

# ENSTA-Paris 3e année - Cours ROB317

## TP1 : Détection et Appariement de Points

### Caractéristiques

Septembre 2021

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.

Vous devez fournir un rapport, au format pdf, en justifiant autant que possible vos réponses avec des expériences personnelles.

## 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 :

[https://perso.ensta-paris.fr/~manzaner/Cours/ROB317/TP1\\_Features.zip](https://perso.ensta-paris.fr/~manzaner/Cours/ROB317/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 nécessitent 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 :

[https://perso.ensta-paris.fr/~manzaner/Cours/ROB317/TP1\\_Features\\_Images.zip](https://perso.ensta-paris.fr/~manzaner/Cours/ROB317/TP1_Features_Images.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://opencv24-python-tutorials.readthedocs.io/en/stable/index.html>

## 2 Format d'images et 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. Déchiffrer les fonctions OpenCV utilisées pour la lecture et la copie d'images, ainsi que la fonction Matplotlib utilisée pour l'affichage.

Q2 EXPLIQUER pourquoi le noyau de convolution fourni en exemple réalise un réhaussement de contraste par rapport à l'image originale. Pour cela, montrer l'effet du calcul d'une convolution 1d approximant une dérivée seconde sur une fonction échelon (Heaviside), et utiliser les propriétés de la convolution.

Q3 MODIFIER le code pour calculer, avec les deux méthodes, les convolutions qui approximent les dérivées secondes  $I_{xx} = \frac{\partial^2 I}{\partial x^2}$ ,  $I_{xy} = \frac{\partial^2 I}{\partial x \partial y}$ , et  $I_{yy} = \frac{\partial^2 I}{\partial y^2}$ . Quelles précautions doivent être prises pour obtenir un affichage correct des valeurs de dérivées ?

## 3 Détecteurs

Q4 COMPLÉTER le code fourni dans le script *DoH.py* pour calculer la fonction d'intérêt correspondant au déterminant de la matrice hessienne, et les points d'intérêt correspondants. Quelle est l'interprétation géométrique de ces points d'intérêt ? Expliquer comment le code fourni, qui utilise la dilatation morphologique, permet de calculer les maxima locaux de la fonction d'intérêt *Theta*.

Q5 COMMENTER les résultats obtenus avec votre détecteur DoH. Donner deux méthodes permettant de réaliser ce calcul sur plusieurs échelles. Pourquoi est-il intéressant d'étendre la notion de maxima locaux pour faire en sorte que deux points d'intérêt soient toujours distants d'au moins  $r$  pixels, et comment peut-on réaliser cette fonction avec le code fourni ?

Q6 EXPÉRIMENTER et comparer les deux détecteurs ORB et KAZE en lançant le script *Features\_Detect.py*. Rappeler le principe de chacun de ces détecteurs. Expliquer 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

Q7 EXPLIQUER brièvement le principe des descripteurs attachés aux points ORB et ceux attachés aux points KAZE. Quelles propriétés des détecteurs et/ou des descripteurs (bien distinguer les deux aspects dans la réponse), permettent de rendre l'appariement invariant par changement d'échelles et invariant par rotation ?

Q8 EXPLIQUER et comparer qualitativement les performances des trois stratégies d'appariement de points d'intérêt réalisées dans les trois scripts *Features\_Match\_CrossCheck.py*, *Features\_Match\_RatioTest.py* et *Features\_Match\_FLANN.py*. Expliquer pourquoi les distances utilisées sont différentes pour les deux descripteurs. Expliquer pourquoi la stratégie FLANN (et dans une moindre mesure la stratégie RatioTest) ne fonctionne pas bien avec les points ORB.

Q9 PROPOSER une stratégie pour évaluer quantitativement la qualité des appariements en déformant une image avec une transformation géométrique connue (on pourra utiliser une fonction telle que *cv2.warpAffine*).