TP Filtrage

December 10, 2014

1 Introduction

Rappel... toujours les mêmes Avant tout, nous vous rappelons, même si ce n'est pas nécessaire, que la documentation et des exemples concernant les fonctions que vous allez utiliser sont disponibles soit dans le logiciel via la commande *help* soit en cherchant sur le web via votre moteur de recherche préféré.

1.1 À rendre

De votre part, nous attendons deux éléments :

- Un compte–rendu court : 1 ou 2 pages pour expliquer/commenter les codes de la première partie concernant les sommes de Minkowski et les filtrage moyenneur et dérivateur.
- Des scripts : un pour chaque question avec des figures si nécessaire (toujours commentées). Pensez a bien commenter vos scripts et a mettre en évidence vos résultats.

Vous avez jusqu'au 14 décembre pour rendre votre TP sous format électronique. Un dépôt est ouvert sur le e-campus. Vous devrez y déposer une archive (zip ou tar) contenant les éléments ci-avant només.

1.2 Les moyens

Vous trouverez toutes les ressources à l'adresse suivante : `https://github.com/matthewozon/TPfiltrage4ETI`
Consigne à observer : Le TP doit se dérouler sous OCTAVE que vous pouvez lancer soit en mode terminal via la commande *octave* soit en démarrant l'interface graphique *QtOctave*. **NEW** une autre possibilité pour une interface graphique plus user friendly : dans le terminal, vous lancez la commande *octave –force-gui*. Pour l'affichage des figures vous avez des choix qui sont listés dans *available_graphics_toolkits* (variable octave) qui contient souvent *fltk* et *gnuplot*. Vous pouvez sélectionner celui que vous voulez pour modifier le type d'affichage des figures en tapant : *graphics_toolkit* suivi de la valeur que vous souhaitez et qui est contenue dans *available_graphics_toolkits*.

Conseil pour les figures : Utiliser la commande *print('-depsc','nom_du_fichier')* qui "imprime" ce que vous voyez dans la figure sans en diminuer la qualité (contrairement aux captures d'écran). Pour plus d'information sur cette commande, nous vous invitons à taper après votre prompt octave : *help print*.

2 Addition de Minkowski :

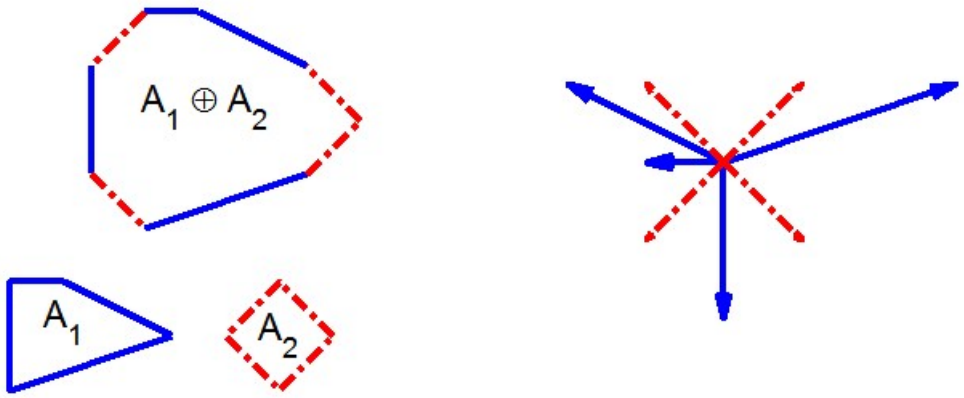

Figure 1: *Addition de Minkowski de deux polygones par interclassement des vecteurs d'arêtes.*

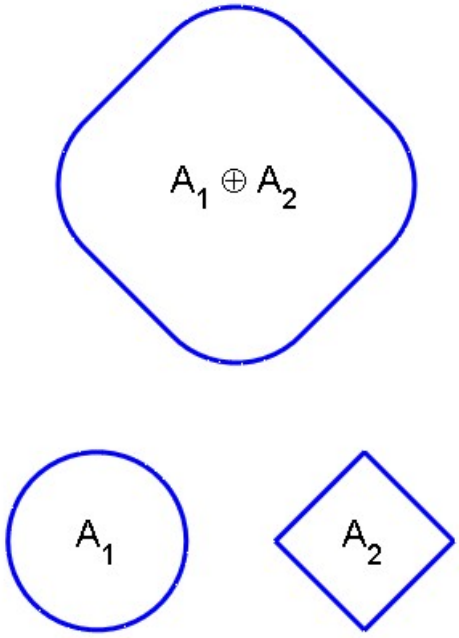

Figure 2: *L'addition de Minkowski d'un polygone et d'un disque de rayon r est le dilaté du polygone de taille r.*

Etant données deux "formes" *convexes* A et B, la somme de Minkowski de ces deux formes,

notée $C = A \oplus B$, est l'ensemble :

$$C = \{\vec{w} = \vec{u} + \vec{v} \text{ avec } \vec{u} \in A \text{ et } \vec{v} \in B\}$$

Propriétés de l'opérateur $\oplus$:
a) commutativité : $A \oplus B = B \oplus A$.
b) associativité : $(A \oplus B) \oplus C = A \oplus (B \oplus C)$.
c) existence d'un neutre : $A \oplus \emptyset = A$.

Lorsque les formes en question ont des contours polygonaux, il en est de même de leur somme de Minkowski. On montre que ce contour peut être obtenu en procédant à un interclassement des vecteurs $\overrightarrow{V_k} = \overrightarrow{M_k M_{k+1}}$ définissant le contour, cet interclassement étant fondé sur les angles polaires des $\overrightarrow{V_k}$, comme on peut le voir sur la fig.1.

Plus précisément, à un polygone défini par ses n sommets $M_1, ...M_n$ dans le plan "à translation près", on peut associer le "bouquet de n vecteurs" de somme nulle définis ci-dessus (convention $M_{n+1} = M_1$). Cette association fonctionne aussi (bijectivement) dans l'autre sens : le cumul progressif des $\overrightarrow{V_k}$ redonne les M_k. Si l'on note B cette association de "prise en bouquet", on a donc l'identité $B(P_1 \oplus P_2) = B(P_1) \cup B(P_2)$. Si bien que l'on peut résumer l'algorithme à ceci :

En entrée P_1 et P_2 (décrits par leurs sommets).
$Bouquet = B(P_1) \cup B(P_2)$.
$P_3 = B^{-1}(Bouquet)$ est la somme de Minkovski de P_1 et P_2.

Cette façon de "redéfinir" l'opérateur $\oplus$ permet notamment de prouver sa commutativité et son associativité de façon quasi immédiate.

Nous allons travailler dans le plan complexe, car il s'avère que les fonctions que l'on peut utiliser y sont globalement plus simples.

Vous trouverez ci-dessous un programme qu'il s'agit dans un premier temps de comprendre, en le commentant abondamment, puis, dans un deuxième temps, de compléter au niveau des lignes centrales (utiliser l'opérateur 'diff' et les fonctions 'abs' et 'angle') .

```
close all;clear all;hold on;axis equal
i=complex(0,1);
P1=[0;1;1+i;i;0];
P2=[2;4;3+i;2];
...
...
...
...
...
[An,Ind]=sort(A);
S=R(Ind).*exp(i*An);
P3=cumsum(S);P3=[P3;P3(1)];
P=[P1;NaN;P2;NaN;(4.5+i*4.5)+P3];
plot(real(P),imag(P)',LineWidth',2,'linesmoothing','on');
text([0.4,2.9,4.6],[0.5,0.3,3.5],...
```

```
{‘A_1’,‘A_2’,‘A_1 \oplus A_2’},‘fontsize’,14);
```

On pourra (voir Fig. 2) prendre pour P_1 le polygone régulier des racines n-ièmes de l'unité pour réaliser une dilatation approchée de P_2 par un disque.

3 Addition de Minkowski par convolution de 2 images :

Que fait le programme suivant ? Quel rapport voyez-vous avec la convolution de Minkowski ?

```
clear all;close all;
n=256;
I=zeros(n);
for i=1:n
    for j=1:n;
        I(i,j)=((i+j)>(n-20))&&((i+j)<(n+20))&&(i>110)&&(i-j<130);
    end;
end;
figure(1)
imshow(I);
J=zeros(n);
for i=1:n
    for j=1:n;
        J(i,j)=(i-n/2)^2+(j-n/2)^2<=400;
    end;
end;
figure(2)
imshow(J);
K=conv2(im2double(I),im2double(J),’same’);
K(100:160,100:160);% permet de se rendre compte des valeurs des pixels
figure(3);
imshow(K>0);
%imagesc(K);% alternative a la ligne ’imshow(K>0)’
S=strel(’disk’,20,0);% l’indice 0 pour avoir un vrai disque... sur matlab seulement
L=imdilate(I,S);
imshow(L)
L(100:160,100:160)
figure(4)
imshow(abs((K>0)-L));
```

4 Filtrage moyenneur :

Commentez le programme suivant ; comparer le résultat à celui d'un filtre gaussien
w=fspecial('gaussian',100,sigma), avec par exemple sigma = 3.

```
clear all;close all;
I=imread('barbara.png');% on pourra prendre galement I=checkerboard(100,5,5);
whos I;%  dans le cas de ''barbara'', c'est une image 512 x 512 de type uint8
figure(1);imshow(I)
n=10;
w=(1/(2*n+1)^2)*ones(2*n+1);
%filtre moyenneur 21 x 21 ; notez les bords; essayez d'autres valeurs de n
figure(2);J=imfilter(I,w,'same');imshow(J);
figure(3);J=imfilter(I,w,'replicate');imshow(J);% sans bords
```

5 Filtrage dérivateur :

On a vu dans le TP précédent qu'un filtre dérivateur selon l'axe Ox est donné par la matrice :

$$\begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$$

En faisant de même avec sa transposée, on peut en déduire l'image de la norme du gradient $\sqrt{Gx.^2 + Gy.^2}$, à moins que l'on ne préfère, plus simplement, $abs(Gx) + abs(Gy)$.

a) Ecrire un programme qui compare, sur l'image "Barbara", la norme du gradient avec le filtre de Sobel.

Indications : utiliser :
Gx=im2double(imfilter(I,m,'same'));
imfilter(I,fspecial('sobel'),'replicate');

b) Comment pourrait-on imaginer un filtre "dérivée seconde" ?

Solution :
a) Voici les instructions manquantes au premier programme :

```
V1=diff(P1);
V2=diff(P2);
Vf=[V1;V2];%fusion
A=angle(Vf);
R=abs(Vf);
```

Comparaison entre filtres dérivateurs ($\sqrt{Gx.^2 + Gy.^2}$) et filtre de Sobel :

```
clear all;close all;
I=imread('barbara.png');
m=[-1,0,1;
   -2,0,2;
   -1,0,1];
mr=rot90(m);
```

```
Ip=im2double(imfilter(I,m,'same'));
Iq=im2double(imfilter(I,mr,'same'));
figure(1);J=sqrt(abs(Ip).^2+abs(Iq).^2);imshow(J);
Isobel=imfilter(I,fspecial('sobel'),'replicate');
figure(4);imshow(Isobel)
```

6 Déconvolution

6.1 À lire

6.1.1 Produit de convolution discret

On rappelle que pour deux matrices I et w (correspondant resp. à l'image et au filtre), on note leur produit de convolution au point (n, m) :

$$I_w(n,m) = \sum_{k=-K}^{K} \sum_{l=-L}^{L} I(n+k, m+l)\, w(-k,-l)$$

avec w de taille $(2K+1, 2L+1)$ et en numérotant ses lignes et colonnes de $-K$ à K et de $-L$ à L respectivement. Cette opération revient á faire une symétrie centrale de w puis á placer son centre en (n, m) et faire la somme des produits terme á terme.

6.2 FFT et Transformée de Fourier

La figure 3 illustre le lien entre la FFT et la transformée de Fourier. Dans le cas de la transformée continue, la formule est donnée par :

$$G(f_x, f_y) = \mathfrak{F}\{g(x,y)\}(f_x, f_y) \propto \iint_{\mathbb{R}^2} g(x,y)\, e^{-2i\pi(f_x x + f_y y)} \mathrm{d}x\mathrm{d}y$$

Propriété

1 Produit de convolution : $\left(g \underset{(x,y)}{*} h\right)(\tau_x, \tau_y) = \mathfrak{F}^{-1}\{\mathfrak{F}\{g(x,y)\}(f_x, f_y)\, \mathfrak{F}\{h(x,y)\}(f_x, f_y)\}(\tau_x, \tau_y)$

2 Décalage : $(g(x,y) * \delta(x - r_x, y - r_y))(\tau_x, \tau_y) = (g(x + r_x, y + r_y) * \delta(x,y))(\tau_x, \tau_y) =$ $g(\tau_x + r_x, \tau_y + r_y)$ équivalent dans le domaine de Fourier à une multiplication par $e^{2i\pi(f_x r_x + f_y r_y)}$ et donc $\mathfrak{F}\{g(x + r_x, y + r_y)\}(f_x, f_y) = e^{2i\pi(f_x r_x + f_y r_y)} \mathfrak{F}\{g(x,y)\}(f_x, f_y)$

3 Séparabilité : si $f(x,y) = g(x,y)\, h(x,y)$ alors, on a $\mathfrak{F}\{f(x,y)\}(f_x, f_y) = \mathfrak{F}\{g(x,y)\}(f_x, f_y)\, \mathfrak{F}\{h(x,y)\}(f_x, f_y)$

Dans l'autre cas (discret), la formule de la transformée discrète pour un signal périodique $2D$ de période (N_x, N_y) :

$$G(u,v) = \mathfrak{F}_{\mathrm{d}}\{g(n,m)\}(u,v) \propto \frac{1}{N_x}\frac{1}{N_y} \sum_{n=0}^{N_x-1} \sum_{m=0}^{N_y-1} g(n,m)\, e^{-2i\pi\left(\frac{un}{N_x} + \frac{vm}{N_y}\right)}$$

Dans ce cas, les fréquences dans le domaine de Fourier sont comprises dans le carré de sommets : $\{(0.0, 0.0), (0.0, 1.0), (1.0, 1.0), (1.0, 0.0)\}$ avec :

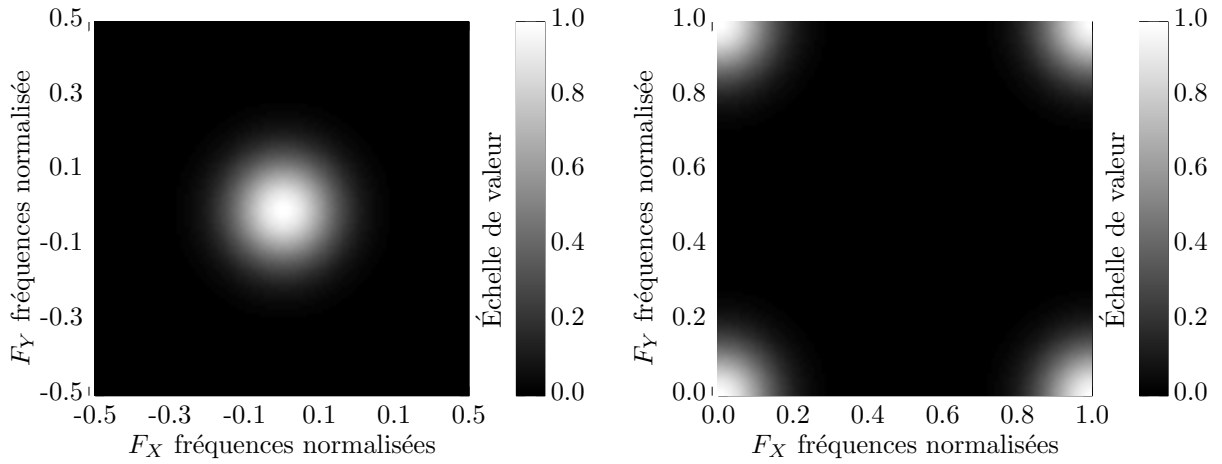

Figure 3: Dualité transformée de Fourier et FFT. À gauche la Transformée de Fourier d'une gaussienne et à droite sa FFT du fait de l'hypothèse de périodicité

.

$(0.0, 0.0)$ le point des basses fréquences en f_x et f_y

$(0.5, 0.0)$ le point des hautes fréquences en f_x et basses en f_y

$(0.0, 0.5)$ le point des basses fréquences en f_x et hautes f_y

$(0.5, 0.5)$ le point des hautes fréquences en f_x et f_y

On remarque la périodicité du domaine de Fourier et donc :

$(1.0, 0.0)$ le point des basses fréquences en f_x et f_y

$(0.0, 1.0)$ le point des basses fréquences en f_x et f_y

$(1.0, 1.0)$ le point des basses fréquences en f_x et f_y

Quand on exécute la fonction *fft2*, on fait l'hypothèse implicite que l'on traite un signal 2D qui est périodique, et que son spectre l'est aussi, par conséquent. Donc, pour des raisons de visualisation, on utilise *fftshift* sur le résultat de *fft2*. Cependant si on inverse le résultat de *fftshift* avec la commande *ifft2*, on ne retrouve pas le signal d'origine, mais le signal modulé. En effet, on comprend visuellement que l'on a décalé le signal d'une demi période dans les deux dimensions. On peut modéliser ce décalage par une convolution dans le domaine de Fourier par un Dirac centré en $\left(\frac{1}{2}, \frac{1}{2}\right)$, ce qui résulte comme :

$$g_{\text{décalé}}(n,m) = \mathfrak{F}_{\text{d}}^{-1}\left\{G_{\text{décalé}}(u,v)\right\} = G(u,v) * \delta\left(u - \frac{N_x}{2}, v - \frac{N_y}{2}\right) = g(n,m)\, e^{2i\pi\left(\frac{n}{2}+\frac{m}{2}\right)}$$

En conclusion, prêtez attention au nombre de fftshift que vous utilisez et aussi à quel moment vous les faites.

6.3 Qu'est-ce que la déconvolution

La déconvolution est le fait de défaire une opération de convolution. Par exemple, lorsqu'on fait une mesure avec un appareil, cet appareil peut se modéliser par sa réponse impulsionnelle (réponse à un dirac) qui caractérise comment l'appareil déforme la grandeur mesurée. De plus, il ne faut pas oublier qu'il peut y avoir du bruit qui vient s'ajouter à notre mesure. Plus formellement, si on définit i la mesure que l'on souhaite avoir, h la réponse impulsionnelle, η du bruit de mesure et y la mesure observée, on peut écrire le problème direct comme :

$$y = i * h + \eta$$

où $*$ représente le produit de convolution. La déconvolution consiste à retrouver i... ce qui n'est pas forcément une opération facile surtout dans le domaine spatial. Une approche plus simple est de voir le problème dans le domaine frquentiel, ce qui s'écrit :

$$Y = I \cdot H + N$$

avec Y, I, H et N les transformées de Fourier de y, i, h et η respectivement. On vous propose de tester trois méthodes de déconvolution (vous verrez que deux sont vraiment similaires). On commence par un filtrage inverse, puis filtrage inverse avec seuil et enfin une méthode un peu plus avancée, le filtrage de Wiener.

Simulation de l'acquisition d'une image :

Pour simuler une acquisition, nous vous proposons de d'utiliser une réponse impulsionnelle gaussienne, ce qui dans beaucoup de cas est une bonne approximation (ex : appareil photo) :

$$h\left(x,y\right) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

avec σ un paramètre de la modélisation. Pour la simulation, procédez à la première étape :

- chargez (*imread*) l'image de Lena et visualisez–la (ou une autre image en niveaux de gris... il faut toujours vérifier la dimension de l'image en profondeur)
- calculez la transformée de Fourier de Lena et affichez–la à côté de Lena dans l'espace direct
- créez le modèle de l'appareil. Utiliser la fonction *meshgrid* pour générer les matrices X et Y. N'oubliez pas de fixer σ. Visualisez le modèle
- prenez la transformée de Fourier du modèle et visualisez sa valeur absolue et sa phase. Utilisez les fonctions *fft2* et *fftshift* pour calculer et ré-arranger la transformée de Fourier
- multipliez, dans le domaine de Fourier, l'image de Lena et le modèle.
- revenez dans le domaine direct et observez l'influence de σ. Servez vous de *ifft2* et *ifftshift*
- pour finir la simulation, ajouter du bruit gaussien centré dont vous fixerez la variance σ_{b}^2

Filtrage inverse

On suppose que l'on connaît le modèle de notre appareil (σ), et on cherche à faire la déconvolution sans prendre de précaution.

- calculez le filtre inverse, qui est littéralement l'inverse point à point du modèle H (dans le domaine de Fourier)

- multipliez dans le domaine de Fourier l'image et le filtre inverse.
- visualisez la transformée de Fourier de l'image d'origine (I) et le résultat précédent
- revenez dans le domaine direct, et comparez votre résultat à l'image d'origine et celle acquise.
- testez avec différents paramètres du modèle et du niveau de bruit.

Filtrage inverse avec seuillage

Ce filtrage est la suite logique du filtrage inverse. Il suffit de refaire les étapes précédentes en seuillant le filtre inverse pour ses valeurs trop faibles qui peuvent induire une instabilité lors de l'inversion, par exemple à cause du bruit. Essayez de gérer les situations où le filtre inverse est trop faible... comment faire?

Filtre de Wiener

Ce filtre vise à répondreà la question : comment adapter la force du filtre en fonction du niveau de bruit ? Pour le définir, on cherche la fonction G telle que

$$\hat{i} = g * y, \qquad g = \arg\min_{g} \quad \mathbb{E}\left[\left(i - \hat{i}\right)^2\right]$$

où G la transformée de Fourier de g et son expression est

$$G = \frac{\overline{H}S}{|H|^2 S + N} = \frac{1}{H}\frac{|H|^2}{|H|^2 + \frac{1}{\mathrm{SNR}}}$$

ou $\overline{H}$ est le complexe conjugé de H, et S est la densité spectrale de puissance définie par :

$$S_{i,i}(f_x, f_y) = \mathfrak{F}\left\{(i * i)(-\tau_x, -\tau_y)\right\}(f_x, f_y)$$

et le SNR est le rapport signal à bruit défini comme le rapport de la puissance du signal sur celle du bruit

$$\mathrm{SNR} = \frac{P_{\mathrm{signal}}}{P_{\mathrm{bruit}}}$$

avec la puissance définie comme :

$$P_i = \lim_{x_1 \to \infty} \lim_{x_2 \to \infty} \frac{1}{2x_1}\frac{1}{2x_2}\int_{-x_1}^{x_1}\int_{-x_2}^{x_2} |i(x_1, x_2)|^2 \,\mathrm{d}x_1 \mathrm{d}x_2$$

Pour une image, on considère que c'est le motif de base d'un signal périodique 2D. La formulation de la puissance devient :

$$P_i = \frac{1}{N_x}\frac{1}{N_y}\int_{-\frac{N_x}{2}}^{\frac{N_x}{2}}\int_{-\frac{N_y}{2}}^{\frac{N_y}{2}} |i(x_1, x_2)|^2 \,\mathrm{d}x_1 \mathrm{d}x_2$$

avec (N_x, N_y) période 2D du signal (taille de l'image).

Pour tester ce filtre, suivez les étapes :

- Estimez le SNR de l'image. Rappel : la puissance d'un bruit gaussien est sa variance
- Calculez la deuxième formule du filtre
- Appliquez le filtre à l'image

- Observez le résultat

On se propose de regarder la première formulation du filtre

- Estimez la densité spectrale de puissance de l'image en utilisant les fonctions *corr2* et *fft2*
- Calculez la première formule du filtre
- Appliquez le filtre à l'image
- Observez le résultat

7 Segmentation

7.1 À lire

Pour faire le calcul du nombre de pixels ayant une valeur supérieure à 128 dans une image en niveau de gris codée sur 8 bits contenue dans M, on peut utiliser la commande suivante

$$sum(double(M(:)>=128))$$

De même, pour faire la somme de tous les pixels ayant une valeur supérieure à 128, on peut utiliser :

$$sum((M(M >= 128)))$$

$M >= 128$ renvoie une matrice de booléens vrais là où la condition est rencontrée et faux sinon. Ainsi, $M(M >= 128)$ renvoie un vecteur avec les valeurs de M supérieures à 128.

7.2 K-moyenne

La segmentation consiste à associer une classe d'appartenance à des échantillons $x(i,j)$. Dans notre cas, l'échantillon est l'image et les classes repérées par les labels c_k sont caractérisées par leur moyennes μ_k pour $k \in [\![1, K]\!]$. On cherche à segmenter l'image au sens du minimum de la valeur :

$$T = \sum_{k \in [\![1,K]\!]} \sum_{\text{pixel}(i,j)} (x(i,j) - \mu_k)^2$$

L'algorithme qui minimise cette valeur est l'algorithme des k-moyennes. Il fonctionne comme indiqué ci après :

0 Initialisation des centres μ_k des classes c_k

1 Affectation de chaque pixel à sa classe la plus proche

2 Calcul des nouvelles valeurs μ_k

3 Refaire les étapes 1 et 2 jusqu'à convergence

Construire une fonction qui implémente cet algorithme et l'utilise sur l'image de Lena.