# Traitement d'images sous Matlab

Ecole d'Ingénieurs du Pas-de-Calais

## EI3
## année 2009-2010




**Vandenbroucke Nicolas** **R11**

# 1 – Acquisition d'images sous Matlab

L'objectif de ce TP est de prendre en main les outils de traitement d'images les plus classiques à l'aide du logiciel Matlab. Matlab est un logiciel de calcul scientifique permettant de développer des solutions à des problèmes techniques. Il permet de réaliser du calcul numérique et de tracer des graphiques pour visualiser et analyser les données. Il dispose d'un langage et d'un environnement de programmation interactifs ainsi que d'outils pour concevoir des interfaces utilisateur graphiques. Matlab est associé à des boites à outils appelé TOOLBOX permettant d'accéder à des fonctions spécifiques à un domaine d'application comme le traitement d'images par exemple.

Les TP de traitement d'images réalisés avec Matlab nécessitent ainsi la toolbox **Image Acquisition** et la toolbox **Image Processing**. Les fonctions de cette dernière toolbox peuvent être listée en tapant la fonction `help images` dans l'éditeur de commande de Matlab. Pour obtenir un descriptif détaillé des ces fonctions, utiliser l'aide contextuelle en tapant la fonction `help` suivi du nom de la fonction ou utiliser l'aide en ligne en ouvrant l'application **Helpdesk** (en tapant la fonction `helpdesk`, par exemple). Avant de parcourir directement le sujet, vous pouver découvrir les démonstrations associées au toolbox **Image Processing** en tapant la fonction `demos` (ou directement le nom de la démonstration à découvrir).

## 1 Prise en main

Ce premier exercice est destiné à prendre en main les toolbox **Image Processing** et **Image Acquisition**.

### 1.1 Rappel sur Matlab

La figure 1 montre la décomposition de Matlab en plusieurs fenêtres :

- une fenêtre d'édition des commandes (**Command Window**),
- deux fenêtres contenant un onglet de visualisation de l'espace des variables (**Workspace**), un onglet de visualisation des fichiers du répertoire de travail (**Current Directory**) et un onglet de visualisation de l'historique des commandes (**Command History**).
- une fenêtre d'édition avec un onglet permettant de visualiser le contenu de fichiers (**Editor**) et un onglet permettant de visualiser le contenu des variables (**Array Editor**).

Les fonctions sont éditées dans la fenêtre de commandes et exécutées en appuyant sur la touche **Entrée**. Le point virgule à la fin d'une fonction permet d'éviter d'afficher les données résultats de la fonction exécutée ou de séparer plusieurs fonctions sur une même ligne de commande. Plusieurs fonctions et commandes peuvent être saisies dans un fichier qui sera enregistré avec l'extension **.m**. En éditant le nom de ce fichier dans la fenêtre de commande, l'ensemble des fonctions déclarées dans ce fichier seront exécutées. L'édition de ce fichier peut s'effectuer en sélectionnant dans le menu **File** : **New ▶ M-file**. Il est également possible d'y créer des fonctions en utilisant la commande `function`.

La fonction `figure` permet de générer une fenêtre graphique permettant de visualiser les données (courbes, images, ...). la fonction `close` permet de fermer la fenêtre active et `close all` permet de fermer toutes les fenêtres. Les fonctions `title`, `xlabel`, `ylabel` permettent respectivement d'afficher un titre à la figure, le nom de l'axe des abscisses et le nom de l'axe des ordonnées. La fonction `subplot` permet d'afficher plusieurs graphiques ou images au sein d'une même figure.

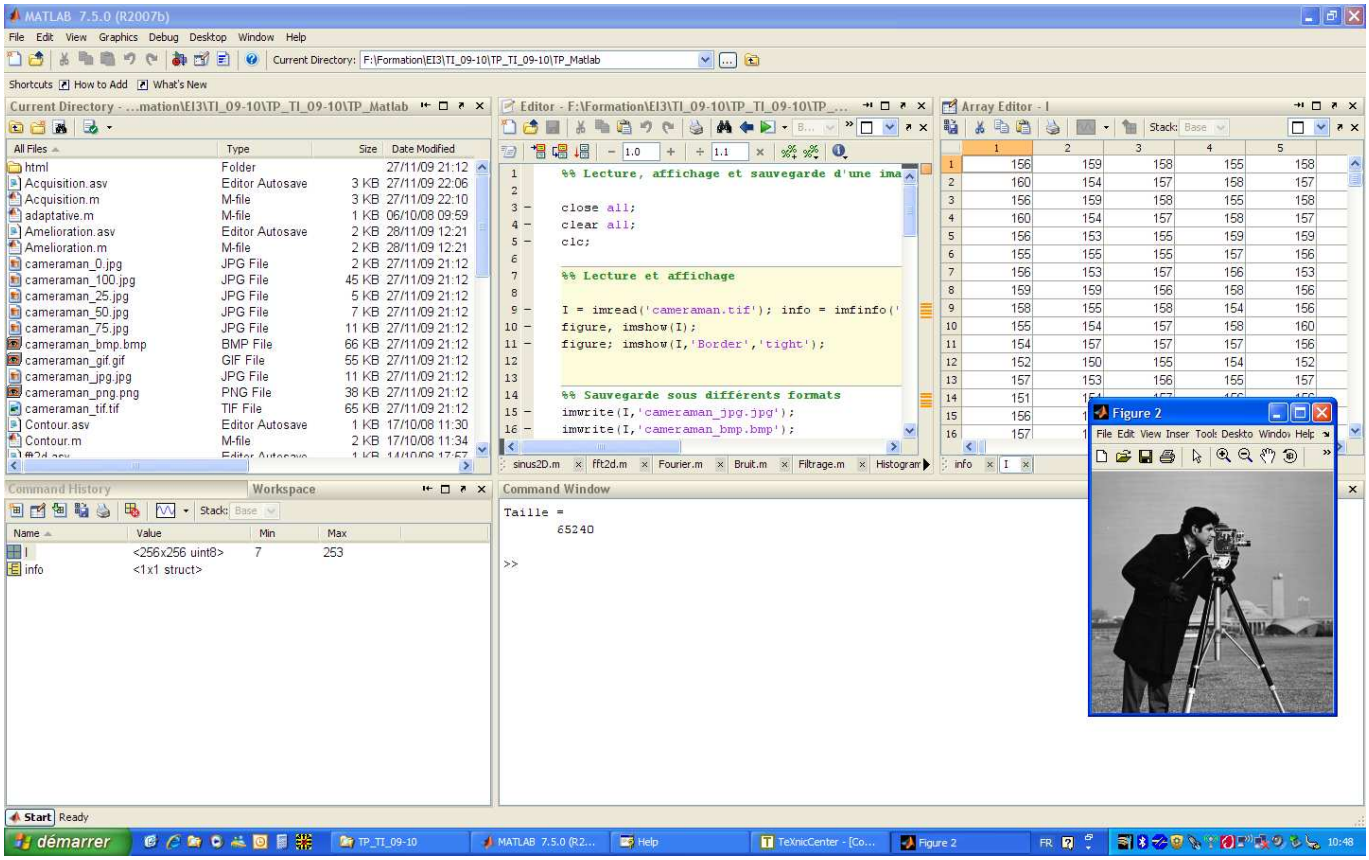


Figure 1 – Le logiciel de calcul scientifique Matlab

La fonction `clear` efface les variables mises en mémoire durant une session Matlab tandis que la fonction `clc` efface le contenu de la fenêtre de commande. La fonction `pause` permet de mettre en veille la fenêtre de commandes de Matlab et les fonctions `disp`, `display` ou `celldisp` permettent l'affichage de textes ou de données dans la fenêtre de commandes de Matlab.

Chaque variable déclarée dans Matlab est stockée dans l'espace des variables à partir duquel il est possible de consulter la taille et le type de la variable ainsi que d'éditer son contenu par un double-click sur le nom de la variable. Les variables Matlab sont des objets de type structure ou des tableaux à $n$ dimensions. Ainsi, un scalaire est un tableau de taille $1 \times 1$ ; un vecteur est un tableau à 1 dimension de taille $1 \times n$ ; une matrice est un tableau à 2 dimensions de taille $m \times n$... Il est ensuite possible d'accéder facilement au tableau, à un élément du tableau, à une ou plusieurs dimensions particulières du tableau.

Une image en noir et blanc est donc un tableau à 2 dimensions. Si $I$ est la variable contenant les données d'une image à niveaux de gris, l'instruction `I(1,1)` renvoie la valeur du pixel de coordonnées $(1,1)$. L'instruction `I(:,1)` renvoie les valeurs des pixels de la première colonne et l'instruction `I(1,:)` renvoie les valeurs des pixels de la première ligne...

Pour créer le vecteur $V = \begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix}$ par exemple, il faut saisir l'instruction `V=[1 2 3 4]`. Pour créer la matrice $M = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ par exemple, il faut saisir l'instruction `M=[1 2; 3 4]`...

Certaines opérations arithmétiques sur les tableaux peuvent être effectués de deux manières :

- de manière matricielle avec les opérateurs : `*`, `/` ou `^`,
- éléments par élément avec les opérateurs : `.*`, `./` ou `.^`.

## 1.2 Lecture, affichage et sauvegarde d'une image

Le chargement en mémoire d'une image se fait avec la fonction `imread`. Par exemple, la fonction suivante permet de lire une image et de placer son contenu dans une variable de type matrice :

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

Cette variable est alors visible dans le **Workspace** (espace des variables) de Matlab. La fonction `whos` permet d'afficher toutes les informations relatives aux variables en mémoire et la fonction `imfinfo` affiche les informations relatives à un fichier image.

L'affichage de l'image (ou de la variable) est réalisé par la fonction `imshow`. Ainsi les fonctions suivantes ouvrent une nouvelle fenêtre pour y afficher l'image *I*.

```
figure; imshow(I);
```

**1)** Afficher l'image 'cameraman.tif' et donner les caractéristiques de cette image. Étudier les options de la fonction `imshow`

Les fonctions `imwrite` et `print` permettent la sauvegarde, respectivement, des images et des figures sous différents formats (tif, jpg, bmp, pcx, png, gif, emf, eps, ...).

**2)** Enregistrer l'image 'cameraman.tif' sous les formats suivants : JPEG, BMP, PNG, GIF et le format TIFF sans compression. Ouvrir et afficher ensuite chacune de ces images, observer leurs différences et comparer les avec l'image d'origine. Mesurer ces différences avec l'image d'origine en calculant l'erreur quadratique moyenne.

**3)** Enregistrer l'image 'cameraman.tif' au format JPEG avec différents niveaux de compression (mettre le paramètre `'Quality'` à 0, 25, 50, 75 et 100). Ouvrir et afficher ensuite chacune de ces images, observer leurs différences en taille et en qualité et comparer les avec l'image d'origine. Mesurer ces différences avec l'image d'origine en calculant l'erreur quadratique moyenne.

## 1.3 Acquisition d'une image

Le matériel d'acquisition est une simple WebCam posée sur un portique et reliée au PC par le port USB. Un pilote Windows spécifique permet de communiquer entre le PC et la Webcam.
La fonction `imaqhwinfo` permet l'obtention d'informations sur le matériel et les pilotes installés.

**4)** En utilisant cette fonction, déterminer les caractéristiques matériel et logiciel du système d'acquisition installé.

L'acquisition d'une image sous Matlab passe par la création d'un objet d'entrée vidéo en utilisant la fonction `videoinput`. Cette fonction affiche également les principales propriétés de l'objet d'entrée vidéo créé. Un objet d'entrée vidéo est un objet de type structure sur laquelle il est possible de régler plusieurs propriétés :

- les paramètres liés au périphérique comme :
  - le format de l'image,
  - l'espace de codage de l'image (RGB, YCbCr, niveaux de gris, ...)...
- Les paramètres liés à l'acquisition comme :

- la luminosité (Brightness),
- le contraste (Contrast),
- le temps d'exposition (Exposure),
- la correction gamma (Gamma),
- la netteté (Sharpness)...

La fonction `propinfo` (ou les fonctions `get` et `set`) ainsi que la fonction `inspect` permettent d'accéder et de connaître les caractéristiques détaillées de chaque propriété. Une propriété peut également être un objet de type structure avec ses propres propriétés. L'accès à cet objet s'adresse de la façon suivante : *nom.propriete*. Attention, certaines propriétés ne sont accessibles qu'en lecture seule selon qu'une acquisition est en cours ou non.

**5)** Donner les différents formats d'images possible de la WebCam. Attention, cette propriété est toujours en lecture seule dès que l'objet d'entrée vidéo est créé. C'est pourquoi il faut la changer au moment de la création avec la fonction `videoinput`.

**6)** Associer un objet d'entrée vidéo *vid* au périphérique matériel correspondant à la WebCam en transmettant le numéro d'identification de ce périphérique avec un format d'image $1280 \times 960$. A l'aide de la fonction `inspect`, accéder aux différentes propriétés de la variable *vid* et modifier l'espace de représentation de la couleur de telle sorte que les images puissent être acquises dans l'espace RGB. Cette propriété peut également être modifiée à l'aide de l'instruction :

```
set(vid,'ReturnedColorSpace','rgb');
```

La fonction `preview` permet de créer une fenêtre d'aperçu afin de visualiser la scène observée et la fonction `closepreview` permet de fermer cette fenêtre. C'est ainsi qu'il est possible de régler certain paramètre d'acquisition.

**7)** Orienter la caméra sur l'objet à observer et visualiser l'objet d'entrée vidéo *vid* à l'aide de la fonction précédente. Effectuer la mise au point de l'objectif de la WebCam.

**8)** Accéder à la propriété de source vidéo *vid.Source* (la fonction `getselectedsource` peut également être utilisée) et relever ses propriétés de luminosité, de contraste, de temps d'exposition, de correction gamma, de teinte, de saturation et de netteté à l'aide de la fonction `propinfo` ainsi que les valeurs minimales et maximales de réglage. Ajuster ces paramètres à l'aide de la fonction `set` qui permet de changer les propriétés d'un objet ou de façon interactive grâce à la fonction `inspect`.

L'acquisition d'UNE seule image se fait avec la fonction `getsnapshot`.

**9)** Utiliser cette fonction pour réaliser l'acquisition de l'image et afficher cette image dans une figure. Enregistrer cette image au format TIFF.

**10)** Changer l'espace de représentation RGB en un espace de luminosité et faire l'acquisition d'une nouvelle image sans modifier les réglages précédent et sans déplacer l'objet observé. Enregistrer cette image au format TIFF.

**11)** Faire une nouvelle acquisition en niveau de gris avec un format $320 \times 240$ sans modifier les réglages précédent et sans déplacer l'objet observé. Enregistrer cette image au format TIFF.

## 1.4 Types des images en mémoire

Matlab supporte 4 formats d'images :

- les images binaires,
- les images d'intensités (à niveaux de gris),
- les images couleur RGB,
- les images couleur indexées.

Il est possible de changer de format en utilisant les fonctions suivantes :

- `ind2gray` : indexé → intensité,
- `ind2rgb` : indexé → RGB,
- `rgb2ind` : RGB → indexé,
- `rgb2gray` : RGB → intensité,
- `im2bw` : intensité, indexé, RGB → binaire : c'est l'opérateur de binarisation. Une image binaire peut être également obtenue en utilisant des opérateurs de comparaison et des opérateurs logiques. Par exemple, les instructions `(I==seuil)` ou `((I>=seuil_bas) & (I<seuil_haut))` permettent d'obtenir des images binaires par comparaison des niveaux des pixels d'une image `I` à des valeurs de seuils.

Le passage d'une image couleur à une image d'intensité correspond à la transformation des composantes R, G, B en la composante Y des systèmes de transmissions YIQ et YUV qui séparent l'information de luminance et de chrominance.

**12)** Ouvrir l'image couleur acquise et enregistrée précédemment et la convertir en une image d'intensité. Comparer cette image avec celle précédemment acquise directement en niveaux de gris en calculant l'erreur quadratique moyenne entre ces deux images.

**13)** Ouvrir l'image couleur acquise et enregistrée précédemment et extraire et afficher en niveaux de gris puis en couleur les images correspondant aux composantes R, G et B. Que peut-on observer ?

Les valeurs des pixels de ces images peuvent être de différents types :

- logique (0 ou 1 pour les images binaires),
- entier non signé codé sur 8 bits (entre 0 et 255),
- entier non signé codé sur 16 bits (entre 0 et 65535),
- réel (entre 0 et 1).

Il est possible de changer le type des variables en utilisant les fonctions suivantes :

- `im2double` : codage d'images en type réel,
- `im2uint8` : codage d'images en type entier non signé sur 8 bits,
- `im2uint16` : codage d'images en type entier non signé sur 16 bits,
- `double` conversion de données en type réel,
- `uint8` conversion de données en type non signé sur 8 bits,
- `uint16` conversion de données en type non signé sur 16 bits.

Les images binaires sont codés en entier logique ou en réel logique et leurs pixels ont des valeurs uniquement égales à 0 ou à 1. L'affichage d'une image peut se faire, soit en réel et les valeurs des pixels sont alors comprises entre 0.0 et 1.0, soit en entier et les valeurs des pixels sont alors comprises, par exemple, entre 0 et 255 pour des entiers non signés codés sur 8 bits.

**14)** Ouvrir l'image monochrome acquise et enregistrée précédemment et la convertir en une image de type double en utilisant successivement les fonctions `double` et `im2double`. Expliquer les différences observées.

## 1.5 Manipulation des images

Certaines fonctions ou certains outils de Matlab permettent des manipulations interactive sur une image contenue dans une figure ou non :

- `imageinfo` : retourne les information de l'image dans la figure ou d'un fichier image,
- `zoom` : zoom sur une zone de l'image de la figure,
- `imcrop` : sélectionne une zone de l'image,
- `improfile` : affiche le profil d'une ligne sélectionnée,
- `impixel` : retourne les valeurs des pixels sélectionnés,
- `impixelinfo` : affiche la position et les valeurs d'un pixel pointé avec la souris,
- `impixelregion` : affiche les valeurs des pixels dans une région sélectionnée avec la souris,
- `imdistline` : affiche la distance entre deux pixels sélectionnés,
- `imdisplayrange` : affiche l'intervalle des valeurs des pixels de l'image,
- `imcontrast` : réajuste une image,
- `imtool` : outil qui utilise les outils précédents.

D'autres fonctions permettent des opérations géométriques sur l'image :

- `imresize` : ré-échantillonnage de l'image (homothétie),
- `imrotate` : rotation de l'image.

**15)** Tester les outils `imtool`, `impixelinfo` et `imdisplayrange` sur l'image monochrome acquise et enregistrée précédemment.

**16)** Ouvrir l'image monochrome acquise et enregistrée précédemment et appliquer une rotation de 45 $^\circ$. Quelle est la taille de l'image ainsi obtenue ?

**17)** Ouvrir l'image monochrome acquise et enregistrée précédemment et diviser sa taille par quatre en utilisant la fonction `imresize`. Comparer cette image avec celle acquise directement à la taille $320 \times 240$ en calculant l'erreur quadratique moyenne.

# 2 – Amélioration et restauration d'images

## 1 Amélioration d'images

La fonction `imhist` permet le calcul et l'affichage de l'histogramme d'une image. En exploitant cet histogramme plusieurs opérations sont possibles en utilisant les fonctions suivantes :

- `imadjust` : recadrage de la dynamique selon une correction gamma,
- `histeq` : égalisation et spécification d'histogrammes,
- `adapthisteq` : égalisation adaptative d'histogrammes,
- `im2bw` : binarisation d'une image.

**1)** Exécuter et tester la démo `imadjdemo` qui utilise principalement les fonctions `imadjust` et `histeq`.

**2)** Acquérir une image en niveau de gris de taille $1280 \times 960$ avec un maximum de luminosité, un minimum de correction gamma et de contraste et visualiser son histogramme. Que peut-on constater ? Après avoir enregistré cette image, déterminer les valeurs de recadrage et réaliser le recadrage dynamique en spécifiant également la valeur de gamma choisie. Quelle est la LUT correspondant à cette opération ?

**3)** Déterminer les valeurs de recadrage en utilisant la fonction `stretchlim` d'abord, puis les fonctions `min` et `max` ensuite. Réaliser respectivement les deux recadrages dynamiques correspondant.

**4)** Acquérir à nouveau une image en niveau de gris de taille $1280 \times 960$ dans des conditions correctes et réaliser l'égalisation d'histogramme. Observer et commenter les différences avec le recadrage dynamique. Visualiser l'histogramme cumulé de cette image ayant servi à l'égalisation. Réaliser une égalisation adaptative et comparer avec le résultat précédent.

**5)** Acquérir une image de taille $1280 \times 960$ en réglant correctement la caméra et seuiller cette image après avoir déterminé le seuil à appliquer en visualisant son histogramme. Effectuer de la même façon le seuillage mais , tout d'abord sans fixer de seuil, puis en calculant automatiquement le seuil avec la fonction `graythresh`. Quelle est la LUT correspondant à cette opération ? Quelles sont les différence entre les méthodes précédentes ?

## 2 Restauration d'images

La fonction de filtrage linéaire propre aux images proposé par Matlab est la fonction `imfilter`. Cette opérateur correspond à une convolution réalisable également avec les fonctions `conv2` ou `filter2` mais qui est spécifique aux images numériques et traite en particulier les bords de l'image. Le noyau du filtre est une matrice de taille quelconque définie par l'utilisateur ou accessibles par la fonction `fspecial`.

Des filtrages non-linéaire peuvent être réalisés avec la fonction `nlfilter` ou `ordfilt2`.

**6)** Exécuter et tester la démo `nrfiltdemo` qui utilise principalement ces fonctions .

## 2.1 Filtre passe-bas

**7)** Acquérir une image en niveau de gris de taille $1280 \times 960$. A partir d'un filtre moyenneur, effectuer le lissage de cette image en utilisant des filtres de tailles $3 \times 3$, $5 \times 5$ puis $7 \times 7$ afin d'étudier l'influence de la taille du noyau du filtre. Comparer les résultats obtenus entre eux et avec ceux obtenus en utilisant un filtre gaussien.

**8)** Appliquer deux fois un filtre moyenneur de taille $3 \times 3$ sur une images acquise en niveau de gris et comparer aux résultats précédents. Quel est le filtre équivalent à ces deux opérations ? Valider ce filtre en comparant le résultat du filtrage correspondant avec celui déterminé précédemment.

La fonction `medfilt2` (ou plus généralement `ordfilt2`) permet de réaliser le filtrage par un filtre médian.

**9)** Réaliser les mêmes filtrages que précédemment en faisant varier la taille du filtre mais en utilisant le filtre médian et comparer les résultats.

La fonction `imnoise` permet d'ajouter du bruit à une image.

**10)** Appliquer un bruit de type "poivre et sel" sur une image acquise en niveau de gris et appliquer un filtre moyenneur de taille $3 \times 3$ puis un filtre median de même taille pour comparer à nouveau les résultats obtenus.

**11)** Appliquer un bruit uniforme sur une image acquise en niveau de gris et appliquer un filtre moyenneur de taille $3 \times 3$ puis un filtre median de même taille pour comparer à nouveau les résultats obtenus.

**12)** Écrire une fonction `imadapt` permettant le filtrage adaptatif de l'image et tester cette fonction sur les deux types de bruit précédent. Discuter des résultats obtenus et comparer les temps de traitement. Comparer les résultats obtenus avec ceux obtenus en utilisant la fonction `wiener2`.

## 2.2 Filtre passe-haut

### 2.2.1 Utilisation de la dérivée première

**13)** Acquérir une image en niveau de gris de taille $1280 \times 960$ et filtrer cette image en utilisant un filtre de Prewitt horizontal. Comparer les résultats obtenus avec les fonctions `imfilter`, `filter2` et `conv2`.

**14)** Filtrer l'image précédente en utilisant le filtre de Prewitt horizontal opposé. Comparer les résultats obtenus avec les précédents en utilisant les fonctions `imfilter`, `filter2` et `conv2`.

**15)** Utiliser des orientations supplémentaires afin d'étudier leur effet sur le filtrage. Pour cela on pourra utiliser l'opérateur de transposition " $'$ " ainsi que les fonctions de rotation `fliplr`, `flipud` et `rot90`.

**16)** Fusionner les résultats précédents pour obtenir une image filtrée unique.

### 2.2.2 Utilisation de la dérivée seconde

**17)** Utiliser un filtre Laplacien 4-voisins puis 8-voisins pour effectuer le filtrage passe-haut et comparer les résultats avec ceux obtenus précédemment. Comparer les résultats obtenus avec les fonctions `imfilter`, `filter2` et `conv2`.

**18)** Réaliser la même opération en utilisant le laplacien de la fonction `fspecial`. Que peut-on remarquer ?

### 2.2.3 Vers la détection de contours...

La fonction `edge` permet d'appliquer un filtrage passe-haut en utilisant soit la dérivée première dans différentes directions, soit la dérivée seconde. Cette fonction exploite ensuite le résultat obtenu en fixant un seuil aux maximums locaux de la dérivée première ou aux passage par zéro de la dérivée seconde.

**19)** Exécuter et tester la démo `edgedemo` qui utilise principalement la fonction `edge`.

### 2.2.4 Réhaussement de contraste

Le filtre rehausseur est défini comme le filtre Laplacien mais le coefficient du centre du noyau est égale à 5 pour un Laplacien 4-voisins (ou 9 pour un Laplacien 8-voisins).

**20)** Acquérir une image à niveau de gris avec une mauvaise netteté et étudier l'effet de ce filtre sur cette image. Réaliser la même opération en utilisant le laplacien de la fonction `fspecial`. Que peut on remarquer ?

## 2.3 Utilisation de la transformée de Fourier

La fonction `fft2` permet de calculer la transformée de Fourier d'une image et la fonction `ifft2` la transformée inverse. La fonction `fftshift` permet de centrer une transformée de Fourier sur la fondamentale (fréquence spatile nulle).

**21)** Calculer la transformée de Fourier centrée d'une image acquise en niveau de gris. Afficher le module ou la partie réelle de cette transformée soit dans une image, soit dans un graphique (fonction `surf`, `image`, `imagesc`, `imshow`, ...).

Afin de réaliser un filtrage en utilisant les transformées de Fourier, on remplacera les valeurs correspondant aux fréquences à filtrer par des zéros (filtre idéal). On réalisera ensuite la transformée inverse en faisant attention d'en prendre son module ou sa partie réelle.

**22)** Réaliser le filtrage passe-bas, passe-haut puis passe-bande d'une image acquise en niveau de gris en utilisant la transformée de Fourier. Proposer des fréquences de coupures adéquates. Comparer les résultats à ceux obtenus avec des noyaux de convolution.

**23)** Exécuter et tester la démo `firdemo` qui utilise le principe du filtrage fréquentiel ainsi que la démo `dctdemo` qui utilise la transformée en cosinus discrète pour compresser les images.

# 3 – Morphologie mathématique

## 1 Acquisition des images et pré-traitement

**1)** Ouvrir ou acquérir une image couleur de taille $1280 \times 960$. Transformer cette image en image monochrome et enregistrer cette image.

**2)** Binariser l'image acquise précédemment de telle sorte à obtenir des objets en blanc et un fond en noir. Si plusieurs binarisations sont nécessaires, utiliser les opérateurs logiques (fonctions `imcomplement`, `or`, `xor` et `and`) pour recomposer l'image.

## 2 Érosion et dilatation

Les fonctions `imerode` et `imdilate` permettent respectivement de réaliser des opérations d'érosion et de dilatation sur des images à niveaux de gris à partir d'un élément structurant. Cet élément structurant peut être défini soit par une matrice de 0 et de 1, soit en utilisant la fonction `strel` qui permet de configurer des éléments structurants élémentaires.

### 2.1 Érosion

**3)** Appliquer une érosion sur l'image binaire en utilisant comme élément structurant un carré de taille $3 \times 3$ pixels. Commenter les résultats.

**4)** Appliquer une érosion sur l'image binaire en utilisant comme élément structurant un carré de taille $5 \times 5$ pixels. Commenter et comparer les résultats.

**5)** Appliquer **deux** érosions consécutivement sur l'image binaire en utilisant comme élément structurant un carré de taille $3 \times 3$ pixels. Commenter et comparer les résultats.

**6)** Appliquer une érosion sur l'image binaire en utilisant comme élément structurant un disque de taille $5 \times 5$ pixels. Commenter et comparer les résultats.

### 2.2 Dilatation

**7)** Appliquer une dilatation sur l'image binaire en utilisant comme élément structurant un carré de taille $3 \times 3$ pixels. Commenter les résultats.

**8)** Appliquer une dilatation sur l'image binaire en utilisant comme élément structurant un carré de taille $5 \times 5$ pixels. Commenter et comparer les résultats.

**9)** Appliquer **deux** dilatations consécutivement sur l'image binaire en utilisant comme élément structurant un carré de taille $3 \times 3$ pixels. Commenter et comparer les résultats.

**10)** Appliquer une dilatation sur l'image binaire en utilisant comme élément structurant un disque de taille $5 \times 5$ pixels. Commenter et comparer les résultats.

# 3 Ouverture et fermeture

Une ouverture est une érosion suivie d'une dilatation et une fermeture est une dilatation suivie d'une érosion.

## 3.1 Ouverture

**11)** En utilisant les fonctions définies précédemment, réaliser une ouverture sur l'image binarisée. Comparer et commenter les résultats. Réaliser la même opération avec la fonction `imopen`. Que se passe-t-il lorsque cette opération est répétée ou lorsque la taille ou la forme de l'élément structurant change ?

## 3.2 Fermeture

**12)** En utilisant les fonctions définies précédemment, réaliser une fermeture sur l'image binarisée. Comparer et commenter les résultats. Réaliser la même opération avec la fonction `imclose`. Que se passe-t-il lorsque cette opération est répétée ou lorsque la taille ou la forme de l'élément structurant change ?

## 3.3 Squelettisation

La fonction `bwmorph` permet plusieurs opérations morphologiques sur des **images binaires** dont l'érosion, la dilatation, l'ouverture, la fermeture, l'amaincissement, ...

La squelettisation consiste à effectuer récursivement l'opération d'amincissement jusqu'à ce que l'image ainsi créée ne change plus.

**13)** Appliquer et observer l'opération de squelettisation sur l'image précédente correctement reconstruite en utilisant la fonction `bwmorph`.

# 4 Autres opérations sur images binaires

La fonction `imclearborder` est une fonction morphologique qui permet de supprimer des régions qui sont au contact des bords de l'image binaire. La fonction `bwareaopen` permet de supprimer des régions de trop petites tailles dans une image binaire. La fonction `imfill` est une fonction morphologique qui permet de combler les "trous" dans les régions d'une image binaire.

**14)** En utilisant ces trois fonctions, traiter l'image acquise et binarisée afin d'obtenir une image dans laquelle les formes correspondent au mieux aux objets de la scène réelle. Utiliser d'autres fonctions morphologiques si nécessaire.