TP Stéréoscopie&Segmentation

Consignes

Ce TP sera évalué et est à rendre dans un dossier archivé (.zip, .rar, .tar.gz) nommé "PI_TP4_Nom_Prenom" avant le vendredi 27 octobre 23h59 par mail : **arthur.longuefosse@u-bordeaux.fr**

Dans ce TP, il vous est demandé de créer un Notebook Jupyter dans lequel vous coderez le ou les algorithmes demandés pour chaque partie. Le bon fonctionnement et **la propreté du code ainsi que ses justifications (sous forme de texte Notebook ou de commentaires) seront évalués**.

Vous pouvez réutiliser des fonctions codées dans les TPs précédents, en faisant attention à leur propreté. Vous pouvez également, si vous le souhaitez, créer des fichiers .py que vous importerez dans le Notebook principal. Ces fichiers seront à joindre dans l'archive et compteront dans l'évaluation.

Dans le dossier "images" se trouvent deux images d'une scène, vues par l'œil gauche (cat_l.jpg) et par l'œil droit (cat_r.jpg). Vous pouvez trouver d'autres couples similaires d'images en cherchant des images stéréoscopiques sur internet. Il est également possible d'utiliser des photos que vous avez vous-même capturées. Pour cela, prenez 2 photos d'une même scène, l'une en plaçant votre téléphone devant votre œil gauche et l'autre devant votre œil droit (sans bouger entre temps).

1. Anaglyphe

Il existe plusieurs méthode pour générer de la stéréoscopie à partir de plusieurs images. L'une de ces méthodes est l'anaglyphe, qui consiste à superposer les images en modifiant les couleurs d'affichage des deux images. Il est alors possible de voir les différents niveaux de profondeur de l'image à l'aide de lunettes avec des filtres colorés.

Implémentez une fonction anaglyphe prenant en entrée deux images gauche et droite, et retournant une nouvelle image anaglyphe dans laquelle l'image de gauche est vue en rouge et celle de droite en cyan (vert+bleu).

Deux versions sont demandées ; une avec des boucles **for** (parcours des pixels), et une sans boucle **for** (NumPy est votre ami).

Comparez les temps d'exécution de chaque implémentation.

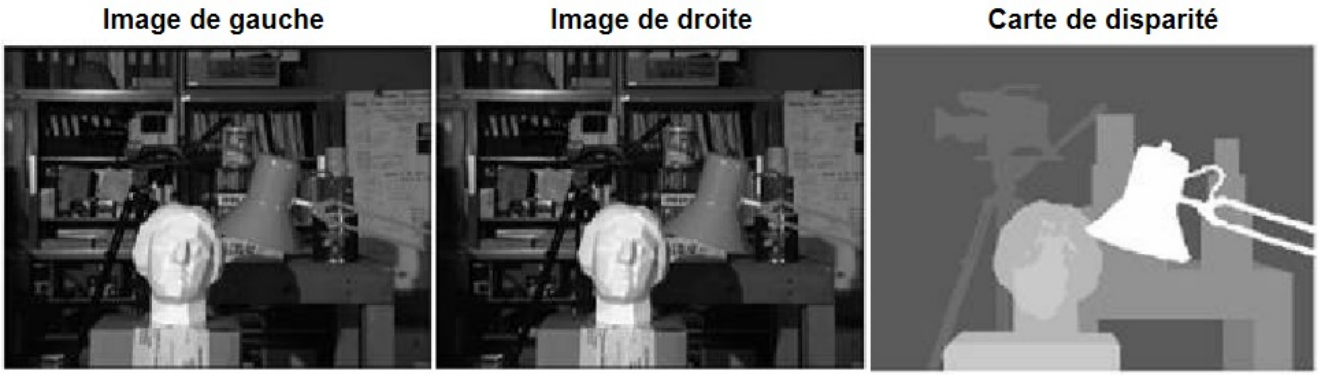
2. Carte de disparité

On cherche désormais à obtenir une carte de disparité de l'image de gauche. Une carte de disparité est une image dans laquelle chaque pixel contient une valeur du déplacement des pixels d'une image à l'autre. Il faut donc d'abord trouver, pour chaque pixel de la première image, quel est le pixel correspondant dans la deuxième image, puis calculer la distance entre les deux.

Il existe plusieurs types d'algorithmes pour cela. Les premiers correspondent aux détecteurs, que l'on a pu voir à plusieurs reprises lors des TPs précédents, tels que SIFT, SURF ou ORB (**O**riented FAST and **R**otated **B**RIEF). Les deuxièmes sont composés des Algorithmes de Block-Matching (ou BMA, voir https://en.wikipedia.org/wiki/Block-matching_algorithm). Les algorithmes utilisés par OpenCV pour un certain nombre de tâches de vision par ordinateur appartiennent à cette deuxième catégorie.

Pour obtenir la carte de disparité, vous pouvez utiliser les fonctions déjà installées dans OpenCV (particulièrement celles des algorithmes BMA). Vous aurez alors besoin de tester un certain nombre de méthodes différentes et de beaucoup modifier les paramètres afin d'obtenir un résultat convaincant (peu bruité, bords lisses...). *Attention, la plupart de ces algorithmes considèrent des images en niveaux de gris (1 canal) et non des images couleur.*

Afin d'affiner les résultats des différentes méthodes, il est possible d'appliquer un flou ou une succession d'opérations d'érosion et de dilatation (voir "fermeture morphologique" et "ouverture morphologique")



3. Segmentation

On cherche désormais à segmenter les différents objets de l'image (sur les images cat_l.jpg et cat_r.jpg, il s'agit donc du mur, du chat, du banc et du reste de l'image).

Il existe 3 types d'algorithmes de segmentation :

- Par pixel (par exemple en sélectionnant des seuils dans l'histogramme de l'image) ;
- Par voisinnage (bords/coins), par exemple avec une convolution (en utilisant un noyau approprié !) ou d'autres méthodes codées dans les TPs précédents. Des exemples sont trouvables sur cette page ;
- Par région, tel qu'avec des méthodes de watershed.

Codez au moins une méthode de segmentation. Appliquez cette segmentation sur l'image originale (de gauche) et affichez le résultat.

Améliorez ensuite la segmentation obtenue grâce la carte de disparité. À vous de décider de la méthode finale que vous utiliserez pour cela. *Note : Il est à nouveau possible d'affiner les résultats à l'aide de flous et d'ouvertures/fermetures morphologiques.*

Les différentes régions ainsi obtenues seront affichées sur une image finale, avec une couleur différente par région et l'arrière-plan en noir.