M4201Cin TP 3

Filtres de convolution

31 janvier 2018 – B. COLOMBEL

Les réponses aux questios des exercices 1 à 6 seront écrites dans l'éditeur Scinote dans un seul fichier.
Les réponses autres que des fonctions Scilab seront commentées. Il sera enregistré sous le nom :

`Nom_Prenom-TP3.sci`

et déposé dans le dossier prévu à cet effet dans AMETICE (module M3201Cin, TP3)

Si le module SIVP de **Scilab** est bien installé, vous verrez au démarrage :

```
Initialisation :
  Chargement de l'environnement de travail

SIVP - Scilab Image and Video Processing Toolbox
        load macros
        load gateways
        load help
        load demos
```

Sinon, installez le :

Applications > Gestionnaire de modules - ATOMS > Traitements des images > SIVP

Cliquer sur installer puis redémarrer **Scilab**.

1 Calculs statistiques — Rapport signal/bruit

Soit I une image de dimension $p \times n$.

— La *luminance* d'une image I :

$$\overline{I} = \frac{1}{p \times n} \sum_{i=1}^{p} \sum_{j=1}^{n} I(i,j)$$

est donnée en **Scilab** par la commande `mean(I)`.

— Le **contraste** peut-être modélisé par l'écart-type :

$$\sigma = \sqrt{\frac{1}{p \times n} \sum_{i=1}^{p} \sum_{j=1}^{n} \left(I(i,j) - \overline{I}\right)^2}$$

et est donné par `stdev(I)`.

— Le **rapport signal sur bruit** d'une image I obtenue par traitement d'une image de référence I_0 est :

$$SNR = 20 \log_{10} \left(\frac{\sum_{i=1}^{p} \sum_{j=1}^{n} I_0(i,j)^2}{\sum_{i=1}^{p} \sum_{j=1}^{n} \left(I(i,j) - I_0(i,j)\right)^2} \right)$$

Si l'image I est stockée dans la variable `image` et I_0 dans la variable `image0`, le SNR s'obtient avec la commande **Scilab** :

```
--> 20*log10(sum(image0.^2)/sum((image-image0).^2))
```

⚠ attention aux opérations termes à termes!

Exercice 1 :

1. Ouvrir les images `lena.pgm` et `lena_bruit.pgm` et les stocker dans les variables `image0` et `image`.
2. (a) Afficher l'histogramme de l'image `lena.pgm`.
   (b) Calculer sa luminance, son contraste et comparer avec l'histogramme.
3. Calculer le niveau de gris moyen et l'écart-type pour l'image `lena_bruit.pgm` et comparer avec les résultats obtenus pour `lena.pgm`.
4. Calculer le rapport signal Bruit de l'image `lena_bruit.pgm`.
5. Écrire une fonction `s = SNR(image0, image)` qui calcule le rapport signal bruit de image par rapport à l'image de référence `image0`.

2 Bruiter une image

Un bruit est une image dont les niveaux de gris $B(i,j)$ ont une répartition aléatoire, nous allons donc nous servir de `grand` pour ajouter du bruit à une image. On distingue plusieurs sortes de bruit :

— **Bruit gaussien additif** (ou bruit uniforme) : les valeurs de $B(i,j)$ sont répartis uniformément autour d'un niveau moyen; ce type de bruit simule très bien le bruit thermique des capteurs CCD et CMOS.
Pour créer un bruit uniforme de p % avec **Scilab** et l'ajouter à `image0` :

```
--> bruitunif = (1+p*grand(image0,'unf',-1,1));
--> image = image0.*bruitunif;
--> image = max(image,0);
--> image = min(image,255); //intervalle ramene a [0;255]
```

ou plus simplement en utilisant la fonction `imnoise` de SIVP :

```
-->image = imnoise(image0, 'speckle', p) //avec sivp
```

— **Bruit poivre et sel** (ou bruit binaire) : les valeurs de $B(i,j)$ valent 0 ou 1 suivant que le pixel est affecté d'un bruit ou pas, ce type de bruit simule très bien le bruit lié aux pixels défectueux 1 d'un capteur CCD ou CMOS. Pour créer un bruit binaire de p % avec **Scilab** (cas des pixels blancs seulement) et l'ajouter à `image0` :

```
--> bruitbin = grand(image0,'bin', 1, p);
--> image = max(image0, 255*bruitbin);
```

ou plus simplement en utilisant la fonction `imnoise` de SIVP :

```
--> image = imnoise(image, 'salt␣&␣pepper', p); //avec sivp
```

Exercice 2 :

1. Créer une image `image1` bruité uniformément à 20 %, calculer le SNR de `image1`, commenter.
2. De même créer une image `image2` bruité uniformément à 50 %, calculer le SNR de `image2`, comparer avec les résultats obtenus pour `image1`.
3. Créer une image `image3` avec un bruit binaire de 10 %, calculer le SNR de `image3`, comparer avec les résultats obtenus pour `image1` et `image2`.

3 Filtres passe-bas

la fonction `imfilter` permet de faire une convolution d'une image par un « filtre » défini par une matrice F, cette matrice pouvant être créée à partir de la fonction `fspecial`.

Par exemple :

```
--> F = fspecial('average', 3); //filtre moyenneur
--> image_filtree = imfilter(image0, F); //filtrage
```

ou encore :

```
--> F = fspecial('gaussian',3); //filtre gaussien
```

Exercice 3 : **Filtres linéaires**

1. **Filtre de moyenne**
   (a) Appliquer le filtre de moyenne de dimension 3×3 aux images bruitées `image2` et `image3`.
   (b) Sur quel type de bruit ce filtre semble-t-il le plus efficace ?
2. **Filtre gaussien**
   (a) Créer un filtre gaussien `F` de dimension 3×3.
   (b) Le visualiser :
   ```
   --> n = 3; plot3d(1:n, 1:n, F); //graphe
   ```
   (c) Vérifier que la somme des coefficients du filtre `F` vaut 1
   (d) Filtrer les images bruitées `image2` et `image3` à l'aide de ce filtre.
   (e) Sur quel type de bruit ce filtre semble-t-il le plus efficace ?

Exercice 4 : **Filtrage non-linéaire**

1. **Filtre médian**
   (a) Rappeler comment fonctionne un filtre médian.
   (b) Écrire une fonction `res = f_median(image0, n)` qui retourne l'image obtenue après filtrage par le filtre médian de dimension $n \times n$ (on pourra ne pas prendre en compte les bords).
   (c) Filtrer les images bruitées `image2` et `image3` à l'aide de ce filtre.
   (d) Sur quel type de bruit ce filtre semble-t-il le plus efficace ?
2. **conservative smoothing** : filtrage par le maximum
   Ce filtre de lissage supprime bien le bruit de type « poivre et sel » c'est à dire qu'il « adoucit » les pixels isolés ayant un niveau de gris très différent des niveaux de gris de leur voisinage et il a la particularité de bien préserver les contours très marqués.
   Ce filtre s'assure en fait que tout pixel a son niveau de gris placé dans la gamme de ses voisins.

   Méthode. On considère le niveau de gris du pixel à traiter, et d'autre part tous ses voisins (à l'exception de lui même) . Sur les voisins on calcule le niveau min et le niveau maximum, si le niveau de gris du pixel à traiter est compris entre le minimum et le maximum alors on le laisse inchangé sinon on le remplace par le maximum
   (a) Écrire une fonction `res = f_max(image0, n)` qui retourne l'image obtenue après filtrage par le filtre médian de dimension $n \times n$ (on pourra ne pas prendre en compte les bords).
   (b) Filtrer les images bruitées `image2` et `image3` à l'aide de ce filtre.
   (c) Sur quel type de bruit ce filtre semble-t-il le plus efficace ?

Exercice 5 : **SNR**
En calculant les SNR de vos différentes images obtenues par rapport à l'image de départ, vérifier l'efficacité des différents filtres sur les différents type de bruit.