RAPPORT TRAVAUX PRATIQUES

TP2: **Reconnaissance d'objets avec SIFT** Date : 13/12/2018

Par : HAMIDULLAH Yasser Prof : Dr NGUYEN Thi Oanh
DAOUDA Saidi Kadri

INTRODUCTION

Les différentes recherches faites sur le domaine de la vision par ordinateur, menant vers les approches et des résultats de plus en plus optimale. Dans notre premier TP on a utilisé une technique traditionnelle qui consiste à utiliser les histogrammes des pixels créés à base d'une base d'image et utiliser utilisant le théorème de Bayes afin de prédire la classe d'appartenant des nouveaux pixels d'images inconnus. Certes, cette dernière peut déjà être utilisé pour divers problème tels que la détection de peau, détection des régions d'objets, etc. Mais cela avec une performance très sensible et peut être affecté par la qualité des images de la base et en entrée, et aussi l'homogénéité de ces dernières. Nous allons utiliser maintenant le descripteur dans la reconnaissance d'objet.

1. Présentation de l'application

Pour la mise en marche
Commande 1 : pip install -r requirements.txt (installation des dépendances)
Commande 2 : python detection.py nom_fichier (pour la reconnaissance)
D'autre fichier sont fournis pour l'apprentissage ***_train.py,** le test ***_test.py**, confusion matrix ***_confusion.py**, et exporter des résultats en csv **csv_export_stats.py**.
Pour utiliser l'application, il suffit juste lancer la détection en donnant en paramètre une image.

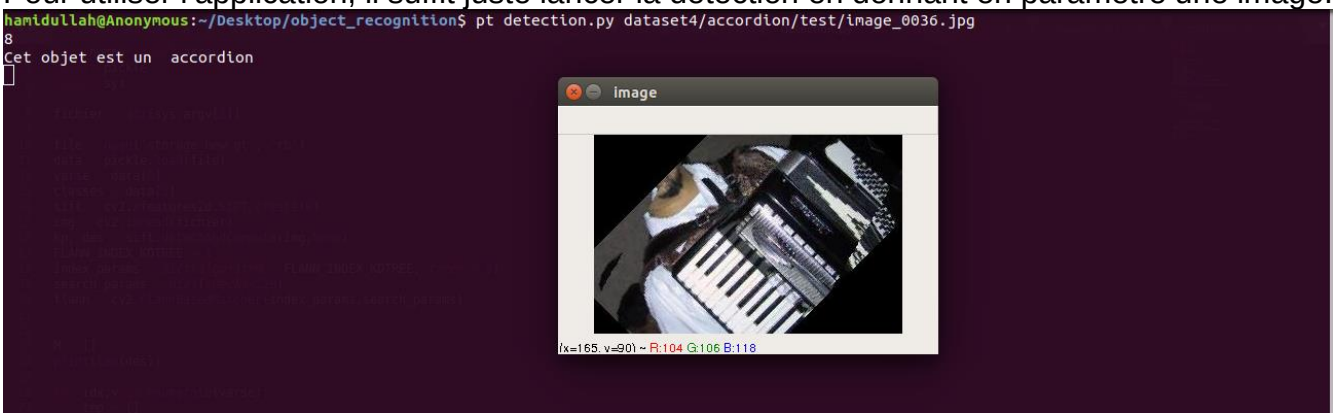


Figure 1: Exemple de détection

2. Présentation du jeu de données

Afin de tester la robustesse de cette approche et voir plus de résultat à interpréter, on a utilisé deux jeux de données :
1 : Données très variées, dans lesquelles, les images de la même classe sont très différentes les unes des autres au niveau de la taille couleur, forme, etc. Ce sont des différences naturelles.

2 : Données plus ou moins homogènes, dans lesquelles seules différences entre les images de la même classe, sont fait d'une manière manuelle en tournant, rognant, zoomant, en ajoutant des bruits.

3. Préparation de la base d'apprentissage

Pour préparer notre base d'apprentissage, on a d'abord effectué une dissection de la base en 2 parties, une pour l'entrainement et une pour le test. Il y a des classes qui ont plus d'image, donc on normalisé la taille des données dans chaque dossier. Normalement on doit faire des expérimentations avec des différentes taille de disectionnement mais comme ce n'était pas le but de ce TP d'étudier le partitionnement des données d'apprentissages, on a fixé la taille des données à 70% pour l'entrainement et 30%. Au sein de chaque classe on a deux dossiers, **train** et **test** respectivement pour l'entrainement et le test.
Pour la base numéro 1 : on a 35 images de train et 15 de test ***(dataset4)***
Pour la base numéro 2 : on a 52 images de train et 20 de test ***(dataset3)***

4. L'entrainement

Dans cet étape on a créé notre base de référence de descripteur. Les étapes sont les suivantes :

1. Détecter les points d'intérêts avec SIFT
2. Récupérer les descripteurs associés à ce point.
3. Stocker les descripteurs dans une liste [classe][image].
4. Sérialisation dans un seul fichier (toutes les images dans un fichier).

On fait la même chose pour chacune des images du dossier **train** de chaque classe.

4.1. Dection des points d'intérêts

Dans opencv SIFT et bien d'autres, et pour l'utiliser, on crée l'objet SIFT et peut prendre en paramètre le nombre de point maximale à détecter. Parfois des images ont plus de point d'intérêt et d'autre moins, et ça peut jouer vraiment sur le temps de traitement de toute la base. Pour avoir normaliser ça on a fixé le nombre de point détecté pour chaque image.

4.2. Récupérer les descripteurs associés à ce point

Les descripteurs s'obtiennent avec avec les points en faisant le detectAndCompute dans opencv. On les stocke dans un tableau à double dimension [classe][image]. Exemple : pour 10 classes avec 35 images d'entrainement on aura 10x35 = 350.

4.3. Sauvegarde des données

Pour stocker nos données d'entrainement, on a utilisé la sérialisation pour conserver les états de notre tableau.

5. Le test

Pour tester, on parcourt le dossier test de chaque classe, et pour chaque image on fait :
1. Récupère ses points d'intérêt
2. Pour fait le Matching avec chacune image dans la base.
3. On trie les images de la base avec le nombre de correspondance de point d'intérêt.
4. En fonction du nombre de voisin le plus proche qu'on veut avoir on prends les n premiers éléments du tableau trié.
5. On prend la classe dominante comme la classe de l'objet en entrée
6. Si ça correspond à la classe de cette dernière c'est vrai sinon c'est faut.

5.1. Le matching avec chacune des images dans la base

Le matching consiste à comparer les points d'intérêt entre deux images données. Pour affirmer qu'un point d'intérêt X d'une image A et correspond à un point dans une autre image B, on calcule le carré de distance entre ce point (de A) avec tous les point dans B et on trie puis prends 2 premiers points Y et Z qui ont le carré de distance plus petit. Si la différence entre XY

et XZ est inférieur à 0.6 on dit que celui qui a la plus petite distance avec X est le point correspondant de X dans dans B, sinon il n'y a pas de correspondance. Pour faire ça dans opencv il existe plusieurs algorithmes de KNN implémenté comme le Flann KNN, BrutForce, BestMatcher, et on n'a pas testé plusieurs algorithmes de ce type mais on a directement le flannKNN based Matcher.

5.2. Création du tableau trié sur le nombre de correspondance

A chaque image trouvée dans les dossiers tests on essaye de faire la correspondance et on compte le nombre de correspondance avec chaque image, et on trie à base de celle-ci pour avoir un tableau de plus grand vers le plus petit.

On peut prendre après le nombre de voisin le plus proche qu'on veut mais pour nous on a effectué plusieurs expérimentations. Cette étape permet de ne prendre que les top n dans le tableau de ci-dessus.

5.3. Choix de la classe d'appartenance

Avec les n voisins les plus proches on a une liste contenante peut être de différents types, et on prend la classe la plus répétées dans cette liste pour prédire la classe de l'image en entrée.

5.4. La matrice de confusion

La matrice de confusion est une sorte d'un résumé sous forme de matrice qui nous permet de voir et mesure la précision de notre prédiction. Pour avoir ce tableau, on a créé un tableau F de taille [nbclasse][nbclasse] double dimension avec la taille de nombre de classe de chaque. On initialise ses valeurs à 0 au début du test. Après chaque image on prend la classe de l'image qui est c et la classe prédite c1. On incrémente la valeur à l'emplacement F[c][c1] +=1. Par exemple si on a 3 classes, le tableau F va être :

```
      [0,0,0],
F =  [0,0,0]
      [0,0,0]
```

, et après avoir passé une image de classe 1, on a prédit la classé 2, le tableau dévient.

```
      [0,0,0],
F =  [0,0,1],
      [0,0,0]
```

et ainsi de suite. [

Pour le visualiser, on transforme ce tableau en une matrice et dessinant chaque valeur par une couleur en fonction du nombre de correspondance.

5.5. Evaluation du modèle

Pour évaluer notre modèle de prédiction on a utilisé le pourcentage des images biens classifiée sur le nombre totale des images.

Taux de correction = Nombre de prediction correcte / Nombre totale des images

6. L'expérimentations

Pour nos expérimentations, on fait varier beaucoup de paramètre:

1. La base *symbol* **b {1,2}, pour les deux base on a numéroté de 1 et 2.**
2. Nombre de descripteur *symbol* **d {2,4,8,16,32,64,128}, on a utilisé des différentes valeurs de descripteur pour voir comment ça affecter le resultat de prédiction.**
3. Le nombre de voisin le plus proche lors de prédiction, symbole **k { 1,2,3,5,10,15}, comme on a pas la valeur de paramètre la plus adapté et laquelle montre plus de bon resultat.**
4. Nombre de classe **N** = {10 **classes**}

Et on affiche la matrice de confusion pour le paramètre qui donne le meilleurs resultat pour chaque base. On recueille les résultats issus de chaque expérimentation et met dans un tableau on crée un graphe pour voir une interprétation à celui-ci.

6.1.Expérimentation 1 :
Base b = 1 ; **dataset4**
N= 10 classes

Descripteurs d = 2

K Voisins	Taux de correction%
1	21.62
2	21.62
3	24.32
5	27.03
10	24.32
15	24.32

Descripteurs d = 4

K Voisins	Taux de correction %
1	16
2	16
3	15.33
5	11.33
10	14.67
15	15.33

Descripteurs d = 8

K Voisins	Taux de correction %
1	20
2	20
3	17.33
5	14.67
10	16.67
15	18.67

Descripteurs d = 16

K Voisins	Taux de correction %
1	14
2	14
3	15.33
5	16
10	18
15	15.33

Descripteurs d = 32

K Voisins	Taux de correction %
1	22.67
2	22.67
3	22
5	20.67
10	18
15	18

6.2.Expérimentation 2 :
Base b = 2 ; **dataset3**
N= 10 classes

Descripteurs d = 2

K Voisins	Taux de correction %
1	61.9
2	61.9
3	57.14
5	52.38
10	52.38
15	33.33

Descripteurs d = 4

K Voisins	Taux de correction %
1	63
2	63
3	62.5
5	61
10	58.5
15	60.5

Descripteurs d = 8

K Voisins	Taux de correction %
1	76
2	76
3	76
5	73.5
10	75
15	73.5

Descripteurs d = 16

K Voisins	Taux de correction %
1	80.5
2	80.5
3	79.5
5	82
10	78
15	76

Descripteurs d = 32

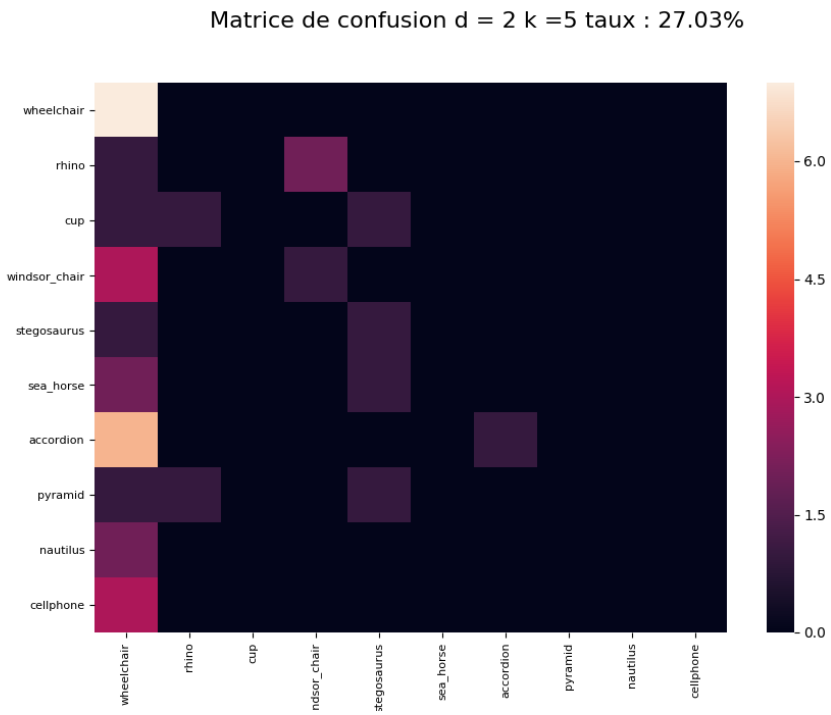
K Voisins	Taux de correction %
1	85
2	85
3	83.5
5	83
10	79.5
15	80.5

7. RESULTATS

Les deux meilleurs résultats des deux expérimentations sont :

7.1.Expérimentation 1

k=5 avec descripteur = 2 avec le taux : 27.03



7.2. Expérimentation 2

k=1 et k = 2 avec descripteur = 32 avec le taux : 85%

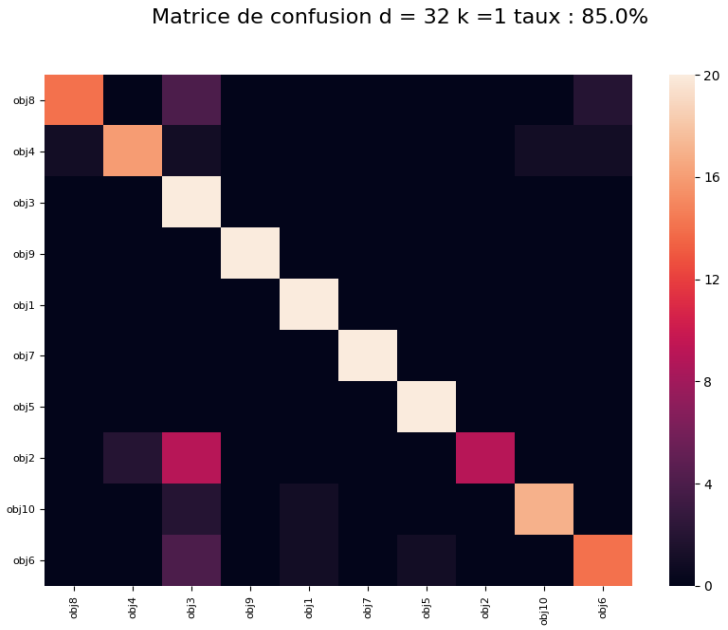


Figure 2 : Matrice de confusion k = 1 d = 2

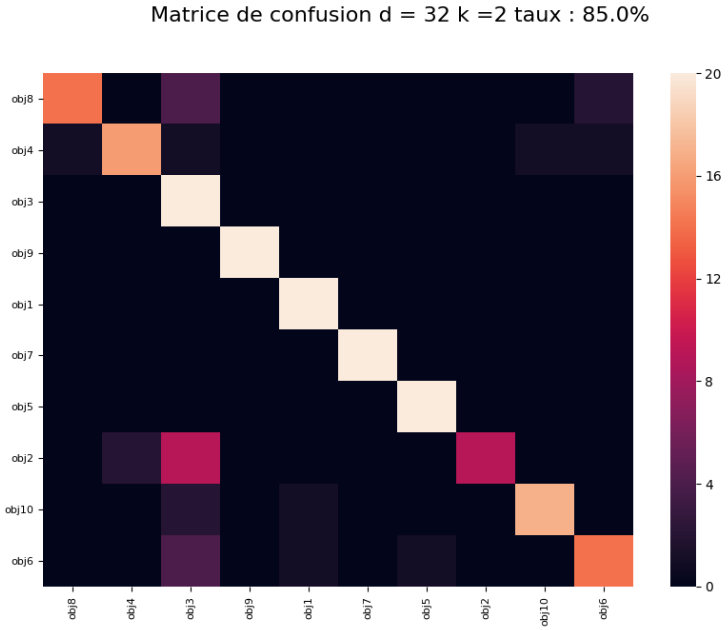


Figure 3 : Matrice de confusion k = 2 d = 2

7.3. Récapitulation

Ci-dessous le graphe de variation de taux de correction en fonction du nombre de descripteur. La barre orange la ligne marquant le max de taux.

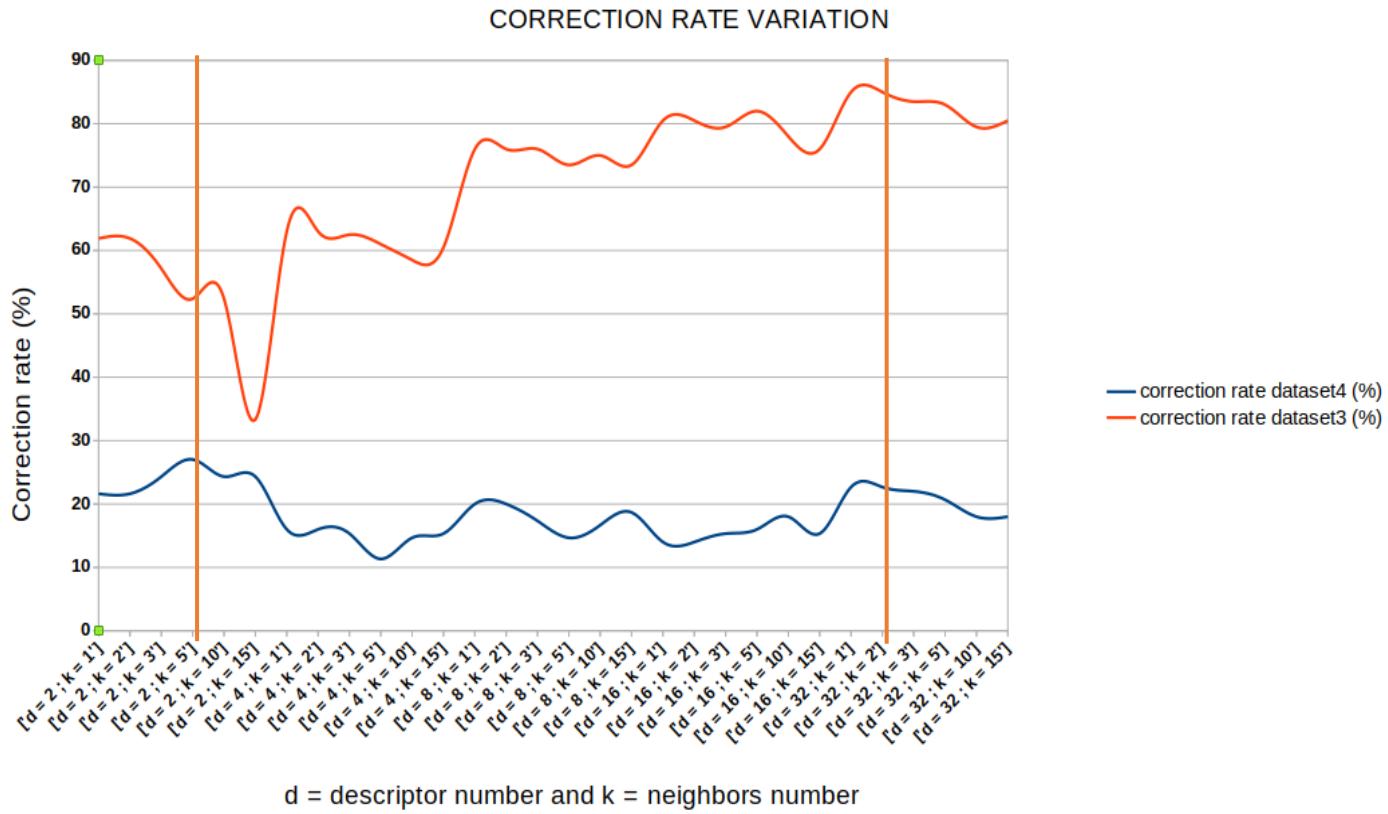


Figure 4 : Variation de taux en fonction de descripteur

8. INTERPRETATION ET ANALYSE DES RESULTATS

Base 1

Pour la base 1 (dataset3), nos résultats sont à 27.03% de taux correction au maximum, cela est due aux différences naturelles des images, c'est-à-dire les différences entre les images ne sont pas dues aux transformations manuelles telles que les rognages, rotations, zoom, etc, elles ont donc des tailles, de forme, sources différentes. Et ça résulte que notre model distingue mal les images de différentes classes, comme montre la matrice de confusion dans le résultat de l'expérimentation 1. Il y a des images qu'il arrive à bien classifié et d'autres non. On a essayé de voir la matrice de plus près les images qui ont plus de confusion entre eux et on a remarqué que vraiment ces images ont une ressemblance au niveau de la forme par exemple la classe **wheelchair et windsor_chair (images ci-dessous)** il y a une forte confusion entre eux :

Figure 5 : windor_chair --- wheelchair

Même SIFT trouve beaucoup de point sur l'image peut seulement va correspondre avec les ceux des images dans la base, d'où on trouve le max de taux de correspondance lorsqu'on a

moins de descripteur (d = 2), car si on a trouvé X point d'intérêt et qu'on a trouvé Y des correspondances, le taux de correction va être :
Taux = Y/X d'où le résultat décroit lorsqu'on prend plus de descripteur.
La variation de k voisins dans cette base affecte aussi le taux de correction puisqu'on voit que plus on augmente le nombre de voisin plus ça descend puisque le point de correspondance est très peu et ça ne laisse pas beaucoup de différence entre la bonne et la mauvaise classe pourtant on prend la classe dominante dans les voisins. Par exemple :
On a :

Classe	**a**	**a**	**c**	**c**	**c**	**c**	**d**	**d**	**c**
Correspondance	2	1	1	1	1	1	1	1	1

Si on va déduire la classe par la classe dominante on aura surement une mauvaise prédiction de classe car la bonne classe c'est le « **a ».**

Raison : **Car dans cette base peu de correspondance entre les points.**

Base 2
Pour la base 2 (dataset4), nos résultats sont à 85% de taux correction au maximum, cela est vraiment normale car les images au sein de la même classe sont déformées par exprès, avec les rognages, rotations, agrandissement, réduction, ajout des bruits (comme expliqué par le propriétaire du dataset. Et d'où l'image dans le dossier d'entrainement et le test sont les mêmes images mais rajouté des modifications très légères. Et on a aussi une matrice de confusion assez bien car on a presque la diagonale en couleur très claire ce qui signifie un meilleur taux de correction. Mais en observant la matrice de confusion d'entre **obj2 et obj3.**

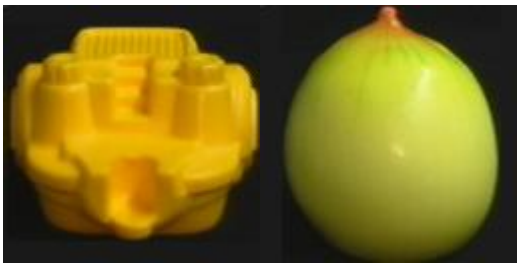


Figure 6 : obj2 -- obj3

On aperçoit une légère correspondance entre ces deux classes car, si on le met à niveau de gris ou si on applique un flou de ces deux, elles deviennent de plus en plus indiscernables. On sait que SIFT dans sa robustesse, n'est pas sensible à des transformations courantes, mais dans le cas ou on a n'a pas assez de variété au sein de notre classes (comme le cas de la base 1) ces genres d'erreur (confusion) n'est pas inévitable.
Ce qui nous mène à définir la raison de cette confusion, c'est le manque de variété dans chaque classe de notre dataset. Le résultat de celle-ci est plutôt contrairement au celui de la première base car on a assez de correspondance des points d'intérêts et ce qui nous résulte ici que le maximum de taux de correspondance a été trouvé avec le plus grand nombre de descripteur. Supposons qu'on a trouvé Z descripteurs, et K correspondance trouvées, si le K est sensiblement égal à Z le résultat se rapproche plus de 1 puisqu'on rappelle que le
Taux = K/Z ; donc plus K augmente plus le résultat tend vers le taux de 100%. Par contre dans la base 1 ce même propos ne donne pas le même résultat.
Le graphe de récapitulation des résultats
Ce graphe nous donne une vue globale sur nos résultats, en générale le paramètre k voisin nous permet d'avoir une variété aux résultats obtenu avec le même nombre de descripteur.

Avoir moins de k voisins et dans le cas où on n'a pas assez de variété nous donne des meilleurs résultats, contrairement dans le cas où on a trop de variété.
Le paramètre d nombre de descripteur améliore le résultat dans le cas où on a une base assez homogène comme la base 2 (dataset3) et fait décroitre le résultat dans le cas de la base 1 (dataset4) car tout simplement peu de correspondance des descripteurs diminue le taux de précision qui se calcule par le rapport de ce dernier avec le nombre total.

Remarque :

Si on ne limite pas le nombre de descripteur (si on ne normalise pas) on aura trop de décalage dans les données d'apprentissage stockées. Ça peut aussi causer la lenteur des tâches, car parfois SIFT détecte jusqu' plusieurs centaines voir de millier sur une même image en fonction de la taille et la qualité de l'image d'après SIFT.
Plus de descripteur c'est plus lent mais augmente la précision et moins de descripteur plus rapide mais affecte plus ou moins le résultat et il y a un très fort décalage comme on a vu dans nos graphiques et matrice de confusion.

N.B : Lorsqu'on parle de descripteur ici c'est le descripteur associé au point d'intérêt détecté.

CONCLUSION

Dans ce TP on a fait une petite application qui peut reconnaître les objets dans une image en utilisant SIFT, donc ça prend entrée une image et donne en sortie le nom de l'objet contenu dans l'image. En sachant que cela a été faite en utilisant une base d'image, et afin de voir l'efficacité de SIFT on a utilisé deux bases qui sont complètement différentes. Une première qui contient des images totalement variées de même type, et une autre contient des images de même classes très homogène. D'après nos résultats, on peut conclure que SIFT est une méthode très robuste pour l'utilité dans la reconnaissance d'objet. Mais même étant ainsi, elle a ses limites, qui est souvent exposé c'est le temps, dans nos expérimentations aussi on a vu qu'elle donne plus de meilleurs résultats pour les images modifiés manuellement qui son de même type. Elle donne un résultat moins bon sur des images de même type d'objet mais acquise d'une manière, formes, tailles, traitements très différents.

REFERENCES

[1] **Feature Matching opencv**
https://docs.opencv.org/3.4/dc/dc3/tutorial_py_matcher.html
[2] **Dataset 3 (base 2)**
http://www1.cs.columbia.edu/CAVE/software/softlib/coil-100.php
[3] **Dataset 4 (base 1)**
http://www.vision.caltech.edu/Image_Datasets/Caltech101/