Since 1906

ĐHQGHN

VNU

ĐẠI HỌC QUỐC GIA HÀ NỘI

oi

V]

INSTITUT FRANCOPHONE INTERNATIONAL

Thème: Détecteur de points d'intérêt SIFT et descripteur SIFT

Résumé

Auteurs : **Altema Emaude, Lindor Bernard, Vaillant Hyggens**

Enseignante: Madame Nguyen THi Hoan

Option: **Systèmes Intelligents et Multimédia**

Promotion : 24eme

Année Académique 2020-2021

Plan

Introduction

Dorénavant, la vision par ordinateur est un domaine scientifique interdisciplinaire qui traite la manière dont les ordinateurs peuvent être conçus pour comprendre ou plus profondément à partir d'images ou de vidéos numériques. Sur le plan d'ingénierie, elle facilite l'automatisation des différentes tâches que le système visuel humain peut effectuer tout naturellement. L'un des intérêts principaux de cette discipline qu'est la vision par ordinateur c'est la théorie des systèmes artificiels qui permet de faire l'extraction des informations relatives aux images, dont les résultats sont de plus en plus favorables.

Dans notre TP, nous avons implémenté la méthode " scale-invariant feature transform" communément appelé (SIFT), que l'on peut traduire par « transformation de caractéristiques visuelles invariantes à l'échelle ». C'est un algorithme utilisé dans le domaine de la vision par ordinateur pour détecter et identifier les éléments similaires entre différentes images numériques (éléments de paysages, objets, personnes, etc.). Il a été développé en 1999 par le chercheur David Lowe. L'étape fondamentale de la méthode proposée par Lowe consiste à calculer ce que l'on appelle les « descripteurs SIFT » des images à étudier. Il s'agit d'informations numériques dérivées de l'analyse locale d'une image et qui caractérisent le contenu visuel de cette image de la façon la plus indépendante possible de l'échelle (« zoom » et résolution du capteur), du cadrage, de l'angle d'observation et de l'exposition (luminosité).

Certainement, on utilise cette méthode pour résoudre divers problèmes tels que la détection de la peau, la détection des régions d'objet, etc. Mais cela implique une performance très sensible et peut agir sur la qualité des images de base et d'entrée, ainsi que par l'homogénéité de celle-ci.

Objectif

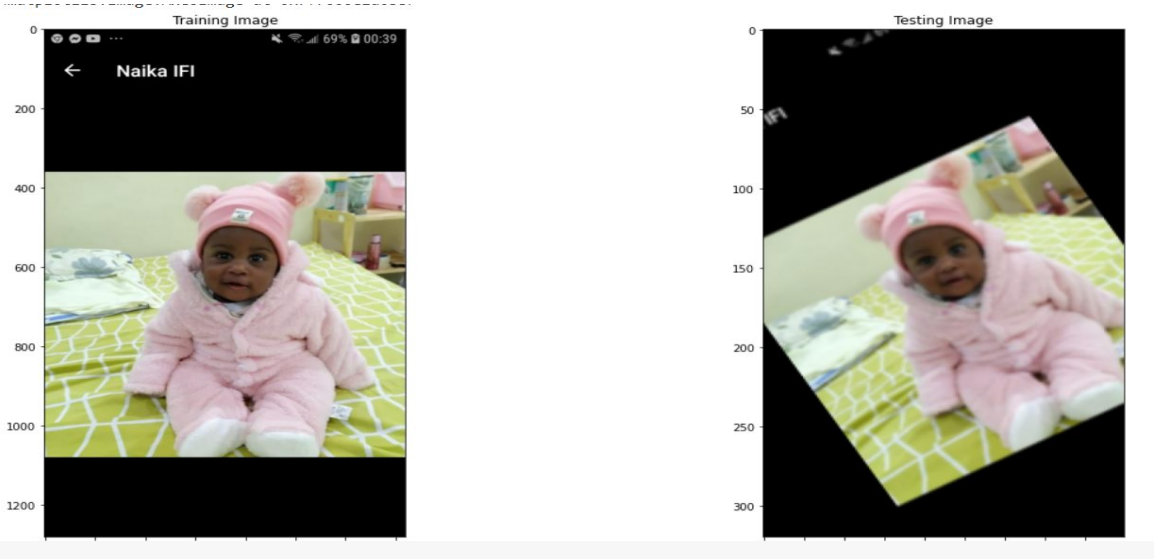
Notre but principal c'est de faire usage du descripteur SIFT dans la reconnaissance d'objets tout en implémentant un programme qui sera capable de :

- De faire la reconnaissance d'objets à l'aide du descripteur SIFT.
- Détecter des points d'intérêt SIFT et Descripteur SIFT sur une image de notre choix.
- Détecter de points d'intérêt SIFT et Descripteur SIFT sur une base d'images pour lesquels nous avons choisi le jeu de données COIL-100.
- Calcul de descripteurs SIFT et leurs mises en correspondance(Matching) des points d'intérêt et entre des images.
- Compter le score de correspondances réussies.

Implémentation du premier modèle.

En tout premier lieu nous avons importé toutes les librairies dont nous aurons besoin (cv2, matplotlib, numpy etc) nous chargeons notre image de référence, étant donné qu'on travail sur colab on met tout sur drive, On affiche l'image par un plt.imshow sur la variable contenant le lien vers cette image pour verifier la presence dans le drive.

Puis on crée l'image de test en ajoutant l'invariance d'échelle et l'invariance rotationnelle. ce qui fait pivoter un peu l'image de test et la rend un peu floue. En voici le résultat.



Pour utiliser la méthode SIFT déjà implémentée dans opencv pour pouvoir faire la détection des point clés ou key points en Anglais et la création du Descripteur SIFT, nous devrions avoir une version opencv compatible pour cela nous avons installé la version (“pip install opencv-python==3.4.2.16”) et le contrib python qui lui est compatible en faisant (“ pip install opencv-contrib-python==3.4.2.16”).

A ce stade nous pouvons faire appel à la méthode SIFT et détecter les points clés, en voici une image des point détectés pour notre image;

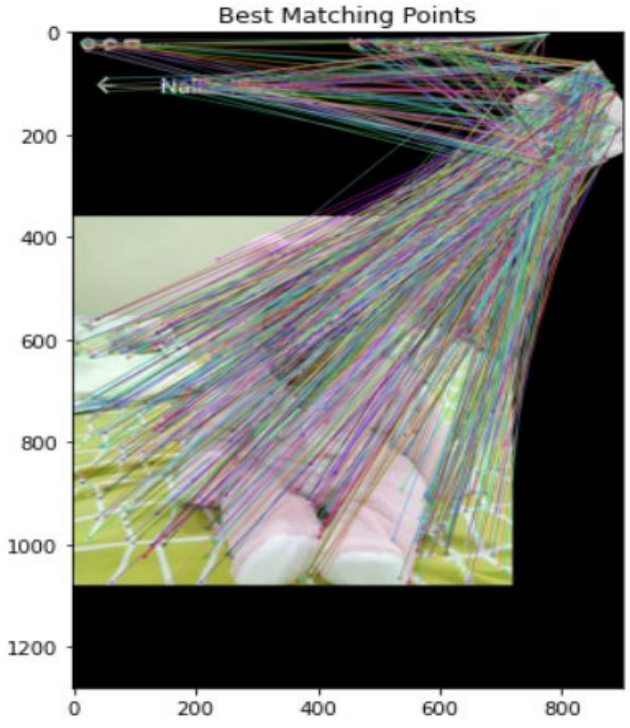
```
array([[[0, 0, 0],
        [0, 0, 0],
        [1, 1, 1],
        ...,
        [0, 0, 0],
        [0, 0, 0],
        [0, 0, 0]],

       [[0, 0, 0],
        [0, 0, 0],
        [1, 1, 1],
        ...,
        [0, 0, 0],
        [1, 1, 1],
        [0, 0, 0]],

       [[0, 0, 0],
        [0, 0, 0],
        [0, 0, 0],
        ...,
        [0, 0, 0],
        [0, 0, 0],
        [0, 0, 0]],
       ...,

       [[0, 0, 0],
        [0, 0, 0],
        [0, 0, 0],
        ...,
        [0, 0, 0],
        [0, 0, 0],
        [0, 0, 0]],

       [[0, 0, 0],
        [0, 0, 0],
        [0, 0, 0],
        ...,
        [0, 0, 0],
        [0, 0, 0],
        [0, 0, 0]],

       [[0, 0, 0],
        [0, 0, 0],
        [0, 0, 0],
        ...,
        [0, 0, 0],
        [0, 0, 0],
        [0, 0, 0]]], dtype=uint8)
```

A partir de là on fait apparaître les points clés d’une part avec une certaine taille plus ou moins visible sur l’image de l'entraînement et d’autre part sans la taille. en voici le résultat en question.

```
[13] Number of Keypoints Detected In The Training Image:  1046
     Number of Keypoints Detected In The Query Image:  235
```

On constate qu'il y a 1046 points clés sur l'image d'entrée et 235 points sur l'image de la requête.

- Puis on fait la correspondance entre les descripteurs SIFT de l'image de formation et l'image de test. Le nombre de points de correspondance entre l'image d'entraînement et l'image de requête est 1046.




```
Number of Matching Keypoints Between The Training and Query Images:  1046
```

Implémentation du deuxième modèle.

Pour la deuxième partie du tp ou nous devons utiliser SIFT sur un jeu de données qui contient des milliers d'images le travail a été plus fastidieux car il s'agit ici de développer un modèle

qui devait faire la reconnaissance d'objets dans un jeu de données très volumineux. Nous avions besoin d'une bibliothèque en plus qui est le scikit learn-model-selection pour pouvoir utiliser le "train_test_split" pour découper votre dataset en données d'entraînement et données de test. Après avoir partagé les données en 80% pour entrainement et 20% pour test, Ce qui nous donne 5760 et 1440 pour les deux parties précitées.

```
# Division des images en train et test
X_train, X_test, y_train, y_test = train_test_split(list_image, list_label, test_size=0.2, random_state=100)
print("Nombre d'image d'entrainement : {}".format(X_train.shape[0]))
print("Nombre d'image de test : {}".format(X_test.shape[0]))
```

```
Nombre d'image d'entrainement : 5760
Nombre d'image de test : 1440
```

Puis on calcule les point clés et le descripteur pour chacune des images de l'entraînement;

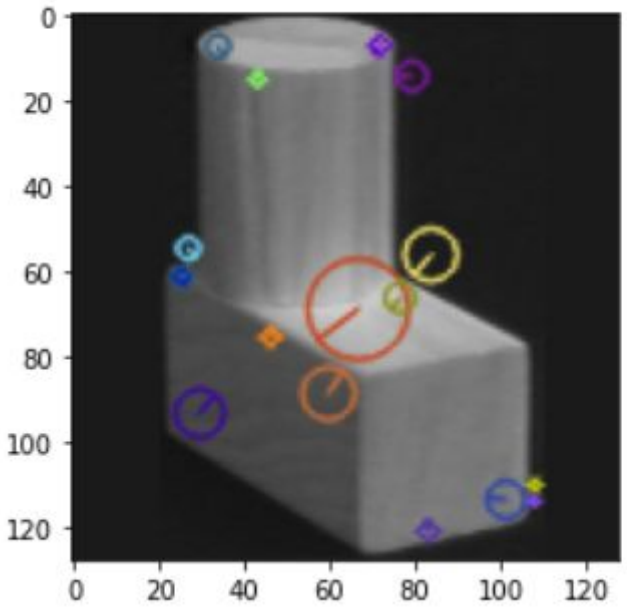
On rappelle qu'il y a 5760 images représentant les 80% de notre dataset et pour la troisième image on trouve qu'il a 39 descripteurs et chaque descripteurs a 128 valeurs. l'image ci-dessous nous en dit plus.

```
(5760,)
(5760,)
```

```
[ ]  print(X_descriptor[2].shape)

(39, 128)
```

On fait apparaître ici les points clés de la 10eme image de notre jeu de données.

```
<matplotlib.image.AxesImage at 0x1bcaec30c40>
```



Ensuite nous avons défini trois fonctions: une pour calculer le score, une autre pour calculer les k meilleurs scores et une autre pour calculer les matching points puis on définit notre fonction de prédiction**.**

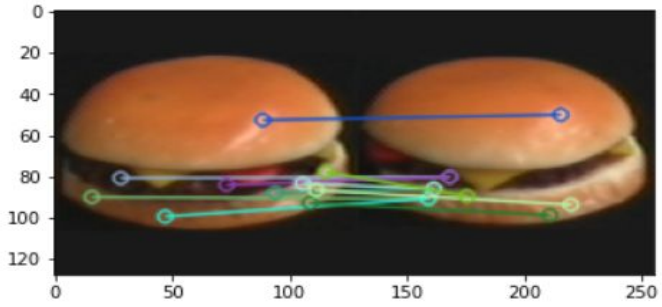
Matching

Le matching ou encore la mise en correspondance est très utilisée dans le domaine de la vision par ordinateur, c'est un processus de comparaison qui consiste à calculer les distances entre les caractéristiques. On peut même en déduire que c'est l'une des étapes les plus importantes lors de la reconstruction 3D à partir d'une paire stéréo ou d'une séquence d'images.

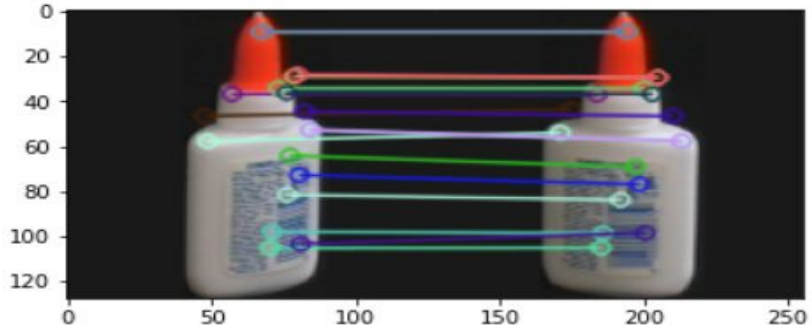
Test

On teste avec les images de test en voici quelques résultats de reconnaissance d'objets de notre deuxième modèle.

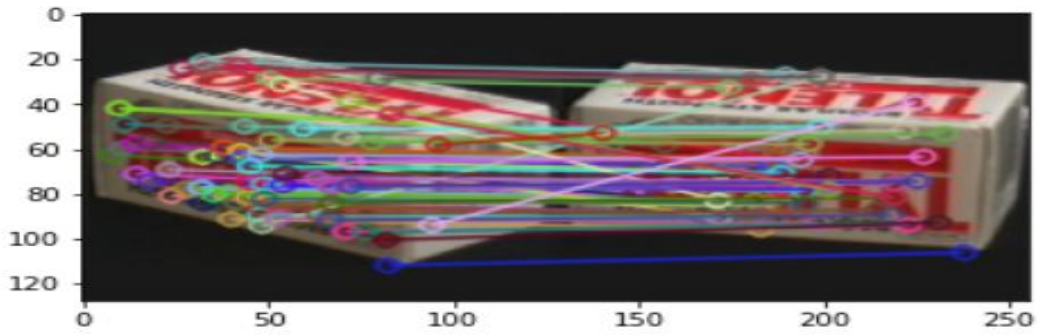
```
In [297]: img_test = X_test[10]
          predc, img_match = predict(img_test)
          print("La prédiction est : {}".format(predc))
          plt.imshow(img_match)

          La prédiction est : obj73

Out[297]: <matplotlib.image.AxesImage at 0x1bca0a9f610>
```



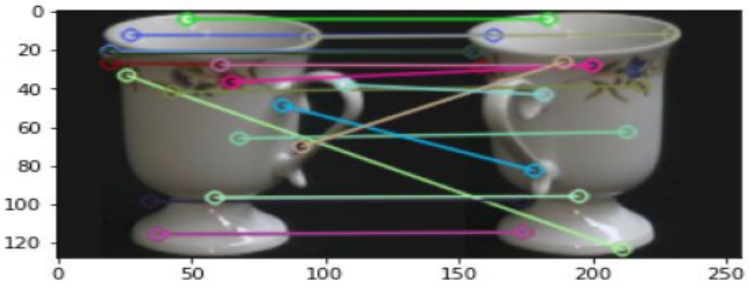

```
In [299]: img_test = X_test[110]
          predc, img_match = predict(img_test)
          print("La prédiction est : {}".format(predc))
          plt.imshow(img_match)

          La prédiction est : obj33

Out[299]: <matplotlib.image.AxesImage at 0x1bca0b4db50>
```



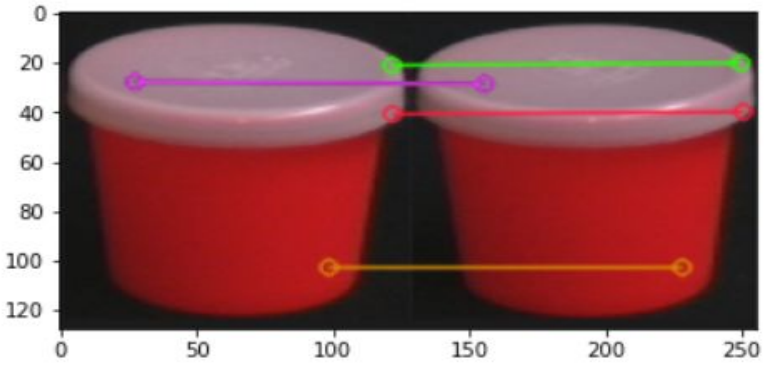
```
<matplotlib.image.AxesImage at 0x1d7b4432370>
```



```
In [298]: img_test = X_test[10]
          predc, img_match = predict(img_test)
          print("La prédiction est : {}".format(predc))
          plt.imshow(img_match)

          La prédiction est : obj89

Out[298]: <matplotlib.image.AxesImage at 0x1bca0afe0d0>
```



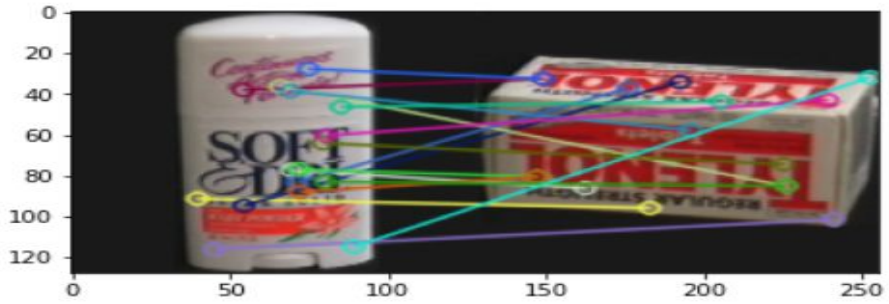
```
In [300]: img_test = X_test[117]
          predc, img_match = predict(img_test)
          print("La prédiction est : {}".format(predc))
          plt.imshow(img_match)

          La prédiction est : obj95

Out[300]: <matplotlib.image.AxesImage at 0x1bca0ba2610>
```



Constat

On note aussi que notre modèle, quoiqu'il est performant, il n'est pas parfait car on a aussi eu de mauvais résultats car la précision n'attendait pas 100% disons ce serait la confiance absolue. En voici un exemple;

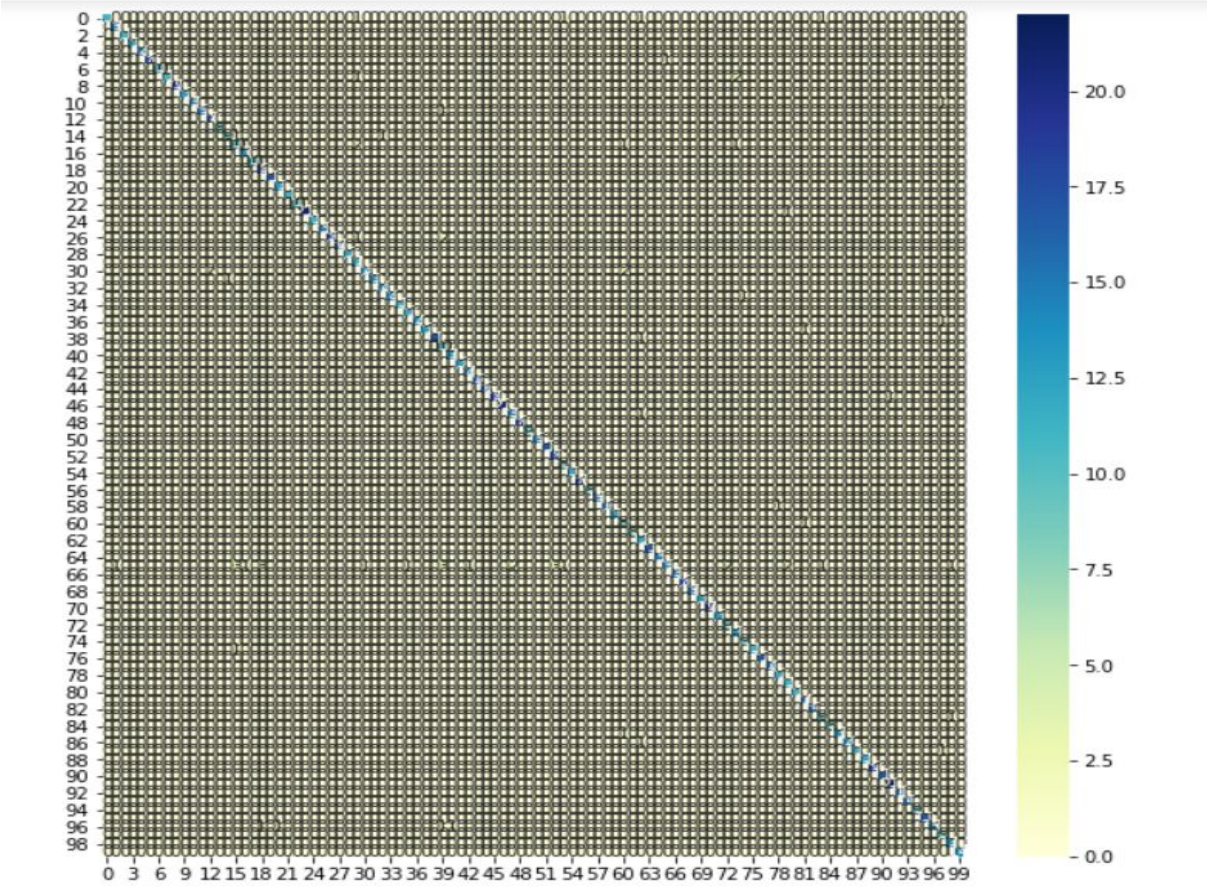
```
In [301]: img_test = X_test[254]
          predc, img_match = predict(img_test)
          print("La prédiction est : {}".format(predc))
          plt.imshow(img_match)

          La prédiction est : obj22

Out[301]: <matplotlib.image.AxesImage at 0x1bca0bfa0d0>
```



4 Matrice de confusion

La matrice de confusion est une matrice qui mesure la qualité d'un système de classification.

Chaque ligne correspond à une classe réelle, chaque colonne correspond à une classe estimée. Brièvement on peut dire qu'une matrice de confusion est un résumé des résultats de prédictions sur un problème de classification. Les prédictions correctes et incorrectes sont mises en lumière et réparties par classe. Les résultats sont ainsi comparés avec les valeurs réelles. Elle permet de comprendre de quelle façon le modèle de classification est confus lorsqu'il effectue des prédictions. Ceci permet non seulement de savoir quelles sont les erreurs commises, mais surtout le type d'erreurs commises.



Conclusion

Ce TP nous a permis d'effectuer la reconnaissance d'objets dans une image en utilisant le descripteur SIFT, en implémentant un programme qui permet de détecter les points d'intérêts et le descripteurs SIFT et la mise en correspondance (matching). Donc ça prend en entrée une image et donne en sortie l'image de l'image et tous les points clés correspondant du même objet contenu dans l'image de requête. En se rappelant que cela a été fait en utilisant une base d'images provenant de Colombia University Image Library (COIL-100) qui contient 7204 images de 100 objets distincts.

D'après les résultats obtenus, on peut conclure que notre programme fonctionne bien car il arrive à effectuer avec succès les points d'intérêts et la correspondance qui existe entre deux images et on peut conclure pour dire que SIFT est une méthode très robuste pour la reconnaissance d'objet, mais présentant des limites, qui le plus souvent est le temps d'extraction des points d'intérêts au sein d'une image au cas où on ne limite pas le nombre.

Perspective

Etant donné que la méthode SIFT offre cette robustesse dans la performance quant à la détection d'objets dans les images, nous comptons l'essayer dans d'autres types d'images par exemple dans des images hyperspectrales ou images satellitaires pour voir si SIFT peut détecter des objets dans des images prises à grande distance. Alors nous pourrons vraiment témoigner de SIFT comme une méthode très performante dans la détection d'objets.

Références

[1] **https://towardsdatascience.com/computer-vision-an-introduction-bbc81743a2f7**

[2] **https ://www.ibm.com/topics/computer-vision**

https://www.analyticsvidhya.com/blog/2019/10/detailed-guide-powerful-sift-techniqueimage-matching-python/

[3] **https://fr.wikipedia.org/wiki/Matrice_de_confusion**

Dataset (base 1) **http://www1.cs.columbia.edu/CAVE/software/softlib/coil-100.php**

[4]Dataset(base 1) **http://www.vision.caltech.edu/Image-Datasets/Caltech101/**

[5]Feature Matching opencv **https://docs.opencv.org/3.4/dc/dc3/tutorial-py-matcher.html**

Lien vers le code pour la reconnaissance d'objets dans le dataset COIL-100

https://drive.google.com/file/d/1hQn_Vi9dtwB1zWZLQBQP5b9SiQkPMBGN/view?usp=sharing

Lien du descripteur sur une image de notre choix.

https://colab.research.google.com/drive/1qjzV3n9YSHBXHHNcIPbNDg5vG86K1FwC?usp=sharing