

Utiliser des images sous MATLAB

1 Les images à TELECOM ParisTech

L'existence de l'équipe "Image" de TELECOM ParisTech remonte à près de 30 ans. Aussi il existe un format d'image spécifique puisqu'il n'existait pas les standards actuels d'archivage. De plus, nous traitons des images de format peu courant (données complexes en flottant ou en double précision par exemple...). Voilà pourquoi nous continuons à maintenir ce format.

1.1 Images "classiques" sur 8 bits

Une image est archivée avec deux fichiers :

- l'image stricto sensu, dans un fichier sans en tête, en balayage video. Son extension peut dépendre des données. Pour une image Noir et Blanc (NB) classique (8 bits), cette extension est `.ima`.
- des données auxiliaires codées en ASCII (on peut donc éditer le fichier avec n'importe quel éditeur). Le format le plus simple consiste à écrire le nombre de colonnes, puis le nombre de ligne (en entier). L'extension de ce fichier est `.dim`.

Il est très facile donc d'ouvrir une image au format TELECOM ParisTech, d'autant qu'elle n'est donc pas comprimée.

1.2 Autres images

Les images "classiques" Noir et Blanc (celles du grand public) sont archivées en 8 bits. Cependant, les images peuvent être en couleur, sur 16 bits, ...

Nous verrons cet aspect au paragraphe 3, d'autant que des problèmes complexes de plateforme (big endian, small endian) perturbent sévèrement la mise en œuvre. Pour les images en couleur, un exemple est donné 3.3.

2 Quelques ordres Matlab

2.1 Lecture sous Matlab des images NB 8 bits

Soit une image, avec donc deux fichiers. On prend par exemple l'image `jour1` : elle est décrite par ses fichiers `jour1.dim` et `jour1.ima`.

Pour lire ce type de données sous Matlab, on dispose d'une procédure (`.m`) : `ima2mat.m` dont la syntaxe, appliquée par exemple à la donnée `jour1` est la suivante :

```
im1 = ima2mat('jour1');
```

L'image est alors accessible en tant que tableau Matlab `im1`. Les dimensions de cette matrice peuvent se récupérer ainsi :

```
nlig = size(im1,1)
ncol = size(im1,2)
```

2.2 Visualisation d'une image

Ici `image` représente des données codées en entier entre 0 et 256.

- Visualiser une image peut s'effectuer avec l'ordre `image`

```
image(im1);
```

- une image en niveau de gris doit utiliser la colormap `gray` :

```
colormap(gray)
```

Cette colormap par défaut est sur 100 niveaux de gris.

Une image en 256 niveaux de gris nécessite une colormap sur 256 niveaux :

```
colormap(gray(256))
```

Si l'on veut se contenter de nn niveaux :

```
colormap(gray(nn))
```

2.3 Visualisation d'une matrice

Une matrice `imz` a des valeurs pas nécessairement entières, ni réelles (exemple : une transformée de Fourier...).

Aussi, il faut prévoir deux étapes :

- passer en réel si besoin

```
aimz = abs(imz);
```

- modifier linéairement les valeurs pour qu'elles soient comprises entre 0 et 255. Pour cela, on recherche le min et le max, puis on normalise la matrice

```
zmin = min(imz(:));  
zmax = max(imz(:));  
znorme=255/(zmax-zmin);  
nimz = (imz-zmin)*znorme;  
image(nimz);
```

2.4 Histogramme d'une image

L'histogramme d'un tableau (`im1`) peut se calculer directement avec `hist`

```
hist(im1(:))
```

ou, sur 256 niveaux par pas de 1 entre 0 et 255 :

```
hist(im1(:),0:1:255)
```

Cela donne des résultats différents : méfiez vous de `hist` utilisé sans argument complémentaire et consultez le "help" en ligne.

2.5 Transformée de Fourier d'une image

Matlab donne directement la possibilité d'effectuer une transformée de Fourier sur un signal 2-D (matrice Z).

- `fft2` est la procédure de base. A partir d'une matrice de réels, elle donnera une matrice de complexes. On peut utiliser ou non les arguments 2 et 3 :
 - `fft2(Z)` donnera un résultat ayant le même nombre de lignes et de colonnes que la matrice initiale Z .
 - `fft2(Z,p,q)` donnera un résultat sur p lignes et q colonnes, la matrice initiale étant complétée par des valeurs nulles (ce qui donne un pas fréquentiel plus petit, et donc plus de précision sur l'allure du spectre).

Attention, comme la procédure `fft`, le spectre est donné entre 0 et 1 (en fréquence normalisée). Pour l'obtenir entre -0.5 et 0.5, il faut utiliser `fftshift`

- `ifft2` permet d'obtenir la transformée de Fourier Inverse 2-D. Attention `ifft2(fft2(Z))` donne en général un résultat en complexes même si Z est une matrice de réel (erreurs d'arrondis).
- `fftshift` transforme une FFT (1-D ou 2-D) placée entre 0 et 1 (fréquence normalisée) en un résultat entre -0.5 et 0.5. Cette procédure s'applique tant en 1-D qu'en 2-D.
- **RAPPEL** : `abs`. ne pas oublier qu'un spectre est complexe. Pour le visualiser, en règle générale, on prend la norme. `abs` transforme une matrice complexe en une matrice réelle : on peut alors la visualiser comme on le souhaite.

Exemple

Soit une matrice Z réelle 2-D. Pour visualiser sa FFT-2D, avec la fréquence (0,0) au milieu de la représentation, il suffit de taper l'ordre

```
surf(abs(fftshift(fft2(Z))));
```

2.6 Multiplication sous Matlab

MATLAB est un outil dédié aux matrices : a priori, un produit est matriciel et s'effectue avec l'ordre `*`.

Pour effectuer un produit élément par élément de deux matrices, il faut utiliser l'ordre `.*`.

2.7 Autre procédure utile

`conj` donne le conjugué d'une matrice.

3 Images diverses et autres formats

3.1 Lecture sous Matlab des images NB 16 bits : format Solaris

Soit une image 16 bits, avec donc deux fichiers. On prend par exemple l'image `jour1` : elle est décrite par ses fichiers `jour1.dim` et `jour1.imw`.

Pour lire ce type de données sous Matlab, on dispose d'une procédure (.m) : `imw2mat.m` dont la syntaxe, appliquée par exemple à la donnée `jour1` est la suivante :

```
im1 = imw2mat('jour1');
```

L'image est alors accessible en tant que tableau Matlab `im1`. Mais attention, ce tableau a des valeurs comprises entre 0 et 65535. Les dimensions de cette matrice peuvent se récupérer ainsi :

```
nlig = size(im1,1)
ncol = size(im1,2)
```

Pour afficher cette image avec Matlab, il faut se reporter à la section 2.3.

3.2 Lecture sous Matlab des images NB 16 bits : format PC

Soit une image 16 bits, avec donc deux fichiers. On prend par exemple l'image `jour1` : elle est décrite par ses fichiers `jour1.dim` et `jour1.IMW`.

Pour lire ce type de données sous Matlab, on dispose d'une procédure (.m) : `imw2mat.m` dont la syntaxe, appliquée par exemple à la donnée `jour1` est la suivante :

```
im1 = IMW2mat('jour1');
```

L'image est alors accessible en tant que tableau Matlab `im1`. Mais attention, ce tableau a des valeurs comprises entre 0 et 65535. Les dimensions de cette matrice peuvent se récupérer ainsi :

```
nlig = size(im1,1)
ncol = size(im1,2)
```

Pour afficher cette image avec Matlab, il faut se reporter à la section 2.3.

3.3 Images couleur RVB

Matlab peut afficher une image en "vraies couleurs". Pour cela il faut créer un tableau de dimension 3, la troisième composante ayant une taille 3 (3 couleurs, dans l'ordre RVB).

Voici un exemple où les trois canaux sont `ima`, `imb` et `imc`.

```
s=size(ima);
imrvb=uint8(zeros(s(1), s(2), 3));
imrvb(:,:,1) = uint8(ima);
imrvb(:,:,2) = uint8(imb);
imrvb(:,:,3) = uint8(imc);
image(imrvb);
```

3.4 Autres images

Divers formats d'images coexistent à Télécom ParisTech. Ils se reconnaissent principalement par l'extension des fichiers de données, qui est en minuscule lorsque le fichier a été généré à partir d'une plateforme Solaris (big endian) et majuscule lorsque le fichier a été généré par une plateforme PC (small endian).

3.4.1 Architectures Sun/Motorola (big endian)

Voici quelques extensions pour les fichiers big endian :

- .ima : unsigned char (1 octet)
- .imw : unsigned short (2 octets)
- .rvb : 3*unsigned char (couleurs)
- .ims : signed short (2 octets)
- .iml : int (long) (4 octets)
- .imf : float (4 octets)
- .imd : double (8 octets)
- .cxb : complex signed byte (2x 1 octet)
- .cxs : complex signed short (2x 2 octets)
- .xcf : complex float (2x 4 octets)
- .xcd : complex double (2x 8 octets)

Vous trouvez dans ce répertoire les procédures (.m) suivantes, à utiliser exclusivement sur les plateforme big endian :

- **ima2mat.m** : lit une image 8 bits (unsigned char) au format TSI (fichier .ima : les octets, fichier .dim : le nombre de colonnes puis de lignes en ASCII),
- **imw2mat.m** : lit une image 16 bits (unsigned short) au format TSI (fichier .imw : les octets, fichier .dim : le nombre de colonnes puis de lignes en ASCII),
- **imf2mat.m** : lit une image 32 bits (float) au format TSI (fichier .imf : les octets, fichier .dim : le nombre de colonnes puis de lignes en ASCII),
- **rvb2mat.m** : lit une image couleur (3 * unsigned char)
- **cxb2mat.m** : lit une image complexe 8 bits (signed char pour la partie réelle, signed char pour la partie imaginaire) au format TSI (fichier .cxb : les char, fichier .dim : le nombre de colonnes puis de lignes en ASCII).
- **cxs2mat.m** : lit une image complexe 16 bits (signed short pour la partie réelle, signed short pour la partie imaginaire) au format TSI (fichier .cxs : les short, fichier .dim : le nombre de colonnes puis de lignes en ASCII).
- **xcf2mat.m** : lit une image complexe 32 bits (float pour la partie réelle, float pour la partie imaginaire) au format TSI (fichier .xcf : les float, fichier .dim : le nombre de colonnes puis de lignes en ASCII).

3.4.2 Architectures PC/Intel (small endian)

Voici quelques extensions pour les fichiers small endian :

- .IMW : unsigned short (2 octets)
- .IMS : signed short (2 octets)
- .IML : int (long) (4 octets)
- .IMF : float (4 octets)
- .IMD : double (8 octets)
- .CXB : complex signed byte (2x 1 octet)
- .CXS : complex signed short (2x 2 octets)
- .CXF : complex float (2x 4 octets)
- .CXD : complex double (2x 8 octets)

Vous trouvez dans ce répertoire les procédures (.m) suivantes, à utiliser exclusivement sur les plateformes small endian :

- `IMW2mat.m` : lit une image 16 bits (unsigned short) au format TSI (fichier .IMW : les octets, fichier .dim : le nombre de colonnes puis de lignes en ASCII),
- `IMF2mat.m` : lit une image 32 bits (float) au format TSI (fichier .IMF : les floats, fichier .dim : le nombre de colonnes puis de lignes en ASCII),
- `CXS2mat.m` : lit une image complexe 16 bits (signed short pour la partie réelle, signed short pour la partie imaginaire) au format TSI (fichier .CXS : les short, fichier .dim : le nombre de colonnes puis de lignes en ASCII).
- `CXF2mat.m` : lit une image complexe 32 bits (float pour la partie réelle, float pour la partie imaginaire) au format TSI (fichier .CXF : les float, fichier .dim : le nombre de colonnes puis de lignes en ASCII).

Bien évidemment, les images ne faisant intervenir qu'un octet sont lues indifféremment sur les deux plateformes avec les mêmes procédures.

4 Écriture d'images au format TSI

4.1 Plateforme Big Endian

Vous trouvez dans ce répertoire les procédures (.m) suivantes :

- `mat2ima.m` : écrit une matrice en image 8 bits (unsigned char) au format TSI (fichier .ima : les octets, fichier .dim : le nombre de colonnes puis de lignes en ASCII),
- `mat2imw.m` : écrit une matrice en image 16 bits (unsigned short) au format TSI (fichier .imw : les shorts, fichier .dim : le nombre de colonnes puis de lignes en ASCII),

- **mat2rvb.m** : lit une matrice en image 3*8 bits (unsigned char) au format TSI (fichier .ima : les octets, fichier .dim : le nombre de colonnes puis de lignes en ASCII).
- **mat2cxs.m** : écrit une matrice complexe en image complexe 16 bits (signed short) au format TSI (fichier .cxs : les floats, fichier .dim : le nombre de colonnes puis de lignes en ASCII),
- **mat2cxf.m** : écrit une matrice complexe en image complexe 32 bits (float) au format TSI (fichier .cxf : les floats, fichier .dim : le nombre de colonnes puis de lignes en ASCII).

4.2 Plateforme Small Endian

Vous trouvez dans ce répertoire les procédures (.m) suivantes à utiliser depuis une plateforme Small Endian (PC) :

- **mat2ima.m** : écrit une matrice en image 8 bits (unsigned char) au format TSI (fichier .ima : les octets, fichier .dim : le nombre de colonnes puis de lignes en ASCII),
- **mat2IMW.m** : écrit une matrice en image 16 bits (unsigned short) au format TSI (fichier .IMW : les short, fichier .dim : le nombre de colonnes puis de lignes en ASCII),
- **mat2CXS.m** : écrit une matrice complexe en image complexe 16 bits (signed short) au format TSI (fichier .CXS : les float, fichier .dim : le nombre de colonnes puis de lignes en ASCII),
- **mat2CXF.m** : écrit une matrice complexe en image complexe 32 bits (float) au format TSI (fichier .CXF : les float, fichier .dim : le nombre de colonnes puis de lignes en ASCII).