Iris Détection & Recognition

Encadré par
Mr. Abdelmonaime LACHKAR
2022

Ghazi Chaimae
Hssassa Naila

Sommaire

Sommaire

AVANT-PROPOS

Nom et prénom des membres :

Ghazi Chaimae: Elève ingénieur en 2ème année cycle ingénieur en Géni Informatique

Hssassa Naila: Elève ingénieur en 2ème année cycle ingénieur en Géni Informatique

Intitulé du travail :

Réalisation d'un programme de "Iris Détection and recognition " détecter les iris des yeux pour l'authentification OPENCV_Python

Encadrant du projet

Mr. Abdelmonaime LACHKA

Remerciement

Nous avons préparé ce rapport de projet après l'achèvement partiel de notre projet "Reconnaissance des modèles d'iris humains pour l'identification biométrique".

Nous sommes très reconnaissants a notre professeur Mr. Lachkar , notre guide de projet qui nous a aidés à entreprendre le projet en fournissant un soutien et une assistance continus.

et Nous tenons à l'exprimer notre plus sincère gratitude et Nos remerciements particuliers.

Lieu : ENSA TANGER Date : 03/06/2022

CHAPITRE 1:

CONTEXTE GÉNÉRAL DU PROJET

PRESENTATION DE PROJET

1.1. Introduction

Le travail est réalisé dans le cadre d'un projet au sein de l'ENSA de Tanger.
Ceci est un mini projet sur Python qui tourne autour du logiciel libre OpenCV(Open Source Computer Vision Library).
Le programme python s'occupe de détecter l'iris de yeux humain afin de l'identifier **.**

1.2 Problématique et objectif

Le projet va être divisé en trois régions :

1- SEGMENTATION:

La première étape de la reconnaissance de l'iris consiste à isoler la région réelle de l'iris dans une image numérique de l'œil. La région de l'iris peut être approchée par deux cercles, un pour la limite iris/sclère et un autre, intérieur au premier, pour la limite iris/pupille. Les paupières et les cils obstruent normalement les parties supérieure et inférieure de la région de l'iris. De plus, des réflexions spéculaires peuvent se produire dans la région de l'iris, corrompant le motif de l'iris. Une technique est nécessaire pour isoler et exclure ces artefacts ainsi que pour localiser la région circulaire de l'iris.
Le succès de la segmentation dépend de la qualité d'imagerie des images oculaires.

2- NORMALIZATION :

Une fois que la région de l'iris est segmentée avec succès à partir d'une image de l'œil, l'étape suivante consiste à transformer la région de l'iris afin qu'elle ait des dimensions fixes afin de permettre des comparaisons. Les incohérences dimensionnelles entre les images oculaires sont principalement dues à l'étirement de l'iris causé par la dilatation de la pupille à partir de différents niveaux d'éclairage. D'autres sources d'incohérence comprennent la variation de la distance d'imagerie, la rotation de la caméra, l'inclinaison de la tête et la rotation de l'œil dans l'orbite. Le processus de normalisation produira des régions d'iris, qui ont les mêmes dimensions constantes, de sorte que deux photographies du même iris dans des conditions différentes auront des

caractéristiques caractéristiques au même emplacement spatial.

PRESENTATION DE PROJET

1.2 Problématique et objectif

3- MATCHING :

Après avoir enregistré avec succès les caractéristiques uniques de chaque image et les avoir stockées dans une structure de données appropriée, il est temps pour nous de récupérer des informations en recherchant un certain fichier image ou index stocké dans la structure de données (tableau multidimensionnel) et correspondant avec l'image en cours d'examen. C'est-à-dire que différents modèles contenant différentes informations sont stockés dans le tableau, puis nous faisons correspondre l'image sous observation avec tous les modèles stockés dans le tableau et vérifions le pourcentage de correspondance et nous annonçons l'image avec le pourcentage le plus élevé de rapport correspondant, comme l'image probable complémentaire de l'image observée actuellement.

- **Objectif**

C'est de mettre en place un système de reconnaissance de l'iris pour identifier l'Homme (Homo sapiens) de manière unique.

L'outil de développement utilisé est Scipy, OpenCV en Python et d'autres bibliothèques python dépendantes, et l'accent ne sera pas seulement mis sur le logiciel permettant d'effectuer l'identification par reconnaissance, mais également sur l'identification des syndromes biologiques de tout être humain.

Une approche de développement rapide d'applications (RAD) sera utilisée afin de produire des résultats rapidement. Pour effectuer l'expérimentation, un ensemble de données contenant des images oculaires de plus de 100 images oculaires humaines prises sous différents angles.

PRESENTATION DE PROJET

1.3 Motivation et background

-Motivation

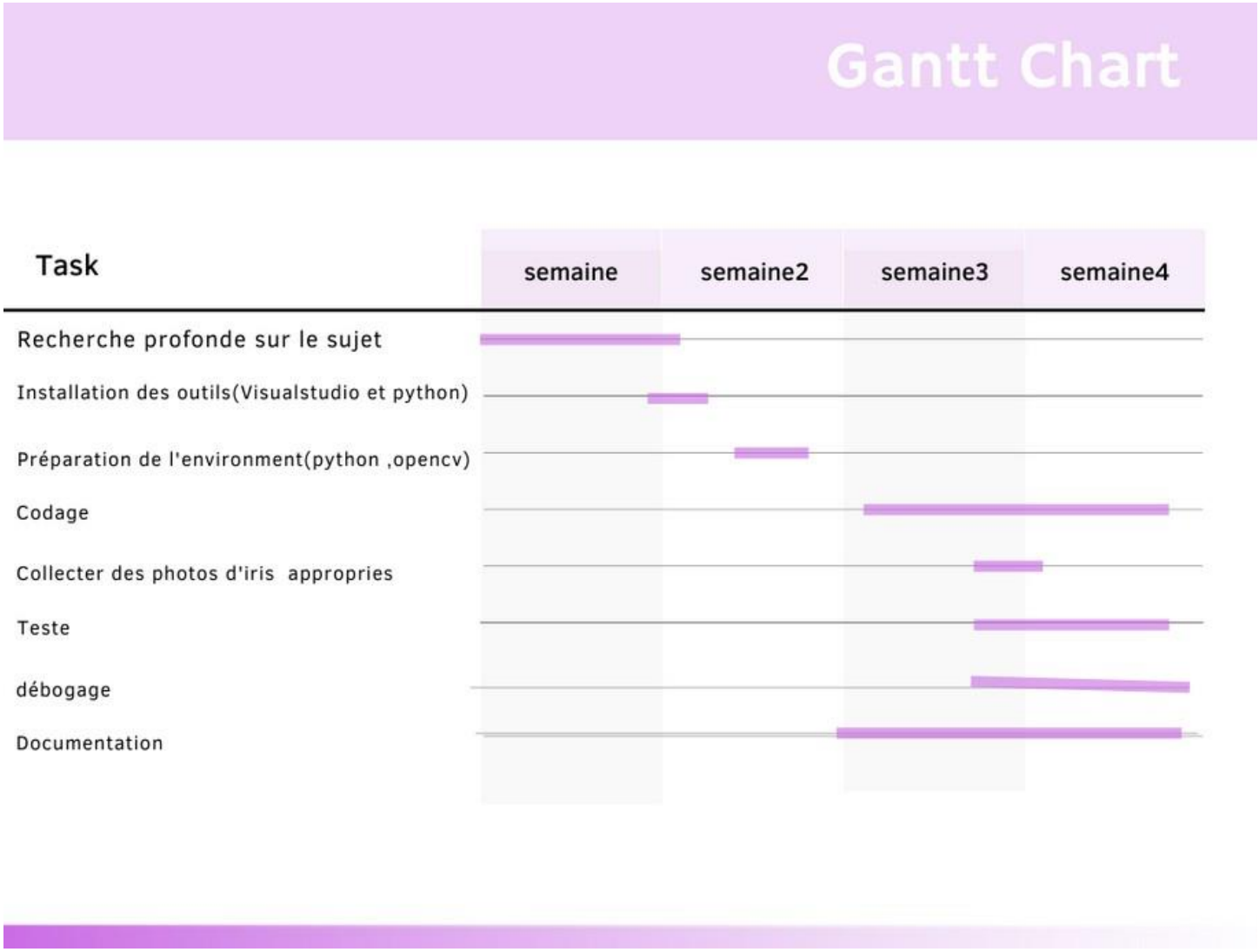
La principale motivation derrière ce projet était le projet lui même et le désir de le réaliser au temp . Nous avons essayé et avons réussi dans une certaine mesure dans ce projet et visons à nous améliorer de plus en plus avec le temps. Nous avons été très motivés par notre professeur encadrant qui a partage avec nous un ensembles de ces projets innovants. Ces dernieres ont été une clé de motivation et d'encouragement a notre decouverte du monde du traitement d'image.

-BACKGROUND :

Le système de reconnaissance humaine a joué un rôle dominant dans la société moderne. Toute personne peut être identifiée par ses traits biométriques comme les empreintes digitales, l'iris, la rétine, l'oreille, etc. Elle peut également être identifiée par son statut comportemental comme l'écriture manuscrite, les empreintes digitales, etc. Mais le motif Iris est considéré comme l'un des meilleurs sinon le meilleur moyen pour identifier n'importe quel être humain. Ainsi, l'arrière-plan du projet est la mise en œuvre d'un traitement d'image opencv réussi avec python pour segmenter, normaliser et extraire des caractéristiques de l'image pour nous permettre d'obtenir une valeur de données à partir de l'image Iris.

Diagramme de GANTT :

LE DIAGRAMME DE GANTT EST UN OUTIL UTILISÉ (SOUVENT EN COMPLÉMENT D'UN
RÉSEAU PERT) EN ORDONNANCEMENT ET GESTION DE PROJET ET PERMETTANT DE VISUALISER
DANS LE TEMPS LES DIVERSES TÂCHES COMPOSANT UN PROJET ….LES DIAGRAMMES DE
GANTT SONT UTILISÉS DANS LA PLUPART DES LOGICIELS DE GESTION DE PROJET.



CHAPITRE 2:
ETUDE FONCTIONNELLE ET CONCEPTUELLE

ETUDE FONCTIONNEL/CONCEPTUEL

1 Etude de l'existant:

DESCRIPTION DE L'EXISTANT :

L'homme peut être identifier par plusieurs manières (empreinte,signature,face....) mais toujours le problème de sécurité et de stabilité sont posés.

SOLUTIONS ENVISAGÉES:

La reconnaissance de l'iris est le meilleur processus d'authentification biométrique disponible aujourd'hui. Alors que beaucoup le confondent avec le scanner rétinien, la reconnaissance de l'iris consiste simplement à prendre une photo de l'iris; cette image est utilisée uniquement pour l'authentification. Mais qu'est-ce qui fait de la reconnaissance de l'iris, un système d'authentification de premier choix?

- Stable – l'image unique de l'iris humain est formée à l'âge de 10 mois et reste inchangée tout au long de la vie
- Unique – la probabilité que deux image d'iris produisant le même code est presque impossible
- Flexible – la technologie de reconnaissance de l'iris s'intègre facilement dans les systèmes de sécurité existants ou fonctionne de manière autonome
- Fiable – une image d'iris distinctive n'est pas susceptible d'être volée, perdue ou compromise
- Sécurisé – contrairement au dépistage rétinien, la reconnaissance de l'iris est sans contact et rapide, offrant une précision inégalée par rapport à toute autre alternative de sécurité, allant de 3″ à 10″

2 Analyse des besoins fonctionnels et non fonctionnels

2.1 Les besoins fonctionnels

L'analyse fonctionnelle est une démarche qui consiste à caractériser les fonctions offertes par un produit pour satisfaire les besoins d'un utilisateur. En effet, chacun de ces besoins reflète les attentes d'un utilisateur envers le système conçu. Nous présenterons ainsi tous les besoins fonctionnels qui serons être implémentés dans notre programme.

Comme nous le savons, pour identifier n'importe qui/quoi , nous devons le détecter et après le traitement et l'analyse on sorte avec une décision(Matching) .

Donc, logiquement, nous aurons besoin des photos des yeux à haute résolution et de type (.bmp)

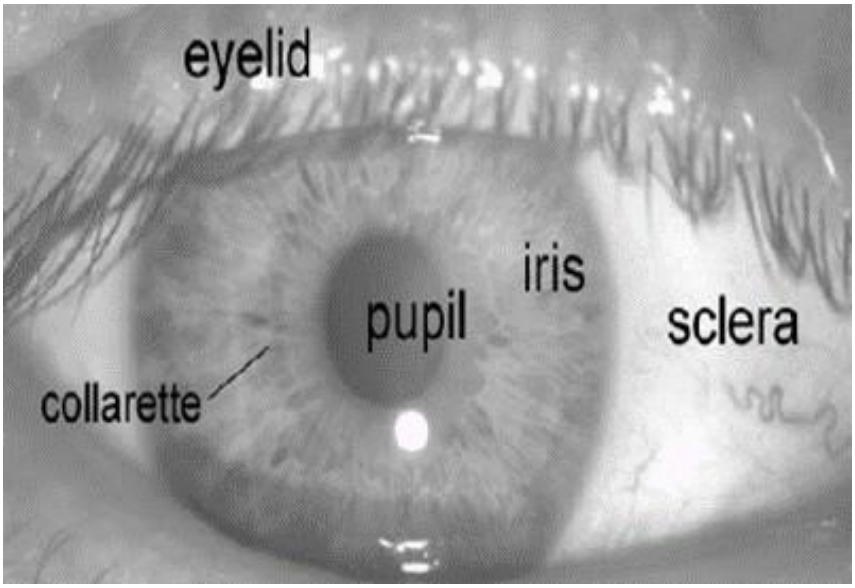
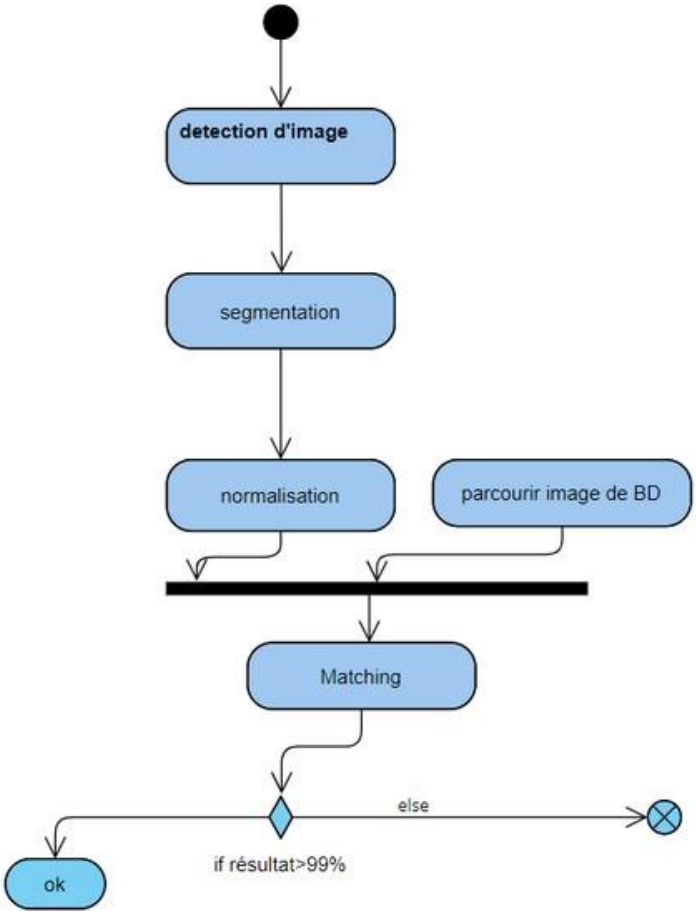


Diagramme d'activité

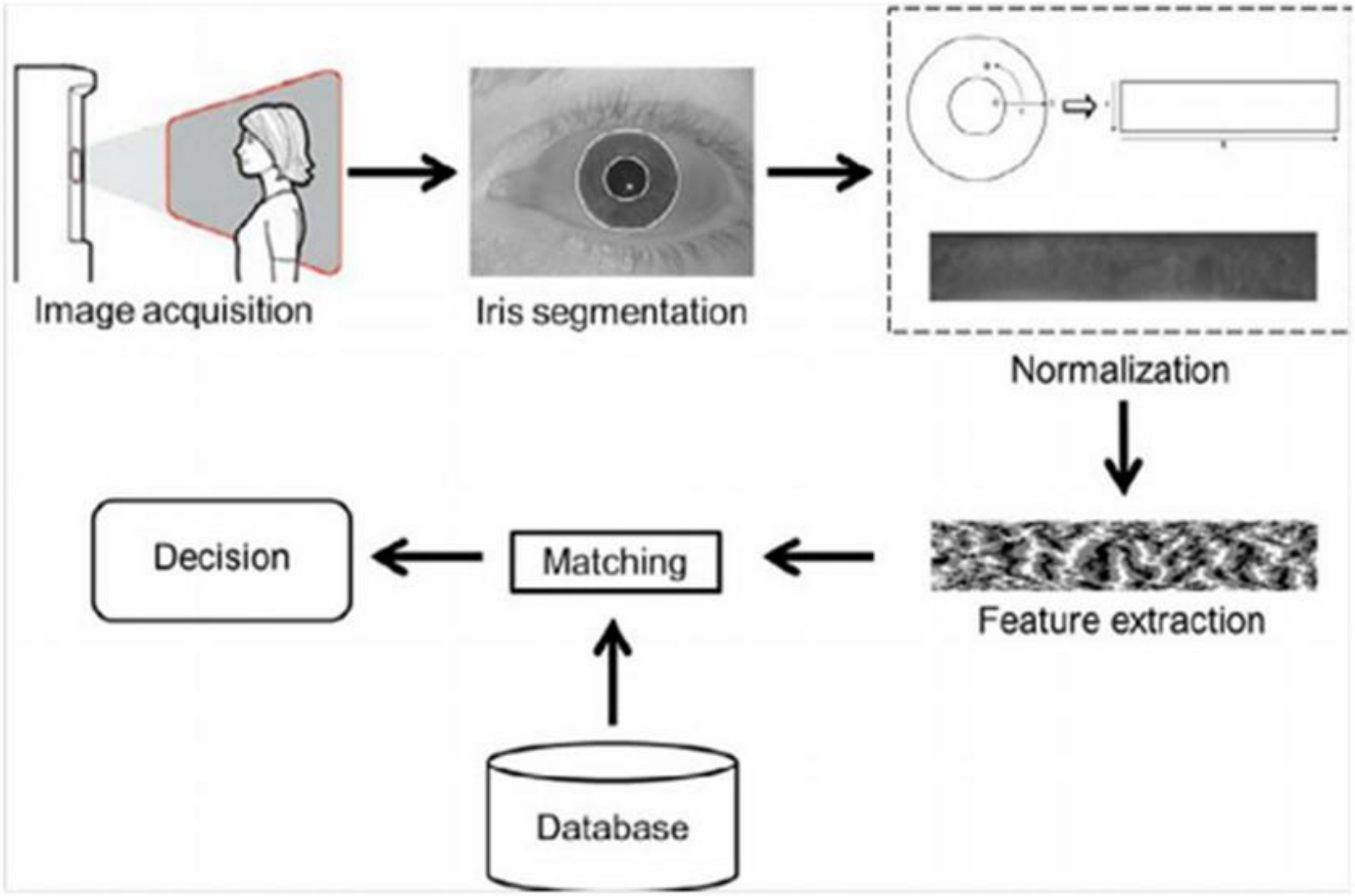
Le diagramme d'activité est un diagramme comportemental d'UML, permettant de représenter le déclenchement d'événements en fonction des états du système et de modéliser des comportements parallélisables (multi-threads ou multi-processus).
Le diagramme d'activité est également utilisé pour décrire un flux de travail (workflow).
Le diagramme d'activité est une représentation proche de l'organigramme ; la description d'un cas d'utilisation par un diagramme d'activité correspond à sa traduction algorithmique. Une activité est l'exécution d'une partie du cas d'utilisation, elle est représentée par un rectangle aux bords arrondis.
Le diagramme d'activité est sémantiquement proche des diagrammes de

communication (appelés diagramme de collaboration en UML 1), ou d'état-transitions, ces derniers offrant une vision microscopique des objets du système.

Le diagramme d'activité présente une vision macroscopique et temporelle du système modélisé :
Le diagramme d'activité de notre projet est comme suit :



Phases du projet



CHAPITRE 3:
ETUDE TECHNIQUE ET ARCHITECTURE

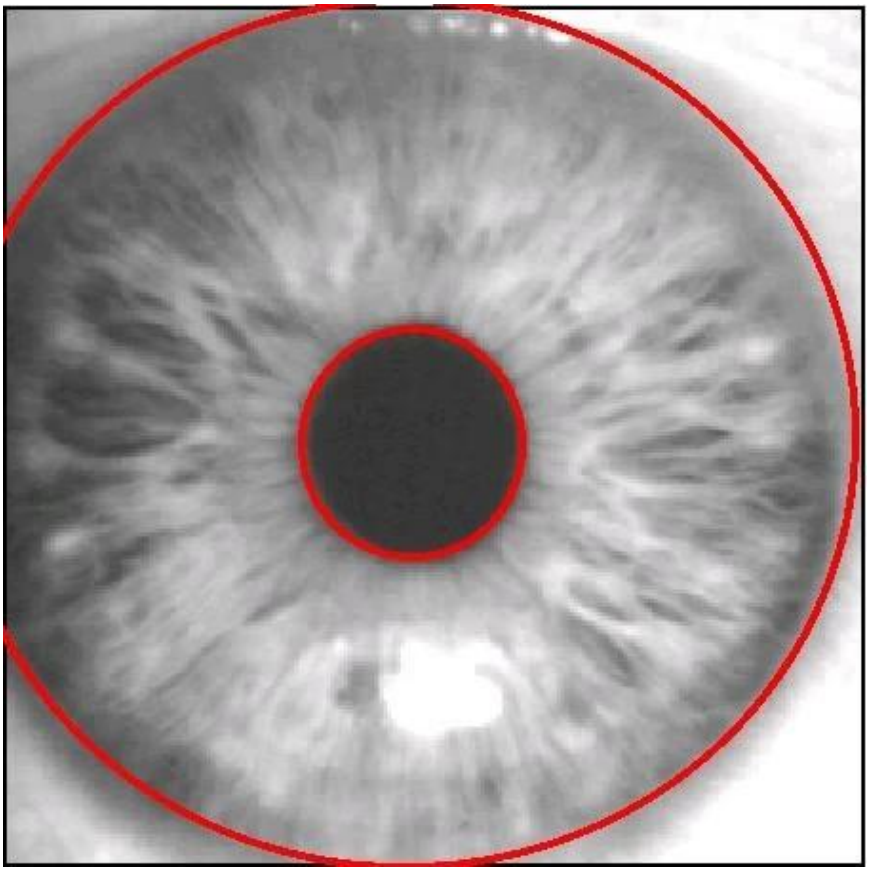
Iris biométrique

L'iris est la membrane colorée de l'œil. Elle est constituée d'un réseau de tubes fins dont le diamètre est inférieur à celui d'un cheveu. La forme de l'iris, c'est-à-dire l'enchevêtrement des tubes ne varie que très peu durant la vie de l'individu.

Par contre, la couleur (des tubes) varie un peu avec le temps (5 à 10 ans) et pour certaines maladies.

L'iris n'est pas lié à l'ADN : Les 2 iris d'un individu ont a peu près la même couleur mais leur forme (enchevêtrement des tubes) est aussi différente que celle de l'iris d'une autre personne. L'iris contient une quantité d'information particulièrement importante, que certains n'hésitent pas à comparer à la quantité d'information contenue dans l'ADN.

L'iris n'occupe qu'une surface très faible. L'observation pratique à travers un système optique ne permet de déceler que des contours macroscopiques et pas de descendre au niveau des tubes élémentaires. Toutefois, ceci évolue avec la précision des capteurs. Mais il n'est pas non plus indispensable de recueillir toute l'information contenue dans l'iris pour authentifier un individu. En effet, les iris sont suffisamment variés pour qu'une approximation de l'information totale suffise à certifier l'identité d'un individu.



Capture d'images

Il n'est pas très simple de capturer l'image de l'iris

Cet organe est très sensible

Sa taille est très variable suivant la lumière ambiante ou l'état de fatigue

Il est très petit, souvent obscurci par les cils, les paupières, les lentilles ou des réflexions de la lumière.

Les utilisateurs ont tendance à bouger.

Le système de prise d'image doit être rapide, précis et ne doit pas utiliser une lumière qui se refléterait sur l'œil (tâche de lumière sur l'image).

Le système de numérisation doit permettre d'obtenir au moins 70 pixels de rayon sur l'iris, et dans la plupart des cas, le rayon est compris entre 100 et 140 pixels. Une caméra CCD monochrome (640 x 480) est employée avec une source de lumière de longueur d'onde comprise entre 700 et 900 nm ; invisible pour les humains. D'autres systèmes emploient une caméra à large vision qui permet la localisation des yeux sur le visage, puis une autre caméra avec une vision étroite prend des images des yeux avec une plus grande résolution.

NB:

C'est pour cette raison les photos utilisées doivent être de bonne qualite sous forme .bmp

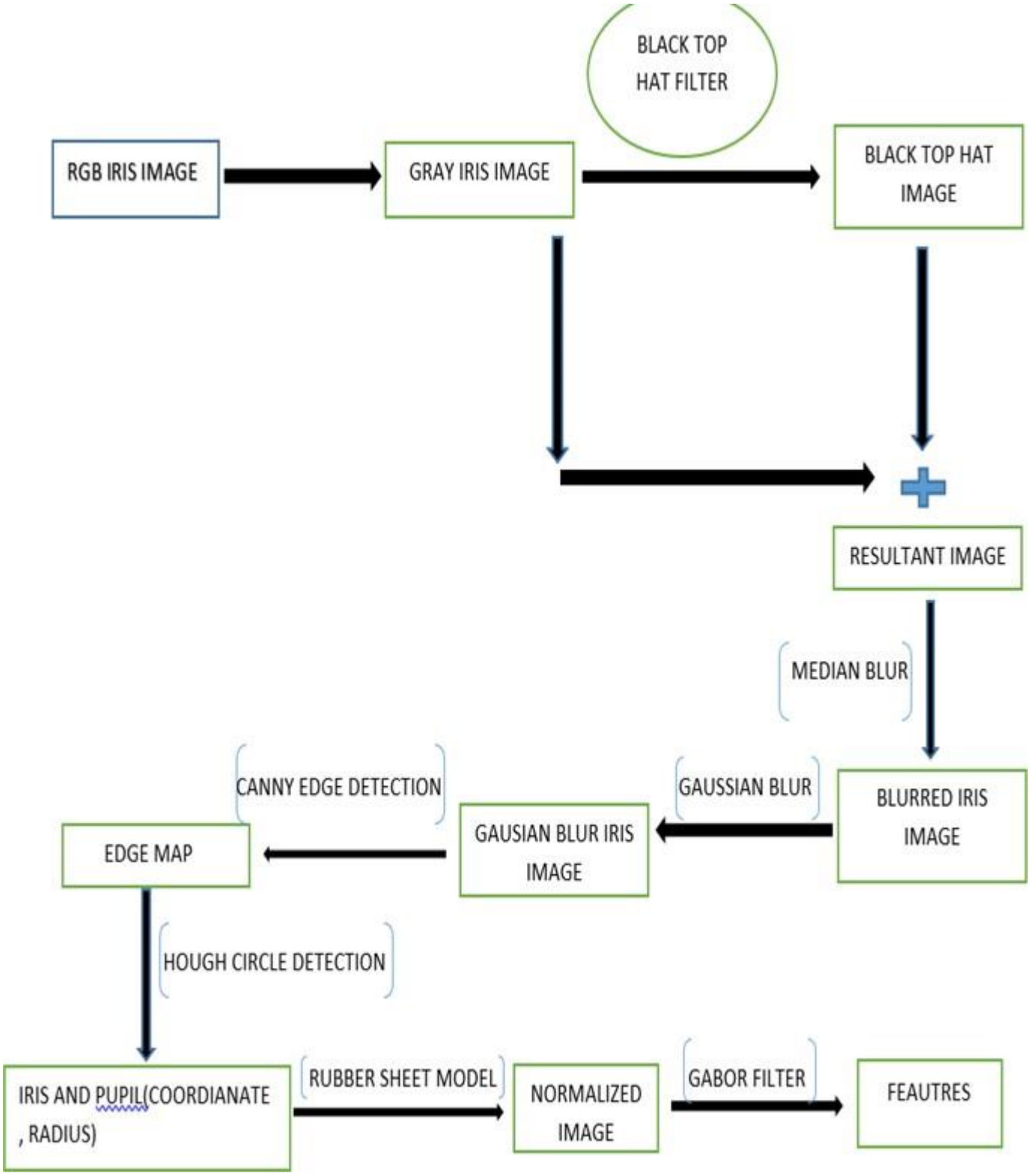
Technologie biométrique et iris

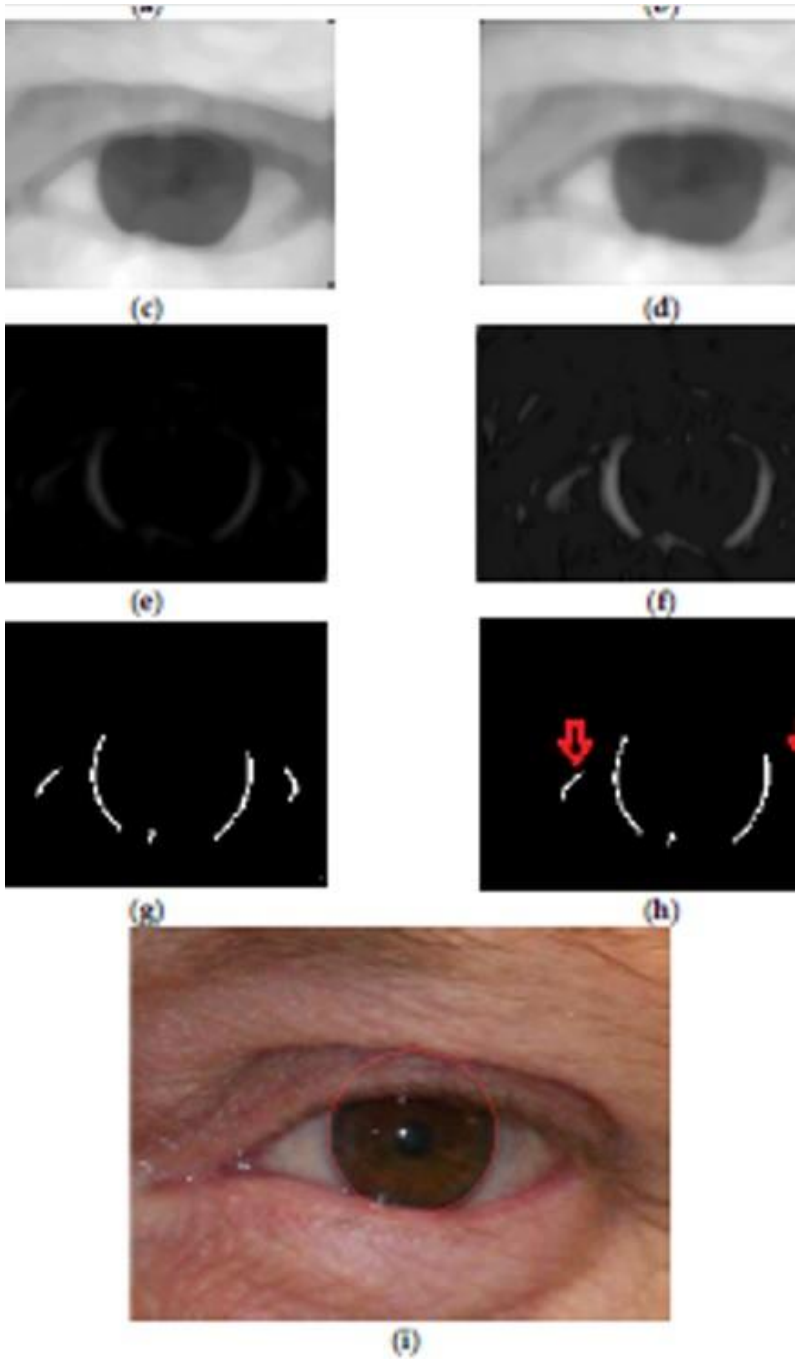
Au cours des dernières années, l'utilisation de l'identification biométrique s'est répandue : de nombreuses entités gouvernementales et privées intègrent désormais des systèmes biométriques dans leurs infrastructures. La biométrie est fréquemment utilisée pour déverrouiller des applications et des appareils, mais aussi pour vérifier l'identité d'une personne qui demande un passeport, ouvre un compte bancaire, embarque dans un avion ou participe à un vote. La liste est sans fin.

La reconnaissance de l'iris utilisée dans le monde entier

Les trois technologies biométriques les plus populaires portent actuellement sur le visage, les empreintes digitales et l'iris. La reconnaissance de l'iris est unanimenment reconnue comme l'une des méthodes d'identification biométrique les plus sûres et les plus précises. Contrairement aux mains et au visage, l'iris est un organe interne, protégé, et donc moins susceptible d'être altéré. Les données biométriques de l'iris de plus d'un milliard de personnes ont ainsi été collectées dans le cadre du programme d'identité Aadhaar en Inde. La vérification de l'identité via l'iris est également utilisée dans les aéroports et les ports des Émirats arabes unis.

Détail du modèle de système




(c)
(d)
(e)
(f)
(g)
(h)
(i)

Hardware & software utilisé

1.1 Hardware Requirements:

1. Intel Core i5
2. 1.9 GHZ processor
3. 8 GB RAM
4. 4 GB of free hard disk space

1.2 Keyboard, mouse

2. Software Requirements:

1. Operating system – windows10
2. Python (OpenCV)
3. Jupyter Notebook/ Anaconda

langages utilises

OpenCV fournit un ensemble de plus de 2500 algorithmes de vision par ordinateur, accessibles au travers d'API. Ce qui permet d'effectuer tout un tas de traitements sur des images (extraction de couleurs, détection de visages, de formes, application de filtres,...). Ces algorithmes se basent principalement sur des calculs mathématiques complexes, concernant surtout les traitements sur les matrices (car une image peut être considérée comme une matrice de pixels).

python

Python est un langage de programmation interprété, interactif, orienté objet et de haut niveau à usage général. Il a été créé par Guido van Rossum entre 1985 et 1990. Comme Perl, le code source Python est également disponible sous la licence publique générale GNU (GPL). Ce tutoriel donne une compréhension suffisante du langage de programmation Python.

NumPy

NumPy est la bibliothèque Python fondamentale pour l'informatique numérique. Son type le plus important est un type de tableau appelé . NumPy offre de nombreuses routines de création de tableaux pour différentes circonstances. est l'une de ces fonctions basées sur des plages numériques. Il est souvent appelé parce que c'est une abréviation largement utilisée pour NumPy.ndarrayarange()np.arange()np

La création de tableaux NumPy est importante lorsque vous travaillez avec d'autres bibliothèques Python qui s'appuient sur eux, comme SciPy, Pandas, Matplotlib, scikit-learn, etc. NumPy convient à la création et à l'utilisation de tableaux car il offre des routines utiles, permet d'améliorer les performances et d'écrire du code concis.

Scipy

Scipy est une bibliothèque scientifique de Python construite sur la base de Numpy et facile à utiliser. Elle contient plusieurs fonctions mathématiques et scientifiques très performantes.

Segmentation

La première étape de la reconnaissance de l'iris consiste à isoler la région réelle de l'iris dans une image numérique de l'œil. La région de l'iris, illustrée à la figure 1.1, peut être approchée par deux cercles, un pour la limite iris/sclère et un autre, intérieur au premier, pour la limite iris/pupille. Les paupières et les cils obstruent normalement les parties supérieure et inférieure de la région de l'iris. De plus, des réflexions spéculaires peuvent se produire dans la région de l'iris, corrompant le motif de l'iris. Une technique est nécessaire pour isoler et exclure ces artefacts ainsi que pour localiser la région circulaire de l'iris.

Le succès de la segmentation dépend de la qualité d'imagerie des images oculaires. Les images de la base de données CASIA iris [13] ne contiennent pas de réflexions spéculaires en raison de l'utilisation de la lumière infrarouge proche pour l'éclairage. Cependant, les images de la base de données LEI [14] contiennent ces réflexions spéculaires, qui sont causées par l'imagerie sous lumière naturelle. De plus, les personnes avec des iris foncés présenteront un très faible contraste entre la pupille et la région de l'iris si elles sont imagées sous la lumière naturelle, ce qui rend la segmentation plus difficile. L'étape de segmentation est essentielle au succès d'un système de reconnaissance de l'iris, car les données faussement représentées comme des données de modèle d'iris corrompront les modèles biométriques générés, entraînant de faibles taux de reconnaissance.

Réduction dimensionnelle

La méthode de luminosité fait la moyenne des couleurs les plus proéminentes et les moins proéminentes : (max(R, G, B) + min(R, G, B)) / 2. La méthode moyenne fait simplement la moyenne des valeurs : (R + G + B) / 3. La méthode de la luminosité est une version plus sophistiquée de la méthode de la moyenne. Il fait également la moyenne des valeurs, mais il forme une moyenne pondérée pour tenir compte de la perception humaine. Nous sommes plus sensibles au vert que les autres couleurs, donc le vert est le plus pondéré. La formule de luminosité est 0,21R +

0,72 G + 0,07 B.

Top-Hat Transformation

En morphologie mathématique et en traitement d'image numérique, la transformation en haut de forme est une opération qui extrait de petits éléments et détails d'images données. Il existe deux types de transformation chapeau haut de forme : La transformée haut de forme blanche est définie comme la différence entre l'image d'entrée et son ouverture par un élément structurant ; La transformation en haut-de-forme noir est définie comme la différence entre l'image de fermeture et l'image d'entrée. Les transformations haut de forme sont utilisées pour diverses tâches de traitement d'image, telles que l'extraction de caractéristiques, l'égalisation de l'arrière-plan, l'amélioration de l'image, etc.

Le haut-de-forme noir renvoie une image, contenant les "objets" ou "éléments" qui :

• Sont « plus petits » que l'élément structurant, et

• sont plus sombres que leur environnement.

La taille, ou la largeur, des éléments qui sont extraits par les transformations haut de forme peut être contrôlée par le choix de l'élément structurant . Plus ce dernier est gros, plus les éléments extraits sont gros.

Les deux transformations haut de forme sont des images qui ne contiennent que des valeurs non négatives à tous les pixels. L'une de ses utilisations les plus importantes dans la segmentation d'image consiste à ajuster les conditions d'éclairage non uniformes sur une image et à fournir une meilleure valeur de seuil pour séparer les objets.

Soit f : E <->R une image en niveaux de gris, mappant des points d'un espace euclidien ou d'une grille discrète E (telle que R2 ou Z2) en une ligne réelle. Soit b(x) un élément structurant en niveaux de gris.

La transformation chapeau haut noir de f (parfois appelée transformation chapeau bas) est donnée par

Tb(f) =f - f * b

Où * est l'opération de fermeture.

Gaussian Smoothing filter

Dans le traitement d'image, un flou gaussien (également appelé lissage gaussien) est le résultat du floutage d'une image par une fonction gaussienne (du nom du mathématicien et scientifique Carl Friedrich Gauss). C'est un effet largement utilisé dans les logiciels graphiques, généralement pour réduire le bruit de l'image et réduire les détails. L'effet visuel de cette technique de flou est un flou doux ressemblant à celui de la visualisation de l'image à travers un écran translucide, nettement différent de l'effet bokeh produit par un objectif flou ou l'ombre d'un objet sous un éclairage habituel. Le lissage gaussien est également utilisé comme étape de prétraitement dans les algorithmes de vision par ordinateur afin d'améliorer les structures d'image à différentes échelles - voir représentation de l'espace d'échelle et implémentation de l'espace d'échelle .
Mathématiquement, appliquer un flou gaussien à une image revient à convoluer l'image avec une fonction gaussienne. Ceci est également connu sous le nom de transformée de Weierstrass bidimensionnelle. En revanche, la convolution par un cercle (c'est-à-dire un flou de boîte circulaire) reproduirait plus précisément l'effet bokeh. La transformée de Fourier d'une gaussienne étant une autre gaussienne, l'application d'un flou gaussien a pour effet de réduire les composantes haute fréquence de l'image ; un flou gaussien est donc un filtre passe-bas.

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

où x est la distance à l'origine sur l'axe horizontal, y est la distance à l'origine sur l'axe vertical et σ est l'écart type de la distribution gaussienne. Lorsqu'elle est appliquée en deux dimensions, cette formule produit une surface dont les contours sont des cercles concentriques avec une distribution gaussienne à partir du point central. Les valeurs de cette distribution sont utilisées pour construire une matrice de convolution qui est appliquée à l'image d'origine. Ce processus de convolution est illustré visuellement dans la figure de droite. La nouvelle valeur de chaque pixel est définie sur une moyenne pondérée du voisinage de ce pixel. La valeur du pixel d'origine reçoit le poids le plus lourd (ayant la valeur gaussienne la plus élevée) et les pixels voisins reçoivent des poids plus petits à mesure que leur distance au pixel d'origine augmente. Il en résulte un flou qui préserve mieux les limites et les bords que d'autres filtres de flou plus uniformes

Canny Edge Detection

Canny Edge Detection est un algorithme de détection de contours populaire. Il a été développé par John F. Canny en 1986. Il s'agit d'un algorithme à plusieurs étapes et nous passerons par chaque étape. Étant donné que la détection des contours est sensible au bruit dans l'image, la première étape consiste à supprimer le bruit dans l'image avec un filtre gaussien 5x5.

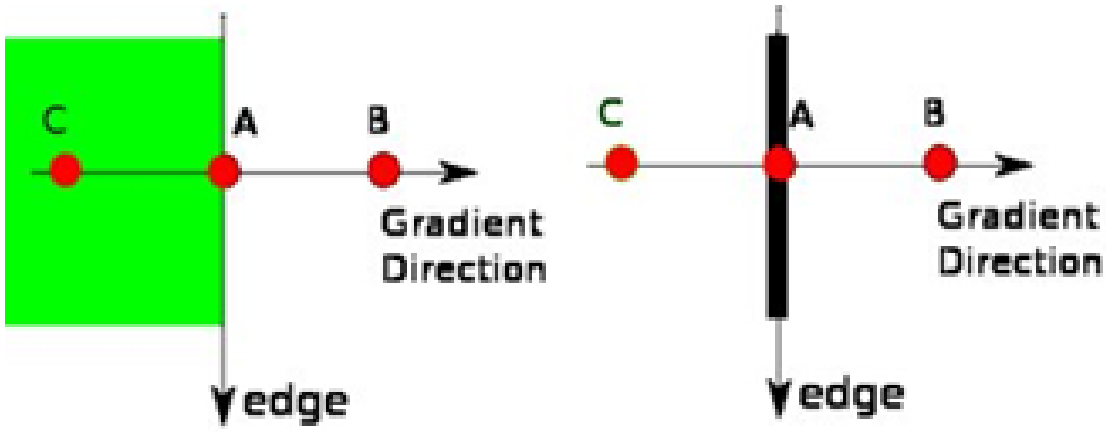
L'image lissée est ensuite filtrée avec un noyau de Sobel dans les directions horizontale et verticale pour obtenir la première dérivée dans la direction horizontale ( ) et la direction verticale ( ). À partir de ces deux images, nous pouvons trouver le gradient et la direction des bords pour chaque pixel comme suit :

$$Edge\_Gradient\ (G) = \sqrt{G_x^2 + G_y^2}$$

$$Angle\ (\theta) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

La direction du dégradé est toujours perpendiculaire aux bords. Il est arrondi à l'un des quatre angles représentant les directions verticale, horizontale et deux diagonales.

Après avoir obtenu l'amplitude et la direction du gradient, un balayage complet de l'image est effectué pour supprimer tous les pixels indésirables qui peuvent ne pas constituer le bord. Pour cela, à chaque pixel, pixel est vérifié s'il s'agit d'un maximum local dans son voisinage dans la direction du gradient. Vérifiez l'image ci-dessous :



Le point A est sur le bord (dans le sens vertical). La direction du dégradé est normale au bord. Les points B et C sont dans des directions de gradient. Ainsi, le point A est vérifié avec les points B et C pour voir s'il forme un maximum local. Si c'est le cas, il est considéré pour l'étape suivante, sinon, il est supprimé (mis à zéro). En bref, le résultat que vous obtenez est une image binaire avec des "bords fins".

Circular Hough Transform

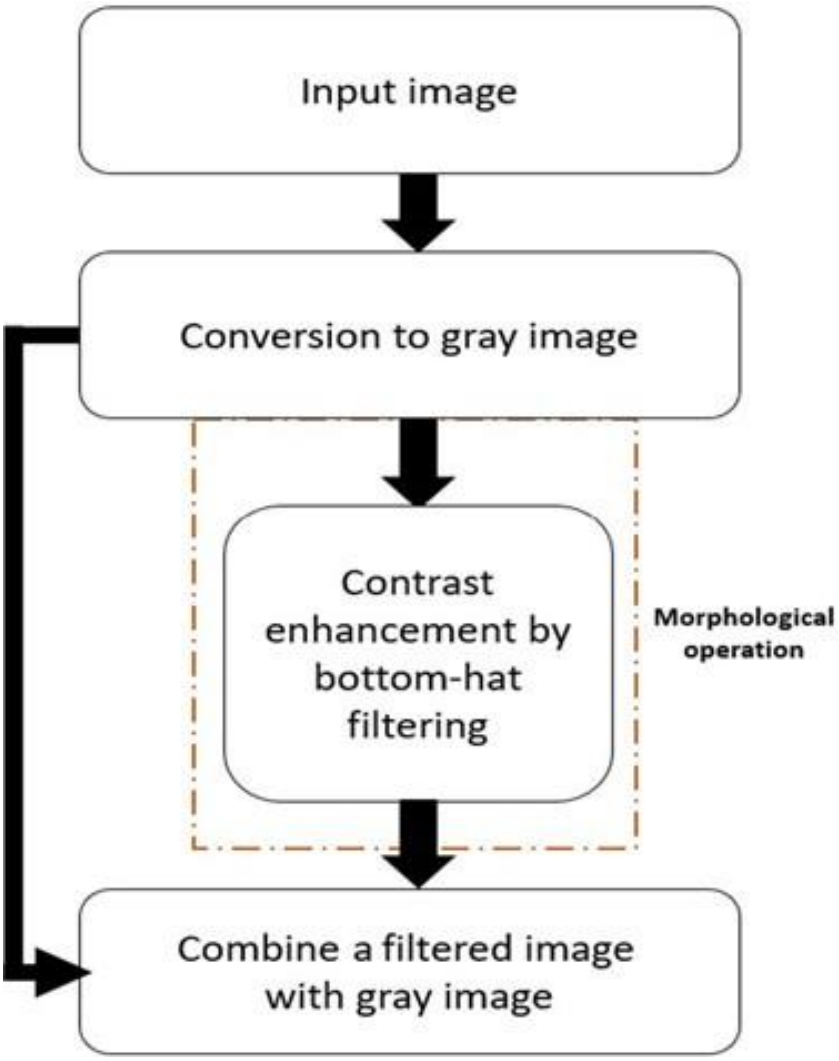
La transformée de Hough circulaire (CHT) est une technique de base utilisée dans le traitement d'images numériques, pour détecter des objets circulaires dans une image numérique.
La transformée de Hough circulaire (CHT) est une technique d'extraction de caractéristiques pour détecter les cercles. C'est une spécialisation de Hough Transform. Le but de la technique est de trouver des cercles dans des entrées d'image imparfaites. Les cercles candidats sont produits en « votant » dans l'espace des paramètres de Hough, puis en sélectionnant les maxima locaux dans une matrice dite d'accumulateur.
Dans un espace à deux dimensions, un cercle peut être décrit par :
$(x-a)2+(y-b)2 = r2$
où (a,b) est le centre du cercle et r est le rayon. Si un point 2D (x,y) est fixe, alors les paramètres peuvent être trouvés selon (1). L'espace des paramètres serait tridimensionnel, (a, b, r). Et tous les paramètres qui satisfont (x, y) se trouveraient à la surface d'un cône à angle droit inversé dont le sommet est en (x, y, 0). Dans l'espace 3D, les paramètres du cercle peuvent être identifiés par l'intersection de nombreuses surfaces coniques définies par des points sur le cercle 2D. Ce processus peut être divisé en deux étapes. La première étape consiste à fixer le rayon puis à trouver le centre optimal des cercles dans un espace de paramètres 2D. La deuxième étape consiste à trouver le rayon optimal dans un espace paramétrique à une dimension.
Si le rayon est fixe, alors l'espace des paramètres serait réduit à 2D (la position du centre du cercle). Pour chaque point (x, y) du cercle d'origine, il peut définir un cercle centré en (x, y) de rayon R selon (1). Le point d'intersection de tous ces cercles dans l'espace des paramètres correspondrait au point central du cercle d'origine. Considérez 4 points sur un cercle dans l'image originale (à gauche). La transformée de Hough du cercle est illustrée à droite. Notez que le rayon est supposé connu. Pour chacun (x, y) des quatre points (points blancs) de l'image d'origine, il peut définir un cercle dans l'espace des paramètres de Hough centré en (x, y) avec un rayon r. Une matrice d'accumulation est utilisée pour suivre le point d'intersection. Dans l'espace des paramètres, le nombre de points de vote par lesquels passe le cercle serait augmenté de un. Ensuite, le point maximal local

(le point rouge au centre sur la figure de droite) peut être trouvé. La position (a, b) des maxima serait le centre du cercle d'origine.

Implementation

Phase 1

l'image d'entrée RVB est convertie en niveaux de gris pour un traitement ultérieur, et l'opération morphologique est appliquée par filtrage à chapeau inférieur avec un disque d'éléments structurants symétriques de taille 5 pour l'amélioration du contraste. Enfin, deux images de l'image grise et de l'image résultante par filtrage bottom-hat sont ajoutées pour obtenir une image rehaussée

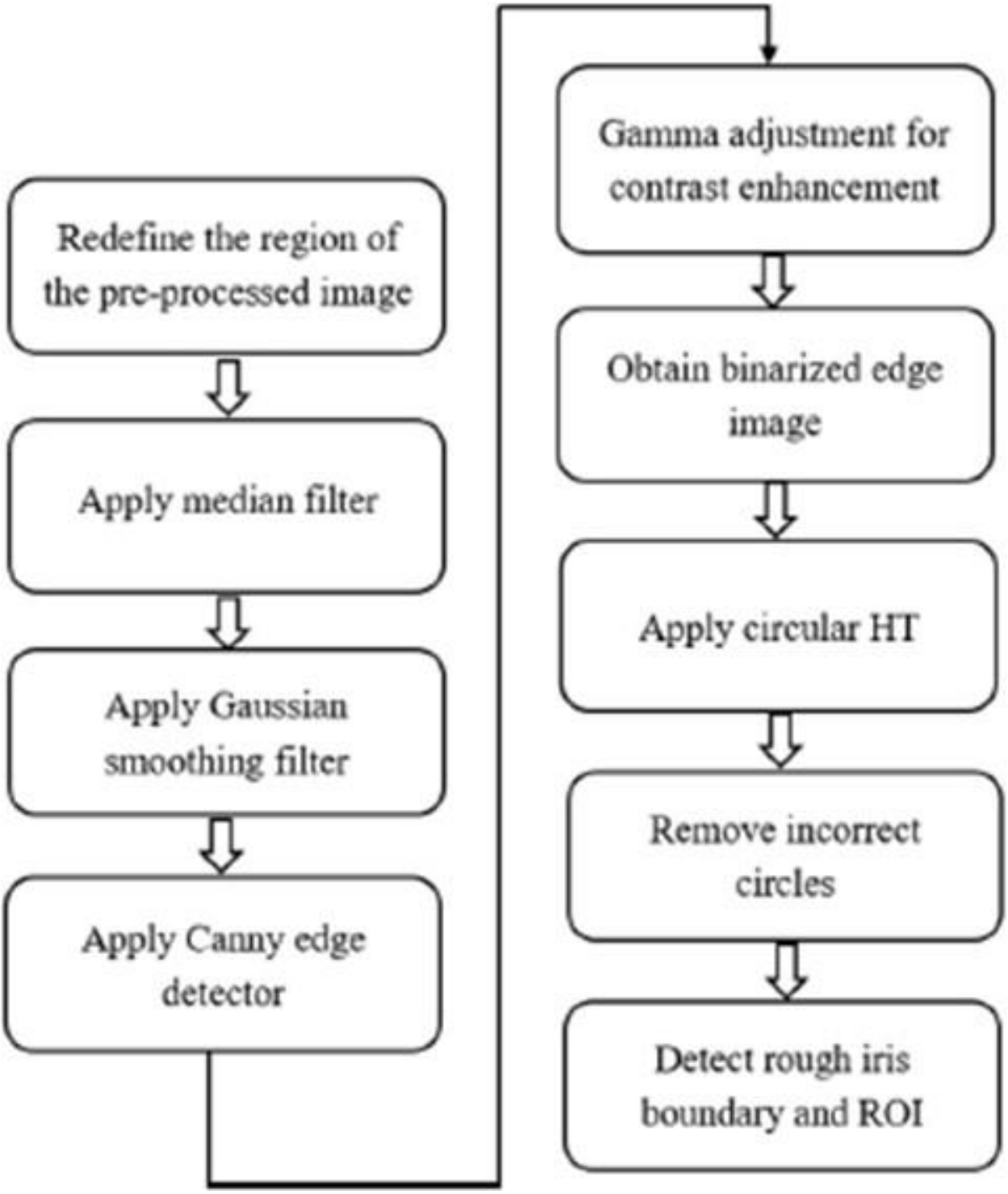


Phase 2

Le processus global de la phase 2 est présenté par l'organigramme de la figure 4. Dans la phase 2, l'image filtrée de la phase 1 est redéfinie comme une image de 280 _ 220 pixels pour réduire l'effet des sourcils dans la détection de la limite de l'iris. Ensuite, un filtre médian 17 _ 17 est appliqué pour lisser l'image des bruits de sel et de poivre et réduire l'éclairage du teint et de la texture de la peau. 13 _ 13 filtre de lissage gaussien symétrique avec _ de 2 est appliqué à l'image filtrée pour augmenter l'uniformité des pixels, comme illustré sur la figure 5d. Ensuite, le détecteur de bord Canny avec la même valeur _ est utilisé pour détecter les bords de la limite de l'iris, comme illustré à la figure 5e. Cependant, les bords ne sont pas très clairs et un ajustement gamma avec g = 1,90 est appliqué pour améliorer le contraste de l'image . Avec l'image gamma améliorée, l'image de bord binarisée est obtenue avec huit connectivités voisines. Dans cette image de bord, il y a plus de bords circulaires le long des bords de la limite de l'iris, et le HT circulaire peut trouver tous les cercles possibles dans l'image. Cependant, les bords incorrects de type cercle peuvent être supprimés en filtrant les bords dont le rayon est hors de la plage des rayons minimum et maximum de l'iris humain. Ensuite, les bords les plus connectés sont sélectionnés comme bords de l'iris, et la limite rugueuse de l'iris est détectée. Compte tenu de la possibilité d'erreur de détection de la limite de l'iris, le retour sur investissement est défini légèrement supérieur à la limite détectée. Pour des comparaisons équitables, tous les paramètres optimaux pour le fonctionnement en détection de retour sur investissement, y compris

le filtre médian, le filtre de lissage gaussien, le détecteur de bord Canny, l'ajustement gamma et la binarisation, etc., ont été sélectionnés empiriquement uniquement par des données d'apprentissage sans

données de test.


Redefine the region of the pre-processed image
Apply median filter
Apply Gaussian smoothing filter
Apply Canny edge detector
Gamma adjustment for contrast enhancement
Obtain binarized edge image
Apply circular HT
Remove incorrect circles
Detect rough iris boundary and ROI

Code réalisant la segmentation

segmentation.py

```python
import matplotlib.pyplot as plt
import numpy as np
import cv2
import os
from skimage.morphology import erosion, dilation, opening, closing, white_tophat
from skimage.morphology import black_tophat, skeletonize, convex_hull_image
from skimage.morphology import disk
import scipy
from scipy import ndimage
def rgb2gray(rgb):
    return np.dot(rgb[...,:3], [0.299, 0.587, 0.114])
plt.figure(figsize=(15,15))
def center(gaussian_img):
    edges = cv2.Canny(gaussian_img,50,100)
    cimg = cv2.cvtColor(edges,cv2.COLOR_GRAY2BGR)
    circles = cv2.HoughCircles(edges,cv2.HOUGH_GRADIENT,1,20,
                               param1=50,param2=30,minRadius=0,maxRadius=0)
    circles = np.uint16(np.around(circles))
    center=circles[0,:]
    x=int(center[0,0])
    y=int(center[0,1])
    r=int(center[0,2])
    return (x,y,r)
```

segmentation.py > center

```python
def radius(gaussian_img):
    edges = cv2.Canny(gaussian_img,20,20)
    cimg = cv2.cvtColor(edges,cv2.COLOR_GRAY2BGR)
    circles = cv2.HoughCircles(edges,cv2.HOUGH_GRADIENT,1,20,
                               param1=50,param2=30,minRadius=0,maxRadius=0)
    circles = np.uint16(np.around(circles))
    circles=circles[0,:]
    circles = np.uint16(np.around(circles))
    radiuses=circles[:,2]
    x=[]
    for i in range(len(radiuses)):
        if(radiuses[i]<115 and radiuses[i]>100):
            x.append(radiuses[i])
    return int(np.mean(x))

path=os.getcwd()
path=os.path.join(path,r"IITD Database")

for i in range(100,125):
    img_path=os.path.join(path,str(i))
    img_path=os.path.join(img_path,"01_L.bmp")
    print(img_path)
    image=np.array(cv2.imread(img_path))
    gray_img=np.uint8(rgb2gray(image))
    selem = disk(6)
    b_tophat = black_tophat(gray_img, selem)
    resultant_img=b_tophat+gray_img
    median_img = cv2.medianBlur(resultant_img,5)
    gaussian_img=scipy.ndimage.filters.gaussian_filter(median_img,
                                                       sigma=1.90, order=0,output=None,
                                                       mode='reflect',
                                                       cval=0.0, truncate=4.0)
    x,y,r=center(gaussian_img)
    R=radius(gaussian_img)
    cv2.circle(image,(x,y),r,(255,0,0),2)
    cv2.circle(image,(x,y),int(R*1.01),(255,0,0),2)
    plt.grid(False)
    plt.subplot(5,5,i-99)
    plt.imshow(image,cmap ='gray')
plt.show()
```

Explication des méthodes prédéfinis utilises

-OpenCV-Python est une bibliothèque de liaisons Python conçue pour résoudre les problèmes de vision par ordinateur. La méthode cv2.cvtColor() est utilisée pour convertir une image d'un espace colorimétrique à un autre. Il existe plus de 150 méthodes de conversion d'espace colorimétrique disponibles dans OpenCV. Nous utiliserons certains des codes de conversion d'espace colorimétrique ci-dessous.

-OpenCV utilise donc une méthode plus délicate, la méthode Hough Gradient qui utilise les informations de gradient des bords. Nous utilisons la fonction : cv.HoughCircles (image, cercles, méthode, dp, minDist, param1 = 100, param2 = 100, minRadius = 0, maxRadius = 0) Paramètres

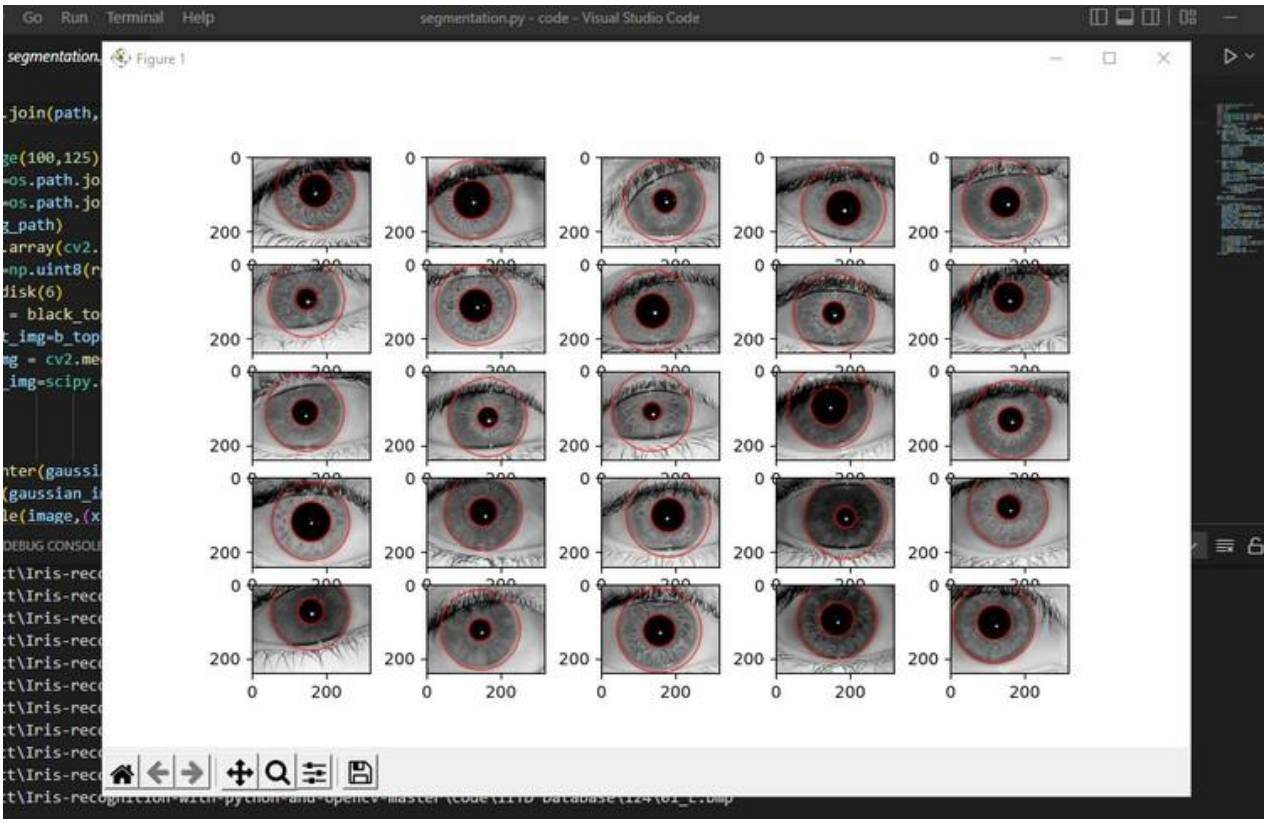
_Nous pouvons utiliser la fonction d'OpenCV pour détecter les cercles présents dans une image. La fonction utilise la transformée de Hough pour trouver les cercles présents dans une image en niveaux de gris.HoughCircles()HoughCircles()

La transformation de Hough extrait les entités d'une image puis, à l'aide d'une procédure de vote, détermine la forme des objets présents dans une image.
Le premier argument de la est l'image dans laquelle nous voulons détecter les cercles, et il devrait être en niveaux de gris. Le deuxième argument est la méthode utilisée pour la détection des cercles.HoughCircles()

-La méthode os.getcwd() en Python renvoie le répertoire de travail courant d'un processus. Chaque processus s'exécutant sous un système d'exploitation a un répertoire de travail associé, appelé répertoire de travail actuel du processus. Le

répertoire de travail courant d'un processus peut être modifié en appelant la méthode Python os.chdir().

Results(Localization and Segmentation)



Normalisation

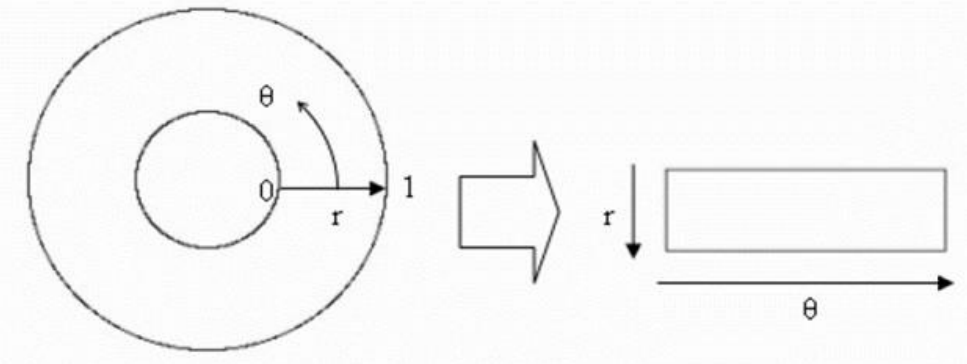
Aperçu

Une fois que la région de l'iris est segmentée avec succès à partir d'une image de l'œil, l'étape suivante consiste à transformer la région de l'iris afin qu'elle ait des dimensions fixes afin de permettre des comparaisons. Les incohérences dimensionnelles entre les images oculaires sont principalement dues à l'étirement de l'iris causé par la dilatation de la pupille à partir de différents niveaux d'éclairage. D'autres sources d'incohérence comprennent la variation de la distance d'imagerie, la rotation de la caméra, l'inclinaison de la tête et la rotation de l'œil dans l'orbite. Le processus de normalisation produira des régions d'iris, qui ont les mêmes dimensions constantes, de sorte que deux photographies du même iris dans des conditions différentes auront des caractéristiques caractéristiques au même emplacement spatial.

Un autre point à noter est que la région de la pupille n'est pas toujours concentrique dans la région de l'iris et est généralement légèrement nasale **[2]**. Ceci doit être pris en compte si vous essayez de normaliser la région de l'iris en forme de "beignet" pour avoir un rayon constant.

Modèle Daugman's Rubber Sheet

Le modèle de feuille de caoutchouc homogène conçu par Daugman **[1]** remappe chaque point de la région de l'iris sur une paire de coordonnées polaires (r, θ) où r est sur l'intervalle **[0,1]** et θ est l'angle **[0,2π]**.



Pour remapper les coordonnées cartésiennes en coordonnées polaires, les équations utilisées sont

$$I(x(r, \theta), y(r, \theta)) \rightarrow I(r, \theta)$$
$$x(r, \theta) = (1 - r)xp(\theta) + rxl(\theta)$$
$$y(r, \theta) = (1 - r)yp(\theta) + ryl(\theta)$$

Ici, I(x, y) est l'image de la région Iris et les coordonnées cartésiennes d'origine sont (x, y) et la
les coordonnées polaires correspondantes sont (r,θ). De plus, (xp,yp) et (xl
, yl
) sont la pupille et l'iris
coordonnées aux limites le long de la direction θ. La dilatation et la taille des pupilles sont également incohérentes
considération pour produire une région de l'iris normalisée. Cependant, cela ne correspond pas à la rotation
incohérences. Dans l'algorithme de reconnaissance de l'iris et d'extraction de caractéristiques de Daugman, la rotation est compensée en déplaçant le gabarit de l'iris dans la direction θ [1].

Implementation

Pour la normalisation des régions de l'iris, une technique basée sur le modèle de feuille de caoutchouc de Daugman a été utilisée. Le centre de la pupille a été considéré comme le point de référence et les vecteurs radiaux traversent la région de l'iris, comme le montre la figure 3.2. Un certain nombre de points de données sont sélectionnés le long de chaque ligne radiale et ceci est défini comme la résolution radiale.

Le nombre de lignes radiales faisant le tour de la région de l'iris est défini comme la résolution angulaire. Étant donné que la pupille peut être non concentrique à l'iris, une formule de remappage est nécessaire pour redimensionner les points en fonction de l'angle autour du cercle. Ceci est donné par:

$$r' = \sqrt{\alpha}\,\beta \pm \sqrt{\alpha\beta^2 - \alpha - r_I^{\,2}} \qquad (3.4)$$

with

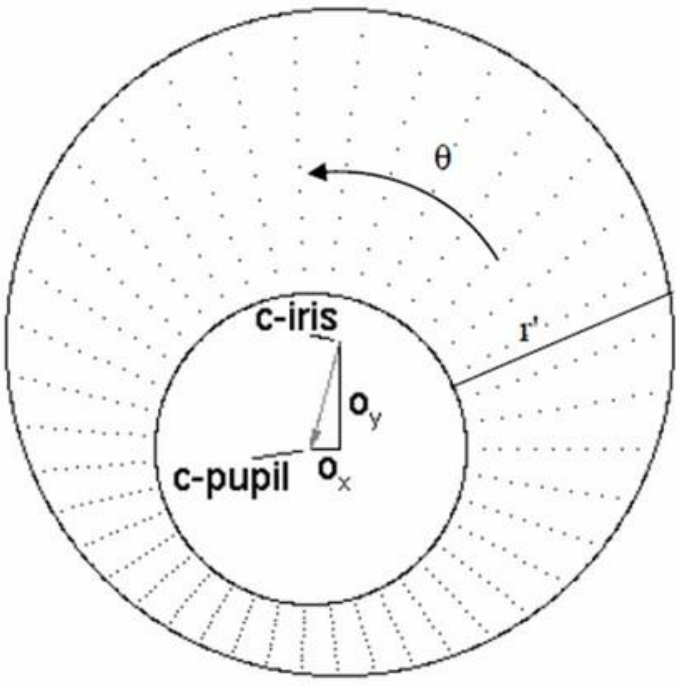
$$\alpha = o_x^2 + o_y^2$$

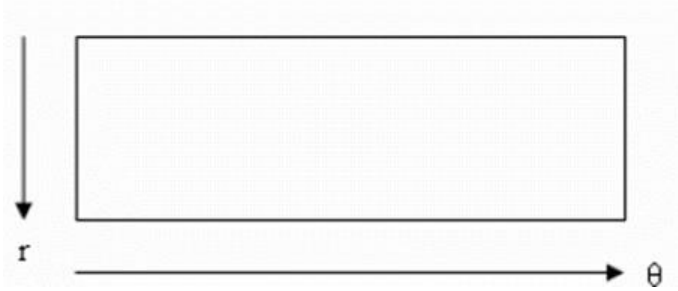
$$\beta = \cos\left(\pi - \arctan\left(\frac{o_y}{o_x}\right) - \theta\right)$$

où le déplacement du centre de la pupille par rapport au centre de l'iris est donné par ox , oy et r' est la distance entre le bord de la pupille et le bord de l'iris à un angle, θ autour de la région, et rI est le rayon de l'iris. La formule de remappage donne d'abord le rayon de la région de l'iris "beignet" en fonction de l'angle θ.

Un nombre constant de points est choisi le long de chaque ligne radiale, de sorte qu'un nombre constant de points de données radiaux sont pris, indépendamment de l'étroitesse ou de la largeur du rayon à un angle particulier. Le motif normalisé a été créé en revenant en arrière pour trouver les coordonnées cartésiennes des points de données à partir de la position radiale et angulaire dans le motif normalisé. À partir de la région de l'iris « beignet », la normalisation produit un tableau 2D avec des dimensions horizontales de résolution angulaire et des dimensions verticales de résolution radiale.

Un autre tableau 2D a été créé pour marquer les reflets, les cils et les paupières détectés lors de l'étape de segmentation. Afin d'empêcher que des données de région autre que l'iris ne corrompent la représentation normalisée, les points de données qui se produisent le long de la bordure de la pupille ou de la bordure de l'iris sont ignorés. Comme dans le modèle de feuille de caoutchouc de Daugman, la suppression des incohérences de rotation est effectuée à l'étape d'appariement et sera discutée dans le chapitre suivant.





Code réalisant la normalisation

iris_match.py

```python
import matplotlib.pyplot as plt
import numpy as np
import cv2,os,scipy
from skimage.morphology import black_tophat,disk
from scipy import ndimage as ndi
from skimage import data
from skimage.util import img_as_float
from skimage.filters import gabor_kernel

def rgb2gray(rgb):
    return np.dot(rgb[...,:3], [0.299, 0.587, 0.114])

def center(gaussian_img):
    edges = cv2.Canny(gaussian_img,50,100)
    cimg = cv2.cvtColor(edges,cv2.COLOR_GRAY2BGR)
    circles = cv2.HoughCircles(edges,cv2.HOUGH_GRADIENT,1,20,
                               param1=50,param2=30,minRadius=0,maxRadius=0)
    circles = np.uint16(np.around(circles))
    center=circles[0,:]
    y=int(center[0,0])
    x=int(center[0,1])
    r=int(center[0,2])
    return (x,y,r)

def radius(gaussian_img):
    edges = cv2.Canny(gaussian_img,20,20)
    cimg = cv2.cvtColor(edges,cv2.COLOR_GRAY2BGR)
    circles = cv2.HoughCircles(edges,cv2.HOUGH_GRADIENT,1,20,
                              param1=50,param2=30,minRadius=0,maxRadius=0)
    circles = np.uint16(np.around(circles))
    circles=circles[0,:]
    circles = np.uint16(np.around(circles))
    radiuses=circles[:,2]
    x=[]
    for i in range(len(radiuses)):
        if(radiuses[i]<115 and radiuses[i]>100):
            x.append(radiuses[i])
    return int(np.mean(x))
def norm_image(path):
    path = os.path.join(os.getcwd(),path)
    image=np.array(cv2.imread(path))
    temp_img=gray_img=np.uint8(rgb2gray(image))
    selem = disk(6)
    b_tophat = black_tophat(gray_img, selem)
    resultant_img=b_tophat+gray_img
    median_img = cv2.medianBlur(resultant_img,5)
    gaussian_img=scipy.ndimage.filters.gaussian_filter(median_img,
                                                       sigma=1.90, order=0,output=None,
                                                       mode='reflect',
                                                       cval=0.0, truncate=4.0)
    xc,yc,r=center(gaussian_img)
    R=radius(gaussian_img)
    theta = np.arange(0.00, np.pi*2, 0.01) #theta
    rng=np.arange(0,100)
    norm_img=np.zeros((rng.size,theta.size))
    for t in theta:
        for rc in rng:
            mc=(R-r)*(rc)/100+r
            x=int(xc+mc*np.cos(t))
            y=int(yc+mc*np.sin(t))
            try:
                norm_img[rc,np.where(theta==t)]=temp_img[x,y]
            except Exception as e:
                pass
    return norm_img
```

Explication des méthodes prédéfinis utilises

numpy.zeros()

La fonction numpy.zeros() renvoie un nouveau tableau de forme et de type donnés, avec des zéros. Syntaxe:

numpy.zeros(shape, dtype = None, order = 'C')

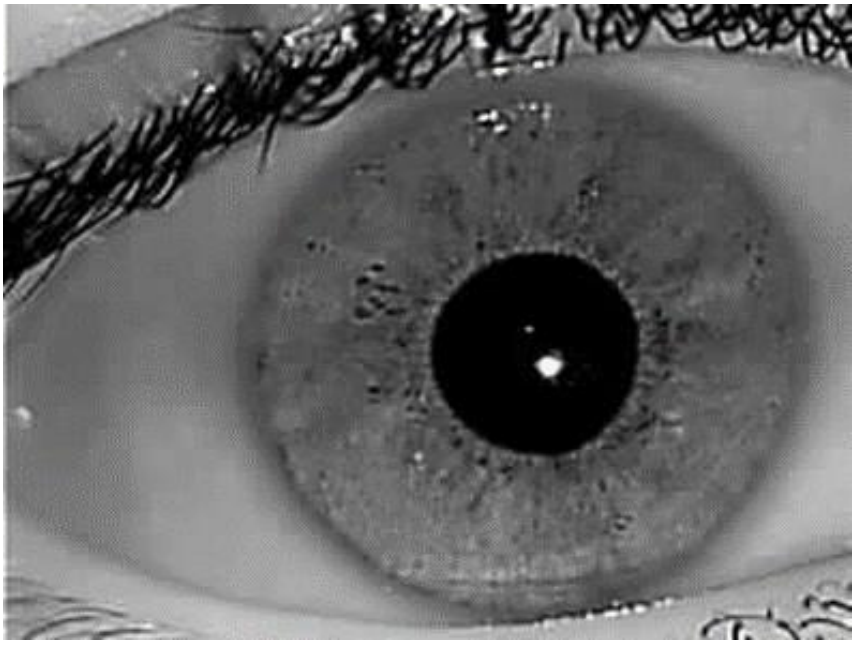
.arange()

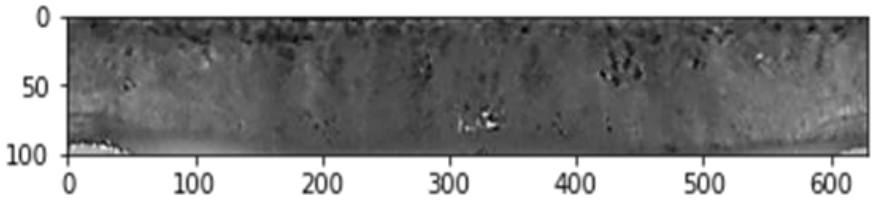
Fondamentalement, la méthode du module NumPy en Python est utilisée pour générer une séquence linéaire de nombres sur la base des points de départ et d'arrivée prédéfinis ainsi que d'une taille d'étape constante.arange()

scipy.ndimage.filters.gaussian_filter

Pour générer une image avec le filtre gaussien, nous avons utilise la fonction gaussian_filter() du module ndimage qui met en œuvre un filtre gaussien multidimensionnel. Cette fonction prend l'image originale en paramètres et génère une image filtrée. Le script suivant applique le filtre gaussien à notre image de test, affiche l'image filtrée et compare les histogrammes des deux images:

Implementation and Results (Normalization)





Matching

Après avoir enregistré avec succès les caractéristiques uniques de chaque image et les avoir stockées dans une structure de données appropriée, il est temps pour nous de récupérer des informations en recherchant un certain fichier image ou index stocké dans la structure de données (tableau multidimensionnel) et correspondant avec l'image en cours d'examen. C'est-à-dire que différents modèles contenant différentes informations sont stockés dans le tableau, puis nous faisons correspondre l'image sous observation avec tous les modèles stockés dans le tableau et vérifions le pourcentage de correspondance et nous annonçons l'image avec le pourcentage le plus élevé de rapport correspondant, comme l'image probable complémentaire de l'image observée actuellement.

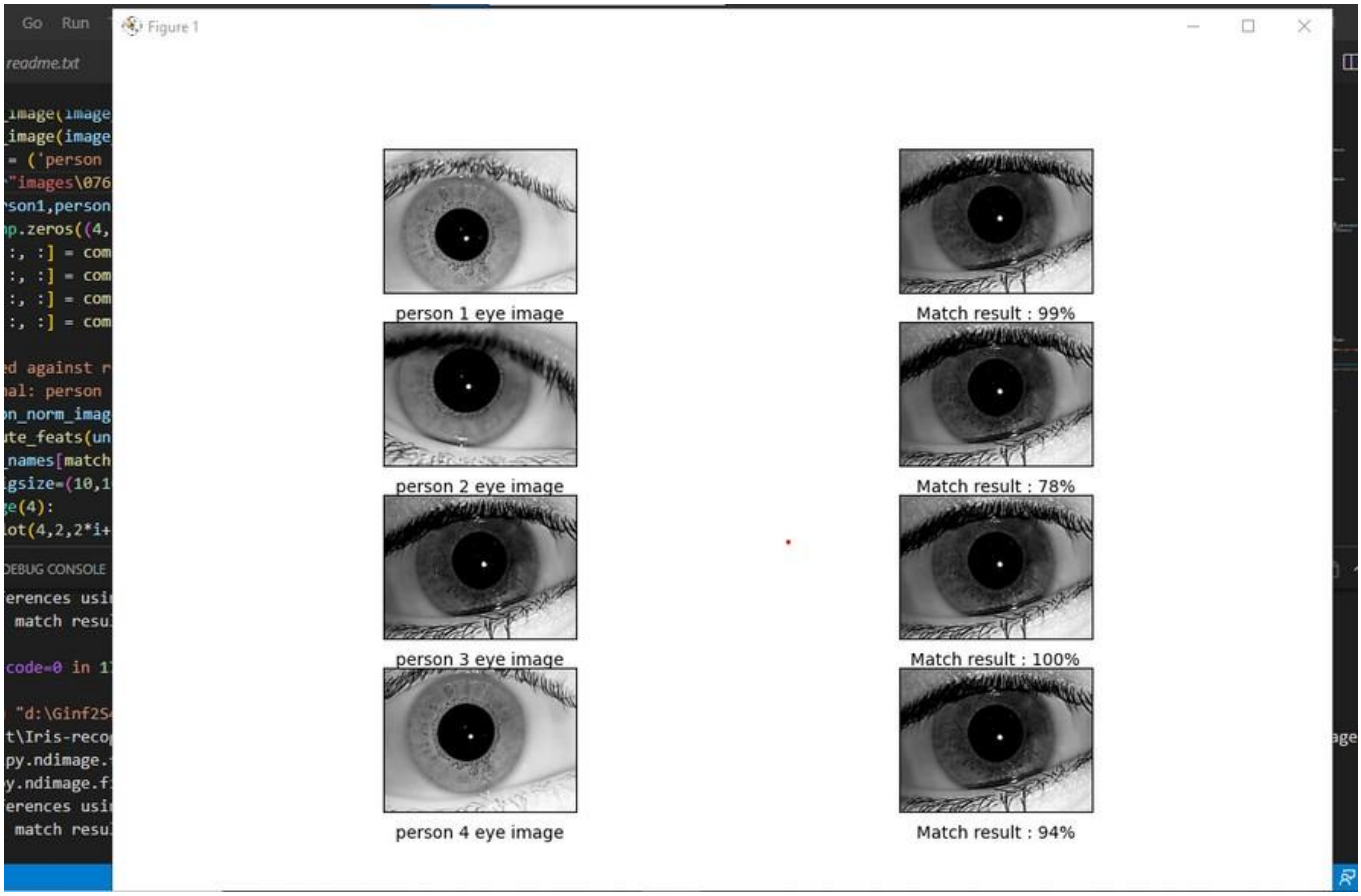
Maintenant, cette méthode s'avère efficace car nous comparons avec tous les modèles d'image disponibles et nous comparons les pourcentages des rapports de correspondance et nous pouvons donc être assez assurés que l'image est identifiée avec précision, car nous pouvons garantir que même l'œil droit et l'œil gauche les images d'une seule personne donneraient d'énormes différences dans le rapport des pourcentages correspondants, donc cela fonctionne presque comme la méthode de distance de Hamming.

Code réalisant le matching

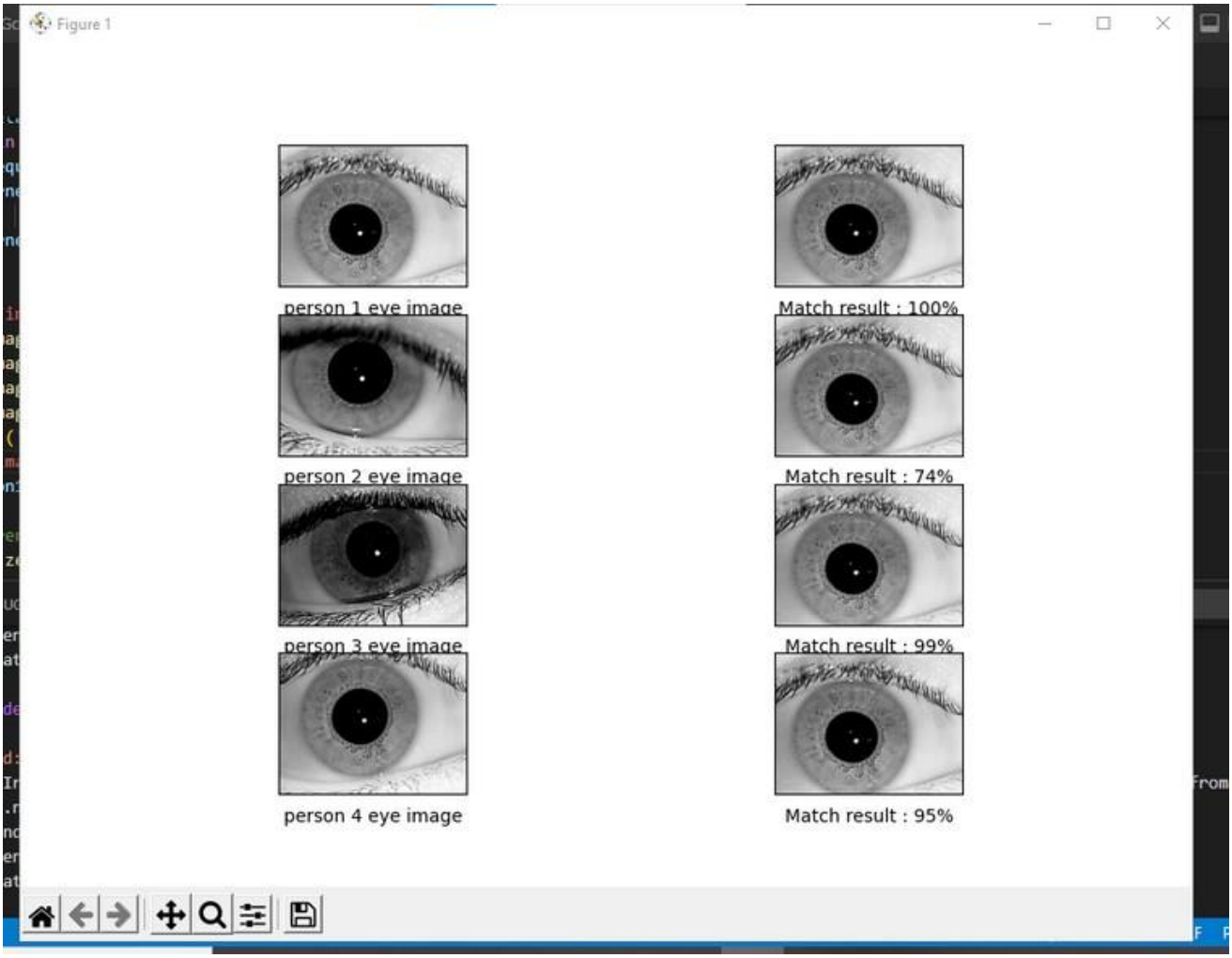
```
def compute_feats(image, kernels):
    feats = np.zeros((len(kernels), 2), dtype=np.double)
    for k, kernel in enumerate(kernels):
        filtered = ndi.convolve(image, kernel, mode='wrap')
        feats[k, 0] = filtered.mean()
        feats[k, 1] = filtered.var()
    return feats

percentage_array=[]
def match(feats, ref_feats):
    max_error=15000000
    min_error = np.inf
    min_i = None
    for i in range(ref_feats.shape[0]):
        error = np.sum((feats - ref_feats[i, :])**2)
        percentage=int((max_error-error)*100/max_error)
        percentage_array.append(percentage)
#         print(str(percentage)+" %")
        if error < min_error:
            min_error = error
            min_i = i
    return min_i
kernels = []
for theta in range(4):
    theta = theta / 4. * np.pi
    for sigma in (1, 3):
        for frequency in (0.05, 0.25):
            kernel = np.real(gabor_kernel(frequency, theta=theta,
                                          sigma_x=sigma, sigma_y=sigma))
            kernels.append(kernel)

image_names=[r"images\Im1.bmp",r"images\002_05_L.bmp",r"images\076_02_L.bmp",r"images\08_L.bmp"]
person1=norm_image(image_names[0])
person2=norm_image(image_names[1])
person3=norm_image(image_names[2])
person4=norm_image(image_names[3])
person_names = ('person 1', 'person 2', 'person 3','person 4')
unknown_img=r"images\Im1.bmp"
images = (person1,person2,person3,person4)
ref_feats = np.zeros((4, len(kernels), 2), dtype=np.double)# prepare reference features
ref_feats[0, :, :] = compute_feats(person1, kernels)
ref_feats[1, :, :] = compute_feats(person2, kernels)
ref_feats[2, :, :] = compute_feats(person3, kernels)
ref_feats[3, :, :] = compute_feats(person4, kernels)

print('Matched against references using Gabor filter banks:')
print('original: person 2, match result: ', end='')
unknown_person_norm_image=norm_image(unknown_img)
feats = compute_feats(unknown_person_norm_image, kernels)
print(person_names[match(feats, ref_feats)])
plt.figure(figsize=(10,10))
for i in range(4):
    plt.subplot(4,2,2*i+1)
    plt.xticks([])
    plt.yticks([])
    plt.xlabel(person_names[i]+" eye image")
    plt.imshow(np.array(cv2.imread(image_names[i])),cmap ='gray')
    plt.subplot(4,2,2*i+2)
    plt.xticks([])
    plt.yticks([])
    plt.xlabel("Match result : "+str(percentage_array[i])+"%")
    plt.imshow(np.array(cv2.imread(unknown_img)),cmap ='gray')
plt.show()
```

Implémentation et Résultats (Matching)

On obtient ainsi après le matching de l'image introduite avec notre base de données existante, le pourcentage de compatibilité avec chaque image de la base de données, et en affiche sur la console la personne qui possède le plus grand pourcentage (le plus compatible).




```
[Running] python -u "d:\Ginf2S4\TIProject\Iris-recognition-with-python-and-opencv-master\code\iris_match.py"
d:\Ginf2S4\TIProject\Iris-recognition-with-python-and-opencv-master\code\iris_match.py:47: DeprecationWarning: Please use `gaussian_
namespace, the `scipy.ndimage.filters` namespace is deprecated.
  gaussian_img=scipy.ndimage.filters.gaussian_filter(median_img,
Matched against references using Gabor filter banks:
original: person 2, match result: person 3
```




```
[Running] python -u "d:\Ginf2S4\TIProject\Iris-recognition-with-python-and-opencv-master\code\iris_match.py"
d:\Ginf2S4\TIProject\Iris-recognition-with-python-and-opencv-master\code\iris_match.py:47: DeprecationWarning: Please use `gaussian_filter` f
namespace, the `scipy.ndimage.filters` namespace is deprecated.
  gaussian_img=scipy.ndimage.filters.gaussian_filter(median_img,
Matched against references using Gabor filter banks:
original: person 2, match result: person 3
```

conclusion

Ainsi, nous concluons que nous avons appliqué avec succès la méthode de reconnaissance de l'iris en partant de la localisation à l'extraction et à la correspondance des caractéristiques et en identifiant avec succès un certain iris individuel qui se démarque des autres. Ainsi, nous prouvons à nouveau que

a) Iris est la biométrie la plus sûre pour l'identification car - même les yeux droit et gauche d'une seule personne ne correspondraient pas de manière convaincante.

b) Cette unicité de chaque iris et le peu de temps nécessaire pour décoder les informations sont le principal phénomène qui rend cela évolutif dans tous les aspects.