Reconnaissance d'objets avec le descripteur SIFT

NGUYEN Quoc Khai

03 Novembre, 2013

1 Introduction

Pour ce tp, je vais utiliser les points d'intérêt calculés par la méthode SIFT qui est implémentée par David Lowe grâce au demo distribué sur (http://www.cs.ubc.ca/ lowe/keypoints/) pour reconnaitre des objets. Tout d'abord, je vais présenter le programme et les explications de ce que j'ai mis en place. Ensuite, je vais expliquer les résultats obtenus. Et enfin, je conclure mon travail dans ce TP.

2 Reconnaissance d'objets avec SIFT

2.1 Présentation du programme

Ce programme contient 3 classes divisé en 6 fichiers : *keypoint.h keypoint.cpp kpimage.h kpimage.cpp control.h control.cpp* et un fichier de la fonction *main*. Un *makefile* sert à faciliter de compiler le programme. Je mets aussi le shell-script pour convertir des images dans le répertoire du programme. Pour exécuter le programme, dans le répertoire du programme, on exécute *make*, ensuite, on tape *./siftimage*, et faire comme ce que le programme proposera. Vous trouverez ci-dessous l'interface du programme :

```
./siftimage
Bienvenue d'utiliser ctimage - l'auteur: NGUYEN Quoc Khai
================================================
Vous pourriez taper:
0) rm repertoire_key number_key -
Pour supprimer des fichiers ayant le nombre de point < number_key
1) fkey repertoire_image repertoire_key - Pour chercher des points intérêts
2) corr 0 img1 key1 img2 key2 img_correct- Pour correct entre 2 images
3) corr 1 key repertoire_des_keys_test -
Pour calculer le score entre l'image de test avec d'images d'apprentassge
4) corr 2 repertoire_des_keys repertoire_des_keys_test -
Pour calculer le score entre 2 bases d'images
5) help - pour aider
6) exit - pour quitter
================================================
Tapper: corr 2 database/k database/k
Tapper: corr 2 database/kt database/kt1
```

Figure 1: Interface du programme

Ce programme a les fonctionnalités suivantes :

1. *rm repertoire_key number_key* sert à supprimer des fichiers ayant le nombre de descripteurs inférieur à *number_key*

2. *fkey repertoire_image repertoire_key* sert à récupérer les points d'intérêts des images dans le répertoire d'images ver le répertoire de clés. Dans cette fonctionnalité, j'ai rappeler le programme SIFT de l'auteur de la méthode SIFT. Le programme va lire toutes les images dans le répertoire d'images. Chaque image va être créer un fichier de point d'intérêt qui va être mis dans le répertoire *repertoire_key* que l'utilisateur a tapé.

3. *corr 0 img1 key1 img2 key2 img_correct* pour la reconnaissance d'un image de test avec un autre image

4. *corr 1 key repertoire_des_keys_test* pour la reconnaissance d'un image de test avec une base d'images.

5. *corr 2 repertoire_des_keys repertoire_des_keys_test* pour la reconnaissance d'une base d'images de test avec une base d'images de références.

6. *conv matrix_text matrix_image* pour convertir un fichier de texte de matrice à une image.

2.2 Bases d'images

Dans ce tp, j'ai utilisé la base d'images *Columbia University Image Library (COIL-100)*. J'ai divisé cette base d'image en deux parties, l'une sert à la référence et l'autre sert à tester. En raison de la complexité de l'algorithme, avec le temps limité, je ne peux pas tester sur la base entière. J'ai donc prend une partie de 2550 images. 1200 images sont pour la base de test, 1350 images sont pour la base de modèle. J'ai converti des images en format pgm par le fichier de shellscript *convertImages.sh*.

2.3 Mise en correspondance (matching) de points d'intérêts

Pour la mise en correspondance (matching) de points d'intérêts, j'ai implémenté les méthodes se basant sur les idées de l'auteur et dans l'exigence du TP. J'ai calculé le ratio entre la distance la plus courte et la deuxième plus courte distance. Si ce ratio est inférieur à un seuil, la correspondance est considérée comme robuste, sinon elle est rejetée :

$$ratio = \frac{d_{plusproche}}{d_{deuxplusproche}} \leq seuil \tag{1}$$

Notes: J'ai mis : seuil = 0.6 dans e code.

2.4 Calcul de correspondance entre images

Quand la correspondance entre tous les points SIFT calculés, j'ai calculé le score comme suit :

$$score = \frac{correspondances\ reussies}{nombre\ de\ descripteurs\ de\ l'image\ modele} \tag{2}$$

L'image ayant le score le plus élevé déterminera la classe de l'objet inconnu.

Pour la matrice de confusion, chaque test, je l'enregistre dans le fichier *matrix_confusion.OUT* et cette matrice est représentée aussi par l'image *matrix_confusion.png*. En fait, cette image est presque en noir à cause du nombre d'images de chaque classe. Dans la base d'images, chaque classe ne contient que quelques images, alors, chaque valeur de la matrice est faible. C'est la raison pour laquelle j'ai utilisé une formule pour que cette image de vient visible :

$$valueur = \frac{valeur\ *\ 255}{MAX} \tag{3}$$

Notes: MAX est la valeur maximale de la matrice.

Donc, dans chaque test le programme donne deux images de la matrice de confusion *matrix_confusion.png* et *matrix_confusion_visible.png*.

3 Explication des résultats obtenus

3.1 Correspondance d'image à image

3.1.1 Image 1

Tout d'abord, on essaie avec la comparaison entre deux images, on tape :
Commande :
corr 0 database/image_test/obj1__5.pgm database/key_test/obj1__5.key database/image_trainning/obj1__0.pgm database/key_trainning/obj1__0.key database/img_correct.pgm

Figure 2: Image de test

Figure 3: Image de références

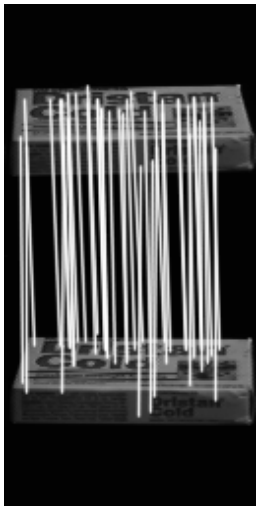

Figure 4: Résutat de correcte

Pour ce test, le score obtenu est de 43.0108%
Explication :
On trouve que ce sont deux images différentes étant dans la même classe d'objet, c'est pour ça que la correspondance est acceptable. Le résultat de correct est seulement de 43% car j'ai mis le seuil = 0.6. Si j'augmenterai le seuil, le pourcentage va être plus élevé. Avec ce seuil, le pourcentage n'est pas élevé mais il ne fait pas beaucoup de faux. Si l'on augmente le seuil, il risque des correspondances faux.

3.1.2 Image 2

Commande :
corr 0 database/basmati.pgm database/basmati.key database/scene.pgm database/scene.key database/img_correct_basmati.pgm

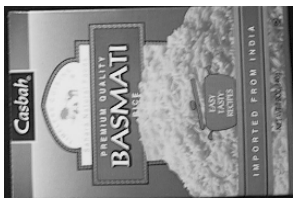

Figure 5: Image de test

Figure 6: Image de références

Figure 7: Image correcte

Explication :
Cet exemple montre que cette méthode permet de reconnaitre un objet dans l'ensemble d'objet. En global, le résultat est exact. Par contre, il y a une correspondance faux à la porte.

3.2 Reconnaissance d'un image dans la base d'images

3.2.1 Exemple 1

Commande : *corr 1 database/test5/key_test/obj13__80.key database/test5/key_trainning*
Classe de l'image:
13
obj13__75.key(0.538462)
obj13__265.key(0.5)
obj13__235.key(0.461538)
obj13__65.key(0.304348)
obj13__225.key(0.285714)
On peut voir des images :

Figure 8: obj13__80.png

Figure 9: obj13__75.png

Figure 10: obj13__265.png

Figure 11: obj13__235.png

Figure 12: obj13__65.png

Figure 13: obj13__225.png

Explication :
Cet exemple est de faire la correspondance l'image *obj13__80.pgm* avec la base d'apprentissage. On trouve que le résultat est bon, il classifie vraiment l'objet à la classe d'objet et 5 images les plus correctes sont toujours dans la classe 13. Le résultat est bon car les côtés de l'objet ne sont pas très différents. C'est à dire des images de la classe 13 dans la base de modèle ont des points d'intérêt correct avec l'objet de test.

3.2.2 Exemple 2

Commande : *corr 1 database/test6/key_test/obj14__10.key database/test6/key_trainning*
Classe de l'image:
14 obj14__15.key(0.768293)
obj14__25.key(0.412698)
obj82__175.key(0.2)
obj10__240.key(0.142857)
obj14__340.key(0.134615)
On peut voir des images :

Figure 14: obj14__10.png

Figure 15: obj14__15.png

Figure 16: obj14__25.png

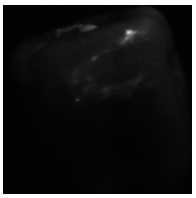

Figure 17: obj82__175.png

Figure 18: obj10__240.png

Figure 19: obj14__340.png

Explication :
Cet exemple est de faire la correspondance l'image *obj14__10.pgm* avec la base d'apprentissage. Le programme peut classifie exactement l'objet dans la classe correspondance, par contre, il y a deux images qui ne sont pas dans la même classe avec l'objet de test mais le programme les donne comme le résultat de *20%* et *14.2857%*. Une question posée est que pourquoi ces deux objets existent dans la liste de 5 objets les plus corrects ? La

répond est que l'objet *obj82__175.png* ne contient que 5 descripteurs. Donc, quand un descripteur correct, le résultat est déjà *20%*. C'est la même principe pour l'objet *obj10__340.png*, il ne contient que 7 descripteurs, quand il y a 1 descripteur correct, le résultat est de *14.2857%*.

Par rapport ce résultat, le problème posé c'est que peut être le programme va se tromper si la base de modèle n'est pas assez de rotation. En fait, le résultat n'est pas bon car j'ai pris seulement 1250 images pour la base de modèle. Alors, chaque classe d'objet, il n'y a que quelques rotations. Si les côtés de l'objet sont beaucoup différentes, il risque de poser des faux. Maintenant, je test avec la base entière :

Commande :
corr 1 database/kt/obj14__10.key database/key 14
obj14__10.key(1)
obj14__15.key(0.768293)
obj14__20.key(0.677966)
obj14__0.key(0.5)
obj14__25.key(0.412698)

Donc, le résultat est exact car les objets dans la liste de 5 objet les plus corrects ont la rotation presque comme l'objet de test.

3.3 Matrice de confusion entre deux bases d'images

Après des tests je trouve que le programme est facile de tromper quand la base d'apprentissage contient des images qui ont seulement quelques descripteurs. A partir de la formule (2), on peut expliquer ce problème. C'est quand le nombre de descripteur est moins, l'image de test est facile d'être classé dans la classe de cette image. Exemple : A = B/C, quand B ne change pas, C est moins, A va être élevé. Donc, pour améliorer le résultat de classifier des images, j'ai essayé de filtrer (supprimer) des images ayant le nombre de descripteur moins de 5.

3.3.1 Version avant le filtrage

Commande : *corr 2 database/test9/key_test database/test9/key_trainning*

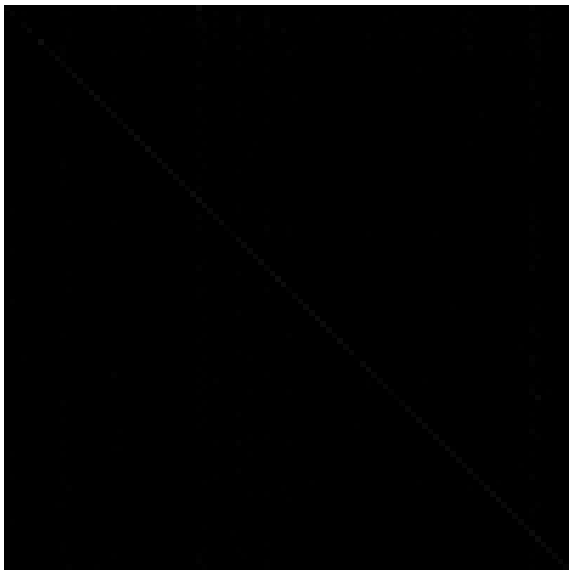

Figure 20: Matrice de confusion

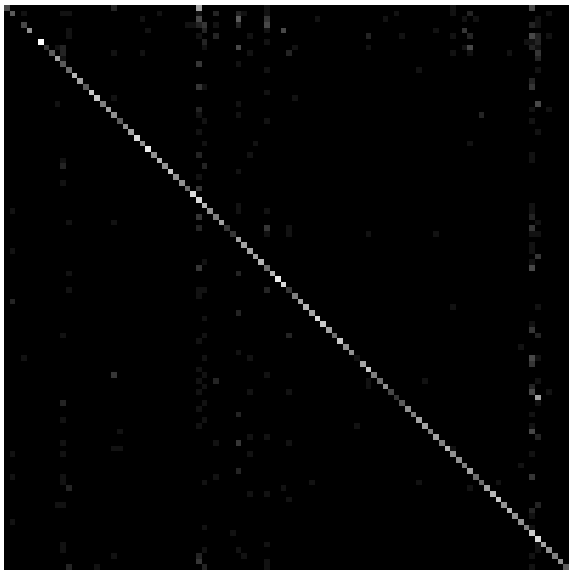

Figure 21: Après la modification appliquée la formule (3)

3.3.2 Version après le filtrage

Commande : *corr 2 database/test10/key_test database/test10/key_trainning*

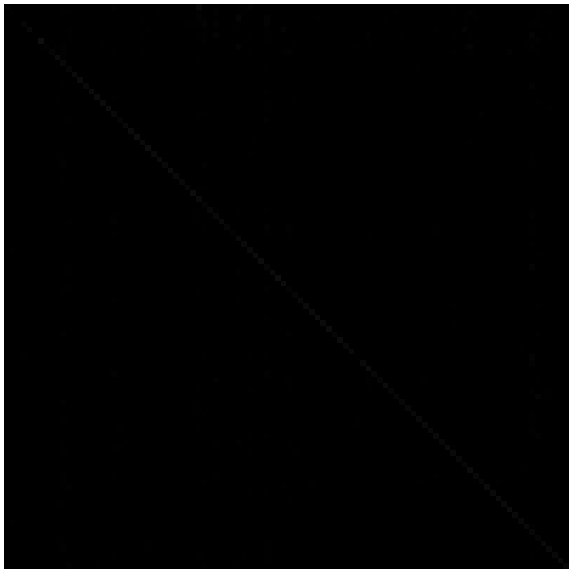

Figure 22: Matrice de confusion

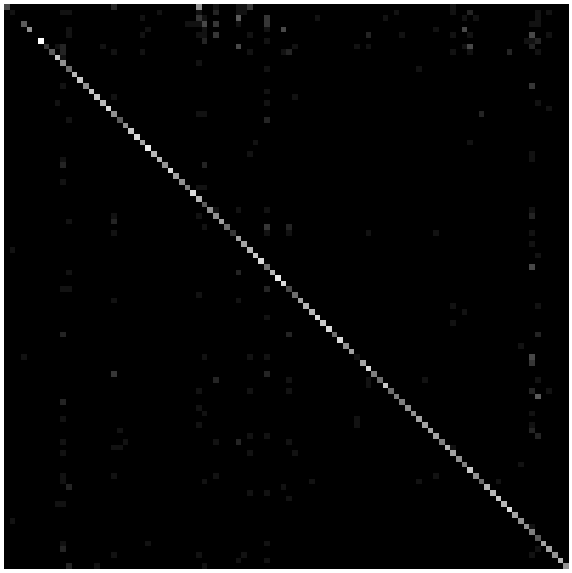

Figure 23: Après la modification appliquée la formule (3)

Explication :

En fait, l'image de gauche est la matrice de confusion, mais c'est difficile pour voir, j'ai donc appliqué la formule (3) pour créer l'image de droit qui sert à être facile de voir.

Dans ces deux image ci-dessus, on voit facilement la ligne en blanc qui traverse chaque image, cette ligne est construite par les éléments corrects entre la base de test et la base de modèle.

On trouve que la version après avoir filtré de la matrice de confusion est moins de faux. Les éléments supérieurs à 0 en dehors de la diagonale sont moins. Pour voir plus clairement, on trace une partie de la matrice de confusion avant et après le filtrage afin de comparer.

3.3.3 Deux versions d'une petite partie de matrice de confusion

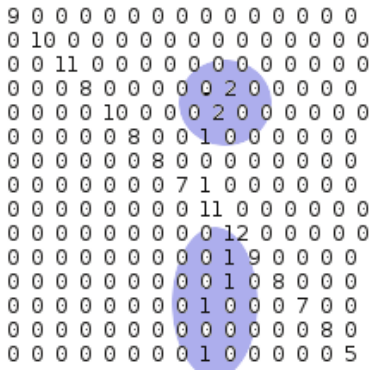

Figure 24: Version avant le filtrage

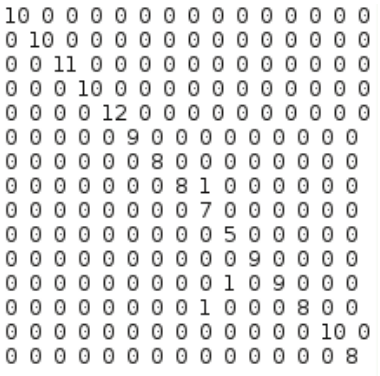

Figure 25: Après la modification appliquée la formule (3)

Explication :
En fait, ce n'est pas le résultat parfait. Si le résultat est parfait, tous les éléments en dehors de la diagonale doivent égale à 0. Ce résultat n'est pas parfait car je ne peux pas tester avec la base d'apprentissage entière. Chaque classe d'objet, j'ai supprimé beaucoup d'objets. Alors, ce n'est pas assez de rotation nécessaire pour chaque objet. Quand on n'est pas suffisant de rotation nécessaire, quand l'objet de test est dans la rotation manquante, c'est risque de poser des faux.

Pour expliquer ce que j'ai dit, on voit le résultat de la matrice de confusion avec seulement 10 classes, mais chaque classe, je ne supprime aucun objet (A cause du temps d'exécuter, mon ordinateur est très chaux, je ne peux pas tester avec la base entière avec 100 classe).
Commande : *corr 2 database/test13/key10 database/test13/key10_trainning*

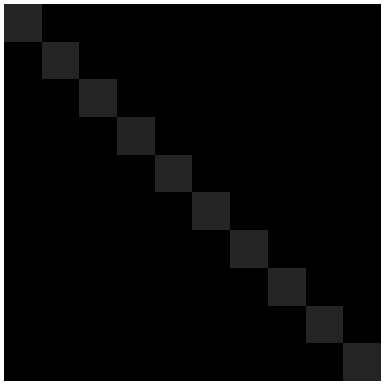

Figure 26: Matrice de confusion

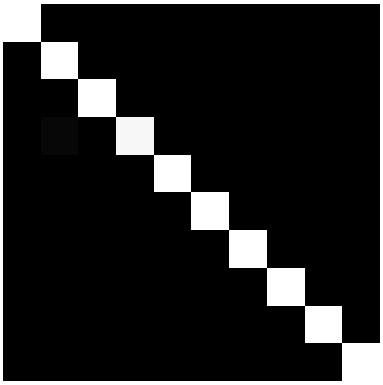

Figure 27: Après la modification appliquée la formule (3)

Ce résultat explique qu'avec la rotation suffisant, le résultat est très bon. La ligne explique que la classification du programme est exacte.

4 Conclusion

En conclusion, je trouve que la méthode de correspondance n'est pas très bonne, il est facile de se tromper quand la base de modèle contient des objets qui ont seulement quelques descripteurs. Par exemple, un objet a seulement 1 descripteur, et ce descripteur correct avec un descripteur de l'objet de test, alors, la correspondance sera de *100%*. Il me semble mieux s'il y a une méthode qui ne dépend pas le nombre de descripteur par exemple.

Pour cette méthode, il est aussi important de choisir le seuil, il nous permet de laisser des points moins intérêts ou prendre des points intérêts.

References

[1] NGUYEN Thi Oanh, *TP2 : Détection de la peau*. IFI, Hanoi, Vietnam, 2013.

[2] David Lowe *Demo Software: SIFT Keypoint Detector*. http://www.cs.ubc.ca/ lowe/keypoints/