Traitement et analyse d'images

Travaux pratiques

TP N°1 : Techniques de prétraitement

---

1. Préliminaires

Le traitement d'images est né de l'idée et de la nécessité de remplacer l'observateur humain par la machine. L'image généralement acquise par caméra puis numérisée doit être traitée pour mettre en évidence l'information recherchée.
Octave est un logiciel qui comprend à la fois des procédures de traitement d'images, des procédures de traitement de signal, des bibliothèques d'analyse et un moniteur de commande. Il peut être utilisé comme un ensemble de librairies que l'utilisateur manipule afin de résoudre ses problèmes spécifiques et d'élaborer ses propres tâches. Il peut également être utilisé comme un environnement fonctionnel où un grand nombre d'opérations sont utilisables sans aucune programmation.
La boîte à outils pour le traitement d'images (Image Processing Toolbox) est une bibliothèque de fonctions Octave. Sont inclus dans ce produit des opérations de prétraitement et de transformation d'images, des commandes d'affichage, des opérations de convolution et de filtrage passe-haut ou passe-bas de base, et des algorithmes standards de morphologie mathématique et de détection de contours.

2. Contexte

La segmentation consiste à effectuer une classification de l'image en régions homogènes par rapport à un ou plusieurs critères. La fiabilité de cet étape d'analyse dépend en grande partie de la qualité de l'image sur laquelle on travaille. La nature des images acquises est généralement sujette à imperfections inévitables liées, soit au système d'acquisition (éclairage non homogène ou mal adapté, bruit électronique des capteurs, bruit d'échantillonnage et de quantification, optique de mise au point, ombre, reflets), soit aux propriétés intrinsèques de l'image elle-même (présence de bruit, de zones texturées). L'amélioration de l'aspect visuel d'une image par une étape dite de prétraitement vient pour faciliter cette segmentation en renforçant la ressemblance entre pixels appartenant à une même région, ou en accentuant la dissemblance entre pixels appartenant à des régions différentes. Dans un traitement de reconnaissance de formes, cette étape demeure complémentaire et tout à fait nécessaire à la phase de segmentation et demande, d'ailleurs comme cette dernière, une paramétrisation, spatiale ou fréquentielle, fonction de l'information pertinente recherchée.

Les prétraitements les plus utilisés sont :

- ➔ les modifications d'histogramme jouant sur la dynamique de l'image,
- ➔ la réduction du bruit ou lissage,
- ➔ le rehaussement de contraste.

3. Lancement du logiciel

Lancer Octave, et charger le paquets des commandes de traitement d'image en tapant :
pkg load image.
Si le paquet n'est pas installé, taper :
pkg install -forge image

4. Lecture et Affichage d'une image

Sous Matlab une image est une matrice associée à une table de couleurs d'affichage. Quatre types de base sont supportés :

- "Indexed images" : tables de pseudo-couleurs bâties en RGB,
- "Intensity images" : images en niveaux de gris,
- "Binary images" : images binaires,
- "RGB images" : combinaisons de 3 matrices (R, G et B).

On s'intéressera ici aux deux types principaux d'images : les images binaires et les images en niveaux de gris.

Matlab fournit un certain nombre de fichiers images (l'image **trees** par exemple) qui sont sous format indexé et qui se chargent à partir de la commande **load** ; pour les convertir en niveaux de gris utiliser la commande **ind2gray**.

load trees; : La matrice-image est stockée dans **X** et la table des couleurs dans **map**. Pour ne pas afficher le résultat d'une sortie, vous ajoutez à la suite de l'instruction un '**;**' (très utile dans le cas d'une matrice de dimension importante). Dans le cas contraire vous pouvez toujours arrêter le défilement en utilisant la commande système **Ctrl C**.

I = ind2gray(X,map); : Convertit l'image (X,map) en niveaux de gris et stocke le résultat dans la matrice I. Les niveaux dans I sont normalisés entre 0 et 1 au lieu de 0 et 255.

brain = imread('brain.png); : L'image en niveaux de gris **'brain.png'** est stockée dans la matrice **brain**.

[ty,tx] = size(brain) : La taille de la matrice est stockée dans les deux paramètres ty et tx.

brain(y,x) permet d'afficher la valeur du pixel (**x**,**y**) (système de coordonnées pixel) correspondant à l'élément de la **ligne y** et de la **colonne x** (système de coordonnées matrice).

Pour afficher cette image il suffit de taper la commande : **figure (1), imshow(brain)** .

Vous pouvez ensuite découvrir d'autres commandes en tapant **help** <**nom de la commande**>.
Taper **help im2bw** puis **help subplot** et **help min** (ou **max**). Taper ensuite **min(brain)** puis **min(min(brain))** .

5. Modification d'Histogramme

▪ *Histogramme d'images :*

✓ charger les images en niveaux de gris **house1.png**, **house2.png**, puis **house3.png**. Tracer leur histogramme au moyen de la commande **imhist**.
Remarques et Interprétations ?

✓ effectuer les mêmes opérations sur l'image **sic.png**. Commentaires ? Comment peut-on segmenter cette image ? Effectuer cette segmentation après avoir cherché l'aide sur les commandes **im2bw** et **graythresh**.

▪ *Sensibilité du système visuel humain :*

✓ charger l'image **house3.png**, image codée sur 256 niveaux avec un numériseur 8 bits. Pour avoir une idée sur le lien entre la profondeur (luminosité) de l'image et le nombre de bits du numériseur, coder **house3** sur 128, 32, puis 8 niveaux de gris en divisant[1] la matrice par 2, 8, et 32. Comparer les images codées et contrôler leur histogramme. Remarques et Conclusions ?

✓ modifier la dynamique de l'image **dna.png** en jouant sur les paramètres d'entrée de la commande **imadjust** : **[low high]**, **[bottom top]** et la correction γ. Commenter images et histogrammes résultant.

✓ égaliser son histogramme en utilisant la commande **histeq**. Commentaires ?

[1] Pour diviser une matrice M par un entier n et stocker le résultat dans R, utiliser la commande :
R = uint8(n*fix(double(M)/n));

6. Réduction du Bruit, Lissage

Matlab fournit la plupart des filtres de lissage standard (Average, Gaussian, Median). Rappeler les principes de base de ces différents filtres spatiaux.

Effectuer et Commenter les séquences suivantes :

Séquence 1 :

```
savoise = imread('savoise.png');
savgaus = imread('savgaus.png'); // image savoise + bruit de type "gaussien"
figure(1), imshow(savoise)
figure(2), imshow(savgaus)
h3      = fspecial('average') // ôter le ';' afin d'afficher la matrice du filtre
h5      = fspecial('average',5)
h7      = fspecial('average',7)
AVE3   = filter2(h3,savgaus);
AVE5   = filter2(h5,savgaus);
AVE7   = filter2(h7,savgaus);
figure(3), imagesc(AVE3), colormap(gray(256)) ;
figure(4), imagesc(AVE5), colormap(gray(256)) ;
figure(5), imagesc(AVE7), colormap(gray(256)) ;
```

Comparer et Interpréter les images AVE3, AVE5 et AVE7. Pour vous aider, comparer des profils d'intensité extraits des différentes images :

```
profileOriginal= savgaus(50,:) ; figure, plot(profileOriginal) ;
profile3=AVE3(50,:) ; figure, plot(profile3) ;
profile5=AVE5(50,:) ; figure, plot(profile5) ;
profile7=AVE7(50,:) ; figure, plot(profile7) ;
```

Pourquoi utilise-t-on la fonction **imagesc** plutôt que **imshow** ?

```
MED3  = medfilt2(savgaus);
figure, imshow(MED3)
```

Comparer et Interpréter les images AVE3 et MED3.

Mesurer l'intensité des pixels de **savgaus** (cf. § 4) dans un voisinage 3x3 centré au pixel (**x=141**, **y=127**). Mesurer l'intensité de ce même pixel central pour **AVE3** et **MED3**. Expliquer les valeurs obtenues.

Séquence 2 :

```
savoise  = imread('savoise.png');
savsap  = imread'savsap.png'); // image savoise + bruit de type "poivre et sel"
figure(1), imshow(savoise)
figure(2), imshow(savsap)
MED3   = medfilt2(savsap);
MED5   = medfilt2(savsap,[5,5]);
MED7   = medfilt2(savsap,[7,7]);
figure(3), imshow(MED3)
figure(4), imshow(MED5)
figure(5), imshow(MED7)
```

Comparer et Interpréter les images MED3, MED5 et MED7.

```
h3       = fspecial('average');
AVE3   = filter2(h3, savsap);
figure(7), imagesc(AVE3)
```

Comparer et Interpréter les images MED3 et AVE3.

7. Filtrage Fréquentiel par Transformée de Fourier Rapide

Après avoir rappelé ce qu'est une transformée de Fourier bidimensionnelle, rechercher l'aide sur les fonctions **fft2**, **fftshift** et **ifft2**.

Appliquer et interpréter la transformée de Fourier recentrée sur les images **interfer.png**, **grain.png** et **camsha.png**. NB. : Afin d'afficher une transformée de Fourier **TF** en couleur, utiliser la commande : **colormap(jet(64)), imagesc(log(abs(TF)));**

Les méthodes de filtrage spatial vues précédemment agissent localement sur une image et de ce fait ne suppriment pas toutes les hautes fréquences si l'image est fortement bruitée. La transformée de Fourier permet d'effectuer rapidement ce débruitage tout en supprimant les hautes fréquences sur la totalité de l'image. Pour illustrer ce propos nous allons appliquer un filtre fréquentiel (masque) rectangulaire sur une image bruitée, texturée et peu contrastée.
Calculer la transformée de Fourier recentrée **TfOt** de l'image **otolithe.png**. Commenter les lignes de la séquence suivante :

```
figure(1), colormap(jet(64)), imagesc(log(abs(TfOt)))
//définition d'une région d'intérêt rectangulaire
roi=zeros(size(TfOt));
dv=50, dh=50;  // les dimensions de bande passante du filtre
xc=size(roi,2)/2; yc=size(roi,1)/2;
roi(yc-dv:yc+dv,xc-dh:xc+dh)=1;
figure(2), imshow(roi)
figure(3), imshow(~roi)
HF = TfOt .* ~roi;
BF = TfOt .* roi;
figure(4), colormap(jet(64)), imagesc(log(abs(HF)))
figure(5), colormap(jet(64)), imagesc(log(abs(BF)))
```

Appliquer ensuite la transformée de Fourier inverse pour obtenir l'image débruitée[2].
Que représente la transformée de Fourier inverse de **HF**.

8. Rehaussement de contraste

Le problème de contraste est étroitement lié à l'élargissement de la zone de transition. Dans quels cas ce problème se produit-il ? Rappeler les différentes méthodes de rehaussement de contraste. Proposer et effectuer une chaîne de traitement[3] permettant le rehaussement de contraste de l'image **floue.png**.

[2] Pour afficher l'Image Débruitée ImD, utiliser la commande **ImD = ifft2(BF);** puis **figure(6), colormap(gray(256)), imagesc(abs(ImD));**

[3] Pour calculer le laplacien de l'image floue, utiliser la commande **Lap = del2(double(floue));** et pour afficher l'Image Réhaussée ImR, la commande **imshow(uint8(ImR));**