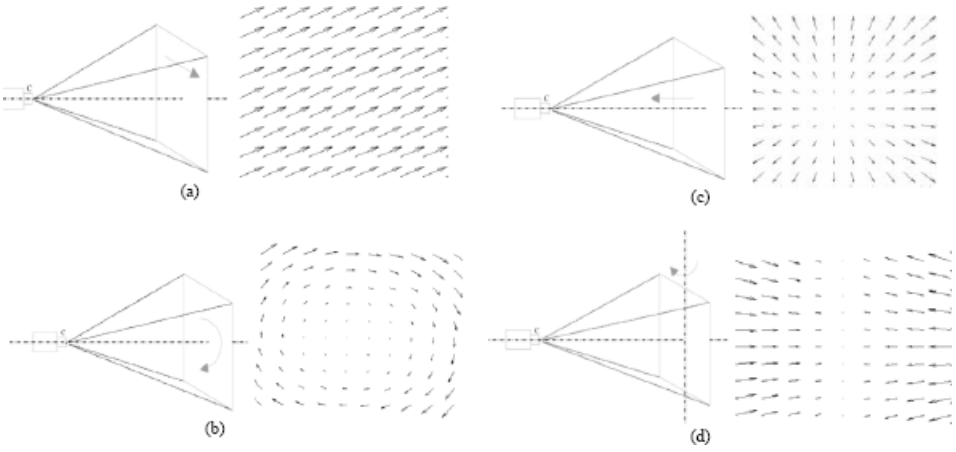

Figure 5 - Optical flow fields generated by four basic types of relative motions. (a) Translation (b) Rotation (c) Moving forward or backward (d) Swing

[1] https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.1080.2915&rep=rep1&type=pdf, https://arxiv.org/ftp/arxiv/papers/1405/1405.2227.pdf

[2] https://ieeexplore.ieee.org/document/5054589

10.2. Texture analysis

Texture analysis is based on the assumption that printed faces contained detectable texture patterns. Here, texture features are extracted from face images (single images or sequences) under the assumption that fake faces are printed, and the printing process or the material (paper) printed on produces certain texture patterns that do not exist in real faces. It's often paired with frequency analysis.

10.3. Life sign detection

Life sign detection tries to analyze signs of life from user images (eye blinking, eye movements, lips movements). The developed algorithms under this approach focus on the movement of a certain identified part of the face.

11. Effectiveness metrics of a biometric system

To determine the effectiveness of a biometric system, it is necessary to have metrics that reflect that effectiveness. Many metrics have been created, we will present here those that are most used and that we have used.

False Acceptance Rate or FAR is the frequency that a non authorized person is accepted as authorized. Because a false acceptance can often lead to damage, FAR is generally a security relevant measure.

On the other side there is False Rejection Rate or FRR which is the frequency that an authorized person is rejected access. FRR is generally thought of as a comfort criteria, because a false rejection is most of all annoying.

Equal Error Rate or EER is when FAR = FRR i.e when we accept the same amount of non authorized person than we reject authorized person.

However, the calculation of these values is dependent on the threshold we set as acceptance and rejection of a person. This is why we use the Receiver Operating Characteristic or ROC, which plots FRR values directly against FAR values, thereby eliminating threshold parameters. An example is shown in fig.6. The ideal ROC only have values that lie either on the x axis (FAR) or the y axis (FRR). As the ROC curves for good systems lie very near the coordinate axis, it is reasonable for one or both axis to use a logarithmic scale. We usually compute the EER while computing ROC.

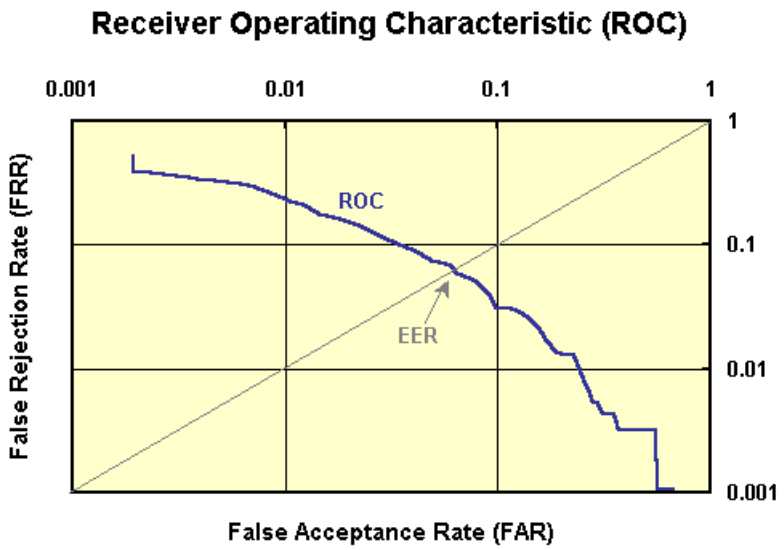

Figure 6 - ROC graph example with EER plotted [3]

[3] http://www.bromba.com/faq/biofaqe.htm#ROC

PERSONAL WORK DONE

12. Technologies

12.1. Programming Language

As we said in the introduction, we decided to use Java in Android Studio. We made this choice because we had already used this environment but we needed to manipulate it more to improve our comprehension and skills. Moreover, we wanted to try to use as more as possible the principles of software engineering like design patterns or SOLID principles.

The performance tests were carried out on Matlab. Our supervisor sent us some codes from his company. We based our tests on it and we wrote some other codes.

12.2. Version manager

We decided to work with GitHub. Before this internship, we had only worked with the GitLab of ENSICAEN, but the two systems are very similar since both are using Git software.

13. Datasets

Two modified datasets were used to benchmark the performance of the various components of our application.

13.1. Yale Face Database

The Yale Face Database[4] contains 165 frontal face gray-scale images covering 15 individuals taken under 11 different conditions: a normal image under ambient lighting, one with or without glasses, three images taken with different point light sources, and five different facial expressions. An example is provided in fig.7.

[4] P. Belhumeur, J. Hespanha, D. Kriegman, ÒEigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection,Ó IEEE Transactions on Pattern Analysis and Machine Intelligence, July 1997, pp. 711-720. http://vision.ucsd.edu/content/yale-face-database

Figure 7 - Yale Face Database samples. From left to right and top to bottom: normal, with glasses, left-light, center-light, right-light, without glasses, happy, sleepy, sad, surprised and wink.

13.2. Cropped Yale Face

To avoid redundant processing, we cropped the images once using our face detection module. This is the dataset that we used, it is referenced as Cropped Yale Face in this report. An example is provided in fig.8.

Figure 8 - Cropped Yale Face samples

13.3. Internal dataset

Our supervisor provided us with a large dataset of more than 35 GB used internally. It consists of 1800 high quality 1080x1920 portrait videos taken by 6 different devices. The frame rate varies between 24 and 30 frames per second and the duration is on average 5 seconds.

It is composed of 20 individuals, for each individual there are 5 videos: one bonafide type, two presentation attack level A type (paper printout) and two presentation attack level B type (video display of face). Each of these videos for each individual exists in 18 copies: 6 different devices and 3 sessions for each device.

13.4. Internal data subset

Since the liveness detection methods we have been working on are not able to detect by design video display of face with movement and blinking, we removed the corresponding videos in the internal dataset, thus reducing the total number of videos to 1080.

As this represents a huge amount of video data and therefore significant processing times when evaluating performance, we decided to split each video into several frames based on a windowing principle: we take one frame and skip the next four. This reduced the dataset size from 21.6 GB to 8.37 GB. The dataset is now composed of a total of 29,000 frames with about 20 to 30 frames for each source video depending on the frame rate. This new dataset is the one used in the following and we will refer to it as internal data subset. Examples are provided in the figures down below.

Figure 9 - Internal data subset sample: same individual, same device, same session, different type. From left to right: bonafide and the other two are paper printout

Figure 10 - Internal data subset sample: same individual, same device, same type (bonafide), different sessions

Figure 11 - Internal data subset sample: same individual, same session, same type (bonafide), different devices

14. Face detection

Face detection is the fact of identifying a human face in digital images. In our case, we do that with Machine Learning Kit (ML Kit), Google's open-source package for mobile developers. This package gives many APIs like face detection, text recognition, or text translation. With the Face Detection API, we can identify faces in the smartphone's camera and the contour of face key details, for instance, the eyes or the mouth as you can see in fig.12.

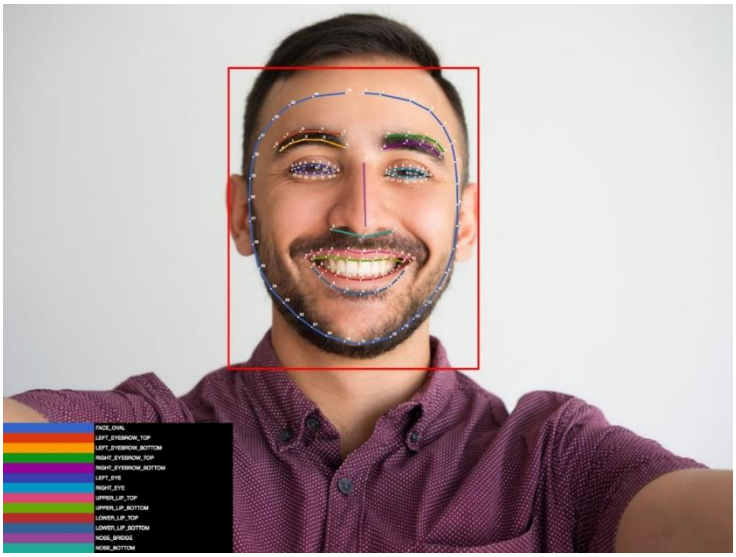

Figure 12 - Face contours by ML Kit

In addition, we can use this API with specific parameters :

- Performance Mode can be set to fast or accurate. The accurate mode is slower than the fast mode but can detect faces more precisely.
- The Landmarks Mode gives us a list of key points.
- The Face Contours Mode gives us a list of points that we can see in figure 4.

- The Classification Mode gives us some probabilities like open eyes or smiling.

15. Face recognition

We decided to develop a simple face recognition module. As mentioned in the state of the art, there are many ways to do face recognition with or without deep learning. One of the simplest way is to use the Local Binary Pattern (LBP) method.

The LBP algorithm sets each pixel in an image by comparing the gray level with the number of neighboring pixels. The first step of this method is to divide the frame in a *n by n* equally sized cells as you can see in fig.13. This parameter is crucial, most of the time we use $n = 7$ or $n = 8$ for face recognition.

Figure 13 - Face divided into 7×7 equally sized cells

In each cell, we apply the LBP algorithm. The algorithm is the following:

- Convert the image in grayscale.
- For each pixel (P is the number of neighbors, g_c represent the graylevel of the pixel, g_i the graylevel of its neighbours i, for i in [0, P-1]):
    - For $i\ in\ [0, P-1], g_i = \begin{cases} 1 & ,if\ g_i > g_c \\ 0 & ,otherwise \end{cases}$
    - For $i\ in\ [0, P-1], g_i = g_i \times 2^i$
    - $g_c = \sum_0^P g_i$

You can see a simplified example of how this algorithm works in fig.14[5].

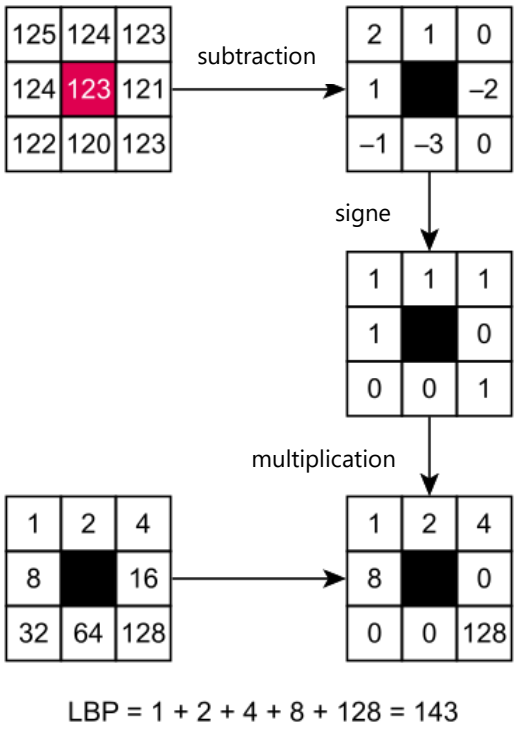

Figure 14 - LBP algorithm example with P = 8 neighbours

After that, we obtain a histogram describing the frame.

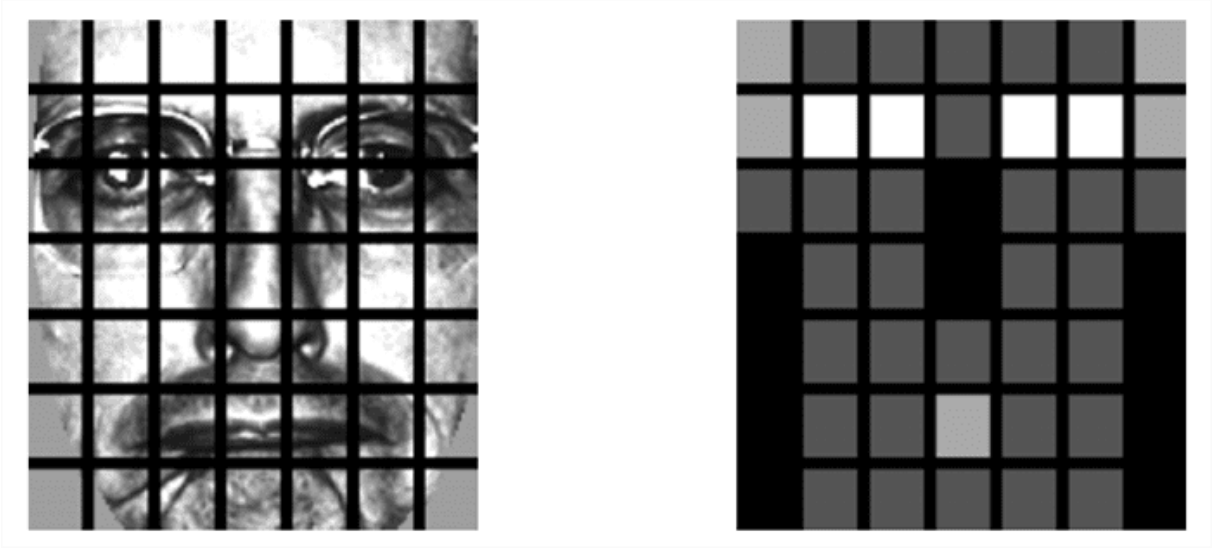

Figure 16 - The original face image (left) followed by its histogram (right)[6]

[5] https://fr.wikipedia.org/wiki/Motif_binaire_local

[6] https://pyimagesearch.com/2021/05/03/face-recognition-with-local-binary-patterns-lbps-and-opencv/

Finally, our system applies functions that merge histogram's values to return a vector of 256 integers to describe the face features. This algorithm is known to have a high discrimination power. Thanks to this property, we will be able to differentiate people.

We have made some performance tests on the Cropped Yale Face dataset to determine the best parameter and the efficiency of our implementation. We've used different grid size and different metrics to get a similarity score between two image's histogram. The metrics used are euclidean distance and cosine distance. An example of the similarity scores obtained by comparing all images in the dataset is shown in fig.17.

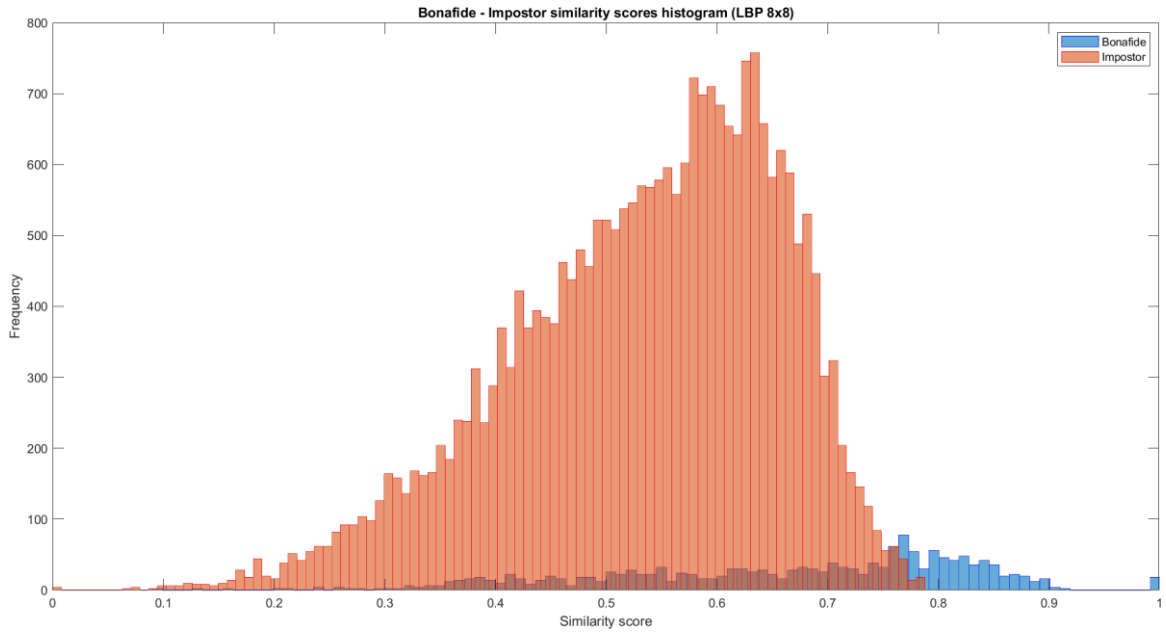

Figure 17 - Bonafide - Impostor similarity scores histogram example. Impostor scores (not the same person) are in red and bonafide (same person) scores in blue.

We made some tests using cosine distance but it did not change the performances compared to euclidean distance, this is why we decided to use euclidean distance. The summarized results with the EER are presented in fig.18.

	LBP 6x6	LBP 7x7	LBP 8x8
Equal Error Rate	34.26%	34.06%	32.6%

Figure 18 - LBP results with EER for different grid size. Euclidean distance was used to compute similarity scores.

As we can see, the most accurate configuration is to use a 8x8 grid in LBP. The results are not good but our goal was not to implement a high performance face recognition method, we just wanted to implement a simple one as our main topic was liveness detection. Since the dataset contains a lot of different illumination variations, we tried to apply a log function on each gray level pixel which lightens the black parts without touching the brightest parts. Nevertheless, the results did not improved which is not surprinsing since LBP is not sensitive to monotic transformation and the log function is monotic.

The ROC curves for the LBP 8x8 configuration is shown in fig.19. The EER is plotted, with this curve you can determine which amount of impostor (FAR) will be accepted for a given bonafide acceptance rate (1-FRR) and vice-versa. For instance, if we decide to have a 50% bonafide acceptance rate, we will have an impostor acceptance rate of about 5%.

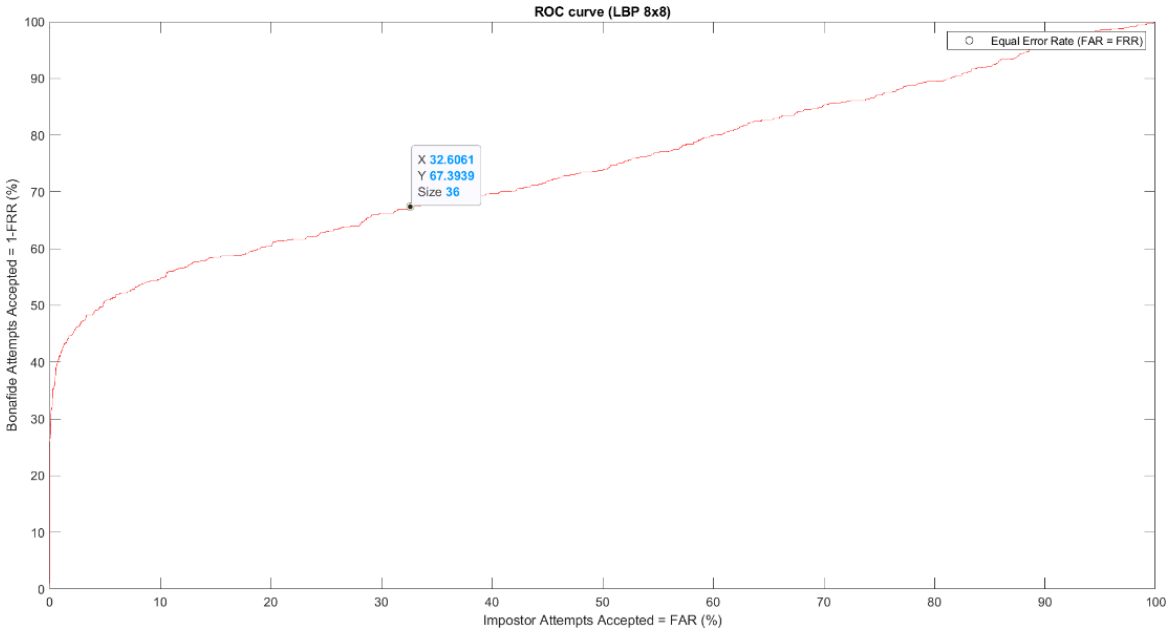

Figure 19 - ROC curve for LBP with 8x8 grid size. x-axis represents impostor acceptance rate (FAR) and y-axis bonafide acceptance rate (1-FRR).

16. Liveness detection

In order to present our progress, we have implemented an activity where the user can try all protocols and methods used in face acquisition and authentication. The user has to choose in the settings, the functionalities he wants to see on the screen. In this part, we will present these functionalities.

16.1. Micro-movements

As mentioned in the state of the art, there are many different ways to detect micro-movements. Here we present the ones we have tried.

16.1.1. Face orientation

ML kit provides different information when detecting a face, one of these is the face orientation. It gives three values, the Euler angles, to describe the orientation of a face as illustrated in fig.20. Euler x corresponds to a pitch, y to a yaw, and z to a roll.

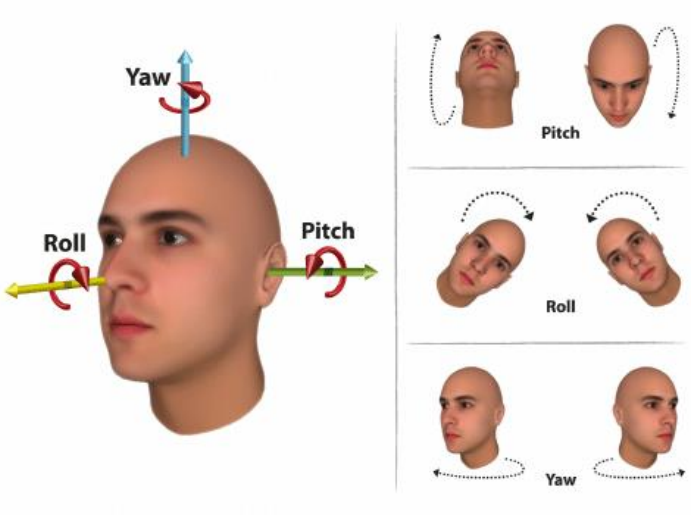

Figure 20 - head orientation angles provided by ML Kit[7]

Our first intuition was to look at the difference in the angle values provided by ML Kit between a real person and a printed photo. To do this, we positioned the capture device, the smartphone, and the printed photo stationery. The smartphone records at 30 frames per second. For the person, we tried to be as static as possible. As you can see in fig.21, which represents the values of each angle for each frame, the results obtained for the person are normal. However in fig.22, for the printed photo, we expected the angles to be constant but it turned out that there were micro-variations (less than one degree). We put this down to the margin of error of the algorithm used by ML Kit.

Figure 21 - head angle orientation for a real person standing still

[7] https://www.researchgate.net/figure/Orientation-of-the-head-in-terms-of-pitch-roll-and-yaw-movements-describing-the-three_fig1_279291928

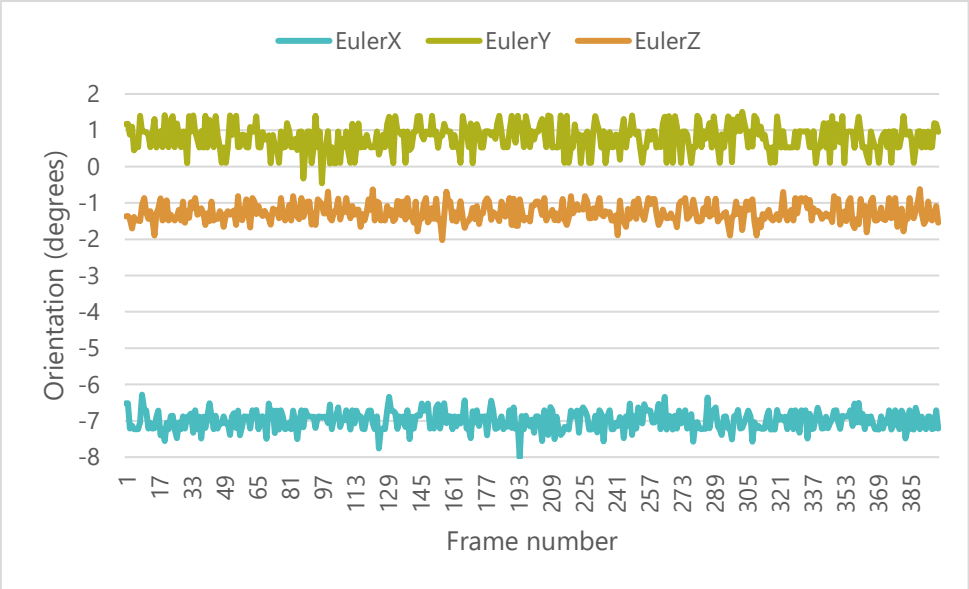

Figure 22 - head angle orientation for a static printed photo

Even if, from a global perspective (the graphs represent about 10 seconds of video), the variations are smaller for a printed photo, when viewed at a smaller scale, the differences in variations are not enough to identify a pattern. Furthermore, this is a still print, but it's possible to simulate larger variations just by holding the print in your hands as shown in fig.23.

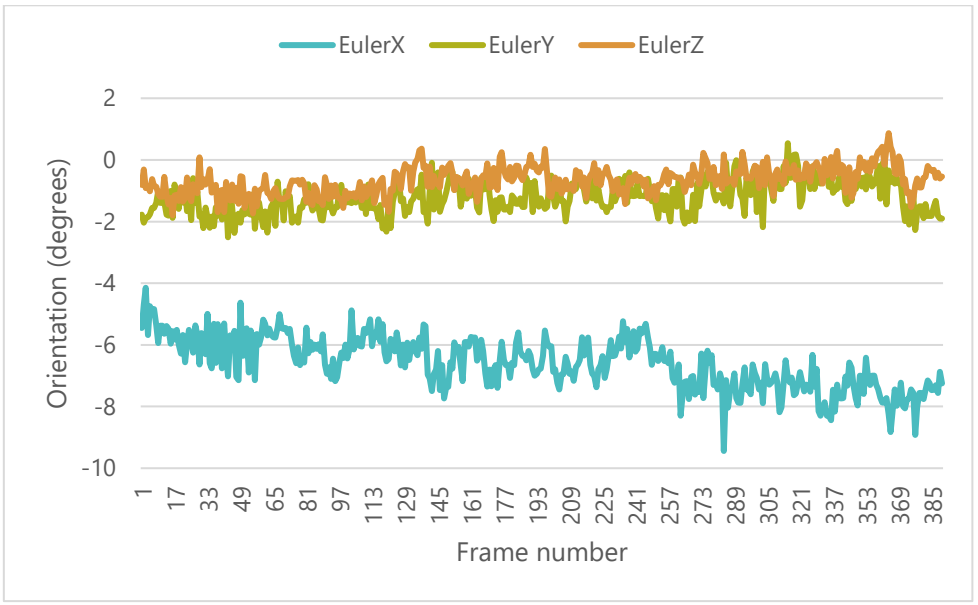

Figure 23 - head angle orientation for a hand-held printed photo

Since the different angles provided by ML Kit are not enough to reliably determine the liveliness of the subject in a micromovement context, we tried to determine the truthfulness of the angle changes using the distance between several points on the face. Indeed, as illustrated in fig.24, the distance between two points on the face during movements varies in different ways depending on whether the surface is planar (b) or not (a).

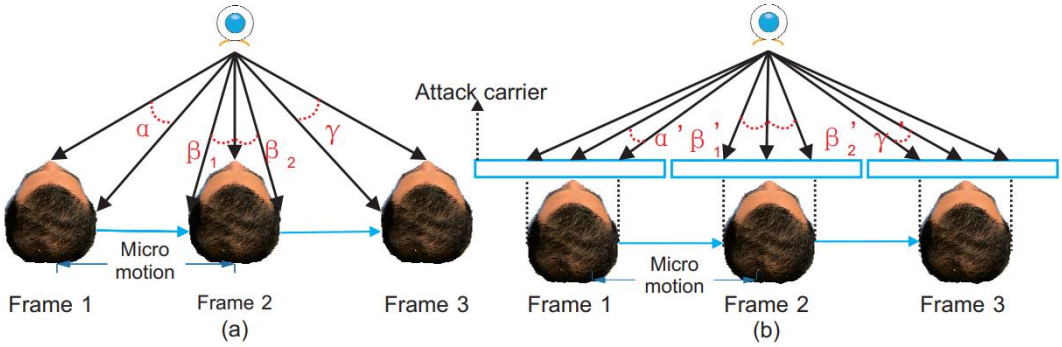

Figure 24 - face points distance difference between non-planar (a) and planar (b) subjects for exaggerated head movements[8]

ML Kit provides the position of a multitude of points on the face, it is therefore theoretically possible to determine whether it is a printed photo or not. However, after several tests, we realized that the algorithm tried to keep a constant distance between the different points even during large head movements as shown in fig.25. Therefore, we cannot use the angles to determine whether a subject is alive or not.

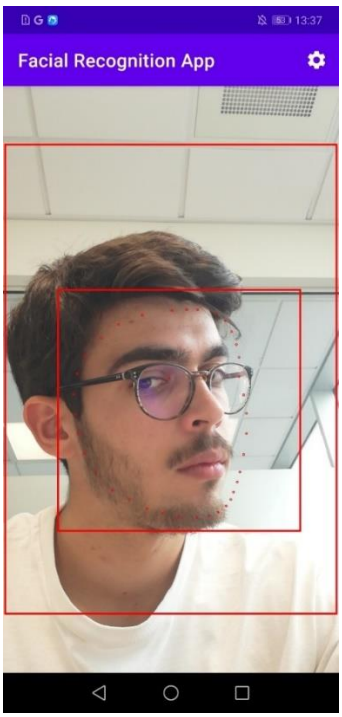

Figure 25 - Screenshot illustrating the constant landmarks provided by ML Kit, at the right of the face, even when we turn the face

16.1.2. Background subtraction

Since we could not use the angles to determine whether a person was alive or not, we fell back on a more common solution, background subtraction. This is a method to determine motion in a video stream through the pixels differences between multiple frames.

[8] https://arxiv.org/abs/1811.05118

First, we convert a frame into a grey-scale image G_t. Then, given the previous image G_{t-1}, we compute the resulting image O_t as follows (x,y are the coordinates of a pixel):

$$O_t(x,y) = \begin{cases} 255 & if\ |G_t(x,y) - G_{t-1}(x,y)| > Threshold \\ 0 & otherwise \end{cases}$$

We are using a threshold to only detect significant pixel changes between frames. By doing so, static area are displayed in black and moving area in white. The challenge here is to determine a value for the threshold. To do so we have done performance tests with different thresholds.

As most devices provide at least 30 frame per second on their front camera, which means one frame every ~33 ms, we decided to not process every frame as the time between two frames is too short to detect any motion. We rather use a window moving mechanism which takes the first frame of a 5 frame window, ignoring the next 4 frames.

In order to measure the performance of this liveness detection method, we calculated the EER. The dataset used is the internal data subset. As our algorithms are implemented in an Android application, the performance tests are performed on a smartphone which poses many constraints when processing large datasets such as the one we use. We therefore decided to use only the images that were captured by device 1 and 2, which reduces the dataset size by 3. We also decided not to crop the images using ML Kit, we therefore apply our method on the whole image.

We applied the background subtraction method to each frame with differents threshold value, in this dataset the frames were extracted from the videos every 5 frames, which corresponds to a duration of ~166 to 200ms depending on the framerate. To calculate a single value expressing the difference between two images we used the root mean square (RMS). Finally, for each video, we took the maximum RMS and categorized it according to whether it was a bonafide or a presentation attack in order to calculate the EER. Once the RMS is normalized, it acts as a liveness score. Nevertheless, for a real-time liveness detection system, we can't wait for an entire video of about 5 seconds before deciding if it's bonafide or a presentation attack. This is why we also computed the maximum RMS for a window of n frames rather than for an entire video. The results are shown in fig.26, for a better understanding, we do not specify the window size in terms of frames but in terms of milliseconds based on the mean framerate. We call it time before decision.

Threshold	Time before decision (ms)	Equal Error Rate (lower is better)
15	500	7.68%
	1000	6.48%
	2000	5.87%
	whole video	near 0%
30	500	8.37%
	1000	7.96%
	2000	7.65%
	whole video	4.18%
45	500	8.94%
	1000	9.26%
	2000	8.02%
	whole video	5%

Figure 26 - Background subtraction performance as EER. Different threshold used.

As we can see, the best performances comes with a threshold set to 15. These results also show us that the difference in grey level, when it exists, between pixels for bonafide images is generally between 15 and 30 because the performance degrades when we use a threshold set to 30 and higher. We reach an EER near 0% simply because, by taking an entire video, the maximum RMS between two frames for a presentation attack video is never higher than in a bonafide video. Furthermore, for a real-time liveness detection, we have good results as we get for a time before decision of 500 and 1000 ms respectively an EER of 7.68% and 6.48%.

However, these good results for a rather simple method, background subtraction, must be put in perspective with the dataset. The presentation attack videos (printed paper) used here have two characteristics that facilitate their detection: the printed paper is quite static but most importantly the background of the image has not been cut out to keep only the face. If the face had been cropped, the method used here would have been ineffective. This is why in a liveness detection system we use several methods to determine the liveness, such as blinking which in this case would have been efficient.

16.2. Eye gaze

This concept is split into different parts. The first step is to detect whether the eyes are open or close and also the eyes blinking. If the eyes are opened, we can continue to the second step, the eyes tracking. This step must detect whether the user is looking at the smartphone or not.

16.2.1. Eyes blinking

In order to detect the eyes blinking, we analyzed the normalized height of each eye *x*. Indeed, thanks to the bounding box around the box we can normalize the height of the eye center. During all the listening, we stock the maximum value of the normalized height *max*. At each frame there are three possible cases:

$$\begin{cases} max = x\,, & if\ x > max \\ eye\ closed\,, & if\ x < threshold * max \\ nothing, & otherwise \end{cases}$$

To fix the value of this threshold, we have did tests with different lights, and we set it to 65% of the maximum value. This value may seem high, but as you can see in fig.27, the less value registered is around 54% when the eyes are completely closed. Indeed, due to the performance of ML Kit, even when the eyes are closed, the points at the top of the eyes do not overlap with those at the bottom.

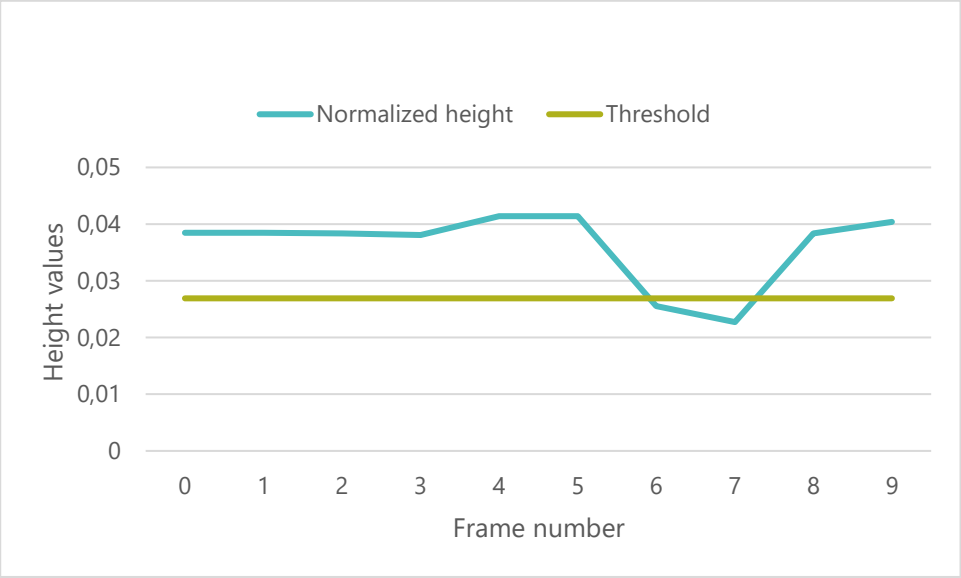

Figure 27 – Normalized eye height obtained during an eye blinking

Thanks to blinking detection we can detect if the face in front of the screen is a photo. Everyone has to blink also if the app doesn't detect blinking the face could be a photo. It is a form of passive interaction between the user and the smartphone less intrusive than asking him to move his head in different directions.

16.2.2. Eyes tracking

If the eyes are open, the application needs to know if the user wants to do face recognition. For instance, in the case that someone forces the user to unlock his phone, if the user doesn't look in the direction of the screen, the smartphone still be locked.

To detect if the user looks at the screen, we need to have the user's face centering and in front of the camera. In this position, we have to verify if the iris are in the center of the eyes, as you can see in fig.28.

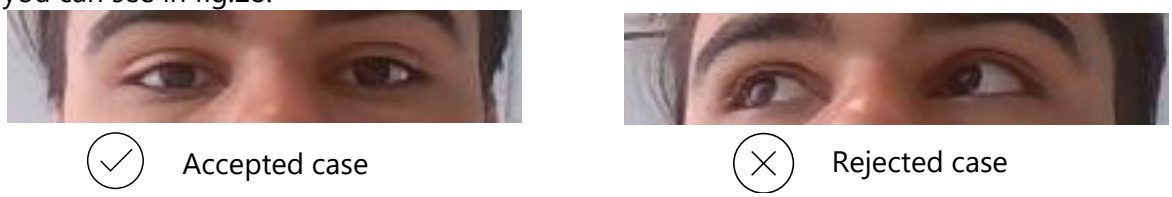

Figure 28 - Different cases of eye tracking

First, to determine the average color of each eye in grayscale. To do that, we create a rectangle with the more extreme eye points as you can see in fig.29,and examine the value of all the pixels inside the rectangle to determine its average color.

Figure 29 - Rectangle used for the eye's average color

After that, we take another rectangle in the center of the eye, as you can see in fig.30, and compare all its pixel values to the average color of the eye already estimated. We can also calculate the percentage of pixels darker than the average color. If this percentage is higher than one threshold, we can conclude that the iris is centered. Thus, the user looks at the camera.

Figure 30 - Iris' rectangle

It was very hard to test eye gaze because we don't succeed to have access to blinking and eye-moving dataset. We can only do tests with our own eyes. We both have dark eyes and we need to do tests on people with blue or green eyes.

17. Android Application implementation

17.1. Specification

The principal goals of our application is to work on our initial topics. We could have developed only a liveness detection solution but we wanted to have a global view of the face recognition process of which liveness detection is part. That is why we choose to do a real time face recognition app which can detect a presentation attack.

17.2. Architecture

We decided to use the MVC model. In this architecture model, the model store and save the data, the controller manages the interactions with the user that are all classes to do face recognition and the view present the information to the user, in general, it is a camera view.

17.3. Modules

17.3.1. Face detector

The class FrameAnalyzer is in charge of instantiating the ML Kit FaceDetector. We used the CameraX library from Google to access device camera and gives ML Kit images. We spent quite

a lot of time understanding how CameraX works and how to link the camera image's buffer to ML Kit.

In order to realize real-time acquisition and authentication, we had to study different configurations of this API. We found a good trade between speed and efficiency. To have a real-time app, we can't use all of the modes simultaneously. Also, we decided to only use the Face Contours Mode with the Accurate Performance Mode. This choice allows us to have a good user-experience with real time camera and precise detection. Moreover, with these settings, we can calculate the landmarks by ourselves.

17.3.2. Face attention detector

Some conditions are needed to allow the program to extract face features. First, it is checked if the user's whole face is on the screen. To check it we use a rectangle centered on the screen and covering 80% of the screen. If the point of the nose is in this rectangle we can consider that the face is quite centered. We need this verification because in some cases ML Kit can do an approximation of points' position even though they are off-screen. It is a problem for us because for instance if our application will be used to unlock a phone, the user can unlock his phone unintentionally and it can raise a problem of security.

Then, the user's face needs to be straight. ML Kit gives us face orientation values. Thanks to this we don't have to manipulate the frame to have a straight face.

Finally, the phone and user have to be the more static as possible to have exploitable frames i.e non-blurry frames for liveness detection. Also, we stack the last ten frames' nose position in a FIFO stack (First In First Out). If the difference in nose position between the frames is less than a certain value, we can consider that the user and the phone are quite a statics to continue. These ten frames (1/3 of a second on a 30 fps camera device) allow enough time for the device to focus in case of sudden movements of the phone or the person.

For each condition, if the user's face doesn't respect it a toast appears on the screen to notify the problem and the border of the screen stays red. The green color means that all conditions are respected and the acquisition or the authentication can start.

17.3.3. Face acquisition

When a frame satisfies all these conditions, it is necessary to resize the frame to have a bitmap around the face. Once this has been done, we can start the acquisition with the LBP algorithm. We do this with the Catalano Framework[9], which is a Java framework including lots of libraries mainly focusing on mathematical tools. We especially use the Imaging library with several Bitmap processing tools. All of the methods used to compute LBP output and compare two histograms are in the util class LBP.

17.3.4. Database

The management of the database is done by the model. Indeed, the class LocalDB has all methods to add a new user or search a user by his name in the database. The creation of the

[9] https://github.com/DiegoCatalano/Catalano-Framework

database is done by a util class MySQLiteOpenHelper which extends SQLiteOpenHelper. This class is instantiated in LocalDB and we use it as a singleton to have only one instance of this class.

As you can see in fig.31, each profile is registered by an auto-set id, his name, his features, and his date of register.

We use a local database because all the users' data is only to do face authentification. Moreover, for user security, it is better to have a local database. The only person who has access to these features is the phone owner.

id	name	features	registered_date
Filtre	Filtre	Filtre	Filtre
1	Tony	247, 171, 83, 198, 143, 2, 170, 168, 3...	Thu Aug 04 13:51:00 GMT+02:00 2022
2	Romaric	429, 69, 50, 53, 208, 8, 140, 165, 60, ...	Thu Aug 04 13:51:30 GMT+02:00 2022
3	Fabien	417, 263, 82, 250, 134, 19, 207, 274, ...	Thu Aug 04 13:51:47 GMT+02:00 2022

Figure 31 - Sample database displayed via DB Browser

17.3.5. Face authentication

As we have seen in the state-of-the-art (part 7.4), a verification system is the fact to decide if the face in front of the screen is a specific person. Also in our case, face authentication is similar to this process. After the user has entered his name for authentication, we check if his profile is in the database. If it is correct, we can start the authentication. We start by using the LBP algorithm to calculate the features. If the features found are quite close to the ones returned by the database and no attacks are detected, we authenticate the user. During the whole process, we use the liveness detection to detect eventual attacks.

17.3.6. Liveness detector

The liveness detector is only used for face authentification. Indeed, this system is here to prevent 2D attacks by checking some conditions to decide if the face in front of the screen is a real person or not.

As described before, we are using background subtraction to detect micromovements. We have implemented the background subtraction module on our own. However, in order to facilitate operations on Bitmaps, we have used some methods of the Catalano Framework.

Another part of liveness detection is the eye gaze. To be sure that the face in front of the screen is a real person we use the eyes blinking. Indeed, if it is a photo it can't blink. Also, we just need to detect one blinking. A second condition is to have opened eyes that look at the screen to be sure that the user wants to authenticate his face.

17.3.7. GUI overlay

The GUI in most of our application's interfaces consists of displaying the video stream returned by the device's camera. The GUI overlay is the module that allows you to draw over it. To do this, GUI overlay implements an interface called FrameListener, the methods of this interface, which provide the coordinates of the points to be drawn, are called by the classes that get the results from ML Kit. A major difficulty encountered was to translate the coordinates of the input image's coordinate system to the view coordinate system. There are many parameters to take into account such as the orientation of the phone, whether the image is taken with the front camera or not, which implies a lot of translation and scaling.

17.4. UI

When the user open the application, he go to the home screen as you can see in fig.32, and can choice to register a new profile, authentificate himself, or just try the liveness detection with his own settings.

Figure 32 - Home screen

If the user choose register or authenticate he will go to a screen to fill in his name as present in fig.33. For the registered screen if the name is already in database the user can't access to the next screen and a toast appears. For the authentification screen if the name is not in the database it is same process with another toast.

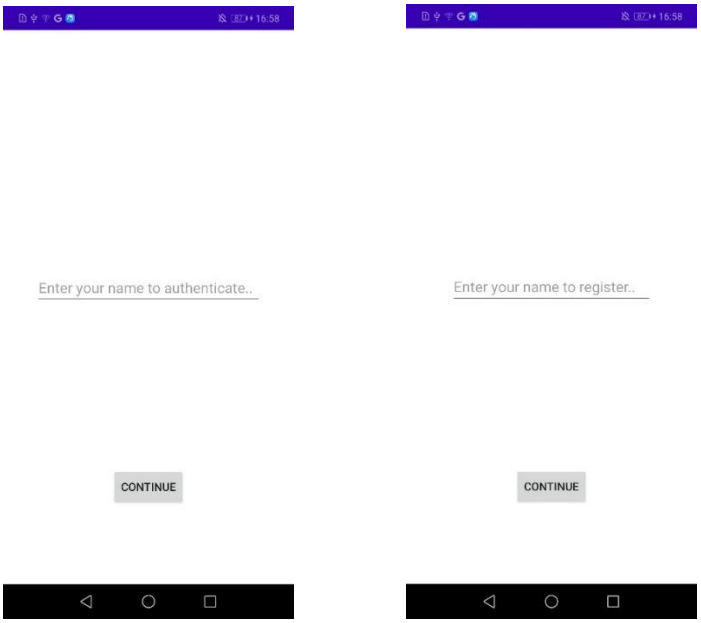

Figure 33 - Name entry screens

If these conditions are completed a new screen with a camera appears as we see in fig.34. The borders stay in red when all conditions are not respect. To register, it is the face attention conditions we have seen in part 16.3.2. To authenticate, it is these conditions combine with liveness detection.

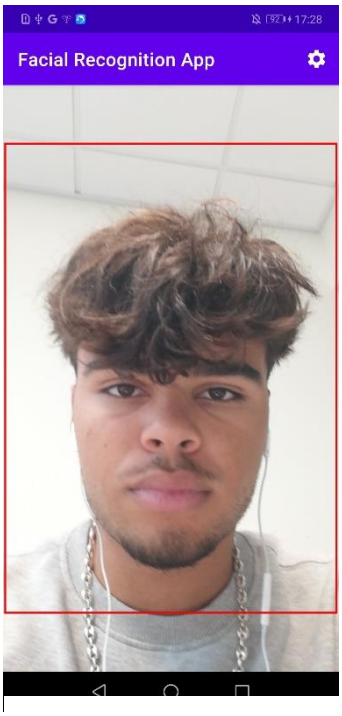

Figure 34 - Camera screen

If all conditions are checked boders become green and a new screen appears to confirm the end of the process fig.35. But if the process is too long the screen will show a rejection message.

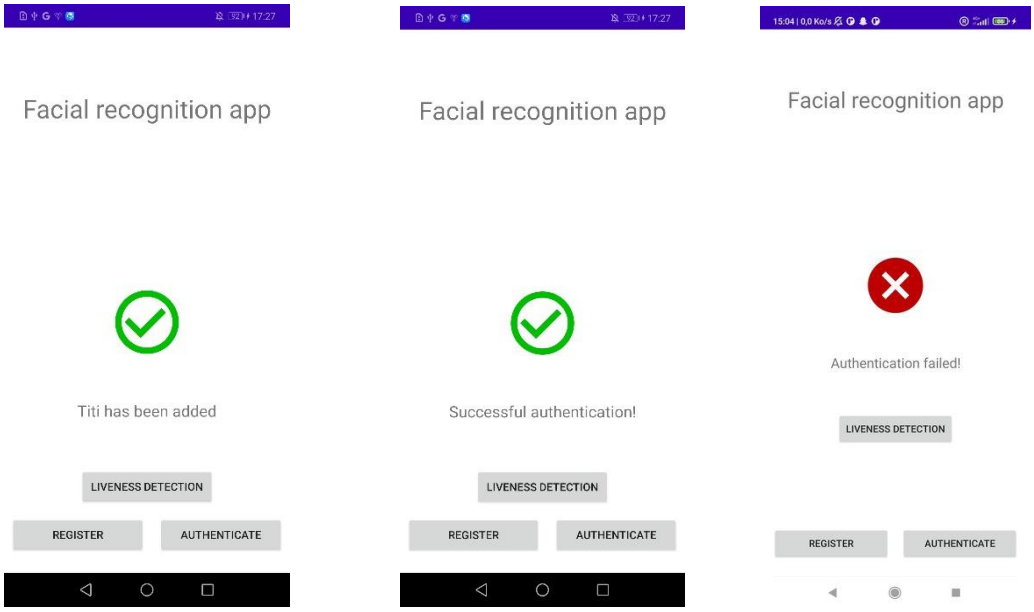

Figure 35 - End of process screens

Finally, the liveness detection screen give to the user different options to show what he want on the camera screen. To do his choices, the user have to click on the logo setting at the right top of the screen.

Figure 36 - Liveness detection screen

CONCLUSION

18. Analysis

This project was very interesting both to develop our hard skills and our soft skills. Indeed, we had to work with a supervisor and his company Mobai. We worked in autonomy during the major part of the internship and also we developed some habits like daily morning discussions between us to split tasks and talk about the work already done. Moreover, at the end of each week we reported on our progress with our supervisor.

In addition, we discovered very interesting fields, face recognition and liveness detection. In fact, that is the first time we work on it and it was a really good introduction to these topics. We worked on it both in Python with the library OpenCV at the beginning and in Java Android with Google ML Kit. This has improved our python skills but more importantly our mobile application development skills.

19. Limitations

Most of the methods used in facial recognition to detect attacks are based on the use of infrared cameras. But our topics deal with frame analysis, we can't use infrared cameras. That is the reason why the only type of attack we can detect is the printed or screen photos and not videos.

20. Future Development

20.1. Improving recognition performance

The performance of our face recognition system is strongly linked to the lighting. Indeed, the LBP algorithm determines the features vector by manipulation of the pixel color. This algorithm is very efficient in the same lighting conditions. However, if one part of the face is more or less lit than the other part, a person can have features values too different to be authentificated as him.

20.2. Use of deep learning

Throughout this project, we did not used machine or deep learning methods. Nevertheless, we think that nowadays it's a must-have to achieve very efficient performance.

20.3. Porting to iOS

We have chosen to create an Android app. This choice has mainly based on the fact that we know Android Studio and Java coding. But we have never used iOS language in our studies. That is why we have chosen to make it for Android. A possible improvement of our app can be to make a porting to iOS.

ENSI CAEN

ÉCOLE PUBLIQUE D'INGÉNIEURS
CENTRE DE RECHERCHE

Ecole Publique d'Ingénieurs en 3 ans

6 boulevard Maréchal Juin, CS 45053
14050 CAEN cedex 04

ministère éducation nationale enseignement supérieur recherche

Normandie Université

Cti Commission des titres d'ingénieur

cdefi Conférence des Directeurs des Écoles Françaises d'Ingénieurs

European Accreditation of Engineering Programmes EUR-ACE®

CONFÉRENCE DES GRANDES ÉCOLES