Traitement et Synthèse d'Image
TP1 – Filtrage

1 Filtrage passe-haut dans l'espace direct : détection de contours

Dans cette partie, on cherche à détecter les contours de cellules à l'aide du filtre de Sobel. On rappelle que le filtre de Sobel est défini par les masques :

-1	0	1
-2	0	2
-1	0	1

-1	-2	-1
0	0	0
1	2	1

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 les filtres de détection de contours, une partie 'lissage' ?

2 – Lire et afficher l'image 'flower.png' à l'aide des commandes Matlab `imread`, `im2double` et `imshow`. Obtenir les dimensions `h` et `w` de l'image.

3 – 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 `imfilter`. On enregistrera la composante verticale du gradient de chaque pixel dans la matrice `Gv` et la composante horizontale dans le tableau `Gh` (sans utiliser de boucle).

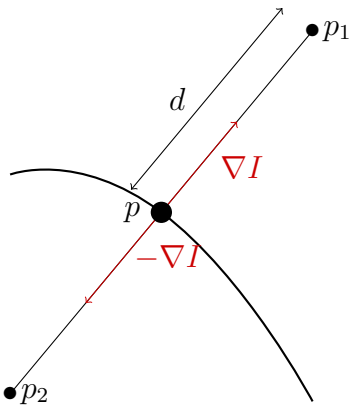
4 – En déduire la norme `G` du gradient en chaque pixel. Afficher sur une même figure, `G`, `Gv` et `Gh`. Pour une meilleure visualisation, on pourra utiliser la commande `imcomplement`. Commenter.

5 – Reprendre les questions précédentes en ayant ajouté à l'image de départ un bruit gaussien centré de variance 0.01 en utilisant la fonction matlab `imnoise`. Qu'observez vous ?

Les contours des objets correspondent à des maxima locaux dans la direction du gradient. Ainsi, pour chaque pixel de l'image, on calcule les coordonnées des deux points p_1 et p_2, qui sont à une distance d dans la direction du gradient ∇I et de l'opposé du gradient $-\nabla I$. Un pixel p appartient alors à un contour si

$$G(p) > G(p_1) \quad \text{et} \quad G(p) > G(p_2) \tag{1}$$

où G est la norme du gradient précédemment calculée.



6 – Calculer les composantes normalisées `Gv_n` et `Gh_n` du gradient de l'image bruitée.

7 – Pour chaque pixel de l'image, calculer les coordonnées p_1 et p_2 à partir de `Gv_n` , `Gh_n` et d. On pourra prendre par exemple $d = 2$. Les coordonnées `X` et `Y` de p seront obtenues à l'aide de la fonction `meshgrid`.

8 – Enregistrer dans un tableau `C` la norme du gradient `G` pour les pixels vérifiant (1). On mettra à zéro les autres pixels.
NB : les calculs étant numériques, on remplacera les inégalités par une condition moins forte de la forme : "si $G(p) - G(p_i) > \varepsilon$", avec, par exemple, $\varepsilon = 0.5$.

9 – Afficher les contours de l'image bruitée et discuter du résultat.

2 Filtrage passe-haut dans l'espace de Fourier

Dans cette seconde partie, on souhaite corriger le biais d'une image (la voie lactée) en vue d'une segmentation ultérieure. Le nuage qu'on souhaite supprimer correspondant aux basses fréquences, on effectue un filtrage passe-haut de l'image. Pour cela, on implémente le filtre de Butterworth, dont on rappelle ci-dessous la définition :

$$H(u,v) = \frac{1}{1 + \left(\frac{f_c}{\sqrt{u^2+v^2}}\right)^{2n}}$$

On pourra par exemple prendre pour paramètres du filtre $n = 2$ et $f_c = 3$.

1 – Lire et afficher l'image 'ngc2175.png'. Obtenir la taille (h, w) de l'image.

2 – Génération du filtre de Butterworth :

- En utilisant la fonction `meshgrid`, obtenir deux matrices donnant respectivement les coordonnées U et V de chaque pixel. On fera de 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)`
- A partir de U et V, obtenir la matrice D, donnant la distance au centre pour chaque pixel de l'image.
- A partir de D, en déduire le filtre H de Butterworth (sans utiliser de boucle).
- Tracer le filtre en 3D en utilisant la fonction `mesh`.

3 – Calculer la transformée 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.

4 – Filtrer l'image par H dans le domaine fréquentiel.

5 – Revenir à l'espace direct par transformée de Fourier inverse et afficher l'image filtrée (de même, ne pas oublier de réarranger les fréquences avant transformée inverse). Commenter.

3 Filtrage non linéaire : filtre médian

L'avantage de filtres définis dans l'espace de l'image par rapport à l'espace de Fourier, est la possibilité de définir des filtres non linéaires. Un exemple typique de filtre non linéaire est le filtre médian. Il permet entre autre, de corriger efficacement un bruit poivre et sel.

1 – Rappeler ce qu'est le filtre médian, et plus généralement, ce qu'est un filtre d'ordre.

2 – Rappeler ce qu'est un bruit poivre et sel et justifier l'usage d'un filtre médian pour corriger une image affectée d'un tel bruit.

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

4 – Ajouter un bruit poivre et sel de densité 0.5 à l'image en utilisant la commande Matlab `imnoise`.

5 – A l'aide d'un filtre médian, corriger le bruit. Pour cela, on utilisera la commande plus générale liée aux filtres d'ordre `ordfilt2`.

6 – Afficher l'image filtrée pour plusieurs tailles de filtre et discuter des résultats.

4 Filtrage à partir d'histogramme : seuillage par K-means

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

2 – Algorithme de K-means à 2 régions :

- Choisir aléatoirement deux intensités m_1 et m_2, comprises entre 0 et 255.
- Assigner à chaque pixel le label 1 si son intensité est plus proche de la valeur m_1, ou le label 2 si son intensité est plus proche de la valeur m_2. On stockera ces labels dans la matrice `labels`.
- Mettre à jour la valeur de m_1 en prenant la moyenne des intensités des pixels de label 1. De même, mettre à jour la valeur de m_2.
- Itérer jusqu'à ce que les moyennes m_1 et m_2 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 – *(Facultatif)* Adapter l'algorithme des K-means pour une segmentation en 3 régions, puis en K régions. Discuter du choix de K pour la segmentation de l'image 'flower.png'.