MT12 - TP04

Antoine GAJAN

P24

L'objectif principal de ce TP est d'apprendre à manipuler et traiter des images sous Scilab, en ajoutant du bruit, en appliquant des techniques de débruitage, et en détectant les contours des objets présents dans les images.

---

1 Travailler avec des images sous Scilab

Dans un premier temps, nous avons installé les modules nécessaires sous Scilab pour effectuer du traitement de données et de la vision par ordinateur. Nous avons vérifié le bon fonctionnement des modules sur l'ordinateur à l'aide des commandes suivantes :

```
U=double(imread("C:\Users\antoi\Desktop\UTC\GI04\MT12\TPs\TP5\Images\mandrill.
bmp"))
imshow(mat2gray(U))
```

1. Nous souhaitons d'abord comprendre la manière d'effectuer des manipulations élémentaires d'images sous Scilab. Pour cela, nous allons modifier la valeur des coefficients de la matrice U pour créer un carré noir de 50*50 pixels en haut à gauche de l'image (et faire de même pour les autres coins).

   Ceci a été effectué à l'aide du code suivant :

```
[nRows, nCols] = size(U)
// En haut a gauche
U(1:50, 1:50) = 0
// En haut a droite
U(1:50, nCols-49:nCols) = 0
// En bas a gauche
U(nRows-49:nRows, 1:50) = 0
// En bas a droite
U(nRows-49:nRows, nCols-49:nCols) = 0
```

   Nous aurions également pu afficher des carrés blancs en remplaçant la valeur 0 par 255. En ajoutant des carrés noirs à chaque coin de l'image initiale du mandrill, nous avons alors obtenu le résultat suivant :

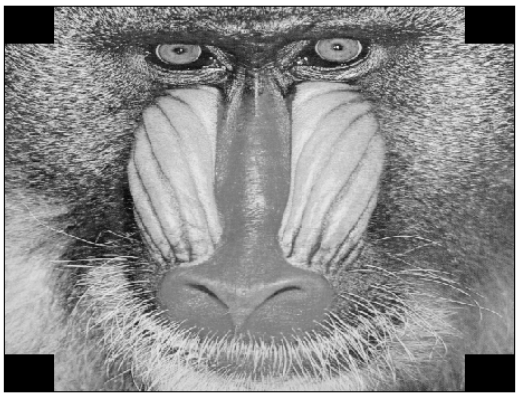

FIGURE 1 – Ajout de carrés noirs de 50*50 pixels à chaque coin de l'image

2 Bruiter une image

Dans cette partie, nous allons créer différentes versions bruitées de l'image étudiée. Nous nous intéresserons à l'ajout d'un bruit gaussien additif et à celui d'un bruit impulsionnel.

2.1 Bruitage gaussien additif

L'ajout d'un bruitage gaussien additif est réalisé en ajoutant à tous les pixels de l'image initiale un bruit gaussien.

2.(a) A l'aide de la fonction `grand` de `Scilab`, nous pouvons écrire une fonction `bruitgaus` qui renvoie une matrice $U_b = U + B$ où B est une matrice de taille 512 * 512 pixels composée de nombres aléatoires suivant la loi normale $\mathcal{N}(0, s^2)$.

```
function Ub = bruitgaus(U, s)
    // Recuperation des dimensions de l'image
    [nRows, nCols] = size(U)
    // Creation d'une matrice de meme dimension que U composee de nombres
aleatoires suivant la loi normale
    B = grand(nRows, nCols,'nor',0,s)
    // Ajout du bruit gaussien
    Ub = U + B
endfunction
```

Nous obtenons les résultats suivants selon les valeurs de s.

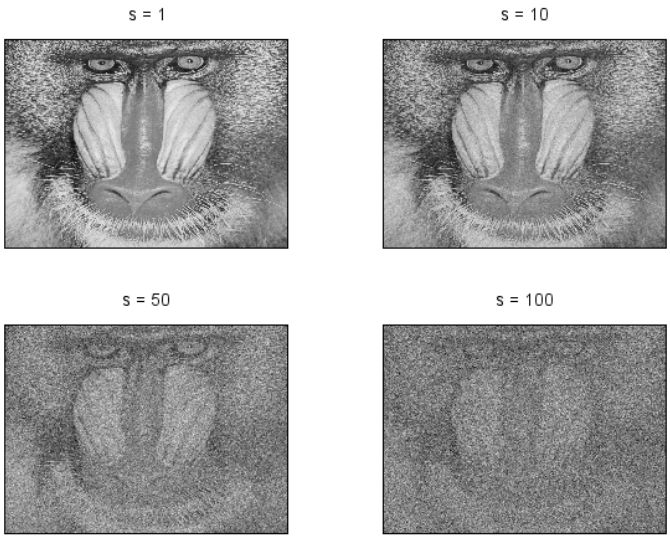

FIGURE 2 – Bruitage gaussien additif obtenu selon les valeurs de s

Nous remarquons que lorsque l'écart type s est faible, le bruit ajouté ne détériore pas significativement l'image du mandrill. En revanche, plus s est grand et plus le bruit ajouté modifiera significativement l'image initiale.

2.2 Bruitage impulsionnel

L'ajout d'un bruitage impulsionnel est obtenu en remplaçant aléatoirement les valeurs de certains pixels de l'image initiale par des valeurs aléatoires tirées uniformément sur l'intervalle $[0, 255]$.

2.(b) Pour réaliser la fonction `bruitimp`, nous pouvons nous appuyer sur la structure de code proposée. Nous obtenons ainsi :

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

Pour différentes valeurs de p, nous obtenons les résultats suivants :

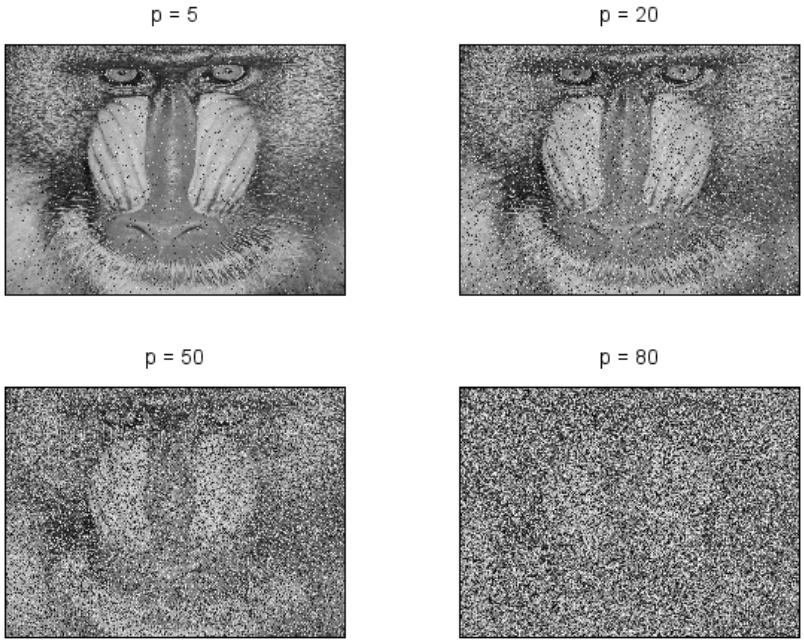

FIGURE 3 – Bruitage impulsionnel pour différentes valeurs de p

On remarque effectivement que plus la valeur de p est élevée (proche de 100), plus le pourcentage de pixels modifié est important, entrainant des difficultés à reconnaître l'image.

3 Filtre moyenne et médian

Dans cette partie, on cherche donc à débruiter les images bruitées (issu d'un bruit gaussien additif ou d'un bruit impulsionnel).

3.1 Intégration de l'image dans une image plus grande

3. Les procédures explicitées dans l'énoncé n'étant pas applicables en bordure de l'image, il est nécessaire de prolonger l'image initiale par 0 en ses bords. Autrement dit, on inclut l'image initiale au centre d'une nouvelle image de taille plus grande.

Ceci peut être effectué grâce au code suivant (ici, on a appliqué $f = 10$) :

```
f=10;
[N,M]=size(U)
Ubig10=zeros(N+2*f,M+2*f)
Ubig10(f+1:N+f,f+1:M+f)=U
```

Pour différentes valeurs de f, on obtient les résultats suivants :

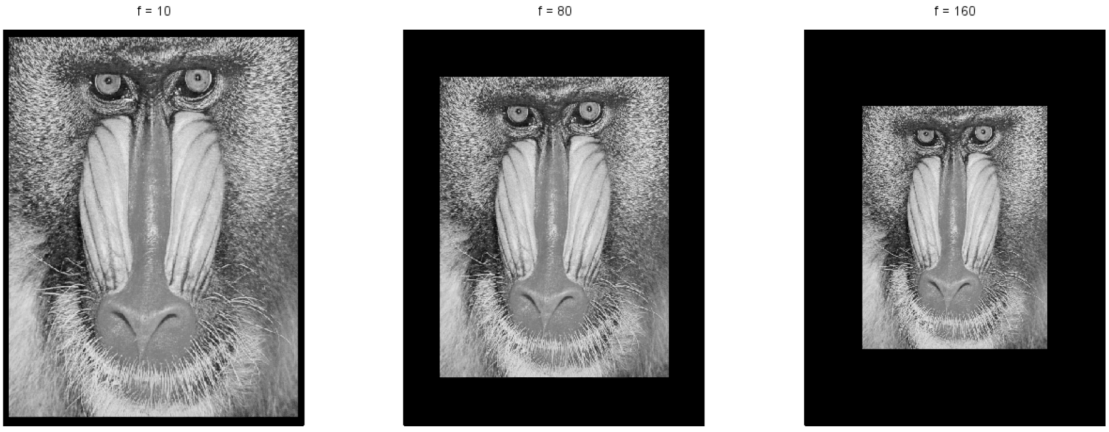

FIGURE 4 – Intégration de l'image au centre d'une image de taille (N+2f) * (M+2f)

3.2 Débruitage d'un bruit gaussien avec un filtre moyenne

4.(a) Dans cette partie, nous voulons essayer de débruiter l'image bruitée par un bruit gaussien additif à l'aide d'un filtre moyenne. Concrètement, nous souhaitons remplacer la valeur de chaque pixel $U(i,j)$ par la valeur moyenne des pixels dans un voisinage centré en (i,j) et de taille $(2f+1, 2f+1)$.

4.(a)(i) Pour cela, nous pouvons coder la fonction `moyenne` suivante :

```
function Umoy = moyenne(U, f)
    // Obtention de U centre
    [N,M]=size(U)
    Ubig=zeros(N+2*f,M+2*f)
    Ubig(f+1:N+f,f+1:M+f)=U
    // Calcul de Umoy
    Umoy = zeros(N, M)
    for i = f+1 : N+f
        for j = f+1 : M+f
            Umoy(i - f, j - f) = mean(Ubig(i-f:i+f, j-f:j+f))
        end
    end
endfunction
```

4.(a)(ii) En reprenant l'image U_b associéé à l'image bruitée obtenue en ajoutant un bruit gaussien additif d'écart-type $s = 30$, nous allons appliquer la fonction moyenne pour différentes valeurs de f afin de débruiter U_b.

Intuitivement, on s'attend à obtenir de meilleurs résultats pour f proche de 1. En effet, en prenant en considération des pixels fortement éloignés du pixel étudié, le programme risque de s'intéresser à des points peu pertinents, n'entrainant pas un débruitage de l'image. Vérifions donc cela pour différentes valeurs de f.

Pour $f \in \{1, 10, 20\}$, nous avons obtenu les résultats suivants :

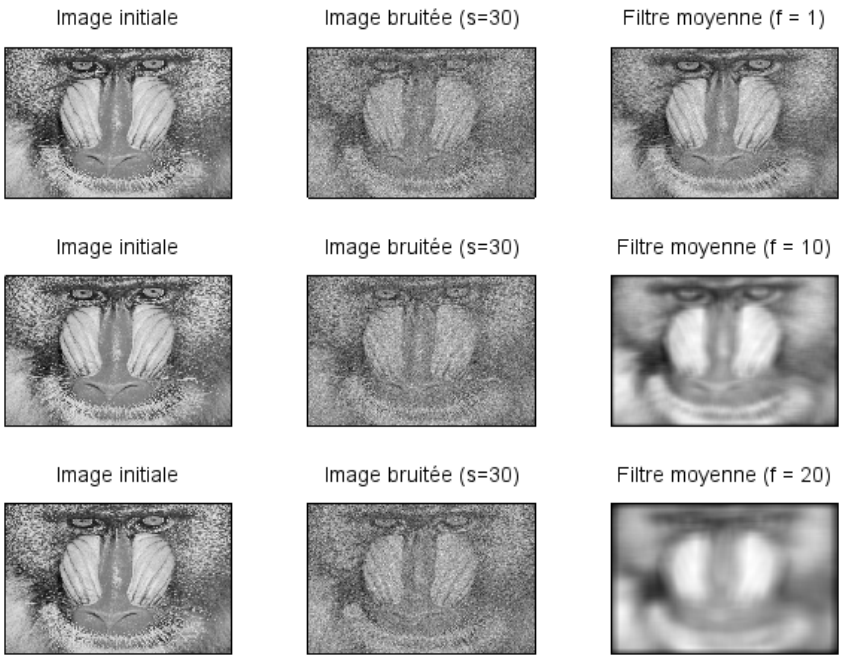

FIGURE 5 – Résultat du débruitage avec filtre moyenne pour $f \in \{1, 10, 20\}$

Comme nous nous y attendions, le débruitage est nettement plus efficace avec des petites valeurs de f.

3.3 Débruitage d'un bruit impulsionnel avec un filtre médiane

Nous allons procéder de manière analogue pour essayer de débruiter l'image associée à la matrice U_{imp}, obtenue en ajoutant un bruit impulsionnel avec $p = 30$, à l'aide d'un filtre médian.

4.(b)(i) A l'aide de la fonction `median` de `Scilab`, nous pouvons écrire une fonction qui remplace chaque valeur $U(i, j)$ par la valeur médiane dans un voisinage centré en (i, j) et de taille $(2f + 1, 2f + 1)$.

Nous obtenons alors le code suivant :

```
function Umed = mediane(U, f)
    // Obtention de U centre
    [N,M]=size(U)
    Ubig=zeros(N+2*f,M+2*f)
    Ubig(f+1:N+f,f+1:M+f)=U
    // Calcul de Umed
    Umed = zeros(N, M)
    for i = f+1 : N+f
        for j = f+1 : M+f
            Umed(i - f, j - f) = median(Ubig(i-f:i+f, j-f:j+f))
        end
    end
endfunction
```

4.(b)(ii) Essayons notre fonction `mediane` pour différentes valeurs de f afin de débruiter l'image associée à la matrice U_{imp}.

Comme avec le précédent filtrage, nous pouvons nous attendre à ce que le résultat soit meilleur pour des valeurs de f petites.

Avec $f \in \{1, 5, 10\}$, nous obtenons les résultats suivants :

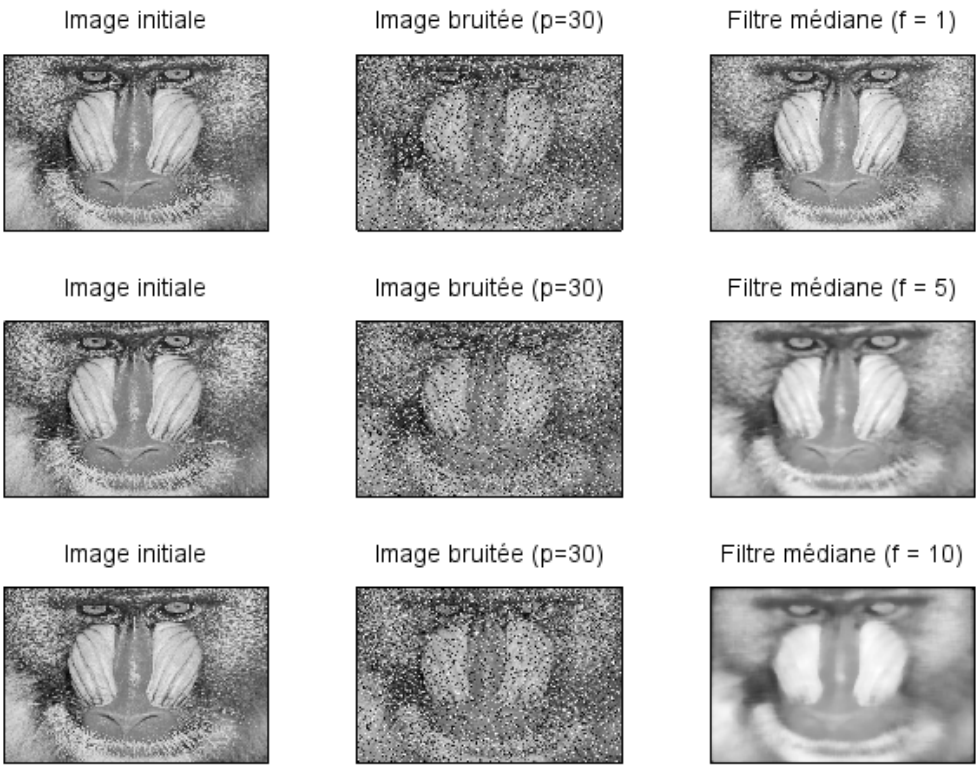

FIGURE 6 – Résultat du débruitage avec filtre médiane pour $f \in \{1, 5, 10\}$

Les résultats obtenus montrent l'intérêt d'utiliser cette technique de filtrage pour débruiter une image. Le résultat est effectivement de meilleure qualité avec des petites valeurs de f.

4 Détecter les contours d'une image

Les contours d'une image sont des zones où les niveaux de gris varient rapidement. Pour les détecter, on peut utiliser la norme du gradient discret.

En chaque pixel (i, j) de la matrice U, on veut calculer la norme du gradient discret comme suit :

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

5. On peut donc écrire une fonction `contour` qui effectue le traitement souhaité.

```
function Ucontour = contour(U)
    // Prolongement de l'image au niveau des bords (f = 1)
    [N,M]=size(U)
    f = 1
    Ubig=zeros(N+2*f,M+2*f)
    Ubig(f+1:N+f,f+1:M+f)=U
    Ucontour = zeros(N, M)
    // Calcul de la norme du gradient discret pour chaque pixel
    for i = 1:N
        for j = 1:M
            Ucontour(i, j) = norme_gradient(Ubig, i+f, j+f)
        end
    end
endfunction
```

En appliquant cette fonction aux matrices associées aux images du mandrill et du bateau initiales, nous obtenons le résultat suivant :

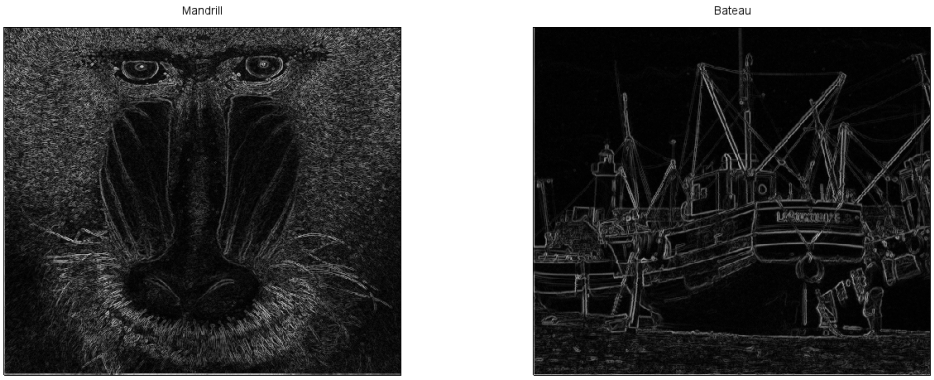

FIGURE 7 – Détection des contours des images du mandrill et du bateau

Le résultat correspond à celui espéré. Nous pouvons détecter les contours du mandrill (visage, yeux,...). J'ai pris le temps d'essayer avec d'autres images pour voir si les résultats seraient similaires. Avec l'image du bateau, on détecte également les contours.

5 Conclusion

Pour conclure, ce TP nous a permis de prendre en main le traitement d'images sous `Scilab`. Nous avons pu appliquer différents types de bruitage pour bruiter l'image et la débruiter. Enfin, nous avons appliqué une technique pour détecter les contours d'une image.