Niveau: Master1 Télécommunication, 2017

Responsable du cours: Zoubeida MESSALI

Enseignante de TP: Soumia SID AHMED

TP2 : Traitement numérique des images par MATLAB

Après ce TP, vous serez capable de:

I. Prise en main des images : lecture, écriture affichage
II. Transformations ponctuelles sur l'image
III. Traitement sur l'histogramme
IV. Transformations Géométriques sur l'image

I. Prise en main des images : lecture, écriture affichage

Manipulation 1:

Ecrire un script qui permet de :

1. **Lisez l'image 'onion.png'**
2. **Parcourez la variable dans laquelle vous avez enregistré votre image avec la structure itérative for.**
3. **Bit-plane Slicing : Soit une image A : 'onion.png' de niveau de gris codé sur 8 bits (nuance de gris varient de 0 à 255). La technique du Bit-plane slicing consiste décomposer l'image en un ensemble de plans, dans notre cas, 8 plans. Afficher les 8 plans de l'image. Commenter ces plans de points de vue quantité d'informations. Pour afficher tous les plans dans une seule figure, utiliser subplot. (Lisez l'aide de la fonction *bitget*)**
4. **Enregistrez l'image 'onion.png' sous le format JPEG**

II. Transformations ponctuelles sur l'image

II.1 Principe

À partir des valeurs des composantes d'une image numérique, on calcule de nouvelles valeurs définissant les composantes de l'image transformée.

$$I \xrightarrow{t} J = t(I)$$

Les transformations ponctuelles sont utilisées, souvent en visualisation, pour mettre en évidence des pixels satisfaisant à une propriété donnée.

II.2 Types de transformations

- **Ponctuelles** : les 2 images ont des dimensions identiques et les composantes d'un pixel de l'image transformée ne dépendent que de celles du pixel à la même position dans l'image initiale (ex : ajustement de luminosité, de contraste).
- **Locales :** les composantes du nouveau pixel sont calculées à partir de celles des pixels d'un voisinage du pixel initial. (ex : filtrage).
- **Globales :** afin de calculer les composantes d'un pixel de l'image transformée, toutes les composantes de tous les pixels de l'image initiale peuvent intervenir (ex : transformation de Fourier).

II.3 Exemples de transformations

- **Quantification**

Afin de pouvoir être stockée en mémoire, chacun des coefficients de l'image est quantifiée sur un certain nombre b de bits. Par exemple, si b = 1, alors il y a deux valeurs possibles pour les pixels, 0 ou 1. On dit alors que l'image est binaire. Couramment, les valeurs des pixels sont quantifiées sur b = 8 bits (1 octet), définissant ainsi 256 niveaux de gris, de 0 à 255. Traditionnellement, la valeur 0 correspond à la couleur noire, et la valeur maximale (1 dans le cas binaire, 255 dans le cas 8 bits) à la couleur blanche. La taille d'une image est ensuite calculée en multipliant le nombre de lignes de la matrice par le nombre de colonnes pour avoir le nombre de pixels de l'image. Ensuite, on multiplie le nombre de pixels par 1 octet si le codage est sur 8 bits. Néanmoins, il existe de nombreux autres espaces de couleurs :

- ✓ les espaces YUV et YCbCr qui sont construits de la même manière. Le premier terme est l'information de luminance Y et les deux autres sont dits termes de chrominance.
- ✓ L'espace HSV qui signifie Hue, Saturation, Value. La teinte correspond au type de couleur (rouge, jaune, violet...), la saturation correspond à l'intensité de la couleur et la valeur correspond-elle à la brillance de la couleur.

La signification de l'espace de couleurs HSV est la suivante :

1. H (hue) : la longueur d'onde la plus dominante
2. S (saturation) : la « pureté » d'une couleur (la quantité du blanc dans la couleur)
3. V (variance) la valeur de luminance – brightness

Manipulation 2:

1. **Tout d'abord, ouvrez avec imread de Matlab l'image 'mandrill.bmp'. Calculer la taille théorique de cette image en Kilo Octets. Vérifiez que la valeur obtenue est la même que celle fournie par Windows.**
2. **Décomposez une image RGB en un ensemble de composante de l'espace HSV et affichez-les. (indication : utilisez rgb2hsv).**
3. **Bit-plane Slicing : Soit une image I : 'onion.png' de niveau de gris codé sur 8 bits (nuance de gris varient de 0 à 255). La technique du Bit-plane slicing consiste décomposer l'image en un ensemble de plans, dans notre cas, 8 plans. Afficher les 8 plans de l'image. Commenter ces plans de points de vue quantité d'informations. Pour afficher tous les plans dans une seule figure, utiliser subplot.**

- **Echantillonnage**

Sous MATLAB, par défaut, une image $g(m,n)$ est indexée par les indices m et n (qui sont des entiers). Pour certaines images de synthèse, on définit une image $\tilde{g}(x,y)$ avec des vecteurs spatiaux x et y ne contenant plus nécessairement des valeurs entières. Il existe plusieurs conventions, mais nous allons supposer ici que le vecteur x correspond à la verticale (donc à ce qui se passe sur l'axe m) et le vecteur y correspond à l'horizontale (donc à ce qui se passe sur l'axe m).

En réalité, l'échantillonnage sous MATLAB revient à définir implicitement des vecteurs spatiaux $= 1,2,3,\cdots,M$ *et* $y = 1,2,3,\cdots,N$. On peut néanmoins, si l'on veut générer une image à partir d'une fonction, ou si l'on souhaite redimensionner une image, définir d'autres vecteurs spatiaux. Par exemple, si l'on considère $x = 1,3,5,\cdots,M$ et $y = 1,3,5,\cdots,N$, on obtient une image de taille deux fois moins grande.

Si l'échantillonnage est le même suivant les 2 directions, on parle d'isotropie. Dans le cas contraire, on parle d'anisotropie.

Manipulation 3:

1. **Nous allons dans un premier temps sous-échantillonner une image en ne prenant qu'un point sur deux dans les deux directions. Ainsi, les nombre de lignes et de colonnes vont être divisés par deux et par conséquent la taille sera divisée par quatre. Pour cela, ouvrez l'image 'mandrill.bmp'. Créez une nouvelle image sous échantillonnée en vous rappelant de l'assignation particulière des éléments d'une matrice avec Matlab. (indication : utilisez dyaddown).**
2. **Ensuite, utilisez la fonction interp2 pour sur-échantillonner par interpolation l'image précédemment sous-échantillonnée.**
3. **Afficher les deux images**

III. Traitement sur l'histogramme

L'histogramme d'une image donne la répartition de ses niveaux de gris. Par exemple, si l'on considère une image codée sur 8 bits, il s'agit de savoir combien de pixels de l'image sont égaux à 0, à 1, etc... jusqu'à 255.

Exemple:

```
X = imread('cameraman.tif'); % Ouvre l'image et la stocke dans une matrice X
X = double(X); % Convertit les pixels de l'image du format int au format  double (nécessaire pour la renormalisation)
% Pour calculer et tracer un histogramme sous MATLAB, on peut utiliser :
bins = 0:255; % Liste des niveaux de gris que l'on veut considerer
[h]=hist(X(:),bins); % Cree l'histogramme pour les niveaux definis dans le vecteur bins
bar(bins,h); % Trace l'histogramme sous forme de barres
title('Histogramme') % Donne un titre a la figure
xlabel('Niveaux de gris') % Donne un nom à l'axe des abscisses
```

- **Egalisation d'Histogramme « Histogram equalization »:**

Manipulation 4:

1. **Réaliser l'égalisation d'histogramme d'une image, puis afficher ces 2 images et leur histogramme en niveaux de gris sur une même figure.**
2. **Expliquer l'effet de l'égalisation d'histogramme sur l'image originale et son**

histogramme.

3. **Pour ce faire, Lisez l'aide des fonctions histeq et imhist.**

IV. Transformations Géométriques sur l'image

Chaque pixel de l'image est défini par sa position (i, j) et son amplitude (intensité) k dans l'image. Il existe deux types de transformations sur les pixels de l'image: les transformations géométriques qui modifient les positions des pixels (Translations, rotations, homothéties, changements d'échelle, symétries …), et les transformations qui modifient les intensités des pixels.

Exemple:

- **Rotation**

```
A=imread('flower.jpg');
figure, imshow(A);
C(:,:,1)=rot90(A(:,:,1),1);
C(:,:,2)=rot90(A(:,:,2),1);
C(:,:,3)=rot90(A(:,:,3),1);
figure, imshow(C);
```

Rotation 90°

Manipulation 5:

- **Translations**

1. **Écrire un programme qui affiche l'image 'flower' après l'application de la fonction *imtranslate* ' décalez l'image de 15 pixels dans la direction x et à 25 pixels dans la direction y '.**

Solutions

Manipulation 1 :

Lisez l'image 'onion.png'

```
A=imread('coins.png');
```

Parcourez la variable dans laquelle vous avez enregistré votre image avec la structure itérative for

```
for x =1 : size(I1,1) % I1 est une matrice, size(I1,1) veut dire : retourner le nombre de lignes de la matrice
  for y=1 :size(I1,2) % I1 est une matrice, size(I1,2) veut dire : retourner le nombre de colonnes de la matrice
        A(x,y)= 1 ; %Affectation d'une valeur dans une cellule de l'image
  end
end
```

Afficher les 8 plans de l'image. Commenter ces plans de points de vue quantité d'informations. Pour afficher tous les plans dans une seule figure, utiliser subplot. (Lisez l'aide de la fonction *bitget*)

```
B=bitget(A,1);
figure,
subplot(2,2,1);imshow(logical(B));title('Bit plane 1');
B=bitget(A,2);
subplot(2,2,2);imshow(logical(B));title('Bit plane 2');
B=bitget(A,3);
subplot(2,2,3);imshow(logical(B));title('Bit plane 3');
B=bitget(A,4);
subplot(2,2,4);imshow(logical(B));title('Bit plane 4');
figure
B=bitget(A,5);
subplot(2,2,1);imshow(logical(B));title('Bit plane 5');
B=bitget(A,6);
subplot(2,2,2);imshow(logical(B));title('Bit plane 6');
B=bitget(A,7);
subplot(2,2,3);imshow(logical(B));title('Bit plane 7');
B=bitget(A,8);
subplot(2,2,4);imshow(logical(B));title('Bit plane 8');
```

Enregistrez l'image 'onion.png' sous le format JPEG

```
imwrite(A,'onion.jpg','jpg');
```

Manipulation 2 :

Tout d'abord, ouvrez avec imread de Matlab l'image 'mandrill.bmp'. Calculer la taille théorique de cette image en Kilo Octets. Vérifiez que la valeur obtenue est la même que celle fournie par Windows.

La taille de l'image :

Chaque pixel est codé sur 8 bits. On 256x256 pixels. Alors, la taille est égale à 256x256x8 = 524288 bits

Décomposez une image RGB en un ensemble de composante de l'espace HSV et affichez-les. (indication : utilisez rgb2hsv).

```
A=imread('flower.jpg');
```

```
HSV = rgb2hsv(A);
H = HSV(:,:,1); %Hue
figure,imshow(H);colorbar;
S = HSV(:,:,2); %Saturation
figure,imshow(S);colorbar;
V = HSV(:,:,3); %variance
figure,imshow(S);colorbar;
```

Afficher les 8 plans de l'image. Commenter ces plans de points de vue quantité d'informations. Pour afficher tous les plans dans une seule figure, utiliser subplot.

```
A=imread('coins.png');
B=bitget(A,1);
figure,
subplot(2,2,1);imshow(logical(B));title('Bit plane 1');
B=bitget(A,2);
subplot(2,2,2);imshow(logical(B));title('Bit plane 2');
B=bitget(A,3);
subplot(2,2,3);imshow(logical(B));title('Bit plane 3');
B=bitget(A,4);
subplot(2,2,4);imshow(logical(B));title('Bit plane 4');
B=bitget(A,5);
figure,
subplot(2,2,1);imshow(logical(B));title('Bit plane 5');
B=bitget(A,6);
subplot(2,2,2);imshow(logical(B));title('Bit plane 6');
B=bitget(A,7);
subplot(2,2,3);imshow(logical(B));title('Bit plane 7');
B=bitget(A,8);
subplot(2,2,4);imshow(logical(B));title('Bit plane 8');
```

Manipulation 3:

Ouvrez l'image 'mandrill.bmp'. Créez une nouvelle image sous échantillonnée en vous rappelant de l'assignation particulière des éléments d'une matrice avec Matlab.

```
load mandrill
X_E=dyaddown(X,1,'m');
```

Ensuite, utilisez la fonction interp2 pour sur-échantillonner par interpolation l'image précédemment sous-échantillonnée.

```
X_S=interp2(X_E);
```

Afficher les deux images

```
imshow(uint8(X_E)) ;imshow(uint8(X_S)) ;
```

Manipulation 4:

Réaliser l'égalisation d'histogramme d'une image, puis afficher ces 2 images et leur histogramme en niveaux de gris sur une même figure.

```
I = imread('tire.tif');
```

```
J = histeq(I);
figure, imshow(I), figure, imshow(J)
```

Expliquer l'effet de l'égalisation d'histogramme sur l'image originale et son histogramme.

ça améliore le contraste.

Manipulation 5:

Écrire un programme qui affiche l'image 'flower' après l'application de la fonction imtranslate ' décalez l'image de 15 pixels dans la direction x et à 25 pixels dans la direction y '

```
A=imread('flower.jpg');
R= imtranslate(im2double(A(:,:,1)),[5, 25]);
G = imtranslate(im2double(A(:,:,2)),[5, 25]);
B= imtranslate(im2double(A(:,:,3)),[5, 25]);
 AT=cat(3,R,G,B);
imshow(AT) ;
```