Tutoriel de Traitement d'Image – Partie II

Ce TP continue le TP précédent de traitement d'image. Vous pourrez fournir un compte rendu de TP sous la forme d'un tutoriel. Questions 3 et 4 : pour aller plus loin...

Partie I

1 Transformée de Fourier

1. Afficher la représentation de l'amplitude (magnitude) et de la phase de l'image en utilisant la fonction *fft2*.
2. Utilisez la fonction *ifft2* pour retrouver l'image de départ. Visualisez là.
3. Retrouvez le résultat présenté en cours : calculez la transformée de Fourier de deux images, récupérez les amplitudes et les phases de ces images. Inversez les amplitudes des deux images (mais pas les phases) et reconstituez les images obtenues par transformée de Fourier inverse. Visualisez les images de départ et les images finales obtenues. Qu'observez vous ?

2 Filtrage

2.1 Filtrage par convolution

1. Acquérir une image en nuance de gris et de résolution 256 * 256 (jpeg).
2. Appliquer un filtre passe-bas sur cette image (vous pouvez utiliser la fonction *convolution* que vous avez développée précédemment), avec la matrice de convolution suivante :

$$M_c = \begin{pmatrix} 1 & 3 & 1 \\ 3 & 5 & 3 \\ 1 & 3 & 1 \end{pmatrix} \tag{1}$$

3. Quel résultat produit le filtre ?
4. Mêmes questions pour le filtre pass-haut défini par la matrice de convolution suivante :

$$M_c = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix} \tag{2}$$

2.2 Filtrage par transformée de Fourier

Dans cet exercice vous prendrez une image de synthèse et une photographie de votre choix (pas trop grandes en nombre de pixels). Vous remarquerez que les fréquences basses dans le spectre de Fourier de l'image sont au centre de l'image.

1. Calculez la transformée de Fourier de votre image (utilisez *fft2*), vous obtenez *im_fft*.
2. Enlevez les fréquence hautes de votre image. Vous obtenez *im_fft_passe_bas*.
3. Appliquez la transformée de Fourier inverse sur *im_fft_passe_bas* et visualisez le résultat obtenu.
4. Comparez avec le filtre passe-bas obtenu dans à la question précédente.
5. Mêmes questions pour un filtre passe-haut (en enlevant les fréquences basses).
6. Qu'observez-vous ?

Partie II – à réaliser à la maison

3 Réduction de bruit et filtrage non linéaire

1. Ajouter du bruit *Salt & Pepper* de paramètre 0.05 (ou 0.1) dans votre image. Pour ce faire, vous fixerez un paramètre *saltValue* égal à 0.05 * 100 ou 0.1 * 100, et vous créerez une matrice de bruit (de 0 à saltValue) grâce à la fonction *randint*. Puis, dans l'image originale, vous remplacerez les pixels qui valent *saltValue* en des pixels blancs.
2. Développer une fonction *filtre_median* afin de corriger le bruit induit dans la question précédente.

4 Segmentation et comptage d'objet

Le seuillage d'image est une technique simple de binarisation d'image, elle consiste à transformer une image en une image dont les valeurs de pixels ne peuvent avoir que la valeur 255 ou 0. On parle alors d'une image binaire ou image en noir et blanc.

1. Acquérir une image couleur dont on souhaiterait compter les objets (par exemple grains de riz très espacés entre eux pour commencer)
2. Écrivez une fonction *seuillage* avec en entrée votre image et le niveau de seuillage souhaité.
3. Écrivez un programme permettant la segmentation de votre image en utilisant la fonction *seuillage* précédemment écrite. Les étapes de votre programme seront les suivantes :
   - Décomposer l'image en trois images correspondant à ses composantes de couleur (Rouge, Vert et Bleu)
   - Seuiller ces images et afficher les ; par exemple pour un niveau de quantification de 255 vous pourrez prendre une valeur de seuil de 150.
   - Afficher l'image *somme des seuillages*
   - Récupérer l'image complémentaire de votre image somme

- Extraire les caractéristiques avec les fonctions *label* et *regionprops* (from skimage.measure import label, regionprops)
- Compter les objets en utilisant le résultat de l'extraction des caractéristiques (encadrement des objets et affichage du compte sur l'image).