ENSTA-Paris 2e année - Cours MI204
TP1 : Détection et Appariement de Points Caractéristiques

L'objectif de ce TP est de (1) se familiariser avec la bibliothèque de traitement d'images OpenCV sous Python, et (2) expérimenter, critiquer et mettre en œuvre différentes techniques de *représentation* pour l'appariement de structures locales dans les images, en suivant les différentes étapes de la problématique :

- DÉTECTEUR : Comment réduire le support de la représentation.
- DESCRIPTEUR : Quelle information attacher à chaque point du support.
- MÉTRIQUE : Quelle mesure utiliser pour apparier des points.
- RECHERCHE : Comment coder et parcourir l'espace des points d'un modèle.

1 Préliminaires informatiques

Une base de code est fournie, qui fonctionne avec une installation standard d'OpenCV (sans opencv-contrib, version testée 4.1.0) et Python3. L'archive contenant cette base peut être téléchargée au lien suivant :
`http://perso.ensta-paris.fr/~manzaner/Cours/MI204/TP1_Features.zip`

Tous les codes fournis sont des scripts Python3 éditables et exécutables en lançant la commande, par exemple :
```
$ python3 Convolutions.py
```

Certains scripts nécessite des arguments, voir le code Python pour plus de détails.

On trouvera quelques paires d'images de test dans le même répertoire que le logiciel de TP :
`http://perso.ensta-paris.fr/~manzaner/Cours/MI204/Image_Pairs.zip`

OpenCV est une bibliothèque très utilisée en analyse d'images et on pourra consulter avec profit de nombreux guides et tutoriels en ligne (attention toutefois aux versions de Python et d'OpenCV fournies dans les exemples). On se limitera ici à mentionner les sites officiels pour le guide de référence des fonctions :
`https://docs.opencv.org/4.1.0/`
ainsi que le tutoriel :
`https://opencv-python-tutroals.readthedocs.io/en/latest/`

2 Convolutions

Q1 Expérimenter le code de convolution fourni en exemple dans *Convolutions.py*. Observer la différence entre le calcul direct par balayage du tableau 2d et le calcul utilisant la fonction *filter2d* d'OpenCV. Expliquer pourquoi le noyau de convolution fourni en exemple réalise un réhaussement de contraste par rapport à l'image originale.

Q2 Modifier le code pour calculer la convolution qui approxime la dérivée partielle $I_x = \frac{\partial I}{\partial x}$, puis le module du gradient $||\nabla I|| = \sqrt{I_x^2 + I_y^2}$, et afficher les images correspondantes. Quelles précautions doivent être prises pour obtenir un affichage correct ?

3 Détecteurs

Q3 Compléter le code fourni dans le script *Harris.py* pour calculer la fonction d'intérêt de Harris (à une seule échelle), et les points d'intérêt correspondants. Commenter les résultats obtenus avec votre détecteur de Harris et l'effet des paramètres utilisés, en particulier la taille de la fenêtre de sommation. Comment peut-on réaliser ce calcul sur plusieurs échelles ?

Q4 Expérimenter et comparer les deux détecteurs ORB et KAZE en lançant le script *Features_Detect.py*. Expliquer le principe de chacun de ces détecteurs, les principaux paramètres propres à chaque détecteur et leur effet sur la détection. Comment peut-on visuellement évaluer la répétabilité de chaque détecteur appliqué sur une paire d'images ?

4 Descripteurs et Appariement

Q5 Expliquer la stratégie d'appariement de points d'intérêt réalisée dans le script *Features_Match.py*. Pourquoi les distances utilisées sont-elles différentes pour les deux descripteurs ?

Q6 Proposer une stratégie pour évaluer quantitativement le meilleur appariement en déformant une image avec une transformation géométrique connue (on pourra utiliser une fonction telle que *cv2.warpAffine*).