Traitement d'images

Travaux pratiques - séance 2

- Avant XX 2018 minuit, envoyez par e-mail la version finale des sources + un compte-rendu où vous détaillez bien les problèmes rencontrés, en justifiant vos solutions. Le sujet du message sera [TI5A]nom-prenom-final2-CR. Evitez de joindre les images de test que j'ai mises à votre disposition au début.

L'objectif du TP est d'obtenir une reconstruction panoramique à partir de plusieurs images.

Exercice 1 — Detecteur FAST

1. Implémentez le détecteur FAST vu en cours. N'utilisez pas la stratégie basée sur la variation de l'entropie (détaillée dans l'article), mais faites un simple parcours séquentiel de la circonférence autour d'un pixel pour décider s'il est un coin. La suppression non-maximale est optionnelle (bonus). Visualisez le résultat de votre détecteur sur l'image synthétique et sur les images naturelles à l'aide d'une fonction `showCorners` que vous implémenterez.
   De même, comparez votre détecteur FAST avec celui déjà implémenté en OpenCV, dans la fonction `FAST(InputArray image, vector keypoints, int threshold, bool nonmaxSuppression=true )`. Cela veut dire :
   - vérifier que les résultats des détections sont identiques (nombre et localisation de points d'intérêt) pour les mêmes paramètres de détection. Les éventuelles différences sont à investiguer car elles sont dues très probablement à des erreurs d'implémentation de votre coté.
   - comparer les temps de calcul nécessaires pour les deux détecteurs. Bonus : proposer des améliorations (relativement simples) qui diminuent le temps de calcul de votre implémentation.

Exercice 2 — Appariement

1. En faisant tourner le détecteur sur les images `set1-1.png` et `set1-2.png`, vous obtenez deux ensembles de détections qu'on doit associer. Pour faire cela, il faut calculer un score entre une détection de la première image et chaque détection de la deuxième image. La solution suggérée est d'utiliser un score de type sum of squared distances (SSD) entre deux petites régions carrées (patchs) p_1 et p_2 centrées sur les coins c_1 et c_2 qu'on analyse :

$$SSD(c_1, c_2) = \sum_{p_1 \in R_1, p_2 \in R_2, p_1 \leftrightarrow p_2} \Big[I_1(p_1) - I_2(p_2) \Big]^2$$

   ou $p_1 \leftrightarrow p_2$ signifie que p_1 et p_2 sont pixels correspondants dans les deux patchs (situés dans la même location par rapport aux coins c_1 et c_2).
   En réalité, pour être robuste aux variations globales d'intensité entre les deux images, on préfère de retirer de chaque coté la moyenne du patch respectif, pour calculer un score ZMSSD (zero-mean sum of squared distances) :

$$ZMSSD(c_1, c_2) = \sum_{p_1 \in R_1, p_2 \in R_2, p_1 \leftrightarrow p_2} \Big[(I_1(p_1) - \mu_1) - (I_2(p_2) - \mu_2) \Big]^2$$

   Dans ce cas, plus le score est petit, mieux c'est, et on va apparier un coin avec le correspondant de l'autre image avec le score le plus petit. Néanmoins, il faudra choisir un seuil (la valeur est à justifier)

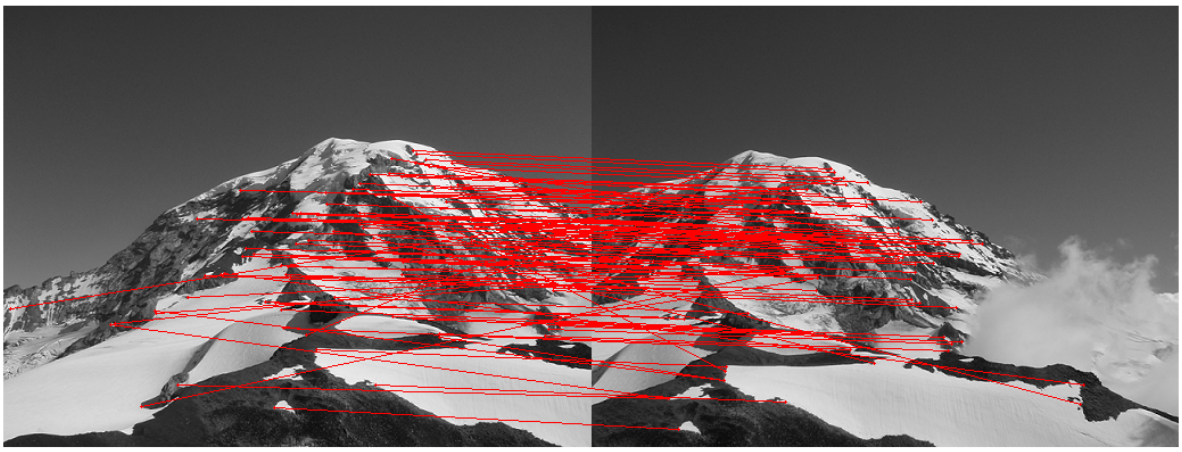

FIGURE 1 – Appariement de coins entre deux images.

au dessus duquel on refuse même le meilleur appariement. C'est le cas par exemple quand un point de l'image 1 n'apparaît pas dans l'image 2, et on évite ainsi de l'apparier de manière erronée. Si vous avez trop de fausses appariements, vous pouvez utiliser deux autres stratégies plus fines :

— married matching : vous appariez image 1 vers image 2, puis image 2 vers image 1 et vous ne gardez que les points qui se sont choisis réciproquement

— vous regardez le meilleur score et le second meilleur ; pour accepter l'appariement il faut que le meilleur score soit bien meilleur que le deuxième

Une fois que vous avez une liste d'appariements, vous pouvez la visualiser avec une fonction `showMatches` que vous implémenterez (Figure 1)

Exercice 3 — Estimation de homographie (1.5 points)

1. Utilisez une stratégie de type RANSAC pour estimer l'homographie entre les images `set1-1.png` et `set1-2.png`. Choisissez au hasard quatre appariements et

   1. formez la matrice A de taille 8×9 vue en cours
   2. calculez la décomposition SVD : $[U \;\; S \;\; V] = svd(A)$
   3. reconstruisez la matrice H à partir de la dernière colonne de V qui contient toutes les valeurs $h_{00}, \ldots, h_{22}$

   (Pour être sûr que votre implémentation est correcte, vérifiez que les quatre points de l'image 1 sont projetés exactement dans les quatre points correspondants indiquées par l'appariement.)
   Cette opération est répétée T fois, et chaque fois vous devez calculer combien d'autres appariements respectent l'homographie calculée (en appliquant un seuil sur la norme de $Hx - x'$).
   Au bout de T essais, vous prenez la H qui a fourni le plus grand nombre n d'inliers, et vous estimez de nouveau H mais de manière plus précise, en construisant une matrice A de taille $2n \times 9$.

Exercice 4 — Reconstruction panoramique

1. Utilisez la matrice d'homographie pour étendre les bords d'une image en échantillonnant dans l'image correspondante. A une coordonnée 2D entière, la matrice H fera correspondre une coordonnée 2D fractionnaire. Vous pouvez effectuer un arrondi au plus proche pixel pour faire l'échantillonnage, mais c'est vivement conseillé de faire une interpolation bilinéaire.
2. (Bonus) Faites la reconstruction en utilisant plusieurs images.