Université Paris Descartes
UFR de Mathématiques et Informatique
45, rue des Saints-Pères 75270 Paris cedex 06

MASTER 1

PROGRAMMATION

Traitement d'images - `Scilab`

Dans ce TP, on va manipuler des images numériques matricielles qui sont composées de *pixels* (de l'anglais *pic*ture *el*ement). Les images numériques en niveaux de gris sont représentées dans Scilab par des matrices. Si I est une image, alors $I(i,j) \in \{0,1,2,\ldots,254,255\}$, 0 représente le noir, 255 le blanc, les autres valeurs entières représentent des intensités intermédiaires.
Dans ce qui suit, nous utiliserons des images au format `BMP`. Commencez par télécharger l'archive `TPimages.zip` à l'adresse
`http://w3.mi.parisdescartes.fr/~jdelon/enseignement/MA106/TPimages.zip`.
Cette archive contient plusieurs images dont on pourra se servir pour le TP. Pour ouvrir et lire des images sous Scilab, on utilisera la toolbox SIVP qui est installée sur les machines. Commencez par charger la toolbox dans l'interface graphique de Scilab. Une fois la toolbox chargée, les lignes suivantes vous permettent d'ouvrir et d'afficher une image dans Scilab :

```
U = double(imread('mandrill.bmp'));        // on ouvre l'image mandrill.bmp
figure(1);imshow(U/255);                 // on affiche l'image
```

Exercice 1 Détection de contours.
Les contours dans les images sont les zones où les niveaux de gris changent rapidement. Ces zones correspondent généralement aux bords des objets, il est donc intéressant d'essayer de les détecter. Pour cela, on utilise par exemple la norme du gradient de l'image. En chaque pixel (i,j) de l'image U, on calcule la norme du gradient discret :

$$NG(i,j) = \frac{1}{2}\sqrt{(U(i+1,j)-U(i-1,j))^2 + (U(i,j+1)-U(i,j-1))^2}. \qquad (1)$$

En visualisant l'image NG de la norme du gradient, on voit ressortir en clair les zones de contours de l'image U.
Ecrire une fonction `[M]=im_contour(U)` qui calcule à partir de U l'image de la norme de son gradient et l'affiche. On pourra supposer que l'image U est prolongée par des 0 pour calculer le gradient aux points (i,j) du bord.

Exercice 2 Ajout de bruit.
Dans ce qui suit, on veut créer différentes versions dégradées de l'image U. La première, U_b, sera obtenue en ajoutant à U une image de bruit gaussien (on parle alors de bruit gaussien additif). La deuxième, U_{imp} sera obtenue en remplaçant aléatoirement les valeurs de certains pixels de U par des valeurs aléatoires tirées uniformément sur $[0,255]$ (on parle de bruit impulsionnel). On utilisera en particulier la fonction **rand**, qui permet de générer une matrice de nombres aléatoires suivant une loi uniforme sur $[0,1]$ ou suivant une loi normale. Par exemple,

```
B=rand(10,15,'normal')
```

crée une matrice B à 10 lignes et 15 colonnes de valeurs (pseudo-)aléatoires suivant la loi normale de moyenne 0 et de variance 1 (si on écrit 'uniform' au lieu de 'normal', la loi utilisée est la loi uniforme). Si U est une matrice, la commande

`B=rand(U,'normal')`

crée une matrice B de même taille que U de valeurs (pseudo-)aléatoires suivant la loi normale de moyenne 0 et de variance 1.

1. **Bruit gaussien additif.**
   Créer une fonction Scilab `[Ub]=bruite(U,s)`, qui prend en entrée l'image U et l'écart-type s, et crée l'image bruitée $U_b = U + B$, où B est une matrice de même taille que U, composée de nombres aléatoires suivant la loi normale de moyenne 0 et d'écart type s donné. Afficher l'image bruitée pour diverses valeurs de s.
2. **Bruit impulsionnel**
   Créer une fonction Scilab `[Uimp]=bruite_imp(U,p)`, qui prend en entrée l'image U et une valeur $0 \leq p \leq 100$, et crée une image U_{imp} en remplaçant $p\%$ des pixels de U par des valeurs aléatoires entre 0 et 255. On pourra utiliser d'abord la fonction `rand` pour engendrer un tableau I de nombres aléatoires suivant la loi uniforme sur $[0, 1[$:

   ```
   I = rand(U);
   ```

   Grâce à ce tableau on remplacera de façon aléatoire $p\%$ des pixels ($p \in [0, 100]$) de l'image U par un niveau de gris aléatoire.

   ```
   Uimp = 255*rand(U).*(I<p/100)+(I>=p/100)*U;
   ```

   Afficher l'image bruitée pour diverses valeurs de p.

Exercice 3 Débruitage.

1. Écrire une fonction Scilab, `[A]=im_extract(U,i,j,f)`, qui extrait le bloc carré centré en (i, j) et de dimensions $(2f + 1) \times (2f + 1)$, où $f \in \mathbb{N}^*$. Prévoir les tests nécessaires en cas de débordement (adapter la taille des blocs extraits lorsque (i, j) est trop proche du bord). Afficher l'image extraite pour quelques valeurs de i, j, f.
2. **Filtre moyenne**
   Écrire une fonction `[M]=im_moyenne(U,f)`, qui remplace en chaque pixel (i, j) la valeur de $U(i, j)$ par la moyenne dans un voisinage centré en (i, j) et de taille $(2f+1, 2f+1)$ (supposer que l'image se prolonge par une fonction nulle aux bords).
   Soit U_b une image obtenue en ajoutant un bruit gaussien additif d'écart-type 10 à U. Utilisez la fonction `im_moyenne` avec $f = 1$ pour débruiter U_b. Afficher le résultat et comparer le à l'image originale U. Qu'observez-vous ?
3. **Filtre médian**
   On rappelle que le médian d'un ensemble de nombres $\{a_1, \ldots, a_n\}$ est la valeur a_k telle qu'il y ait autant de $a_i \leq a_k$ que de $a_i \geq a_k$. On le trouve facilement en triant les a_i par ordre croissant et en prenant la valeur du milieu (dans le cas où n est pair, il y a deux choix possibles pour le médian). La fonction `median` de Scilab permet de trouver le médian d'un vecteur ou d'une matrice.
   Écrire une fonction `[M]=im_median(U,f)`, qui remplace en chaque pixel (i, j) la valeur de $U(i, j)$ par la valeur médiane dans un voisinage centré en (i, j) et de taille $(2f + 1, 2f + 1)$ (supposer que l'image se prolonge par une fonction nulle aux bords).
   Soit U_b une image obtenue en ajoutant un bruit gaussien additif d'écart-type 10 à U. Soit U_{imp} une image obtenue en ajoutant un bruit impulsionnel à un pourcentage $p = 30\%$ des pixels de U. Utilisez la fonction `im_median` pour débruiter ces deux images. Essayez plusieurs valeurs de f. Qu'observez-vous ?
4. **Filtre de Yaroslavski** Le filtre de Yaroslavski est un filtre qui permet de débruiter une image tout en préservant ses contours. Il est efficace sur le bruit gaussien additif. Le principe du filtre est de remplacer en chaque point (i, j), la valeur $U(i, j)$ par

$$V(i,j) = \frac{1}{C}_{i,j} \sum_{(k,l)\in\Omega_{i,j}} U(k,l) g_r(|U(k,l) - U(i,j)|),$$

où $\Omega_{i,j}$ est un voisinage carré centré en (i,j) et de dimensions $(2f+1)\times(2f+1)$, où $g_r(t) = e^{-\frac{t^2}{2\sigma_r^2}}$, et où $C_{i,j}$ est une constante de normalisation qui vaut

$$C_{i,j} = \sum_{(k,l)\in\Omega_{i,j}} g_r(|U(k,l)-U(i,j)|).$$

Ecrire une fonction `[M]=im_yaroslavski(U,f,s)` qui applique le filtre de Yaroslavski à l'image U, avec un voisinage de taille $(2f+1)\times(2f+1)$ et avec $\sigma_r = s$. Si on note nl et nc les nombres de lignes et de colonnes de U, le pseudo-code de la fonction peut s'écrire

Filtre de Yaroslavski

for $i = 1$ **to** nl **do**
  for $j = 1$ **to** nc **do**
    $V = im\_extract(U,i,j,f)$
    $W = e^{-\frac{(V-U(i,j)).^2}{(2*s^2)}}$; // W est une imagette de poids gaussiens
    $B(i,j) = \frac{\sum_{k,l} W_{k,l}V_{k,l}}{\sum_{k,l} W_{k,l}}$;

Soit U_b une image obtenue en ajoutant un bruit gaussien additif d'écart-type 10 à U. Utilisez la fonction `[M]=im_yaroslavski` pour débruiter l'image U_b. Essayez par exemple avec les paramètres $f = 3$ et $s = 20$. Qu'observez-vous ?

5. **Filtre bilatéral** Le filtre bilatéral est une variante du filtre de Yaroslavski, dans laquelle on ajoute une pondération spatiale gaussienne au calcul de l'image filtrée V. On remplace donc ici la valeur $U(i,j)$ par

$$V(i,j) = \frac{1}{C}_{i,j} \sum_{(k,l)\in\Omega_{i,j}} U(k,l) g_r(|U(k,l)-U(i,j)|) g_s(\sqrt{|i-k|^2+|j-l|^2}),$$

où $\Omega_{i,j}$ est un voisinage carré centré en (i,j) et de dimensions $(2f+1)\times(2f+1)$, où $g_r(t) = e^{-\frac{t^2}{2\sigma_r^2}}$, $g_s(t) = e^{-\frac{t^2}{2\sigma_s^2}}$ et où $C_{i,j}$ est une constante de normalisation qui vaut

$$C_{i,j} = \sum_{(k,l)\in\Omega_{i,j}} g_r(|U(k,l)-U(i,j)|) g_s(\sqrt{|i-k|^2+|j-l|^2}).$$

Ecrire une fonction `[M]=im_bilateral(U,f,r,s)` qui applique le filtre bilatéral à l'image U, avec un voisinage de taille $(2f+1)\times(2f+1)$, avec $\sigma_r = r$ et $\sigma_s = s$. Pour gagner du temps, on pourra repartir du code du filtre de Yaroslavski et le modifier.

Tester cette fonction pour débruiter une image bruitée par un bruit gaussien additif.