Dans ce TP, nous proposons d'implémenter et d'étudier le fonctionnement (influence des paramètres, limitations, variantes) d'un détecteur et d'un descripteur dans l'optique de réaliser une application de panorama. L'implémentation pourra être faite en Matlab ou en Python.
N'hésitez pas à vous aider des transparents de cours ou des articles scientifiques décrivant les différentes méthodes pour avoir plus d'informations sur les paramètres ou les méthodes.

Travail de rendu demandé : à l'issu des 2 séances de TP, vous devrez envoyer une archive `zip` qui contiendra :
- Un rapport **au format pdf ou notebook Jupyter** contenant votre analyse de l'approche, obtenue par les tests que vous aurez effectués. Le nombre de pages n'est pas un critère de jugement, tant que le rapport contient une analyse aboutie montrant que vous avez compris l'approche. Plus l'analyse sera aboutie, meilleure sera la note !
- L'ensemble de vos sources (commentées !) avec un petit script permettant de lancer facilement votre programme.

Tout plagia de code source ou de rapport sera sévèrement sanctionné.

EXERCICE 1: DÉTECTION DE POINTS D'INTÉRÊT

L'objectif de cet exercice est d'implémenter et d'étudier deux algorithmes de détection de points d'intérêt.

1. **Détecteur de Harris :**
   (a) Implémenter le détecteur de Harris en utilisant une fenêtre de pondération rectangulaire puis gaussienne.
   (b) Implémenter également une suppression des non-maxima locaux afin d'éviter les "amas" de points d'intérêt.
   (c) Étudier l'influence du type de fenêtre de pondération, de sa taille ainsi que du paramètre k dans la détection de point d'intérêt.
   (d) Utiliser les fonctions existantes de Matlab ou Python pour effectuer une rotation de l'image. Les points d'intérêt détectés sur cette image tournée sont-ils les mêmes que ceux de l'image originale ? Quel paramètre faut-il adapter pour que cela soit le cas ?
2. **Détecteur FAST :**
   (a) Implémenter le détecteur FAST.
   Pour vérifier si n points consécutifs sont à 1, vous pouvez soit utiliser la fonction `circshift` de Matlab (ou son équivalent en Python), soit convoluer votre vecteur par un filtre de taille n rempli de 1.
   (b) Implémenter la suppression des non-maxima locaux. Quelle score utilisé pour cette suppression ?
   (c) Utiliser les fonctions existantes de Matlab ou Python pour effectuer une rotation de l'image. Les points d'intérêt détectés sur cette image tournée sont-ils les mêmes que ceux de l'image originale ? Quel paramètre faut-il adapter pour que cela soit le cas ?
3. **Comparaison :**
   (a) Comparer les points d'intérêt détectés par le détecteur de Harris et par le détecteur FAST. Quelle(s) différence(s) obeservez-vous ?
   (b) **(Bonus)** Utiliser les fonctions existantes de Matlab ou de Python pour comparer vos détecteurs à ceux déjà implémentés (Harris, FAST ou d'autres)

EXERCICE 2: DESCRIPTION ET MATCHINGS DES POINTS D'INTÉRÊT

L'objectif de cet exercice est d'implémenter et d'étudier une méthode de description et de matching des points d'intérêt détectés dans l'exercice précédent.

1. **Description simple :** Utilisation d'un bloc d'intensité
   (a) Pour chaque point d'intérêt, récupérer un bloc de pixels autour de celui-ci et transformer le en vecteur.
   (b) Implémenter une méthode de matching capable de gérer les mauvais matchs (soit via un appariement croisé, soit par comparaison des distances)
   (c) Etudier l'influence de la taille du bloc et des métriques de comparaison utilisées sur la qualité de l'appariement
   Pour les métriques, on utilisera la corrélation (et on étudiera l'intérêt de sa normalisation) ainsi qu'une autre métrique au choix.
   (d) Utiliser les fonctions existantes de Matlab ou Python pour effectuer une rotation de l'une des images. Les appariements obtenus sont-ils les mêmes que précédemment ?
   Implémenter une solution permettant de gérer ce problème
2. **Description "avancé"** (Bonus)
   (a) Implémenter le descripteur LBP et le combiner à votre descripteur FAST.
   (b) Étudier l'influence des différents paramètres de ce descripteur sur la qualité des appariements

(c) **(Bonus^2)** Utiliser les fonctions Matlab ou Python existantes pour comparer votre description à d'autres plus "évoluées" (*e.g.* SIFT).

EXERCICE 3: (BONUS) PANORAMA

L'objectif de cet exercice est d'utiliser les matchs trouvés dans l'exercice précédent pour fusionner les 2 images et former un panorama.

1. Implémenter l'algorithme de la DLT (direct linear transform). Vous pourrez vous aider des documents sur Moodle pour son implémentation.
2. Étudier l'influence de la normalisation des données sur le calcul de la DLT.
3. Utiliser la transformation fournie par la DLT pour déterminer la taille du panorama (transformation des 4 coins de l'image 2 vers le repère de l'image 1 par exemple) puis pour projeter l'image 2 (et l'interpoler) dans le repère de l'image 1 afin de former le panorama.