TD-5 – Extraction de caractéristiques visuelles

Informatique Graphique et Vision

> Les exercices suivants seront réalisés en TD avec le langage Python (3.X), et les bibliothèques NumPy, OpenCV (4.X) et MatPlotlib.

A. Rappel

Installation de la bibliothèque OpenCV

- Ouvrez le terminal « Anaconda Prompt »
- Saisissez le code ci-dessous :

```
pip install opencv-python
```

Lancer un notebook Jupyter

```
Dans le même terminal, saisissez :jupyter notebook
```

Inclusion des bibliothèques nécessaires à ce TD dans votre script

```
01.  import cv2
02.  import numpy as np
03.  from matplotlib import pyplot as plt
```

B. Fonctions OpenCV

Vous trouverez ci-dessous des exemples d'utilisation des principales fonctions OpenCV dont vous aurez besoin afin de réaliser ce TD.

Lecture d'une image en couleur à partir d'un fichier

```
01.  fichier = "image.jpg"
02.  img = cv2.imread(fichier, cv2.IMREAD_COLOR)
```

Affichage d'une image

```
01.  cv2.imshow("Titre", img)   # la variable img contient une image BGR
02.  cv2.waitKey(0) # a ne pas oublier
```

Extraction de coins avec le détecteur de Harris

```
01.  res = cv2.cornerHarris(« imagEnNiveauGris », « TailleBlock »,
     « TailleSobel », 0.04)
```

La fonction « cornerHarris » renvoie une image contenant à chaque pixel un score qui représente la probabilité d'existence d'un coin.

Source : https://docs.opencv.org/3.4/dc/d0d/tutorial_py_features_harris.html

Algorithme SIFT

```
01.  sift = cv2.SIFT_create()  # création de l'objet de la classe SIFT
02.  #Extraction des points et des descripteurs
03.  points, descripteursImg = sift.detectAndCompute(ImageEnNiveauGris, None)
```

Algorithme ORB

```
01.  orb = cv2.ORB_create()  # création de l'objet de la classe ORB
01.  #Extraction des points et des descripteurs
02.  points, descripteursImg = orb.detectAndCompute(ImageEnNiveauGris, None)
```

Affichage de points

```
01.  couleur = (51, 163, 236)
02.  cv2.drawKeypoints(imgEntree, points, imgSortie, couleur,
03.                    cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
```

Source : https://docs.opencv.org/3.4/d4/d5d/group__features2d__draw.html

Algorithme de comparaison entre deux images (force brute)

```
01.  algoBF = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=False)
02.  paires_corresp = algoBF.knnMatch(descripteurImg1, descripteurImg2, k=5)
```

Types de distance : Hamming, L1, L2, *etc*.
Source : https://docs.opencv.org/3.4/d3/da1/classcv_1_1BFMatcher.html

C. Exercices

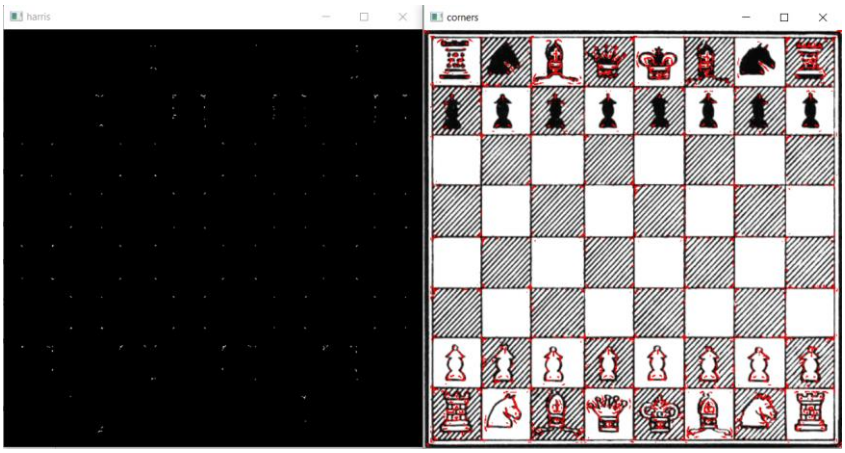
Exercice 1 – Extraction de coins

1.1 Écrivez une fonction qui extrait les coins d'une image passée en argument avec le détecteur de Harris proposée par la bibliothèque OpenCV. Utilisez l'image « chess_board.png » pour tester votre fonction.

- Utilisez une taille de « *23* » pour le filtre de Sobel et de « *2* » pour le voisinage « Block »
- Utilisez le code source ci-dessous afin d'afficher seulement les coins avec un valeur d'au moins 1 % de la valeur maximale de coins retrouvée sur l'image analysée.

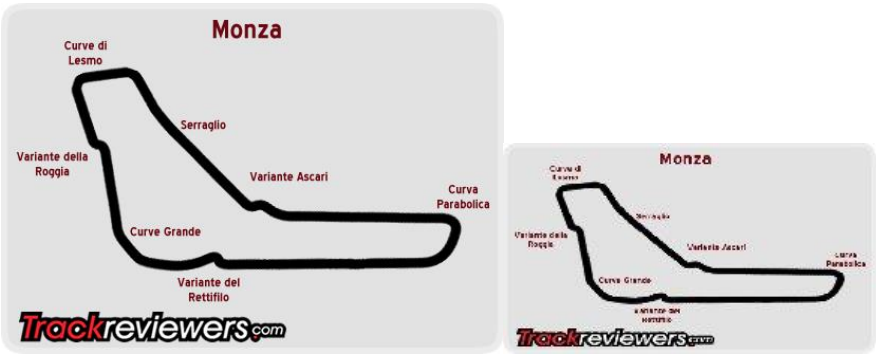
```
image[resultatHarris > 0.01 *resultatHarris.max()] = [0, 0, 255] # couleur rouge
```

Résultat **Résultat sur l'image**



1.2 Répétez la procédure ci-dessus pour les images « *track.png* » et « *track_small.jpg* » et analysez le résultat.

Est-ce que le détecteur de Harris produit le même résultat sur les deux versions de cette image ?



Exercice 2 – Algorithme SIFT

1. Écrivez une fonction qui extrait les caractéristiques visuelles (points remarquables) utilisées par l'algorithme SIFT et affichez-les sur l'image d'entrée. Utilisez l'image « *varese.jpg* » afin de montrer le résultat de votre fonction.

2. Comparez le résultat de l'algorithme SIFT pour les images « *track.jpg* » et « *track_small.jpg* » avec celui de la méthode Harris. Que pouvez-vous conclure ?

Exercice 3 – Algorithme ORB

L'algorithme ORB est plus récent que SIFT et prend en compte des optimisations plus récentes pour la détection et la description de points.

1. Écrivez une fonction qui extrait des caractéristiques visuelles avec l'algorithme ORB et affichez-les sur l'image d'entrée. Utilisez l'image « *varese.jpg* » afin de tester votre fonction.

Exercice 4 – Recherche d'imagette dans une image

Nous allons maintenant utiliser l'algorithme ORB et un algorithme de *matching* par force brute afin de rechercher une imagette à l'intérieur d'une image.

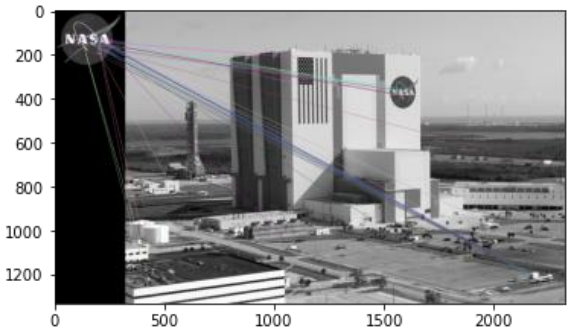
1. Écrivez une fonction qui extrait et compare les caractéristiques visuelles de deux images en prenant en compte les étapes décrites ci-dessous. Testez votre fonction avec les images « *nasa_logo* » et « *kennedy_space_center* ».

Etapes :

```
01. Calculez des descripteurs de l'algorithme OBR de deux images.
02. Utilisez l'algorithme de comparaison exhaustive (force brute) afin de
    chercher de matches de l'image A dans l'image B, en utilisant les
    descripteurs extraits.
03. Triez les paires de correspondances par distance.
04. Affichez les 25 meilleurs matches obtenus entre les images analysées en
    utilisant la fonction « drawMatchesKnn ».
```

Algorithme de tri :

```
Matches_tri = sorted(correspondances, key=lambda x:x[0].distance)
```



Références :

- Joseph Howse, *Learning OpenCV 4 Computer Vision with Python 3*, 3e ésdition, 2022.