TRAVAUX PRATIQUES : Reconnaissance d'objets avec SIFT

Par : Janvier Fabienne et Leon Jean-Bertrand Fritzner

Enseignante : Dr NGUYEN Thi Oanh

Date: 16/12/2020

Introduction

A l'ère où nous vivons, la vision par ordinateur est un domaine scientifique interdisciplinaire qui traite de la façon dont les ordinateurs peuvent être conçus pour acquérir une compréhension de haut niveau à partir d'images ou de vidéos numériques. Du point de vue de l'ingénierie, il cherche à automatiser les tâches que le système visuel humain peut effectuer. La discipline scientifique de la vision par ordinateur s'intéresse à la théorie derrière les systèmes artificiels qui extraient des informations à partir d'images, dont les résultats sont plus en plus optimal.

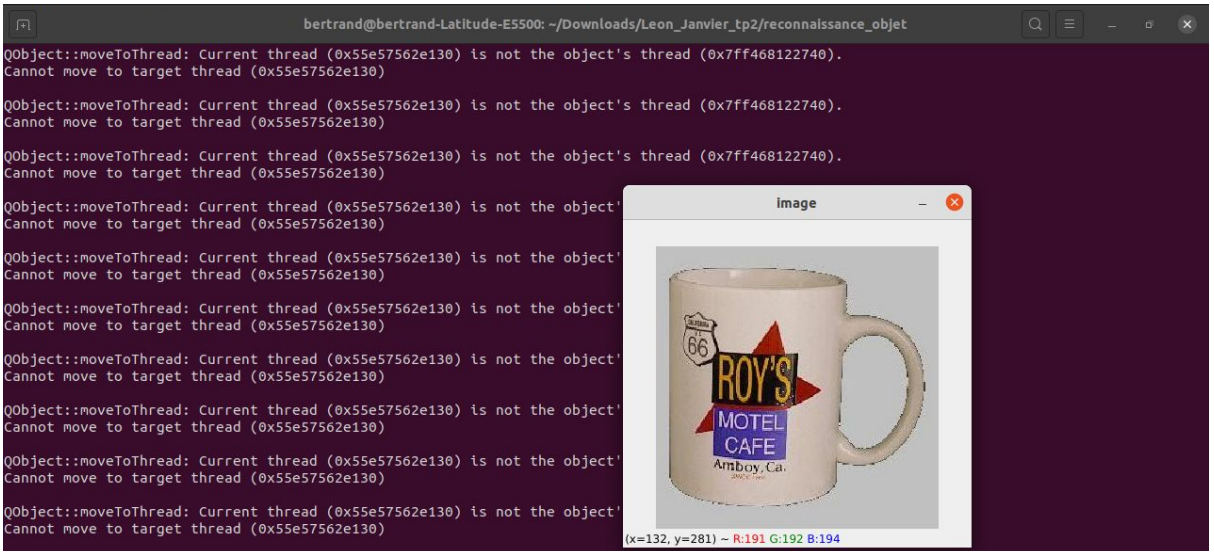
Dans notre premier TP on a utilisé une technique traditionnelle qui consiste à utiliser les histogrammes des pixels créés à base d'une base d'image et utiliser utilisant le théorème de Bayes afin de prédire la classe d'appartenance des nouveaux pixels d'images inconnus. Certes, cette dernière peut déjà être utilisée pour divers problèmes 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 l'exécution de notre projet :

- il faut installer les librairies nécessaires, `pip install opencv-contrib-python`
- A partir du terminal, taper python detection.py nom_fichier(pour la reconnaissance) et d'autres fichiers sont présents pour l'apprentissage tels que: *_train.py, et pour le test *_test.py, matrice de confusion *confusion.py et export des résultats en csv csv_export_stats.py

Pour utiliser l'application, il suffit juste de lancer la détection en donnant en paramètre une image.



2.-Présentation du jeu de données

Afin de tester l'efficacité de notre approche et voir plus de résultat à interpréter, on a utilisé deux jeux de données:

a- 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.

b- Données plus ou moins homogènes, dans lesquelles seules différences entre les images de la même classe, sont faites 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 dissertation de la base en 2 parties, une pour l'entraînement et une pour tester. il y a des classes qui ont plus d'image, donc on normalise la taille des données dans chaque dossier. Normalement on doit faire des expérimentations avec des différentes tailles de sectionnement 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'entraînement et 30%. Au sein de chaque classe on a deux dossiers, train et de test respectivement pour l'entraînement et le test. Ainsi la répartition se fait de telle sorte que le jeu No 1, nommé dataset 4 inclut 25 images de train et 15 de test et le jeu No 2 nommé dataset 3: on a 52 images de train et 20 de test.

4.-L'entraînement

Dans cette étape on a créé notre base de référence de descripteur. les étapes sont les suivants:

A. Détecter les points d'intérêts avec SIFT.
B. Récupérer les descripteurs associés à ce point.
C. Stocker les descripteurs dans une liste[classe][image].
D. 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. Détection 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 d e points 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 points 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'entraînement on aura 10x35 = 350.

4.3. Sauvegarde des données

Pour stocker nos données d'entraînement, 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 f ait

A. Récupère ses points d'intérêt
B. Pour faire le Matching avec chacune image dans la base.
C. On trie les images de la base avec le nombre de correspondance de point d'intérêt
D. En fonction du nombre de voisin le plus proche qu'on veut avoir on prend les n premiers éléments du tableau trié.
E. On prend la classe dominante comme la classe de l'objet en entrée
F. Si ça correspond à la classe de cette dernière c'est vrai sinon c'est faux.

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 inférieure à 0.6, on dit que celui qui a la plus petite distance avec X est le point correspondant de X dans B, sinon il n'y a pas de correspondance. Pour faire ça dans opencv il existe plusieurs algorithmes de KNN implémentés 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 l e 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 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 répandue 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 mesurer la précision de notre prédiction . Pour avoir ce tableau, on a créé un tableau T de taille [n classe][n classe] 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 T va être:

T = [0,0,0]
[0,0,0]
[0,0,0] ,et après avoir passé une image de classe 1, on a prédit la classe 2, le tableau devient.

T = [0,0,0]
[0,0,1]
[0,0,0] 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 correspondances.

5.5. Evaluation du modèle

Pour évaluer notre modèle de prédiction on a utilisé le pourcentage des images bien classifiées sur le nombre total des images.

Taux de correction = Nombre de prédictions correctes / Nombre totales des images.

6.- L'expérimentations

Pour nos expérimentations, on fait varier beaucoup de paramètre:

1. La base symbol b {1,2}, pour les deux bases on a numéroté de 1 et 2.
2. Nombre de descripteurs symboles de {2,4,8,16,32,64,128}, on a utilisé des différentes valeurs de descripteur pour voir comment ça affecte le résultat 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 résultat.
4. Nombre de classe N = {10 classes}

Et on affiche la matrice de confusion pour le paramètre qui donne le meilleurs résultat 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	Dataset 4	N = 10 classes			
Descripteurs d = 2	**K Voisin**	**Taux de correction %**	**Descripteurs d = 4**	**K Voisin**	**Taux de correction %**
	1	21.62		1	16
	2	21.62		2	16
	3	24.32		3	15.33
	5	27.03		5	11.33
	10	24.32		10	14.67
	15	24.32		15	15.33

Base b = 1	Dataset 4	N = 10 classes			
Descripteurs d = 8	**K Voisin**	**Taux de correction %**	**Descripteurs d = 16**	**K Voisin**	**Taux de correction %**
	1	20		1	14
	2	20		2	14
	3	17.33		3	15.33
	5	14.67		5	16
	10	16.67		10	18
	15	18.67		15	15.33

Base b = 1	Dataset 4	N = 10 classes
Descripteurs d = 32	**K Voisin**	**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	Dataset 3	N = 10 classes			
Descripteurs d = 2	**K Voisin**	**Taux de correction %**	**Descripteurs d = 4**	**K Voisin**	**Taux de correction %**
	1	61.9		1	63
	2	61.9		2	63
	3	57.14		3	62.5
	5	52.38		5	61
	10	52.38		10	58.5
	15	33.33		15	60.5

Base b = 2	Dataset 3	N = 10 classes			
Descripteurs d = 8	**K Voisin**	**Taux de correction %**	**Descripteurs d = 16**	**K Voisin**	**Taux de correction %**
	1	76		1	80.5
	2	76		2	80.5
	3	76		3	79.5
	5	73.5		5	82
	10	75		10	78
	15	73.5		15	76

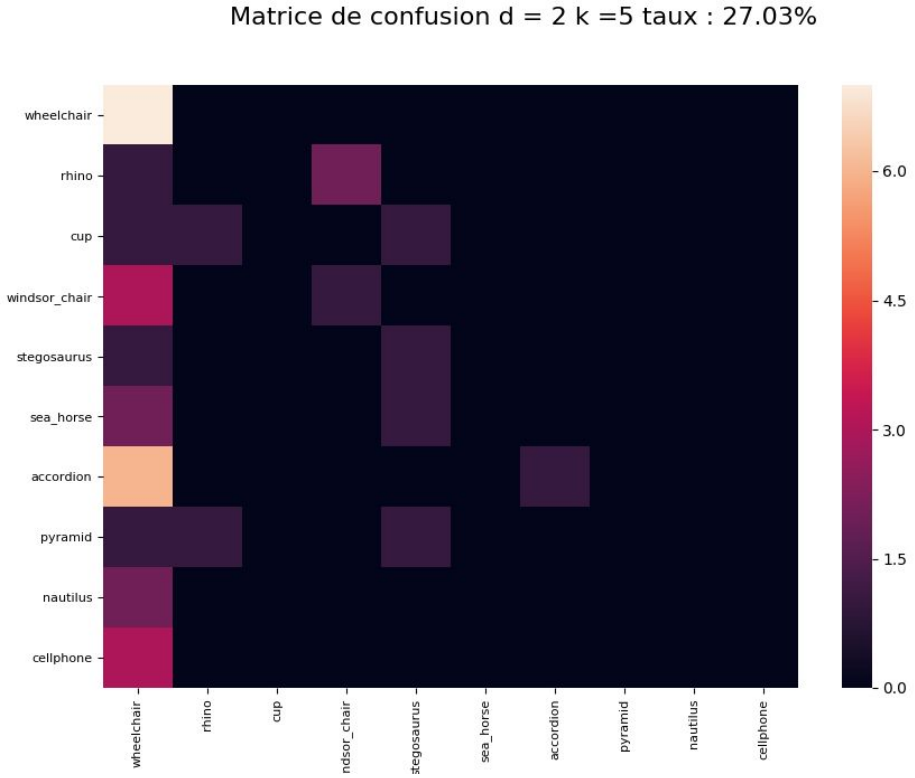
Base b = 2	Dataset 3	N = 10 classes
Descripteurs d = 32	**K Voisin**	**Taux de correction %**
	1	85
	2	85
	3	83.5
	5	83
	10	79.5
	15	80.5

7. Résultats

Les deux meilleurs résultats des deux expérimentations sont:

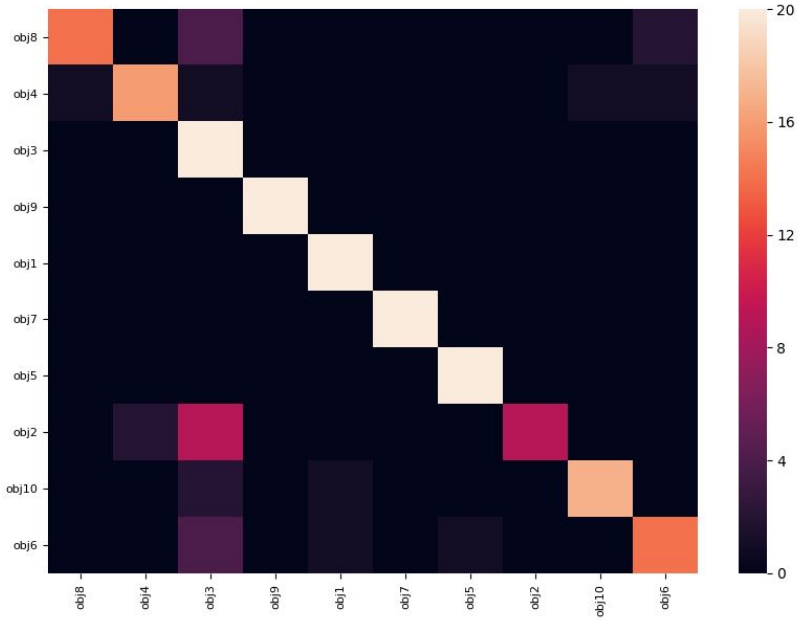
7.1.-Expérimentation 1

k= 5 avec descripteur = 2 avec le taux 27.03



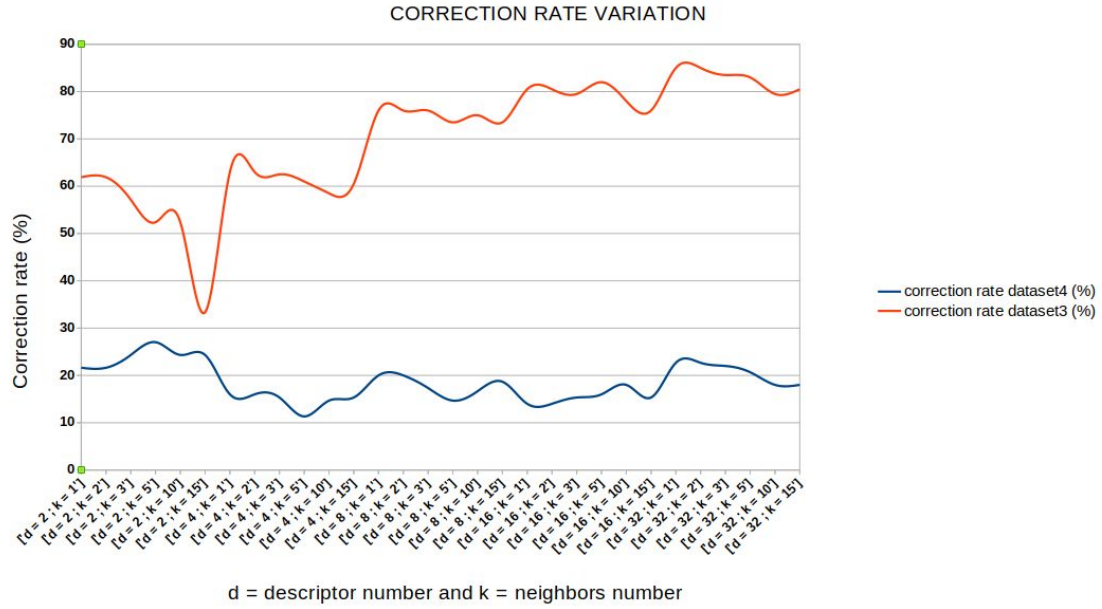
7.2.-Expérimentation 2

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



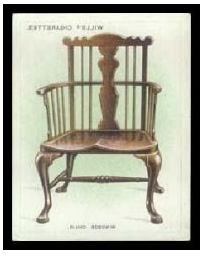
7.3. Récapitulation

Ci dessous le graphe de variation de taux de correction en fonction d' un nombre de descripteurs. La barre orange la ligne marquant le max de taux.



8. Interprétation et analyses des résultats

Pour la base 1 (dataset 4), le résultats de taux de correction maximum est de 27.03 %, cela est due aux différences naturelles des images, ce qui signifie que 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, des sources différentes. Et ça résulte que notre modèle 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 classifier 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 l a classe wheelchair et windsor chair il y a une forte confusion entre eux:



Pour la base 2 (dataset 3), le résultat de taux de correction maximum est de 85 %, 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. Et d'où l'image dans le dossier d'entraînement 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 obj6.

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 où on a pas assez de variété au sein de notre classe (comme le cas de la base 1) ces genres d'erreurs (confusion) n'est pas inévitable. Ce qui nous amène à définir l a raison de cette confusion, c'est le manque de variété dans chaque classe de notre dataset. Le résultat de celle-ci est plutôt contraire à celui de la première base car on a assez de correspondance des points d' intérêt et ce qui nous résulte ici que le maximum de taux de correspondance a été trouvé avec le plus grand nombre de descripteurs.
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 10 0%. 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 obtenus avec l e même nombre de descripteurs. Avoir moins de k voisins et dans le cas où on a pas assez de variété, nous donne de meilleurs résultats, contrairement dans le cas où on a trop de variété.
Le paramètre de nombre de descripteur améliore le résultat dans le cas où on a une base assez homogène comme la base 2 (dataset 3) et fait décroitre le résultat dans le cas de la base 1 (dataset 4) 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 descripteurs, on aura trop de décalage dans les données d'apprentissage stockées. Ça peut aussi causer la lenteur des tâches, car parfois SIFT détecte jusqu'à plusieurs centaines voir de millier sur une même image en fonction de la taille et la qualité de l'image d'après SIFT.
Plus le descripteur est plus lent, plus il augmente la précision et moins il est rapide, ça 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. Lorsqu'on parle de descripteur ici c'est le descripteur associé au point d'intérêt détecté.

CONCLUSION

Pour la réalisation de notre TP on a fait une application qui peut reconnaître les objets dans une image en utilisant SIFT, donc ça prend une image en entrée et donne en sortie l'objet image. En sachant que cela a été fait 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 contient des images totalement variées de même type, et une autre contient des images de même classe très homogène. D'après nos résultats, on peut conclure que SIFT est une méthode très robuste pour l'utiliser dans la reconnaissance d'objet. Dans nos expérimentations on a vu qu'elle donne de meilleurs résultats pour les images modifiées manuellement qui sont de même type et moins bonnes sur des images de même type d'objet mais acquise d'une manière, formes, tailles, traitements très différents.

RÉFÉRENCES

[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/