

# Vision par Ordinateur

## TP 4 : Vision 3D

NGUYEN Van Tho

21 décembre 2013

### 1 Fonctionnement du programme

Dans ce TP, je vis à mettre en correspondance deux images stéréoscopiques. En fait, le travail de mettre en correspondance est divisé en deux étapes. D'abord, les points d'intérêts de deux images sont calculés. En suite, les points correspondants sont calculés à partir de ces points. Pour la deuxième étape, j'ai implémenté deux algorithmes : un algorithme heuristique (proposé dans l'annonce du TP) et un algorithme basé sur l'algorithme RANSAC.

#### 1.1 Algorithme heuristique

Après avoir eu des points SIFT de toutes les deux images, les points les plus similaires entre deux images sont calculés. Cependant, il y a beaucoup des mauvaises correspondances. J'utilise donc un algorithme heuristique pour enlever ces points :

- Trouver la distance minimum
- Effacer toutes les correspondances dont la distance est plus importante que distance minimum multiplié un seuil

```
if correspondance.distance > seuil * min_distance then
    delete correspondance
endif
```

#### 1.2 Algorithme basant sur la matrice fondamentale et l'algorithme RANSAC

Bien que l'algorithme rende des résultats assez bien, trouver un bon seuil pour toutes les images est infaisable. En effet, un seuil qui donne un bon résultat pour une paire d'image stéréoscopique peut donner un très mauvais résultat pour une autre paire. Par exemple, il y n'a pas beaucoup de correspondance (si le seuil est petit) ou beaucoup de mauvaises correspondances (si le seuil est grand). Pour résoudre ce problème, j'utilise la fonction "findFundamentalMat" offerte par OpenCV. Cette fonction permet de trouver la matrice fondamentale. En utilisant l'algorithme RANSAC, elle peut déterminer les points qui sont proches aux droites épipolaires.

Avant d'appliquer cette fonction, j'applique la correspondance symétrique. Autrement dit, les correspondances sont gardées seulement quand il y a une correspondance inverse, d'un point de l'image à droit à un point de l'image de gauche.

### 1.3 Commande pour lancer le programme

```
./matching image-left image-right
```

Où :

- image-left est l'image à gauche
- image-right est l'image à droit

Les images de résultats sont stockés dans le répertoire "output"

### 1.4 Compiler le programme sous linux

```
$make
```

## 2 Expérimentation et Résultat

### 2.1 Détection du mouvement

Pour expérimenter ce programme, j'utilise 50 paires d'image stéréoscopique trouvés sur un site de Nvidia "3dvisionlive.com". Les figures ci-dessous sont les résultats :



FIGURE 1 – Première ligne : Image gauche et image droite

Deuxième ligne : Tracés des correspondances de résultat de l'algorithme heuristique avec seuil = 3 fois min distance (386 correspondances)

et seuil = 5 fois min distance (955 correspondances)

Troisième ligne : Tracés des correspondances de l'algorithme basant sur matrice fondamentale et RANSAC (1160 correspondances)



FIGURE 2 – Première ligne : Image gauche et image droite

Deuxième ligne : Tracés des correspondances de résultat de l'algorithme heuristique avec seuil = 3 fois min distance (231 correspondances) et seuil = 5 fois min distance (810 correspondances)

Troisième ligne : Tracés des correspondances de l'algorithme basant sur matrice fondamentale et RANSAC (1576 correspondances)



FIGURE 3 – Première ligne : Image gauche et image droite

Deuxième ligne : Tracés des correspondances de résultat de l'algorithme heuristique avec seuil = 3 fois min distance (10 correspondances)

et seuil = 5 fois min distance (73 correspondances)

Troisième ligne : Tracés des correspondances de l'algorithme basant sur matrice fondamentale et RANSAC (2173 correspondances)

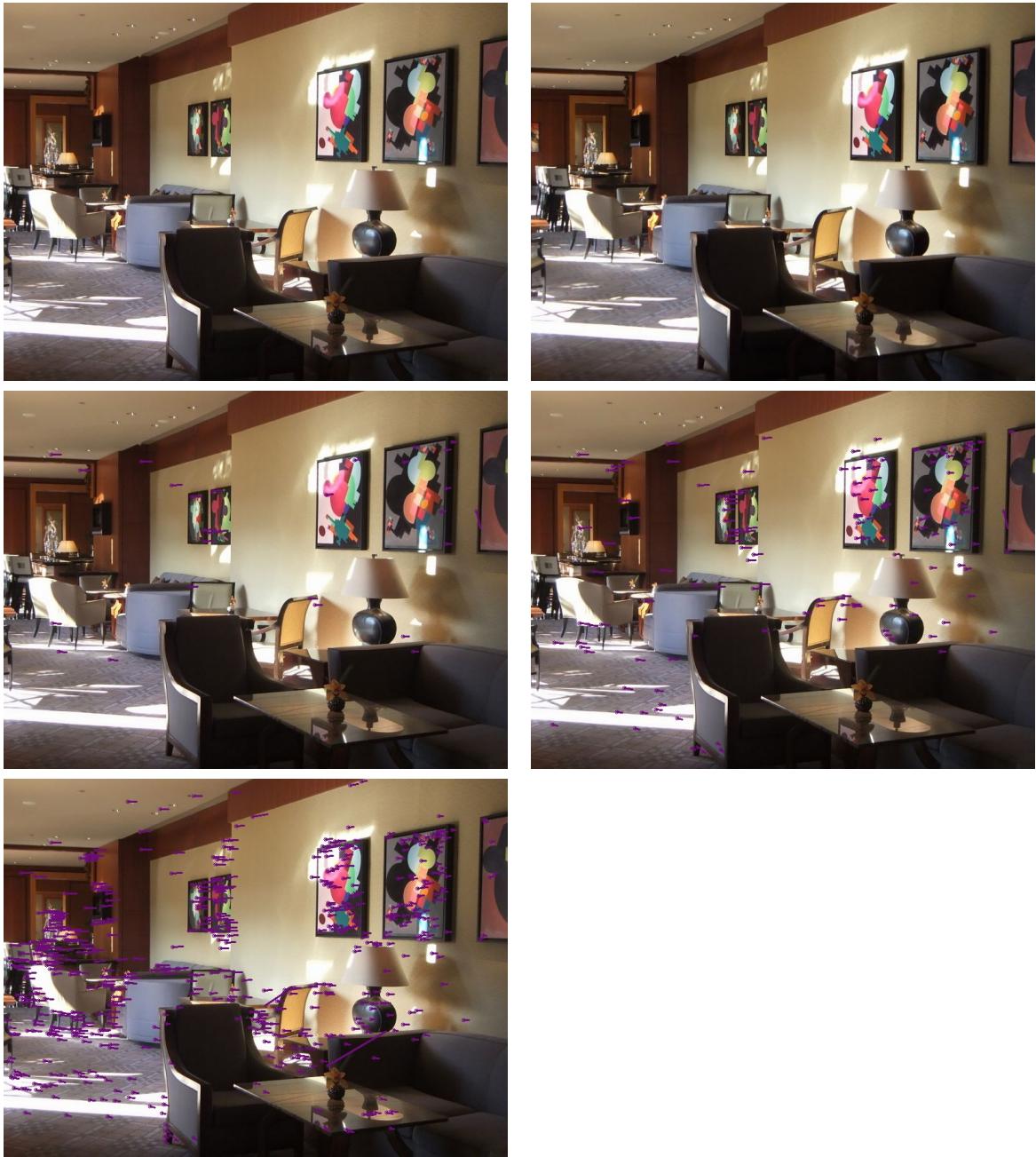


FIGURE 4 – Première ligne : Image gauche et image droite

Deuxième ligne : Tracés des correspondances de résultat de l'algorithme heuristique avec seuil = 3 fois min distance (28 correspondances)

et seuil = 5 fois min distance (158 correspondances)

Troisième ligne : Tracés des correspondances de l'algorithme basant sur matrice fondamentale et RANSAC (658 correspondances)

### 3 Réponses aux questions

À partir de ces résultats, on peut reconstruire les images en 3D. En fait, on peut calculer la matrice fondamentale à partir des points correspondantes trouvés par la méthode heuristique. Cependant, avec la deuxième méthode la matrice fondamentale est rendue par la fonction “findFundamentalMat”.

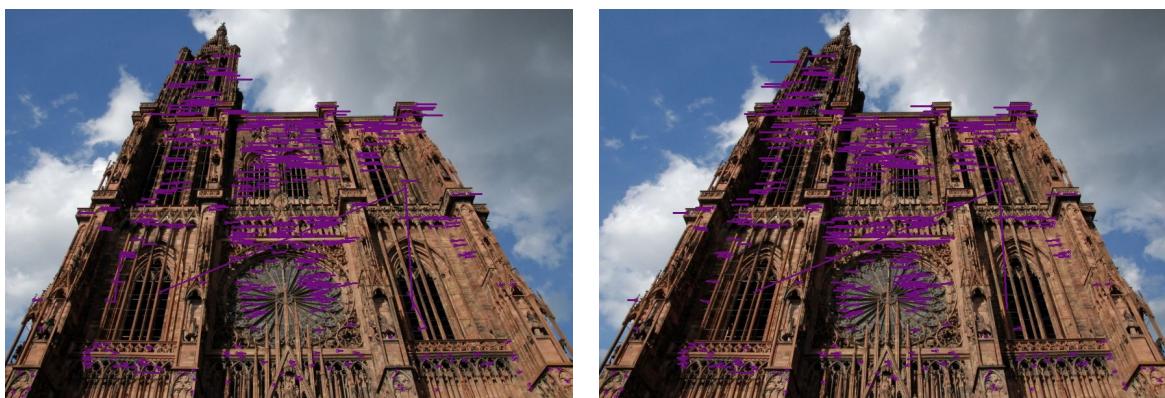
Voir les résultats, on connaît la profondeur des points d'intérêts. En effet, les points avec un long droit ont une profondeur plus petit que ceux qui avec un court droit.

Dans la partie d'expérimentation je mets les résultats de 4 paires d'image stéréoscopique. On constate que la méthode heuristique ne rend pas toujours des bons résultats (Figure 3 et 4). Comme nous avons discuté dans la partie de fonctionnement du programme, il est difficile de trouver un seuil qui rejette les mauvaises correspondances et en même temps garder les bons correspondances. Par exemple, dans exemple 3 (figure 3), quand on utilise un seuil qui est 3 fois plus grand que la distance minimum, le nombre de correspondances rendues est 10. Quand ce seuil est mis à 5 fois plus grand que la distance minimum, le nombre de correspondances est 73. Dans ce cas, l'augmentation du seuil rend un résultat avec beaucoup de correspondances et la qualité de correspondances est toujours bonne. Par contre, on voit dans la figure 1, l'augmentation du seuil rend plus de correspondances mais avec beaucoup d'erreur.

La deuxième méthode (utilisation la fonction findFundamentalMat avec l'algorithme RANSAC) rend des résultats toujours bons. Dans ce TP j'ai expérimenté 50 paires d'image stéréoscopique et dans la plupart de cas cette méthode rend des résultats avec plus de 1000 points correspondants. La qualité de correspondances sont très bonnes (les tracés sont parallèles)

La méthode heuristique marche bien dans le cas dont les images ont beaucoup de points d'intérêts. Autrement dit, les images qui ont beaucoup coins, images de haute fréquence tel que la forêt dans mes exemples. En revanche, les images qui sont assez homogènes, on n'obtient pas beaucoup de correspondances. Ce sont des images de basse fréquence tel que le troisième exemple.

Dans ce TP, j'ai aussi expérimenté avec les inversement des images gauches et droites. Le résultat est le même. Cependant, en terme de visualisation on voit des changements, on peut considérer qu'on change la position de sa vue de gauche à droit. Autrement dit, on affiche les différences dans différentes images mais les changements sont le même. C'est pourquoi en terme de l'information 3D on ne perd rien. Voir la figure ci-dessous pour plus détaillé.



À partir de ces résultats de la méthode heuristique, on peut retrouver les paramètres de la géométrie épipolaire. En fait, on doit trouver la matrice fondamentale. Cette matrice

peut être calculée en utilisant 7 ou plus meilleures correspondances. A partir de la matrice fondamentale on peut calculer les centres de projection de deux caméra et les épipôles, les droites épipolaires.

*Pour être concise, je ne mets pas tous les résultats de mon expérimentation. Par contre, vous pouvez le voir sur ce lien <https://github.com/thonguyen/cv/tree/master/tp4>, les résultats complets.*

## 4 Conclusion et Discussion

Dans ce TP, j'ai implémenté deux méthodes qui permettent de trouver les correspondances entre 2 images stéréoscopiques. J'ai expérimenté ces méthodes avec 50 paires d'image et avec quelques seuils différences.

La méthode basant sur l'algorithme RANSAC est meilleure que celle basant sur les heuristiques. Cependant, on peut obtenir un résultat assez bon avec la méthode heuristique en choisissant un bon seuil.

À partir des correspondances on peut trouver les paramètres de la géométrie épipolaire. Ces paramètres sont les informations fondamentales pour la reconstruction 3D.