Traitement des images numériques

TP 1 : Représentation des images, quantification et échantillonnage

Université Paris 13, Institut Galilée
Master Ingénierie et Innovations en Images et Réseaux - 1ère année

2017-2018

Consignes

- Récupérer le fichier `TP1.zip` sur le site

  `http://www.laurentoudre.fr/tin.html`

- Ouvrir MATLAB et créer un répertoire de travail. Dézipper le fichier `TP1.zip` dans ce répertoire.
- A la fin de la séance, récupérer les scripts que vous avez écrits et les envoyer par e-mail au chargé de TP ainsi qu'à vous même afin de les conserver pour la prochaine séance.

Rendu

- Trois fichiers : `TP1Partie1.m`, `TP1Partie2.m` et `TP1Partie3.m`. Chaque fichier doit contenir votre nom, votre prénom et la date.
- Compte-rendu succinct à rendre à la fin de la séance, contenant les observations, commentaires et réponses aux questions. Le compte rendu doit contenir votre nom et votre prénom.

Plan de l'étude

1 Rappels de cours

1.1 Représentation des images et quantification

Une image g en niveaux de gris et de dimension $M \times N$ est représentée par une matrice de taille $M \times N$ contenant les valeurs des pixels de l'image. Le coefficient $g(m,n)$ contient l'intensité du pixel à la position (m,n) : m est le numéro de la ligne (à partir du coin en haut à gauche) et n le numéro de colonne (à partir du coin en haut à gauche).

$$g = \begin{bmatrix} g(1,1) & \cdots & g(1,N) \\ \vdots & \ddots & \vdots \\ g(M,1) & \cdots & g(M,N) \end{bmatrix}$$

Afin de pouvoir être stocké en mémoire, chacun des coefficients de la matrice est quantifié 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.

$$\text{Image binaire : } g_1 = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \qquad \text{Image 8-bits : } g_2 = \begin{bmatrix} 17 & 20 & 68 \\ 251 & 91 & 173 \\ 129 & 63 & 83 \end{bmatrix}$$

Pour afficher une image sous MATLAB, il est courant de renormaliser les valeurs des pixels pour qu'elles soient comprises entre 0 et 1. Il suffit pour cela, si l'image est quantifiée sur b bits, de diviser l'image par $2^b - 1$ (qui est la valeur maximale possible pour un pixel). On dit alors que l'image est **renormalisée**.

$$\text{Image binaire renormalisée : } g_1^{renorm} = \frac{g_1}{1} = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

$$\text{Image 8-bits renormalisée : } g_2^{renorm} = \frac{g_2}{255} = \begin{bmatrix} 0.0667 & 0.0784 & 0.2667 \\ 0.9843 & 0.3569 & 0.6784 \\ 0.5059 & 0.2471 & 0.3255 \end{bmatrix}$$

1.2 Echantillonnage

L'une des toutes premières étapes du traitement numérique des images est la tâche d'échantillonnage qui réduit l'ensemble continu du monde observable en une série de valeurs discrètes. L'échantillonnage apparaît aussi en de nombreuses autres occasions, par exemple lorsqu'on redimensionne une image, lorsqu'on la convertit en un autre format, etc...

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 $x = 1, 2, 3, \cdots, M$ 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.

Etant donnés deux vecteurs spatiaux x et y, former l'image $\tilde{g}(x,y)$ revient à former l'image $g(m,n)$ telle que :

$$g(m,n) = \begin{array}{c} \\ x(1) \\ \vdots \\ x(M) \end{array} \begin{array}{c} \begin{array}{cccc} y(1) & \cdots & \cdots & y(N) \end{array} \\ \begin{bmatrix} \tilde{g}(x(1),y(1)) & \cdots & \tilde{g}(x(1),y(N)) \\ \vdots & \ddots & \vdots \\ \tilde{g}(x(M),y(1)) & \cdots & \tilde{g}(x(M),y(N)) \end{bmatrix} \end{array}$$

2 Etude sous MATLAB

2.1 Images synthétiques

1. Créer sous MATLAB un script vide nommé `TP1Partie1.m`

> On écrira en haut de chaque script le nom et le prénom de l'auteur, la date, ainsi que les commandes suivantes qui permettent de nettoyer tout l'espace de travail à chaque fois que le script est lancé :
>
> ```
> % NOM Prenom
> % Date
> clear all   % Supprime toutes les variables de l'espace de travail
> close all   % Ferme toutes les figures courantes
> clc         % Nettoie l'historique des commandes
> ```

2. Créer une image `X1` codée sur 8-bits, toute noire et de taille 200×300. L'afficher.

> - Pour créer une matrice sous MATLAB, on pourra utiliser les commandes suivantes :
>
> ```
> X = zeros(M,N);      % Cree une matrice de taille M x N ne contenant que des 0
> % M : nombre de lignes
> % N : nombre de colonnes
> X = ones(M,N);       % Cree une matrice de taille M x N ne contenant que des 1
> ```
>
> - Pour afficher une image sous MATLAB, on utilisera la commande suivante. Attention, on fera attention de bien renormaliser l'image avant de l'afficher (comme expliqué dans l'introduction).
>
> ```
> figure      % Cree une nouvelle figure
> imshow(X/255)   % Affiche l'image X 8-bits (renormalisee)
> ```

3. Créer une image `X2` codée sur 8-bits, toute blanche et de taille 200×300. L'afficher.

4. Créer une image `X3` codée sur 8-bits, unie de couleur gris clair et de taille 200×300. L'afficher.

5. Créer une image `X4` codée sur 8-bits, unie de couleur gris foncé et de taille 200×300. L'afficher.

6. Créer une image `X5` codée sur 8 bits ayant $M = 200$ lignes et $N = 300$ colonnes. Cette image a un fond gris clair et contient une bande horizontale de couleur noire et d'une largeur de 10 pixels. L'afficher.

> Pour modifier les valeurs d'une matrice sous MATLAB, on peut utiliser selon les cas :
>
> ```
> X(3,10)=0;       % Le pixel appartenant a la ligne 3 et a la colonne 10 est mis a 0
> X(4,:)=0;        % Tous les pixels appartenant a la ligne 4 sont mis a 0
> X(:,5)=0;        % Tous les pixels appartenant a la colonne 5 sont mis a 0
> X(4:9,:)=0;      % Tous les pixels appartenant aux lignes de 4 a 9 sont mis a 0
> X(:,3:8)=0;      % Tous les pixels appartenant aux colonnes de 3 a 8 sont mis a 0
> X(5:9,6:10)=0;   % Tous les pixels appartenant a la fois aux lignes de 5 a 9 et
>                  % aux colonnes de 6 a 10 sont mis a 0
> ```

7. Créer une image `X6` codée sur 8 bits ayant $M = 200$ lignes et $N = 300$ colonnes. Cette image a un fond gris foncé et contient une bande horizontale de couleur blanche et une bande verticale de couleur blanche, chacune d'une largeur de 10 pixels. L'afficher.

8. Créer une image `X7` codée sur 8 bits ayant $M = 200$ lignes et $N = 300$ colonnes. Cette image a un fond gris foncé et contient un carré gris clair de taille 30×30. Faire en sorte que le carré soit situé vers le milieu de l'image. L'afficher.

9. Créer une image `X8` codée sur 8 bits ayant $M = 200$ lignes et $N = 300$ colonnes. Cette image a un fond gris foncé et contient une croix blanche (dont vous choisirez la taille et la largeur des bandes). Faire en sorte que la croix soit située vers le milieu de l'image. L'afficher.

2.2 Histogrammes et quantification

1. Créer sous MATLAB un script vide nommé `TP1Partie2.m`

2. Ouvrir l'image `cameraman.tif` et la stocker dans une matrice `Y1`. Quelle est sa taille ? Sur combien de bits est-elle codée ? Afficher l'image.

- Pour ouvrir une image sous MATLAB, on utilise la commande :

```
X = imread('image1.bmp');   % 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 (necessaire pour la renormalisation)
```

- Pour connaître la taille d'une matrice, on utilise :

```
[M,N]=size(X);      % Renvoie le nombre de lignes M et le nombre de colonnes N
```

- Pour calculer les valeurs minimales (ou maximales) d'un vecteur (ou d'une matrice) on utilise :

```
x_min = min(x);     % Valeur minimale du vecteur x
x_max = max(x);     % Valeur minimale du vecteur x
X_min = min(min(x));    % Valeur minimale de la matrice X
X_max = max(max(x));    % Valeur minimale de la matrice Y
```

3. Pour une image en niveaux de gris, on appelle histogramme le fait de représenter le nombre (ou la proportion) de pixels ayant tels niveaux de gris en fonction du niveau 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. Tracer l'histogramme de l'image `Y1` et commenter. En particulier, trouver à quelles parties de l'image correspondent les différents pics de l'histogrammme.

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('Ma Figure')      % Donne un titre a la figure
xlabel('Niveaux de gris')   % Donne un nom a l'axe des abscisses
```

4. Agir sur les pixels pour assombrir l'image (par exemple, retirer 10 à toutes les valeurs de pixels... mais attention on ne peut pas avoir de pixels négatifs !). Afficher l'image obtenue `Y2` et son histogramme. Commenter.

Pour modifier les valeurs d'une matrice selon un critère, on utilise :

```
X(X<0)=0;   % Met toutes les valeurs negatives de la matrice a 0
X(X==9)=5;  % Met tous les pixels de l'image valant  9  a la valeur 5
X(2<=X<=10)=6;  % Met tous les pixels de l'image compris entre 2 et 10 a la valeur 6
```

5. Créer une nouvelle image `Y3` correspondant à l'image `Y1` quantifiée sur 6 bits. Afficher l'image obtenue et son histogramme. Quel est le lien entre l'histogramme de `Y1` et celui de `Y3` ? Commenter.

Etant donné une image quantifiée sur `b1` bits, on peut utiliser la commande suivante pour la re-quantifier sur `b2` bits

```
X = floor(X / 2^(b1-b2));   % Quantification de b1 bits vers b2 bits
% floor permet de calculer la partie entiere d'un nombre
```

6. Créer une nouvelle image `Y4` correspondant à l'image `Y1` quantifiée sur 4 bits. Afficher l'image obtenue et son histogramme. Qu'observe-t-on sur l'image ? Dans quelles zones ce phénomène est-il particulièrement visible ?

7. Calculer la différence `D4 = |Y4 - Y1|` (en utilisant les versions renormalisées de `Y1` et `Y4`) et l'afficher sous la forme d'une image. Confronter avec les observations de la question précédente : ces résultats sont-ils cohérents ? Comment peut-on expliquer cela ?

Pour calculer la valeur absolue d'une matrice, on utilise la commande :

```
X = abs(X); % Valeur absolue
X = abs(X).^2;  % Valeur absolue au carre
```

2.3 Echantillonnage

1. Créer sous MATLAB un script vide nommé `TP1Partie3.m`
2. Ouvrir l'image `cameraman.tif`, la renormaliser et la stocker dans une matrice `Z`. Récupérer le nombre de lignes M et le nombre de colonnes N de l'image.
   (a) Quels sont les vecteurs spatiaux `x` et `y` implicitement définis par MATLAB pour cette image ?
   (b) Si l'on souhaite créer une image de taille deux fois plus petite, quels sont les vecteurs spatiaux `x_sous` et `y_sous` que l'on doit utiliser ? Les créer.

   Pour créer un vecteur `y` ligne contenant toutes les valeurs entre `debut` et `fin` avec un pas de `pas`, on peut utiliser

   ```
   y = debut:pas:fin;
   % Exemples d'utilisation :
   x1 = 0:0.1:1;   % Vecteur ligne 0, 0.1, 0.2, ...
   x2 = 3:8;   % Vecteur ligne 3, 4, 5, ...
   x3 = (0:5)'/2;  % Vecteur colonne 0 1/2 1, ...
   ```

   Si la valeur de `pas` n'est pas spécifiée, MATLAB la fixe automatiquement à 1.

   (c) Créer et afficher l'image `Z_sous` correspondant à une version sous-échantillonnée de `Z` grâce aux vecteurs `x_sous` et `y_sous` définis précédemment. Commenter.

   Pour créer une telle image, on pourra utiliser l'instruction

   ```
   Z_sous = Z(x_sous,y_sous);
   ```

   (d) Si l'on souhaite créer une image de taille deux fois plus grande, quels sont les vecteurs spatiaux `x_sur` et `y_sur` que l'on doit utiliser ? Les créer.

   Pour créer de tels vecteurs, on peut utiliser les commandes suivantes :

   ```
   x_sur = repmat(1:M,2,1); % Repete 2 fois chaque chiffre du vecteur 1:M
   x_sur = x_sur(:);
   ```

   (e) Créer et afficher l'image `Z_sur` correspondant à une version sur-échantillonnée de `Z` grâce aux vecteurs `x_sur` et `y_sur` définis précédemment. Commenter.
3. On souhaite générer l'image de synthèse (normalisée) suivante :

$$\tilde{z}_1(x, y) = 0.5 + 0.5 \cos(2\pi(3x + 4y))$$

   Pour cela, on va utiliser des pas d'échantillonnage $\Delta x = 0.05$ et $\Delta y = 0.05$ respectivement pour les axes x et y. On souhaite générer une image ayant $M = 200$ lignes et $N = 300$ colonnes.
   (a) Définir les vecteurs spatiaux `x` et `y` à utiliser (on prendra `x(1)=0` et `y(1)=0`).
   (b) Nous allons créer une matrice `Z1` correspondant à l'image $\tilde{z}_1$. On stockera dans `Z1(m,n)` le pixel de valeur $\tilde{z}_1(x(m), y(n))$. Afin d'éviter de faire une double boucle `for` pour remplir la matrice, on peut utiliser une commande `ndgrid` qui va générer automatiquement deux matrices `X` et `Y` répliquant les valeurs de `x` et `y`.

```
[X,Y]=ndgrid(x,y);
```

$$\mathtt{X} = \begin{matrix} \mathtt{x(1)} & \cdots & \mathtt{x(1)} \\ \vdots & \ddots & \vdots \\ \mathtt{x(M)} & \cdots & \mathtt{x(M)} \end{matrix} \qquad \mathtt{Y} = \begin{matrix} \mathtt{y(1)} & \cdots & \mathtt{y(N)} \\ \vdots & \ddots & \vdots \\ \mathtt{y(1)} & \cdots & \mathtt{y(N)} \end{matrix}$$

   Cette commande, très utile, permet par exemple de générer automatiquement une matrice $g(m, n) = 3x(m) + 4y(n)$ sans avoir recours à des boucles for. Il suffira alors sous MATLAB d'utiliser la commande `ndgrid` pour générer les matrices `X` et `Y` et de définir `G=3*X + 4*Y`.
   Utiliser la commande `ndgrid` pour générer l'image `Z1` et l'afficher.

Remarque : l'image ici a ses valeurs comprises entre 0 et 1 donc il n'est pas nécessaire de la renormaliser. On pourrait avoir l'impression que cette image de synthèse n'est pas quantifiée. En réalité, par défaut, MATLAB quantifie par défaut chacune des valeurs sous le format `double` *correspondant à 64 bits.*

(c) Faire varier les pas d'échantillonnage Δx et Δy et commenter.

4. On souhaite générer l'image de synthèse (normalisée) définie par l'équation suivante :

$$\tilde{z}_2(x,y) = \begin{cases} 0.8 & \text{si } (x-100)^2 + (y-150)^2 < 120 \\ 0.2 & \text{sinon} \end{cases}$$

Pour cela, on va utiliser des pas d'échantillonnage $\Delta x = 1$ et $\Delta y = 1$ respectivement pour les axes x et y. On souhaite générer une image ayant $M = 200$ lignes et $N = 300$ colonnes. Générer l'image `Z2` et l'afficher. Quelle est la signification du point $(x, y) = (100, 150)$? Et de la valeur 120 ?