CPE LYON

**Année universitaire**
**2021 - 2022**

# Traitement et Synthèse d'Image — S6
# TP1 – Filtrage

**Marion Foare**

**Objectifs.**

- Filtrage dans l'espace direct et dans l'espace de Fourier
- Implémentation de l'algorithme des K-means

**Déroulement.** Ce TP est à effectuer par binôme sous Matlab/Octave. Vous trouverez sur CPe-campus une archive contenant l'ensemble des fichiers nécessaires à la réalisation de ce TP.

**Configuration.** Ce TP nécessite les librairies suivantes :

- Matlab : *Image Processing Toolbox*
- Octave : *Image toolbox*
  puis, en début de script :

```
pkg install -forge image
pkg load image
```

**Évaluation en séance.** Vous expliquerez à l'intervenant :

- votre démarche (algorithme, équations sur lesquelles vous vous êtes appuyés, etc.),
- le choix et l'influence de vos paramètres,
- les résultats obtenus avec votre interprétation.

Nous rappelons que toute tentative de copie entrainera une sanction de l'ensemble des binômes concernés.

# Préparation

1 – Rappeler les fonctions ou écrire un code permettant :

- de lire une image
- de filtrer une image
- d'afficher une image (plusieurs fonctions existent, préciser pour chacune ses spécificités, notamment en terme de gestion de la dynamique)

2 – A quoi sert la fonction `im2double` ? Pourquoi est-elle indispensable lorsqu'on traite/filtre des images ?

3 – Rappeler les fonctions ou écrire un code permettent de calculer :

- le gradient d'une image $I$
- la norme du gradient de $I$
- le gradient normalisé de $I$
- d'afficher les résultats de ces 3 opérations

4 – Que fait la fonction `meshgrid` ? On pourra, par exemple, observer le résultat de `[U V] = meshgrid(-1:2,1:3);`

# 1 Filtrage dans l'espace direct : détection de contours

On cherche à détecter les contours de l'image 'flower.png' à l'aide du filtre de Sobel.

1 – Décomposer le filtre de Sobel en produit (matriciel) de deux filtres 1D, dont un filtre de lissage et un filtre dérivatif. Pourquoi a-t-on, dans ce filtre de détection de contours, une partie 'lissage' (contrairement aux filtres dérivateurs simples) ?

2 – Calculer les composantes horizontale et verticale du gradient en chaque pixel de l'image, à l'aide du filtre de Sobel et de la fonction de filtrage de Matlab. On stockera la composante verticale du gradient de chaque pixel dans la matrice `Gv` et la composante horizontale dans le tableau `Gh` (sans utiliser de boucle).

3 – En déduire la norme `G` du gradient en chaque pixel. Afficher sur une même figure, `G`, `Gv` et `Gh`.

4 – Reprendre les questions précédentes en ayant ajouté à l'image de départ un bruit blanc Gaussien centré d'écart-type 0.1 en utilisant la fonction `randn`. Qu'observez vous ?

# 2 Filtrage spectral : suppression de l'effet de trame

Lorsque l'on souhaite imprimer une image en niveau de gris, il n'est pas possible de réaliser une impression "continue". En effet, les imprimantes ne peuvent réaliser que deux actions : imprimer un point noir, ou ne rien imprimer et laisser apparaitre le papier blanc.

Ainsi, une image imprimée est en fait constituée d'une multitude de points, appelée trame. Afin de recréer les nuances de gris dans l'image, les points imprimés sont d'autant plus gros que la nuance de gris à reproduire est sombre. Un exemple simple est donné ci-dessous :

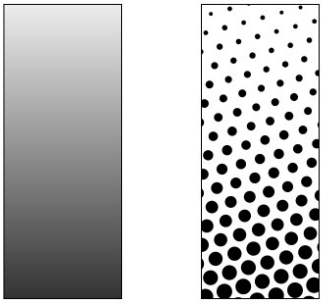

image originale trame

Cet effet de trame n'est en général pas perceptible, car la trame est imprimée à une échelle spatiale très fine. L'oeil/le cerveau intègre l'ensemble de ces variations en faisant de lui-même un lissage, de sorte que nous avons l'illusion de voir une image continue.

Dans cet exercice, on souhaite donc reproduire le fonctionnement du cerveau et supprimer l'effet de trame dans une image.

1 – Lire et afficher l'image 'journal.png'. Obtenir la taille `[h,w]` de l'image.
2 – Calculer la transformée de Fourier 2D de l'image en utilisant la fonction `fft2`. On n'oubliera pas, après avoir effectué la FFT de l'image, de réarranger les données afin que la fréquence nulle se trouve au centre de l'image (`fftshift`). Afficher le module de la transformée de Fourier.
3 – Quelle particularité possède ce spectre ? Comment peut-on l'expliquer à partir de la représentation spatiale de l'image ?
4 – Quel type de filtre faut-il mettre en place pour éliminer l'effet de trame ?
5 – Générer le filtre de Butterworth qui vous semble pertinent à l'aide des étapes suivantes :
   - En utilisant la fonction `meshgrid`, obtenir deux matrices donnant respectivement les coordonnées $U$ et $V$ de chaque pixel. On fera en sorte que le pixel au centre de l'image ait pour coordonnées $(0,0)$ :
     `[U V] = meshgrid(-w/2+1/2:w/2-1/2,-h/2+1/2:h/2-1/2)`
   - À partir de $U$ et $V$, obtenir la matrice $D$, donnant la distance euclidienne au centre pour chaque pixel de l'image.
   - En utilisant $D$, en déduire le gain $H$ de Butterworth discuté à la question précédente (sans utiliser de boucle). On pourra par exemple prendre pour paramètres du filtre un ordre $p = 2$, et un indice de coupure $n_c = 100$, qui devront être ajustés par la suite.
6 – Filtrer l'image par $H$ dans le domaine fréquentiel. Si $I$ désigne la transformée de Fourier shiftée de votre image, on pourra afficher le résultat du spectre filtré avec la commande suivante : `imshow(H.*log10(abs(I)),[]);`
7 – Revenir à l'espace direct par transformée de Fourier inverse 2D et afficher l'image filtrée. Commenter.
8 – Appliquer ce filtre à l'image qui vous aura été désignée, en ajustant les paramètres au besoin.

# 3 Filtrage linéaire vs non linéaire : débruitage

On souhaite comparer les performances de débruitage de plusieurs filtres en présence de différents types de bruit. Pour cela, on peut par exemple utiliser l'Erreur Quadratique Moyenne (MSE), qui permet de quantifier objectivement la qualité d'une estimation en sortie de filtre.

Étant données une image bruitée $I$ de taille $h \times w$, et sa version approchée $\hat{I}$ en sortie de filtre, la MSE est définie par :

$$MSE = \frac{1}{hw} \sum_{i=1}^{h} \sum_{j=1}^{w} \left[ I(i,j) - \hat{I}(i,j) \right]^2$$

Dans cet exercice, on travaillera sur l'image 'flower.png', ou tout autre image de votre choix.

1 – Générer les deux images bruitées suivantes :
   - `I_g` correspondant à l'image originale dégradée par un bruit blanc Gaussien d'écart-type $\sigma = 0.1$, en utilisant la fonction `randn`;
   - `I_sp` correspondant à l'image originale dégradée par un bruit poivre et sel de densité 0.5, en utilisant la commande Matlab `imnoise`.

2 – Comparer qualitativement (ie : en observant les images résultantes) les performances de débruitage de chacun des filtres suivants :
   - filtre moyenneur de taille $n$,
   - filtre de Butterworth passe-bas d'ordre $p = 5$ et d'indice de coupure $n_c$ (cf exercice 2),
   - filtre médian de taille $n$.

   Pour cela on testera, pour chaque filtre, différentes valeurs du paramètre de ce filtre ($n$ ou $n_c$).

3 – Comparer quantitativement les performances de chacun des filtres, en traçant la MSE en fonction de différentes valeurs du paramètre du filtre ($n$ ou $n_c$).

**Préparation pour le TP2**

# 4 Traitement d'histogramme : seuillage par $K$-means

1 – Lire et afficher l'image 'flower.png'.

2 – Écrire une fonction implémentant l'algorithme de $K$-means à $K$ régions :
   - Choisir aléatoirement $K$ intensités $m_i$.
   - Assigner à chaque pixel le label 1 si son intensité est plus proche de la valeur $m_1$, le label 2 si son intensité est plus proche de la valeur $m_2$, ..., le label $K$ si son intensité est plus proche de la valeur $m_K$ . On stockera ces labels dans la matrice `labels`.
   - Mettre à jour les valeurs de $m_i$ en prenant la moyenne des intensités des pixels de label $k$.
   - Itérer jusqu'à ce que les moyennes $m_i$ ne changent plus.

3 – Afficher l'image segmentée en 2 régions avec l'algorithme des $K$-means. Comment pourrait-on remonter à un seuil à partir de cet algorithme ?

4 – Discuter du choix de $K$ pour la segmentation de l'image 'flower.png'.