Cahier des charges

Reconnaissance d'empreintes digitales

Mention Mathématiques et Informatique
2ème année Génie Mathématiques

25/11/2023

Membres du projet
GANZHORN Octave, GOUTH Thomas,
PAULY Alexandre, SABADIE Laura

Encadré par :
Lucie DESPLAT
lucie.desplat@cy-tech.fr

Nelly BARRAU
nelly.barrau@cy-tech.fr

Table des matières

1 Introduction

Contexte du projet :

Dans le cadre du module de Traitement du Signal en 2ème année du cycle ingénieur à CY Tech, un projet de traitement d'images en Python a été mis en place. L'objectif principal de cette initiative pédagogique est de mettre en pratique les concepts théoriques enseignés au cours de ce module, avec une attention particulière portée sur le traitement d'image.

Notre mission s'est alors focalisée sur la reconnaissance d'empreintes digitales. Cette thématique offre une opportunité intéressante d'appliquer les techniques avancées de traitement d'image dans un contexte concret, permettant ainsi d'explorer les aspects techniques de la biométrie à travers la manipulation d'images.

Objectifs du projet :

Le projet vise à répondre aux besoins d'un client fictif, soucieux d'assurer la sécurité au sein de son institution. Cette entité utilise actuellement un système d'authentification basé sur les empreintes digitales. La mission consiste à élaborer un programme efficace capable de vérifier instantanément si une empreinte donnée appartient à la base de données. A terme, l'objectif est d'optimiser le processus d'authentification, et d'assurer la sécurité des données, renforçant ainsi la sécurité de l'institution et offrant une solution fiable et efficace pour la gestion des accès.

2 État de l'art

2.1 La biométrie

La biométrie comprend tout un ensemble de technologies et procédés de reconnaissance, d'authentification et d'identification des personnes à partir de certaines de leurs caractéristiques physiques ou comportementales. Ces caractéristiques doivent être à la fois :

- *Universelles : pour être utilisables par tous* ;
- *Uniques : pour distinguer de manière claire les individus* ;
- *Invariables : pour permettre une utilisation tout au long de la vie* ;
- *Mesurables : pour permettre la comparaison.*

Les trois technologies biométriques les plus utilisées sont la reconnaissance des empreintes digitales, de l'iris et la reconnaissance faciale car elles allient efficacité, fiabilité et facilité de déploiement et d'utilisation.

Dans notre cas, nous allons nous intéresser à la reconnaissance des empreintes digitales. Elle s'appuie sur le fait qu'une seule empreinte digitale comporte une centaine de points caractéristiques, aussi appelés minuties. Pour prouver que deux empreintes digitales sont identiques et établir avec certitude l'identité d'une personne, il est important de souligner que le nombre requis de minuties peut varier en fonction de la législation en place dans chaque juridiction. Par exemple, en France, la norme est d'avoir 12 minuties concordantes, mais dans d'autres pays, ce chiffre peut différer en raison de leurs propres réglementations.

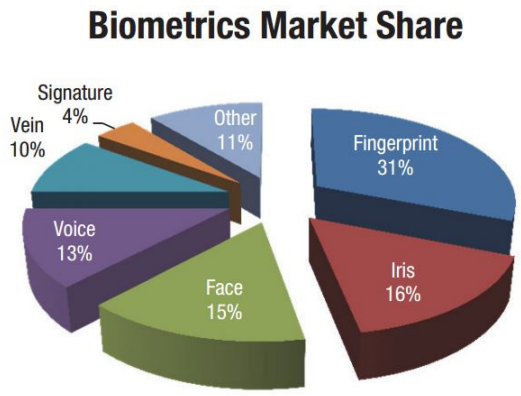

Figure 1: Biometrics market share by system type

FIGURE 1 – Marché mondiale de la biométrie

2.2 Qu'est ce qu'une empreinte digitale ?

Les empreintes digitales sont formées par les crêtes papillaires, des reliefs présents sur la face palmaire des mains et des pieds. Ces crêtes papillaires sont comme de petites collines et vallées sur la peau. Elles commencent à se former très tôt lors du développement

de l'embryon et conservent leurs caractéristiques tout au long de la vie. Chacune de ces empreintes est unique à chaque individu, même chez les jumeaux identiques, et présente des variations entre les doigts d'une même personne. Ces crêtes dermo-épidermiques, formant la structure appelée dactylogramme, jouent un rôle essentiel dans le renforcement de la préhension des doigts comme à agripper les objets. Ainsi, cette singularité offre une opportunité précieuse pour l'identification individuelle.

2.3 Forme générale de l'empreinte

Les empreintes digitales possèdent des motifs différents. En tenant compte de ces derniers, il est possible d'établir un classement. En effet, il existe 3 grandes familles d'empreintes qui regroupent à elles seules 95% des doigts humains :

- *Les « **boucles** » représentent 60% des doigts humains : Dans ce type d'empreinte, les lignes se replient sur elles-mêmes, soit vers la droite, soit vers la gauche.*
- *Les « **tourbillons** » **(aussi appelés en spirale ou verticilles)** correspondent à 30% des doigts humains : Cette empreinte, dite en verticille, comprend des lignes qui s'enroulent autour d'un point, formant un genre de tourbillon.*
- *Les « **arches** » regroupent seulement 5% des doigts humains : Cette empreinte, en arc, contient des lignes disposées les unes au-dessus des autres qui forment des A.*

FIGURE 2 – Différentes empreintes digitales (en boucle, en tourbillons, en arc)

Par ailleurs, même si on retrouve globalement tous les types d'empreintes, certains motifs sont plus présents dans certaines parties du monde. Par exemple, il y a 51% de tourbillons et 47% de boucles en Asie de l'est, tandis qu'en Europe, on trouve 27% de tourbillons et 66% de boucles.

2.4 Minuties ou motifs d'empreinte

On différencie les motifs entre eux grâce à l'aide de "points singuliers" sur les empreintes :

Les points singuliers globaux :

- ***Noyau ou centre** : c'est à dire le lieu de convergence de stries* ;

— ***Delta*** *: lieu de divergence des stries.*

Les points singuliers locaux appelés aussi minuties :

Ce sont des points d'irrégularité se trouvant sur les lignes capillaires. On peut relever jusqu'à 16 types de minuties mais dans les algorithmes on n'en retient que quatre types :

— ***Terminaison à droite ou à gauche*** *(minuties située en fin de stries)* ;
— ***Bifurcation à droite ou à gauche*** *(intersection de deux stries)* ;
— ***Iles*** *: assimilée à deux terminaisons* ;
— ***Lac*** *: assimilée à deux bifurcations.*

FIG. 4 : Noyau

FIG. 6 : Delta

FIG. 5 : Terminaison

FIG. 7 : Bifurcation

3 Objectifs du projet

3.1 Besoins fonctionnels

3.1.1 Description générale du système

Le système de vérification d'identité est basé sur la comparaison de deux ensembles de minuties, correspondants respectivement à deux empreintes à comparer. La méthode la plus répandue implique l'extraction des minuties à partir du squelette de l'image, préalablement préparée par une étape de binarisation.

Pour déterminer la concordance entre deux ensembles de minuties extraits de deux images distinctes, une séquence de prétraitements est effectuée. Cela comprend le prétraitement de l'image, suivi de la binarisation, l'application d'un filtre pour la squelettisation, permettant ensuite l'extraction des minuties. Enfin, la comparaison des minuties des deux empreintes est réalisée pour établir l'identification d'une même personne.

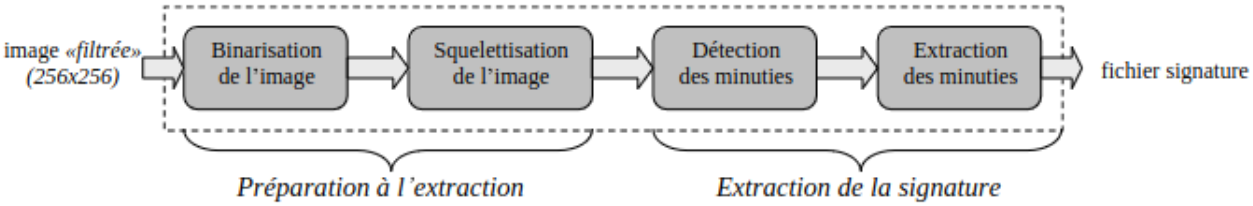

FIGURE 3 – Etapes de traitement d'une empreinte digitale

3.1.2 Fonctionnalités principales

La binarisation est une étape cruciale dans le processus de traitement d'image. Elle consiste à convertir une image en niveaux de gris en une image binaire, où chaque pixel est soit noir (représenté par la valeur 1) ou blanc (représenté par la valeur 0). Cette transformation permet de simplifier l'image et de mettre en évidence les éléments clés pour la détection des minuties.

Par la suite, la squelettisation intervient pour affiner l'image binaire. Elle utilise un filtre pour réduire l'épaisseur de toutes les lignes de crête à une largeur d'un pixel. Cela permet d'extraire les points caractéristiques tout en préservant leur emplacement et orientation par rapport à l'original. Ainsi, une image squelettisée conserve les informations essentielles pour une estimation précise des minuties.

Ces deux étapes de préparation facilitent la détection des minuties. L'image obtenue après binarisation et squelettisation présente des pixels noirs avec une valeur de 1 et des pixels blancs avec une valeur de 0. Pour localiser les points caractéristiques, le nombre de croisements est souvent utilisé. Ce nombre, défini comme la moitié de la somme des différences d'intensité entre deux pixels adjacents, permet de classer les points caractéristiques en terminaisons et bifurcations.

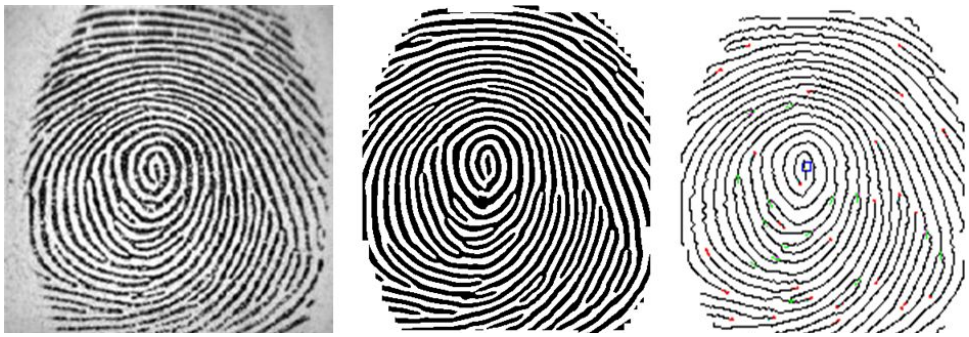

FIGURE 4 – Image originale, image binarisée et image squelettisée

3.2 Besoins non fonctionnels

3.2.1 Sécurité

En ce qui concerne le RGPD (Règlement Général sur la Protection des Données), il n'est pas obligatoire de procéder au chiffrement des données. Cependant, il est requis d'adopter des mesures garantissant la protection des données, englobant la sécurité, la confidentialité, l'intégrité, l'authenticité, la non-violation, et la prévention de la perte de ces données.

C'est pourquoi nous avons choisi de sécuriser les données, notamment les empreintes digitales, en les cryptant. Le cryptage offre divers avantages, notamment la préservation de la confidentialité visuelle et la prévention de l'accès non autorisé à un contenu original. Cette mesure garantira que les données ne puissent être vues que par les personnes habilitées à recevoir ces informations.

3.2.2 Interface

L'interface graphique constitue un aspect suupplémentaire du système. Elle offre une expérience plus intuitive à l'utilisateur, facilitant la navigation et l'interaction avec le système d'identification d'empreintes digitales. Pour des informations détaillées sur son utilisation, référez-vous au manuel d'utilisation fourni.

4 Spécifications techniques

4.1 Langages de programmation et librairies

Dans le cadre de notre projet d'identification et de reconnaissance d'empreintes digitales, le choix des bibliothèques Python revêt une importance cruciale pour assurer l'efficacité de notre application. Chacune de ces librairies a été soigneusement sélectionnée en fonction de sa spécialisation dans des domaines clés du traitement d'images, contribuant ainsi à la réalisation complète du projet.

NumPy est une bibliothèque centrale de la programmation Python, spécialisée dans la manipulation de tableaux. Pour ce projet d'identification et de reconnaissance d'empreintes digitales, une image est simplement un tableau NumPy où chaque élément représente un pixel. Elle facilite également le processus de chargement des images via la librairie **skimage** et leur affichage avec **matplotlib**. En combinant ces fonctionnalités, le projet dispose d'une capacité complète de traitement d'images, de la manipulation initiale des données à la visualisation des résultats obtenus.

OpenCV est utilisé pour l'acquisition d'images et la réalisation de traitement d'images. Dans le fichier **project.py**, OpenCV est utilisé pour charger, afficher et effectuer des opérations de traitement sur les images. Par exemple, la fonction **cv2.imread** est utilisée pour charger les images, et **cv2.threshold** et **cv2.adaptiveThreshold** sont utilisées pour la binarisation.

Le module **PIL (Python Imaging Library)**, également connu sous le nom de **Pillow**, est utilisé pour des opérations élémentaires de traitement d'images. Pillow propose des fonctionnalités de base de traitement d'image, comprenant des opérations ponctuelles, un filtrage avec des noyaux de convolution intégrés, ainsi que des conversions d'espace colorimétrique. Dans ce cas, elle permet d'afficher des images pour l'interface graphique.

SciPy étend les capacités de **NumPy** en fournissant des fonctionnalités supplémentaires pour le traitement de l'image. La bibliothèque SciPy permet d'effectuer une convolution dans le processus de squelettisation de l'image. La convolution est effectuée avec différents filtres Laplaciens pour obtenir des images squelettisées qui sont ensuite utilisées pour détecter et extraire les minuties.

Matplotlib est une bibliothèque de visualisation utilisée pour afficher des images et des graphiques. Elle permet d'afficher les images originales, binarisées, squelettisées et les minuties détectées lors de l'analyse des résultats depuis l'interface graphique.

Time est une bibliothèque de Python utilisée pour mesurer le temps d'exécution des différentes étapes du traitement. Elle est utilisée dans le fichier **project.py** pour mesurer le temps d'exécution des différentes étapes du projet.

Fernet est une bibliothèque de chiffrement utilisée pour le cryptage et le décryptage

d'images. Le fichier **cryptage.py** inclut des fonctions telles que la génération de clés, le chiffrement et le déchiffrement d'images, ainsi que le renouvellement périodique du cryptage pour renforcer la sécurité des données, notamment des empreintes digitales. Cette méthode vise à garantir la confidentialité des informations sensibles. En complément, **base64** aura permis de faire le lien entre la lecture et l'écriture de la clé stockée dans *key.txt*.

io, shutil et csv sont des bibliothèques de Python permettant de manipuler les dossiers et fichiers. Création, lecture, écriture, etc.

Tkinter est la bibliothèque de Python dédiée à la création d'interfaces graphiques. Dans le cadre de ce projet, elle est utilisée pour créer une interface utilisateur permettant d'effectuer la reconnaissance d'empreintes digitales. Le code du fichier **interface.py** utilise des composants Tkinter tels que les cadres (frames), les boutons, les menus déroulants ou encore les onglets pour structurer et organiser l'interface graphique de manière claire. Ces éléments permettent à l'utilisateur d'effectuer des actions telles que charger une image, choisir des méthodes de traitement, et lancer le processus de reconnaissance.

De NumPy à Tkinter, chacune de ces librairies joue un rôle spécifique, allant de la manipulation des pixels à la création d'interfaces graphiques. Pour les étapes détaillées de l'installation de ces bibliothèques, consultez le manuel d'utilisation.

4.2 Base de données

Dans l'idée d'effectuer une reconnaissance, il a fallu intégrer une base de données d'empreintes digitales. Celle provenant de FVC2004, en particulier le fichier DB1 B.zip, comprenant 80 images d'empreintes digitales uniques a été très utile a la mise en place des tests.

Ces empreintes digitales présentent divers degrés de qualité en raison de variations telles que l'insuffisance ou l'excès d'encre lors de la capture. À celà, il a fallut ajouter un fichier csv pour organiser cette base de données. En effet, il associe un identifiant unique à chaque image, facilitant ainsi la gestion et la référence ultérieure. Par ailleurs, les minuties de chaque empreinte sont également stockées au même endroit pour les même raisons.

5 Architecture du système

Cette section se concentre sur la structure générale du système en détaillant les interactions et les relations entre ses différents modules. Son objectif est de faciliter la compréhension de l'organisation et de la conception du système, offrant ainsi une vision claire de la manière dont les éléments du code collaborent pour atteindre les objectifs du projet, qu'ils soient fonctionnels ou non fonctionnels. Destinée spécifiquement aux développeurs, cette section vise à guider efficacement la compréhension et la mise en œuvre du système.

5.1 Architecture globale

À la racine du système, le répertoire "DB" regroupe la base de données d'empreintes digitales et ses informations, déjà cryptées pour garantir la confidentialité des données. Les principaux composants du système sont organisés comme suit :

1. **project.py :** Ce fichier central regroupe les fonctions essentielles pour le traitement d'images et la reconnaissance d'empreintes digitales.
2. **interface.py :** Gère l'interface à partir de laquelle les différentes méthodes codées en Python seront appelées.
3. **cryptage.py :** Ce fichier a pour objectif de contenir le code nécessaire pour débruiter la base de données avant de la crypter à nouveau. Cette manipulation peut être effectuée régulièrement voire à une périodicité constante. Ces méthodes auront besoin de s'appuyer sur le fichier `key.txt`.
4. **key.txt :** Contient la clé de cryptage essentielle à la sécurité des données et à la manipulation des données.

5.2 Modules du système

project.py :

La reconnaissance d'empreintes digitales va pouvoir prendre forme dans ce fichier. Pour cela, il faudra procéder par étapes en commençant par effectuer un prétraitement des images (binarisation et squelettisation), puis par l'extraction de minuties et pour finir, la comparaison avec la base de données. Pour cela, il faut :

1. **binarize_image(image, method) :** Cette fonction va binariser une image donnée qui sera décryptée au préalable à partir de la méthode renseignée. Concernant les méthodes, seuls 3 ont été retenues (Otsu, moyenne adaptative et Gaussienne adaptative) par la fonction `threshold et/ou adaptiveThreshold de cv2`. Ces trois méthodes produisent des résultats très similaires lors de la binarisation d'une même image, comme illustré ci-dessous :

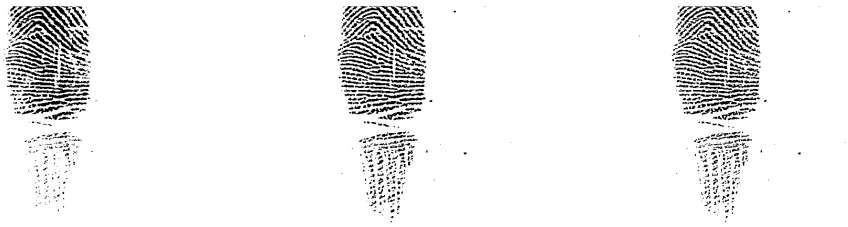

FIGURE 5 – Différentes binarisations avec les méthodes Otsu, moyenne adaptative et Gaussienne adaptative.

2. **skeletonize_image(image, method) :** Cette fonction va squelettiser une image donnée qui sera binarisée au préalable à partir de la méthode renseignée. Concernant les méthodes, une seule aura été retenue, la méthode du filtre Laplacien (M) avec lequel on fera une convolution grâce à la méthode `signal.convolve2d(image, M, mode='same')` de la librairie signal.

   En effet, nous avons expérimenté plusieurs filtres pour effectuer la squelettisation. Dans un premier temps, nous avons testé la squelettisation avec les méthodes `Guo Hall, Zhang Suen et Morphology` mais les résultats obtenus se sont avérés peu cohérents. Ces méthodes ont détecté des contours inexistants sur l'image et tracé des lignes verticales et horizontales dans toutes les directions autour des empreintes digitales. C'est pourquoi nous n'avons pas retenu cette méthode.

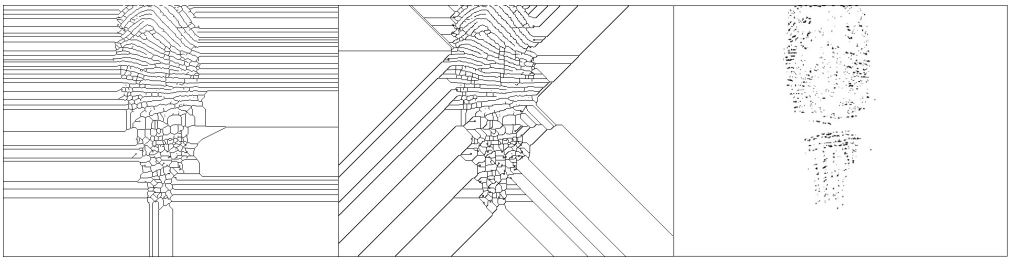

FIGURE 6 – Différentes squelettisations avec les méthodes Guo Hall, Zhang Suen et Morphology.

Dans un second temps, nous avons expérimenté le filtre Laplacien avec ses trois approximations discrètes du Laplacien.

$$M1 = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \quad (1) \qquad M2 = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ 0 & 0 & -1 \end{bmatrix} \quad (2) \qquad M3 = \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 4 \end{bmatrix} \quad (3)$$

Nous avons obtenu des résultats différents pour la squelettisation. En effet, avec les matrices M1 et M2, nous observons des résultats similaires, à quelques détails près. Cependant, avec la matrice M3, nous avons observé une profusion excessive de détails (entraînant l'apparition d'un trop grand nombre de minuties), comme illustré ci-dessous :

FIGURE 7 – Différentes squelettisations avec le filtre Laplacien avec les approximations M1, M2, M3

Nous avons donc opté pour le maintien des approximations M1 et M2, facilitant ainsi la comparaison lors de l'extraction des minuties, et avons décidé de ne pas utiliser M3 en raison d'une surcharge d'informations.

Par ailleurs, nous avons également expérimenté la squelettisation en utilisant le filtre de Sobel avec la matrice suivante, qui s'est révélé moins efficace en comparaison avec la méthode du filtre du Laplacien.

$$M = \frac{1}{4} \times \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & -2 \\ -1 & 0 & 1 \end{bmatrix} \quad (4)$$

FIGURE 8 – Matrice M avec le filtre Sobel et rendu de la squelettisation

3. **crossing_number(image)** : Une fois que l'image a subi un prétraitement, le processus d'extraction des minuties commence en analysant pixel par pixel. En se basant sur le voisinage de chaque pixel, l'indicateur connu sous le nom de crossing number est ajusté. La valeur résultante de cet indicateur détermine le type du point : soit une continuité, soit une discontinuité, qui est identifiée comme une minutie.

Afin de garantir l'authenticité des minuties extraites, nous avons choisi de conserver uniquement celles de type bifurcation. Ces minuties sont privilégiées en raison de leur complexité accrue, ce qui réduit les chances de les trouver dans

d'autres empreintes digitales.

$$\text{CrossingNumber} = \frac{1}{2}\sum_{i=0}^{7}|B(i) - B(i+1)|$$

CN	*Property*
0	Isolated point
1	Ending point
2	Connective point
3	Bifurcation point
4	Crossing point

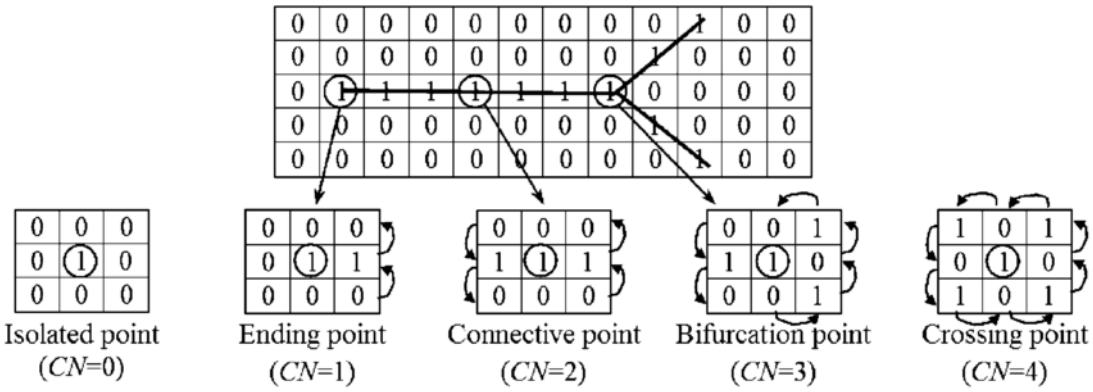

FIGURE 9 – Nature de la minutie

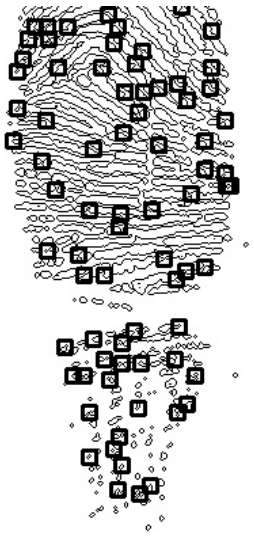

FIGURE 10 – Extraction des minuties

4. **add_minutiae(file_path, image, minutiae)** : Afin de faciliter les temps de calcul, il est préférable de récupérer les minuties de la base de données au préalable. Pour cela, il faudra les extraire à l'aide de la méthode *crossing_number(image)* et les enregistrer dans le fichier *DB.csv* afin de construire une base de données contenant non seulement des images et des identifiants, mais aussi leurs minuties. Rien de bien plus complexe : il suffit de lire l'élément auquel la liste des minuties doit être associée et de les écrire dans le fichier. Cette opération d'extraction doit être effectuée pour chaque combinaison de méthodes entre la binarisation et la squelettisation.

5. **match_template(image, template)** : Cette fonction exploite la méthode *matchTemplate* de cv2 qui retrouve une minutie (template) sur une image avec plusieurs méthodes. L'idée est de combiner les méthodes pour obtenir différents résultats et les comparer.

   Étant donné que la minutie appartienne ou non à l'image, la fonction renverra une matrice comme résultat. Nous allons récupérer ces résultats et calculer pour chacun la valeur moyenne des pixels de l'image de retour pour, par la suite, en calculer la variance afin de connaître la variabilité du résultat car, lorsque la template appartient à l'image, le retour sera très proche du noir. À l'inverse, il sera proche du blanc lorsqu'il ne sera pas correcte.

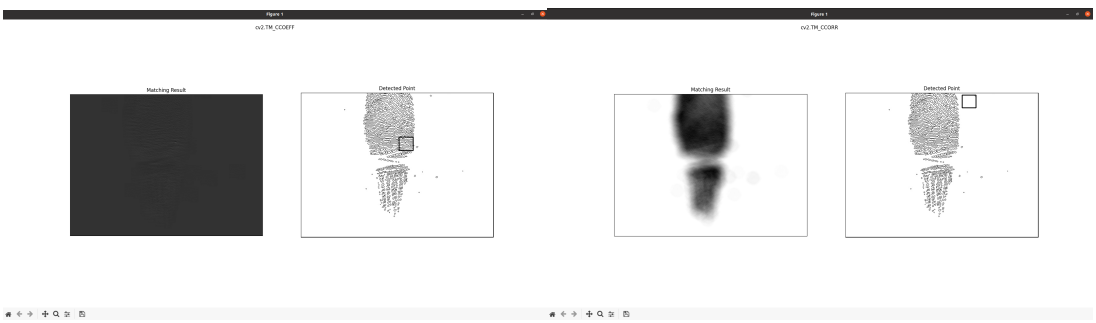

FIGURE 11 – Très bonne correspondance VS très mauvaise correspondance

6. **fingerprint_recognition(skeleton_image, binarization_methods, skeletonize_methods, nb_minutiae)** : La fonction *fingerprint_recognition* prend en entrée une image squelettisée, les méthodes de binarisation et de squelettisation utilisées précédemment, ainsi que le nombre de minuties à rechercher. Maintenant que les étapes de prétraitement et d'extraction des minuties sont complètes, la tâche suivante consiste à localiser ces minuties sur l'image à traiter.

   À l'aide des coordonnées de chaque minutie et des méthodes préalablement employées, la fonction procède à la recherche de leur présence sur l'image cible. Cette comparaison s'effectue avec chaque empreinte digitale enregistrée. Afin d'ajuster

le seuil de reconnaissance, la valeur de *nb_minutiae* peut être ajustée.

Pour valider la reconnaissance, la fonction fait appel à la méthode *match_template(image, template)*, vérifiant ainsi la présence de chaque minutie au sein de l'image cible. Cette étape est cruciale pour garantir la fiabilité du processus de reconnaissance d'empreintes digitales. Sauf que pour s'assurer de la justesse du calcul et exploiter la fonction *matchTemplate* de cv2, nous avons pris la décision de calculer une variance sur le retour de notre méthode *match_template()* afin de vérifier la variabilité du résultat.

7. **main(image_path, binarization_methods, skeletonize_methods, nb_minutiae)** : Cette fonction aura pour but de mettre en place le déroulé de la reconnaissance. De la binarisation à la recherche de la correspondance, elle fera appelle aux différentes méthodes à partir des paramètres renseignés dans l'interface (image à traiter, méthode de binarisation et de squelettisation, ainsi que le nombre de minuties à comparer).

interface.py :

Le fichier interface comprend les codes pour afficher et exploiter l'interface, y compris l'appel des méthodes des autres fichiers.

cryptage.py :

Ce fichier contient les méthodes nécessaires pour la gestion du cryptage des images en utilisant la librairie *Fernet* et *os* pour l'accès aux dossiers/fichiers. Les fonctions spécifiques sont détaillées ci-dessous, accompagnées de leur description :

1. **generateKey()** : Cette fonction génère une nouvelle clé de cryptage.
2. **encryptionImage(key, img_to_encrypt)** : Cette fonction permet d'encrypter une image donnée à partir d'une clé générée par *generateKey()*.
3. **decryptionImage(key, img_to_decrypt)** : Cette fonction permet de décrypter une image donnée à partir d'une clé générée par *generateKey()*.
4. **newEncryption()** : Méthode générale pour décrypter les données avec l'ancienne clé, puis en générer une nouvelle pour crypter à nouveau les données récemment décryptées.
5. **writeFile(path, key)** : Cette fonction écrit la nouvelle clé dans le fichier key.txt.
6. **readFile(path)** : Cette fonction permet de récupérer la valeur de la clé à partir de key.txt.

Les fonctions de la librairie *Fernet*, telles que *Fernet(base64.urlsafe_b64decode(key))*, *cipher_suite.encrypt(image_data)*, et *cipher_suite.decrypt(image_data)*, sont utilisées respectivement pour décoder la clé, encrypter une image, et décrypter une image.

Ces méthodes doivent être implémentées conformément aux instructions fournies afin de garantir l'efficacité du processus de cryptage des images dans le système.

6 Limites et contraintes

La mise en œuvre d'une méthode de comparaison d'empreintes digitales se heurte à diverses limites et contraintes qu'il est essentiel de prendre en considération.

Tout d'abord, l'importance d'une base de données de qualité revêt une importance primordiale. Bien que la base de données ait été établie avec un certain nombre d'images de qualité visuelle satisfaisante, une lacune majeure subsiste : chaque sujet ne dispose que d'une seule empreinte enregistrée. Cette limitation entrave ainsi la capacité du programme à rechercher des empreintes potentiellement similaires d'un même individu.

Par ailleurs, dans le laps de temps imparti, nous avons opté pour une seule méthode de détection des minuties, à savoir le crossing number. Bien que cette méthode soit couramment utilisée, elle aurait pu bénéficier d'une amélioration en étant couplée à des approches plus complexes. L'intégration d'un réseau de neurones aurait permis d'effectuer un apprentissage sur les données existantes, permettant ainsi l'extraction de minuties plus complexes et distinctives. Cette approche aurait potentiellement renforcé la capacité du programme à identifier des correspondances précises entre empreintes.

En outre, le nombre de minuties extraites présente des implications significatives. L'extrapolation de 20 à 100 minuties par méthode (Méthode d'Otsu, Moyenne adaptative et Gaussienne adaptative) soulève la question de l'efficacité du processus de reconnaissance. Une réduction du nombre de minuties extraites aurait pu être avantageuse pour diminuer les temps de calcul lors de la reconnaissance à partir d'une image de test. De plus, classer chaque minutie par son type aurait pu simplifier le processus de comparaison en effectuant des comparaisons intra-types, réduisant de cette façon les calculs.

Afin d'améliorer la précision du programme et aussi pour enlever l'une des contraintes que nous nous étions fixé, soit la "bonne" disposition de l'image, il serait judicieux d'effectuer des rotations de l'image à traiter. Ce serait une fonctionnalité supplémentaire permettant de garantir le succès dans un contexte plus vaste.

En somme, bien que des progrès significatifs aient été réalisés, ces limites soulignent l'importance de poursuivre la recherche de solutions. L'intégration de données plus riches, l'exploration de méthodes de détection de minuties plus avancées, et la mise en œuvre de techniques pour réduire le nombre de minuties extraites sont autant de pistes à considérer pour renforcer la fiabilité et l'efficacité de ce programme de comparaison d'empreintes digitales.

7 Conclusion

Au cours de ce projet axé sur la reconnaissance et l'identification d'empreintes digitales, nous avons pu explorer l'utilisation de Python en mettant en œuvre des techniques de binarisation et de squelettisation. Notre objectif était d'extraire les caractéristiques spécifiques des empreintes digitales pour les rendre identifiables et également de pouvoir les comparer à celles d'une base de données pour trouver la personne correspondante. Nous avons testé différentes approches, constatant que certaines étaient plus efficaces que d'autres.

En résumé, notre démarche a consisté à manipuler des images en utilisant des concepts mathématiques issus du domaine du traitement du signal. À travers cette expérience, nous avons acquis une compréhension approfondie de Python, en particulier dans le domaine du traitement d'image. Malgré les problèmes rencontrés, notre évolution dans le maniement de Python et l'application des divers concepts mathématiques ont contribué à l'aboutissement de ce projet.

8 Bibliographie

Références

[1] Empreintes digitales - Un bref historique des systèmes d'identification criminelle. (s. d.). Thales Group. https://www.thalesgroup.com/fr/europe/france/dis/gouvernement/biometrie/empreintes-digitales-et-identification.

[2] Empreintes digitales. (s. d.). INTERPOL | The International Criminal Police Organization. https://www.interpol.int/fr/Notre-action/Police-scientifique/Empreintes-digitales.

[3] 10 outils de manipulation et traitement d'images en Python. (s. d.). MonCoachData. https://moncoachdata.com/blog/10-outils-de-traitement-dimages-en-python/.

[4] Images cryptées. (s. d.). https://www.apprendre-en-ligne.net/crypto/images/index.html.

[5] FVC2004 - Third International Fingerprint Verification Competition. (s. d.). 403 - Forbidden : Access is denied. http://bias.csr.unibo.it/fvc2004/.

[6] OpenCV : Template Matching. (s. d.). OpenCV documentation index. https://docs.opencv.org/4.x/d4/dc6/tutorial_py_template_matching.html.

[7] Biometrie : Biometrics - Empreintes digitales. (s. d.). https://www.biometrie-online.net/technologies/empreintes-digitales.

[8] Pinto, L. (2023, 1 août). LE HACHAGE D'IMAGE. sfeir.dev. https://www.sfeir.dev/data/le-hachage-dimage/.