UNIVERSITÉ DE RENNES 1

Module VIS - Master MITIC
TP1 : Prise en main de Matlab

O. LE MEUR
olemeur@irisa.fr

2013-2014

But du TP :
- Rappel/initiation au calcul matriciel sous Matlab
- Manipulation des images sous Matlab.

Travail à rendre :
Le travail est à rendre pour la semaine suivante. Pour chaque question, on donnera les commandes et traitements effectués, les images et valeurs obtenues et les commentaires correspondants.

L'ensemble doit être rendu *sous forme électronique*(au format PDF, Word ou Open Office). Ce compte rendu est à envoyer à votre encadrant : `olemeur@irisa.fr`. Si vous travaillez à deux, le nom du fichier doit inclure les noms des binômes.

Avant de commencer :
récupérer l'intégralité de l'archive située sur *http ://people.irisa.fr/Olivier.Le_Meur/teaching/MITIC/*

A savoir :
- N'hésitez pas à vous servir de l'aide par la commande **"help"** sous Matlab, également accessible par le menu, "Help", "Product Help" la connaissance par coeur des instructions d'un logiciel étant illusoire
- Tips and tricks, document intéressant et qui peut aider : *http ://www.ee.columbia.edu/ marios/matlab/tips.pdf*
- Entre chaque exercice, il est conseillé de lancer la commande **"clear"** de manière à vider le workspace et fermer toutes les figures.

1 Rappel/Initiation au calcul matriciel sous Matlab

Le logiciel MATLAB © [matrix laboratory] est à la fois un langage de programmation et un environnement de développement (IDE) utilisé entre autre pour les calculs vectoriels et matriciels.

1.1 Vecteurs, 1 dimension

On peut créer un vecteur très facilement en faisant : "$nom\_vecteur = debut : pas : fin$"

1. Créer un vecteur allant de 0 à 1 par pas de 0.2. : `a=0:0.2:1` Sachant que `b= 3*a+2`, affichez `a` et `b, a(1)` et `a(2)`.
2. Calculer la longueur de `a`.
3. Calculer `a+b` et `a-b`
4. Taper `c = [1,2,3], d= [c,4,5,6]`et `e=d(1:2)` calculer leurs longueurs, expliquez l'utilité de ces opérations
5. Calculer `sum(a), prod(a), mean(a)`.
6. Comparer : `a'*b, a*b',a.*b`. Quelle est la différence entre ces 3 opérations ?

A savoir L'extension classique d'un fichier MATLAB est `.m`. On peut trouver 2 types de fichiers .m : les fichiers de fonctions (qu'on peut utiliser comme en programmation impérative classique) et les fichiers script qui sont un ensemble de commandes enchainées. Une fonction peut être appelé avec des paramètres d'appels soit dans un script soit dans une autre fonction. Un script s'exécute comme suit :
- dans la fenêtre de commande, en tapant `nom_du_fichier` (sans l'extension `.m`) en vous assurant que vous êtes dans le bon répertoire ou que la liste des chemins est correcte.

– en sélectionnant certaines lignes du script dans la fenêtre d'édition et en tapant F9 (pratique pour exécuter une partie isolée d'un script).

1.2 Affichage et visualisation : Courbes 2D

Dans un nouveau fichier `.m`, (fichier, nouveau, M-file), taper les lignes suivantes

```
x=-pi:0.1:pi;
y=sin(x);
plot(x,y);
grid
xlabel('x'),ylabel('y')
title('y = f(x)')
```

Sauvegarder ce fichier sous *essai.m*. Dans la fenêtre de commande, taper `essai`. Expliquer succintement le principe.
On peut diviser l'écran d'affichage en sous-parties avec la commande `subplot`(voir help subplot). Créer un fichier Matlab qui fait les choses suivantes :

1. diviser l'écran en 2 parties
2. tracer la courbe $y = sin(2 * \pi * 5 * t)$ *pour* $0 < t < 1$ dans la partie supérieure
3. tracer la courbe $y = f(t)$ *pour* $t = 1 : 1000$ *avec* $y = randn(size(t))$, soit un vecteur de dimension 1000 contenant des échantillons tirés aléatoirement) dans la partie inférieure.

1.3 Matrices

La manipulation de matrices est pratique et rapide avec Matlab. Le passage d'une ligne à une autre se fait par ‘‘;’’

Création

1. Ainsi, taper `A=[1,2 ;3,4] B=[3,4 ;5,6]`
2. Faire `A+B`, `A-B`
3. Comparer `A*B` et `A.*B`
4. Calculer `inv(A)`, la matrice inverse de A et `A*inv(A)`
5. Calculer `A', size(A), det(A) et trace(A)` (somme des élèments diagonaux). Vérifiez ces résultats par un calcul à la main.
6. **Bonus :** Calculer les vecteurs propres et la valeurs propres de la matrice A (‘‘`eig`’’)
7. **Ultra-Bonus :** Soit C la matrice des vecteurs propres, S la matrice des valeurs propres vérifier que $A = C * S * C^{-1}$

Manipulation de matrices

1. Faire `sum(A),sum(A')`
2. Faire `A(1,:), A(:,1), A(:)` Noter les opérations effectuées
3. Extraire la première colonne et première ligne de `A`.
4. Calculer la moyenne des élèments de `A` et sa valeur max.
5. Taper `D=[A;5,6]`. Que constatez-vous ?
6. On peut extraire les deux dernières lignes de D, en faisant `E=D(2:3,:)`. Extraire les 2 premières de D et la première.

2 Représentation d'une image sous Matlab

Sous Matlab, les images sont représentées par des matrices. Pour une image en niveaux de gris, chaque élément de la matrice représente une intensité, tandis que pour une image couleur, chaque élément de la matrice est un vecteur d'intensité (en général de trois coefficients : rouge, vert, bleu).

A partir de cette section et pour les prochains TP traitants de l'image, nous serons amenés à utiliser la toolbox Image Processing. Encore une fois, n'hésitez pas à consulter l'aide de cette toolbox, les images illustratives parlant d'elle-mêmes.

La plupart des fonctions de cette toolbox (et des logiciels de traitements d'image), travaille avec le type *uint8* pour représenter la valeur des pixels, alors que par défaut sous Matlab, tout est matrice de *double*. Procéder donc au trans-typage autant que nécessaire.(le *type* de vos données étant affiché dans les *workspace*).

2.1 Visualisation et affichage

Commandes et remarques utiles

`imread`, `imshow`, `imfinfo`, `size`, `std2`, `mean2`, `max`, `min`, `sum`

- Pour obtenir des informations sur ces fonctions, vous pouvez consulter l'aide. Pour éviter d'écraser les figures les unes sur les autres (sans passer par `clear`), vous pouvez faire précéder les fonctions d'affichage (`imshow`...) de `figure,` (exemple : `figure, imshow('lena.pgm')`).
- Il est conseillé de créer un nouveau fichier M-file dans lequel vous allez saisir vos commandes et donc chaque ligne (se terminant par un ``;'') sera ensuite interprétée.

Image en niveau de gris

1. Charger et afficher l'image *lena.pgm* sous matlab.
2. Quelle est la taille de cette image ? Sa dynamique ? Quelle place occupe le fichier ?
3. Calculer :
   - La variance de l'image (utiliser la commande `reshape`).
   - Ses intensités moyenne, maximale et minimale.

Couleurs et LUT

1. Charger l'image *clown_lumi.bmp* en niveau de gris.
2. Essayer différents affichages avec les fonctions `image, imagesc` et `imshow`. Observez les différences, faites également un essai en trans-typant vos données.(`image = double(image)`).
3. Pour l'affichage d'une image monochrome (tableau 2-D) Matlab utilise une LUT par défaut qui associe à chaque élément de la matrice image une couleur. Cette LUT par défaut présente 64 couleurs différentes en sortie (commande `colormap` pour afficher les niveaux de la LUT par défaut, voir l'aide).

   Le fichier script `lutndg.m` crée une LUT pout afficher une image monochrome en niveau de gris (ndg). Après avoir ouvert et compris ce fichier, affichez une image monochrome de votre choix et tapez la commande `lutndg.m`
4. Inspirez-vous du fichier pour synthétiser une LUT pour faire l'inversion vidéo.
5. Charger l'image couleur *clown.bmp*, observer le type de données et visualiser l'image. Visualisez-la composante par composante en créant éventuellement les LUTs ad'hoc pour les plans Rouge, Vert, Bleu.

2.2 Histogramme et ajustement de contraste

1. Utiliser la commande `imhist` pour calculer et afficher l'histogramme de différentes images. Visualiser et comparer les histogrammes de plusieurs images. Attention, pour les images couleurs, il faut utiliser la fonction plan par plan.
2. Utiliser la fonction d'égalisation d'histogramme sur des images avec `histeq`. Commenter les modifications apportées sur l'histogramme (`imhist`).