TP Filtrage

Module TSI - 4ETI

Sommaire

1. Filtrage à partir d'histogramme : seuillage par maximisation de la variance interclasses

Dans cette première partie, le filtrage/seuillage est effectué à partir de la seule connaissance de l'histogramme de l'image.

- Lire, convertir en niveau de gris et afficher l'image 'cellules.jpg'.
- Enregistrer dans un tableau h, l'histogramme de l'image et l'afficher.
- Écrire une fonction -prenant en paramètre h- qui affiche d'une part la courbe de la variance interclasses en fonction du seuil et retourne, d'autre part, la valeur de seuil qui maximise celle-ci.

On rappelle ci-dessous l'expression de la variance (voir manuscrit de cours) :

$$V(s) = P_0(s)(m - m_0(s))^2 + P_1(s)(m - m_1(s))^2 \text{ avec}$$

- P_0 (respectivement P_1) : probabilité d'appartenir à la classe fond (resp. objet),
- m_0(respectivement m_1) : valeur moyenne de la classe fond (resp. objet),
- m : valeur moyenne de l'image.

- Seuiller l'image avec la valeur retournée par la maximisation de la variance interclasses et l'afficher.

2. Filtrage d'ordre : 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 filtres non linéaires est le filtre médian. Il permet entre autre, de corriger efficacement d'un bruit poivre et sel.

- Rappeler ce qu'est le filtre médian.
- 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.
- Lire et afficher l'image 'phare_bruit_ps.png'.
- A l'aide d'un filtre médian, corriger le bruit. Pour cela, on utilisera la commande plus générale liée aux filtres d'ordre 'ordfilt2. On rappellera ce qu'est un filtre d'ordre.
- Afficher l'image filtrée.

3. 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 :

$$H(u,v) = \frac{1}{1+(\frac{D_0}{D(u,v)})^{2n}}$$ avec D(u,v) : distance du pixel (u,v) au centre de l'image

- Lire et afficher l'image I de 'ngc.jpg'.
- Obtenir la taille H*W de I.
- Avec la commande 'meshgrid', obtenir deux tableaux donnant respectivement la 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 le tableau D, donnant la distance au centre pour chaque pixel de l'image (aucune boucle requise).
- A partir de D, en déduire le filtre H de Butterworth (à nouveau, aucune boucle nécessaire). On pourra par exemple prendre pour paramètres du filtre, n=2, D0=3.
- Tracer le filtre en 3D : plot3(U,V,H).
- Filtrer l'image I par H (dans l'espace de Fourier). 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).
- Revenir à l'espace direct par fft inverse et afficher l'image filtrée (de même ne pas oublier de réarranger les fréquences avant fft inverse).

NB : la fft retournant des coefficients complexes, on pensera à afficher le module des fft ou fft inverse ('abs').

4. 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 :

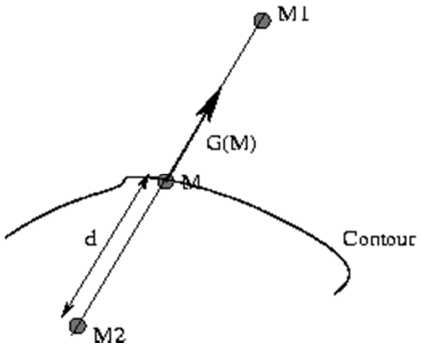
$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \text{ et } \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

- Décomposer (sur papier) 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' ?
- Obtenir les composantes 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 le tableau Gi et la composante horizontale dans le tableau Gj (pas de boucles).
- En déduire la norme G du gradient pour chaque pixel.
- Afficher sur une même figure, Gi, Gj et G. Analyser.
- Reprendre les opérations précédentes en ayant ajouté à l'image de départ, un bruit gaussien centré de variance 0.01 : imnoise(I). 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 calcul ses deux voisins ‘m1’ et ‘m2’ qui sont à une distance d dans la direction du gradient g et de l'opposé du gradient -g. Un point ‘m’ appartient alors à un contour si

$$G(m) > G(m1) \text{ et } G(m) > G(m2) \qquad (1)$$

où G est la norme du gradient précédemment calculée.



- Obtenir le gradient normalisé Gi_n et Gj_n en divisant les composantes Gi et Gj par la norme G du gradient.
- Pour chaque point de l'image, calculer m1 et m2 à partir de Gi_n, Gj_n et d. On pourra prendre d=2.
- 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 de la forme : « si G(m) – G(m1) > epsilon » avec, par exemple, epsilon=0.5.

- Modifier le programme précédant pour ajouter une condition sur la valeur du gradient ; Un point ‘m’ appartient alors à un contour si

$$G(p) > G(p1) \text{ et } G(p) > G(p2) \text{ (1) et } G(p) > seuil$$

- Afficher les contours de l'image affectée du bruit et discuter du résultat.

5. Déconvolution par la psf de l'instrument (facultatif)

Une image étant obtenue via un instrument, elle est affectée des défauts de celui-ci. De manière plus générale, l'image peut être affectée par un flou, un bruit, une dérive d'éclairement etc. Il est assez fréquent qu'on puisse modéliser la psf (qu'on notera h) de l'instrument. L'image étant par ailleurs bruitée, on peut la modéliser par l'équation :

$$i = o * h + n \quad \text{où}$$

- i est l'image obtenue
- o l'objet «vrai »
- h la psf de l'instrument
- n le bruit
- * l'opérateur de convolution

Si on néglige le bruit, on peut reconstruire o grâce au filtrage dans Fourier : O = I./H. Que devient cette équation lorsqu'on tient compte du bruit (dont on notera N la TF) ? Pourquoi l'opération précédente n'est-elle plus possible ?

Un des filtres les plus classiques pour tenter de recouvrir O est le filtre de Wiener. En théorie, le filtre de Wiener retourne l'estimé minimisant l'écart à o, au sens des moindres carrés. Le filtre dépend de la puissance de o et de la puissance du bruit (on rappelle que la puissance d'un signal est le carrée du module de sa transformée de Fourier). Il n'est pas toujours évident de connaître la puissance de l'objet cherché ni celle du bruit. On remplace alors le filtre de Wiener (nécessitant le rapport des puissances signal/bruit) par un filtre paramétré.

Le filtre de Wiener est défini par :

$$W(u,v) = \frac{1}{H(u,v)} \frac{|H(u,v)|^2}{|H(u,v)|^2 + PN(u,v)/PO(u,v)}$$

où PN et PO représentent respectivement les puissances du bruit et du signal o. Les majuscules représentent les transformée de Fourier.

Le filtre de Wiener paramétré est défini par :

$$W(u,v) = \frac{1}{H(u,v)} \frac{|H(u,v)|^2}{|H(u,v)|^2 + \gamma}$$

a) Obtention de l'image floue et bruitée :

- Lire et afficher l'image 'cycliste.jpg'
- Obtenir la psf modélisant un flou de mouvement : h=fspecial('motion',8,0)
- Obtenir l'image affectée de ce flou : i_h=imfilter(i,h,'circular')
- Ajouter un bruit gaussien d'écart type 0.02 :
    - stdn=0.02;
    - n=stdn*randn([L, C]);
    - i_hn=i_h+n;

b) Reconstruction par le filtre de Wiener (connaissant la puissance du bruit et de l'objet vrai) :

Dans un premier temps on suppose connu la puissance du bruit et de l'objet réel (ce qui peut, en pratique, être modélisé).

- n=stdn*randn([L, C]);
- i_hn=i_h+n;
- Obtenir la puissance du bruit et la puissance de l'image de départ :
- N=fftshift(fft2(n));
- PN=abs(N).^2; %puissance du bruit
- Obtenir le filtre de Wiener
- Obtenir la transformée de Fourier de i_hn : I_hn
- Filtrer I_hn par le filtre de Wiener et revenir dans l'espace de l'image.

b) Reconstruction par le filtre de Wiener paramétré :

On suppose maintenant que la puissance du bruit et du signal vrai ne sont pas connus et on remplace le rapport « puissance bruit/puissance signal » par un paramètre constant noté gamma.

- Pour gamma variant de 10^-5 à 2 par pas de 0.05, reconstruire l'image par les filtres de Wiener paramétrés.

6. Quelques commandes matlab

–conversion couleur vers niveau de gris : rgb2gray

–seuiller une image avec le seuil s : im2bw

–tracé d'une courbe en 3D : plot3

–fft d'une image : fft2

–réarrangement des fréquences : fftshift

–réarrangement inverse et fft inverse : ifftshift, ifft2

–ajout d'un bruit : imnoise

http://web.univ-pau.fr/RECHERCHE/SET/LAFFLY/docs_laffly/anova_daf.pdf