MT12 - TP5 : Traitement d'images

1 Travailler avec des images sous Scilab

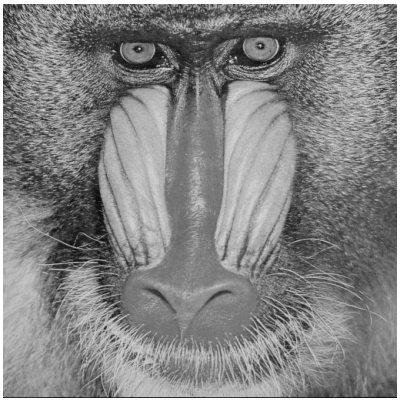

FIGURE 1 – Mandrill

Pour ce TP nous mettons à dispositions différentes images sur le moodle de l'UV. On peut associer à une image en noir et blanc constituée de $N \times M$ pixels une matrice, notée U, de taille $N \times M$ où les coefficients de la matrice sont des entiers compris entre 0 (noir) et 255 (blanc). Sur l'exemple la Figure 1, l'image est constituée de 512×512 pixels. On peut donc associer à cette image une matrice de taille 512×512.

Afin de faire du traitement d'image sous `Scilab`,

A installer comme suit l'application *Image Processing and Computer Vision Toolbox* :

(a) dans la console `Scilab`, cliquer sur les onglets : **Applications** puis **Gestionnaire de Modules - ATOMS** ;

(b) après avoir ouvert le gestionnaire de modules, dans le menu de gauche cliquer sur : **Traitement d'images** puis **Image Processing and Computer Vision Toolbox** et enfin **Installer**.

(c) une fois l'installation effectuée, relancer `Scilab` ;

(d) si tout marche bien, le message suivant doit s'afficher lors de l'ouverture de la console `Scilab` :

Initialisation :
Chargement de l'environnement de travail
Start IPCV 4.X.Y.for Scilab 6.Z etc

B enregistrer sur votre ordinateur l'une des images disponibles sur le moodle (celle du mandrill ou une autre) et utiliser les commandes suivantes pour ouvrir et afficher l'image dans `Scilab` :

```
1 U=double(imread('chemin\mandrill.bmp')); //remplacer le chemin et l'image choisie
2
3 imshow(mat2gray(U))
```

1. Pour modifier une image, il suffit de modifier la valeur des coefficients de la matrice associée U. Dans un premier temps on veut créer un carré noir de 50×50 pixels en haut à gauche de l'image que vous avez choisie. Ecrire en `Scilab` un code permettant d'effectuer cette modification de l'image choisie et afficher l'image ainsi modifiée (on peut de la même manière afficher un carré noir ou blanc en bas à droite de l'image ou en haut à droite de l'image).

2 Bruiter une image

2. Dans cette partie, on veut créer différentes versions bruitées de l'image en modifiant la matrice associée U. La première modification, U_b est obtenue en ajoutant à tous les pixels de l'image initiale un bruit gaussien (on parle alors de bruit gaussien additif). La deuxième modification, U_{imp} est obtenue en remplaçant aléatoirement les valeurs de certains pixels de l'image initiale par des valeurs aléatoires tirées uniformément sur $[0, 255]$ (on parle de bruit impulsionnel).

   (a) En utilisant la fonction `grand` de `Scilab`, écrire une fonction `bruitgaus` qui renvoie une matrice $U_b = U + B$, où B est une matrice de taille 512×512 composée de nombres aléatoires suivant la loi normale de moyenne nulle et d'écart type s donné. Afficher l'image U_b pour différentes valeurs de s.

   (b) Ecrire une fonction `bruitimp` prenant en entrée U et renvoyant une matrice U_{imp} obtenue en remplaçant p% des pixels de U par des valeurs aléatoires entre 0 et 255. Pour ce faire compléter le code suivant :

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

   Afficher l'image U_{imp} pour différentes valeurs de p.

3 Filtre moyenne et médian

Dans cette partie, on cherche à débruiter les deux images associées à U_b et à U_{imp} pour obtenir une image aussi proche que possible de l'image initiale associée à U ; pour ce faire, on va appliquer à U_b et à U_{imp} un masque (ou filtre) en suivant la procédure décrite ci-dessous :

(a) **"U_b avec filtre moyenne"** on remplace la valeur du pixel $U_b(i, j)$ par la valeur moyenne des pixels dans un voisinage centré en (i, j) et de taille $(2f + 1, 2f + 1)$ (avec $f \geq 1$ un entier donné).

(b) **"U_{imp} avec filtre médiane"** on remplace la valeur du pixel $U_{\text{imp}}(i, j)$ par la valeur médiane des pixels dans un voisinage centré en (i, j) et de taille $(2f + 1, 2f + 1)$ (avec $f \geq 1$ un entier donné).

3. Les procédures décrites ci-dessus ne sont pas applicables sur les pixels en bordure de l'image ; afin d'y remédier, on propose d'inclure l'image initiale au centre d'une nouvelle image de taille $(N+2f)\times(M+2f)$ où les bords de cette nouvelle image sont noirs (grossièrement on prolonge l'image initiale par 0 sur les bords de l'image). Compléter le code suivant afin d'effectuer cette opération de prolongement :

```
f=...;
[N,M]=size(U);
Ubig=zeros(N+...,M+...);
Ubig(...:...,...:...)=U;
```

   Afficher `Ubig` pour $f = 10$, 80 et 160.

4. Application des filtres.

   (a) **"U_b avec filtre moyenne"**

(i) En utilisant la question 3 et à l'aide de la fonction `mean` de `Scilab`, écrire une fonction `moyenne` (avec pour arguments U et f), qui remplace chaque valeur de pixel $U(i,j)$ par la valeur moyenne des pixels dans un voisinage centré en (i,j) et de taille $(2f+1, 2f+1)$.

(ii) En notant U_b la matrice associée à l'image bruitée obtenue en ajoutant un bruit gaussien additif d'écart-type $s = 30$ à U, utiliser votre fonction `moyenne` pour différentes valeurs de f afin de débruiter U_b. Afficher le résultat et comparer le à l'image originale associée à U (on pourra utiliser la fonction subplot pour comparer U, U_b et l'image obtenue via le filtre moyenne).

(b) "**U_{imp} avec filtre médiane**"

(i) En utilisant la question 3 et à l'aide la fonction `median` de `Scilab`, écrire une fonction `mediane` (avec pour arguments U et f) qui remplace chaque valeur $U(i,j)$ par la valeur médiane dans un voisinage centré en (i,j) et de taille $(2f+1, 2f+1)$.

(ii) En notant U_{imp} la matrice associée à l'image bruitée obtenue à partir de la fonction `bruitimp` pour $p = 30$, utiliser votre fonction `mediane` pour différentes valeurs de f afin de débruiter cette image et afficher le résultat (on pourra utiliser la fonction subplot pour comparer U, U_{imp} et l'image obtenue via le filtre médian).

4 Détecter les contours d'une image

Les contours d'une image sont des zones où les niveaux de gris varient rapidement. Ces zones correspondent généralement aux bords des objets, on se propose dans cette partie de les détecter. Pour ce faire on va utiliser la norme du gradient discret de l'image. En chaque pixel (i,j) de l'image U on veut calculer la norme du gradient discret :

$$\nabla_h U(i,j) = \frac{\sqrt{(U(i+1,j) - U(i-1,j))^2 + (U(i,j+1) - U(i,j-1))^2}}{2}, \quad \forall 1 \leq i \leq N,\, 1 \leq j \leq M.$$

Ici un problème se pose de nouveau au niveau des bords de l'image. Comme dans la partie précédente on va supposer que l'image initiale est prolongée par 0.

5. Écrire une fonction `contour` (avec en argument une image U de taille $N \times M$) qui calcule à partir de U l'image de la norme de son gradient. Afficher le résultat.