Rapport
Fezoui Yacine
Introduction
Projet d'identification par l'iris de l'œil. La réalisation de ce projet a été faite avec la bibliothèque Opencv qui est spécialisée dans le traitement d'image, utile donc pour la comparaison des matches entre une photo insérée et la base de données. Pour l'interface, j'ai choisi la bibliothèque Flask.

Environnement
pip install Flask opencv-python-headless scipy numpy

Réalisation
outre les lignes de codes qui concerne la communication entre l'interface et la partie traitement, plusieur points sont à retenir de cette dernière

1. Réception de l'image :

```
image= request.files['img']
image= image.read()
```

2. Augmenter le contraste global de l'image :

```
probleme_c = cv2.equalizeHist(probleme_c)
```

3. Cette fonction ensuite vas lire l'image introduite et aussi chaque image de la bdd, pour extraire les keypoints, la Boucle a servie a prendre uniquement les keypoints qui se situe suffisamment loins du centre (383,287) ce qui représente la pupille, et pas assez pour ne pas dépasser l'iris, les valeurs choisit (250 et 70) ont été sélectionnés après plusieur testés.

```
def extract_sift_features(image):
    sift = cv2.SIFT_create()
    keypoints, descriptors = sift.detectAndCompute(image, None)
    K=[]
    D=[]
    for i, j in zip(keypoints, descriptors):
        x, y = i.pt
        distance = euclidean((x, y), (383, 287))
        if distance < 250 and distance > 70 :
            K.append(i)
            D.append(j)
    return K, D
```

4. La boucle ci-dessous, permet de lire chaque photo de la BDD, et d'appliquer les même traitement que la photo insérée. ensuite a chaque itération, nous gardons celle avec le plus de matchs (Explication en 5).

```
for database in database_images:
    image_c2 = cv2.imread(database,cv2.IMREAD_GRAYSCALE)
    database_c = cv2.equalizeHist(image_c2)
    database_k, database_d = extract_sift_features(database_c)
    matches = match_features(probleme_d, database_d)
    print(database,len(matches))

    if len(matches) > max_matches:
        match_sol= matches
        max_matches=len(match_sol)
        solution = database
        solution_c = database_c
        solution_k = database_k
```

5. Cette fonction est appelée à chaque itération, elle prend deux images avec leurs key point, elle renvoie ensuite ce qui est considéré comme étant des good matches. En prenant les deux meilleurs match pour chaque keypoint de l'image insérée, on calcule la différence entre les deux distances. Un vrai good matches sera unique, donc il ne peut pas y avoir 2 bon matches, alors le rapport entre les 2 est forcément plus petit que 0,76, cette valeur est inspirée des nombreux travaux cités en bibliographie.

```python
def match_features(probleme_d, database_d, ration_distance=0.76):
    bf = cv2.BFMatcher()
    probleme = np.array(probleme_d)
    database = np.array(database_d)
    matches = bf.knnMatch(probleme, database, k=2)
    good_matches = []
    for m, n in matches:
        if (m.distance/n.distance) < ration_distance:
            good_matches.append(m)
    return good_matches
```

Sources

La partie du code ou nous calculons le rapport entre les 2 matches est inspiré de plusieurs projet trouvés sur le net, plus exactement en cherchant des mot clé lié au sujet du projet sur github et youtube, même chose pour l'utilisation de cv2.equalizeHist.

Guide

Après avoir exécuté le script python execute.py, le lien vers le site web sera affiché dans le terminal: http://127.0.0.1:5000. 3 cas d'utilisation sont a tester avec les photo qui se trouve en dehor du fichier database 1 (008L_1.png, ident.png, sign.png, echec.png):

1. Reconnaissance: j'ai sorti la photo "008L_1.png" de la base de données, la page d'accueil permet donc de l'insérer et d'opérer la reconnaissance. logiquement, le résultat doit être "008L_2.png" ou "008L_3.png".
2. Insertion dans la base de données: j'ai fait sortir les trois photos de la personne 038R, pour insérer une d'entre elles dans la page Enregistrer "sign.png", puis la reconnaître avec une autre d'entre elles. "ident.png"
3. Cas d'échec: le code permet de récupérer l'oeil le plus proche, donc J'ai assigné la valeur 100 à la variable SEUIL en ayant observé une large différence entre les nombre de matches de deux yeux different (8,9,10…) et entre le même oeil (250, 900…). Des valeurs similaires ont été observées chez d'autres projets. Pour le tester, j'ai sorti une photo d'un œil et supprimé les deux autres. donc il suffit de faire la même opération dans la page d'accueil avec "echec.png" pour avoir un message de non reconnaissance.

Conclusion

Ce projet m'a permis de prendre connaissance de différentes techniques d'identification des individus, l'iris étant plus précis que l'identification par l'index. La recherche bibliographique m'a permit ensuite d'avoir une idée sur la conception du code python et des différentes fonctionnalités qu'offre la bibliothèque opencv. et enfin, la réalisation d'une interface web a permis de rendre l'identification simple à utiliser.

Références web

Iris Recognition Based on SIFT Features : https://arxiv.org/pdf/2111.00176

Efficient Iris Recognition Based on Optimal Subfeature Selection and Weighted Subregion Fusion : https://www.hindawi.com/journals/tswj/2014/157173/

Enhancing Computer Vision with SIFT Feature Extraction in OpenCV and Python

FLANN Feature Matching Example with OpenCV

andreibercu / iris-recognition : https://github.com/andreibercu/iris-recognition