TP1 : Initiation à Matlab®

Manipulation d'une image

Image en couleur

Image en couleurs indexés

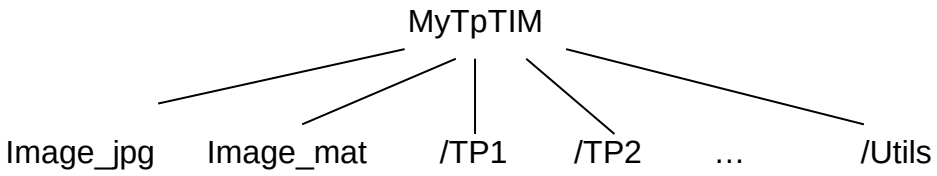
Préliminaires

Matlab® est un environnement de travail pour le calcul scientifique. Il est associé à un langage interprété de programmation impérative, orienté pour la manipulation des tableaux (vecteurs, matrices). Il offre de nombreuses fonctionnalités graphiques pour la représentation de courbes. Les commandes (instructions) peuvent être (i) entrées directement, une à une dans la fenêtre de travail, ou bien le plus pratique, (ii) écrites dans un fichier texte portant l'extension .m. Le programme ainsi décrit dans ce fichier sera exécuté, en entrant dans la fenêtre de travail, le nom du fichier.
Nous allons présenter les principales fonctionnalités qui seront utiles pour la suite des TPs de l'UE.

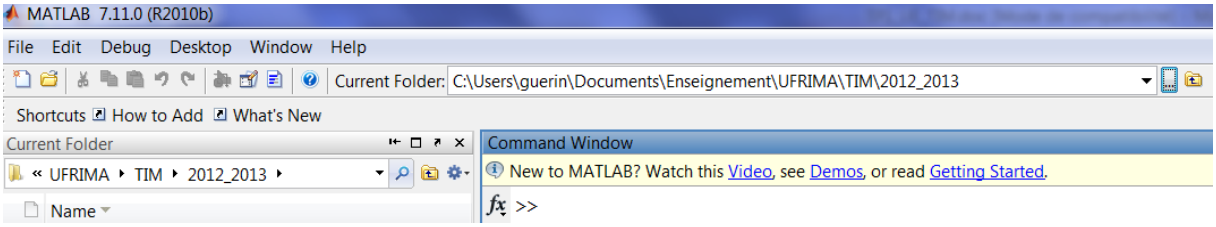
Remarque importante :

Créer sous Windows un répertoire de travail qui sera utilisé à chaque TP. Se positionner dans ce répertoire à chaque session Matlab. Télécharger depuis l'environnement « Moodle » le répertoire 'Utils' comportant des fonctions utiles aux TPs, et les 2 répertoires d'images. De même récupérer le texte de TP et les programmes Matlab qu'il faudra compléter durant cette première séance.

Vous devez avoir l'arborescence suivante dans votre espace de travail :



Lancer l'environnement Matlab®.
Dans la boîte de dialogue en haut «Current Folder », naviguer pour désigner l'emplacement du répertoire TP1, dans lequel il y a les fichiers téléchargés depuis Moodle.



Représentations des scalaires et des tableaux (matrices, vecteurs)

Entrer une à une les instructions suivantes dans la fenêtre de travail (« return » à la fin de chaque instruction). Observer le résultat et noter ce qui a été réalisé.

`a=5`	
`b=6 ;`	
`b`	
`a=[1 2 ; 3 4 ; 5 6]`	
`[lig, col]=size(a) ;`	
`lig`	
`col`	
`a(1, 2)`	
`a(1,  :)`	
`a( :, 2)`	
`t=(0 :0.1 :10);`	
`size(t)`	
`length(t)`	
`t(1 :10)`	
`b=[10 ;20 ;30]`	
`b'`	
`a=a(1 : 2, 1 : 2)`	
`b=10*ones(2, 2)`	
`c=a*b`	
`c=a.*b`	

Format d'image PGM

Les deux fichiers `image_a.pgm` et `image_b.pgm` représentent la même image sous deux formats différents. Ces deux fichiers seront placés dans le répertoire de travail où se trouvent tous les programmes Matlab (fichiers d'extension .m). Noter la taille respective de ces 2 fichiers et éditer ces deux fichiers sous Windows. Comment l'image est-elle sauvegardée dans chacun des deux fichiers ?

Lire, transformer et afficher une image

Sous Windows, placer les images dans un répertoire dédié à cela. Appeler ce répertoire `Image_jpg` et placer le au même niveau que votre répertoire de travail. Télécharger le programme `TP_lire_image.m`. Ce programme ne peut pas s'exécuter, il faut d'abord le

compléter. Editer le programme et le lire pour comprendre les différents traitements qui sont réalisés. Ecrire l'instruction qui permet d'accéder à la valeur de luminance d'un pixel sur une ligne et une colonne particulière. Ecrire les deux instructions qui permettent d'accéder à un profil « ligne » et à un profil « colonne » pour un numéro de ligne et un numéro de colonne fixé. Ecrire ces trois instructions aux endroits indiqués dans le programme. Exécuter alors le programme et répondre aux questions ci-dessous :

- Quelle est la taille de l'image, en ligne et en colonne ?
- Entre quelles valeurs minimale et maximale varient les niveaux de gris ?
- Quelle est la valeur du niveau de gris du pixel à la ligne 25 et la colonne 82 ?
- Relever les profils de niveaux de gris de la ligne 25 et de la colonne 82 et affichez-les.
- Comment a été réalisée l'image appelée `imaOut` ? Que représente-t-elle ?

Transformation d'une image couleur en niveaux de gris

Télécharger et ouvrir le fichier `TP_couleur.m`. Laisser vous guider par le programme que vous devez compléter, puis répondre aux questions.

En particulier :

- Quelle est la taille de l'image renvoyée par l'instruction `size(ima)`? Comment comprenez-vous ce résultat ?
- Expliquer l'instruction pour créer l'image en niveau de gris à partir de l'image couleur.
- Transformer ces 2 images (en couleur et en niveau de gris) par le procédé de « vidéo inversé ». Une image en vidéo inversée se construit en affectant une forte valeur (niveau clair) à un niveau sombre et inversement et cela selon une loi linéaire telle que illustrée ci-dessous. Expliquer à quoi s'applique cette fonction. A quoi est égal Out_{Max} ? Même question pour In_{Max} ? Donner l'équation de cette droite. Ecrire les instructions nécessaires dans le programme pour effectuer les transformations demandées. Afficher et commenter le résultat obtenu.

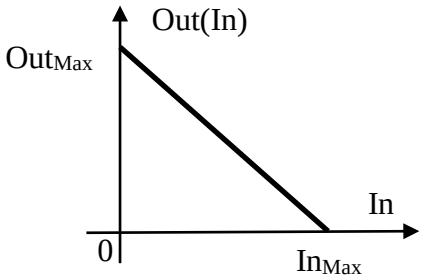
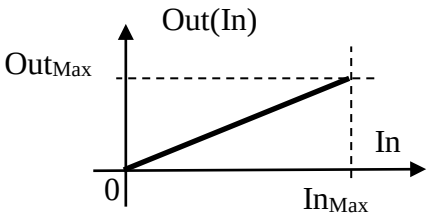


Table des niveaux de gris

Télécharger le fichier `TP_colormap.m`. Compléter le chemin d'accès, sur le répertoire des images au format jpeg .
Laisser vous guider par le fichier. L'objectif est de comprendre l'utilisation des tables de couleurs « gray » avec l'instruction `colormap` :

1. Comprendre le contenu de la table de couleurs « gray »
2. Afficher l'image en 32 niveaux de gris.

Pour écrire correctement la transformation d'image permettant d'obtenir une image sur 32 niveaux de gris, il faut comprendre préalablement la fonction de transformation telle que ci-dessous à appliquer à tous les pixels de l'image. Donner l'équation de cette transformation et en déduire l'instruction à compléter dans le programme.



Quantification et Indexation des couleurs

Télécharger le fichier TP_quantif.m. Vérifier que le répertoire 'Utils' est bien au même niveau que votre répertoire de travail.

L'objectif de ce programme est de comprendre le mécanisme de la quantification des couleurs et celui des couleurs indexées. La quantification va consister à définir un nombre restreint de couleurs choisi au mieux pour repésenter une distribution donnée de couleurs. Cette opération de quantification est réalisée par l'algorithme des Kmoyennes (fonction `KmeansCluster`) qui consiste à calculer les K (paramètre à fournir) couleurs les plus représentatives à partir d'un ensemble d'apprentissage (tableau à constituer, à fournir en entrée de la fonction). Les K couleurs obtenues sont numérotés de 1 à K. Soit C, cet ensemble des K couleurs prototypes: $C = \left\{ \vec{c}_k \in \Re^3; k = 1..K \right\}$. L'opération d'indexation des couleurs consiste à remplacer pour chaque pixel (i,j) de l'image, son triplet {r(i,j); v(i,j); b(i,j)} par le numéro de la couleur représentative la plus proche au sens d'une métrique fixée à l'avance (ici la distance euclidienne).

$$k^* = f(i,j) = Arg\left\{ Min_k \left[dist\left(\overrightarrow{rvb}(i,j), \vec{c}_k \right) \right] \right\}$$

Cette équation veut dire que l'on va calculer la distance euclidienne entre la couleur du pixel (i,j) avec chacune des K couleurs prototypes et que l'on va associer au pixel (i,j) le numéro k* de la couleur protoype qui est la plus proche de la couleur du pixel (i,j).

Dans ce TP, le nombre de couleurs sera d'abord fixé à 64, variable « `nbCoul` », puis le faire varier et interpréter le résultat obtenu.

1°) La première étape est de former la base d'apprentissage des couleurs. C'est un tableau dont le nombre de lignes est égal au nombre choisi de pixels pour former la base et le nombre de colonnes est égal à 3. Les pixels pour cette base d'apprentissage seront pris aléatoirement dans l'image. On n'en sélectionnera que 10% des pixels (sous-echantillonnage aléatoire). Pour cela, on utilise les fonctions `reshape` et `randperm`. Il est demandé ici de bien comprendre le code fourni qui permet de créer cette base d'apprentissage. Expliquer le contenu de la variable `couleur` en sortie de la fonction `KMeansCluster`.

2°) Comprendre la séquence d'instruction réalisant l'indexation des couleurs de l'image :

```
d = dist2(data, couleur);
[temp,indexCoul]=min(d,[],2);
```

```
imaIndexCoul = reshape(imaIndexCoul, lig, col);
```

Que représentent les informations contenues dans la variable `imaIndexCoul`? Quelles différences y a-t-il entre les variables `indexCoul` et `imaIndexCoul` ? Quelle est la valeur minimale ? Quelle est la valeur maximale ?

3°) Créer alors une table de couleur utilisable par la fonction `colormap`
L'image originale est affichée à la figure 1, l'image en couleurs indexées en utilisant cette table de couleurs est affichée à la figure 4. Comment interprétez-vous la différence de rendu entre ces 2 affichages ? Expliquer l'affichage réalisé à la figure 3.

4°) Créer une image au même format que l'image originale (sur trois composantes rouge, vert, bleu) avec seulement les K couleurs choisies. Utiliser comme nom de variable `imaQuantifRGB`. Afficher cette image. Le résultat sera obtenu à la figure 5. Avec quelle table de couleurs cette image sera-t-elle affichée ? Comparer le résultat rendu perceptif avec l'affichage à la figure 4 ? Le résultat était-il attendu ? Expliquer. Comparer la taille en octets entre l'image `imaIndexCoul` et l'image `imaQuantifRGB`.

5°) Exécuter de nouveau le programme complet en faisant varier le nombre de couleurs. Conclure.