MA323 : Techniques avancées en Signal et Image

BE Markov et texture

Auteur:
Gweltaz LEVER

Responsable du cours:
X. DESCOMBES

ISAE - Supaero
10 Avenue Edouard Belin, 31400 TOULOUSE

février 2015

Sommaire

Table des figures

BE 1

Modélisation markovienne

L'objectif de ce bureau d'étude est d'effectuer de la segmentation d'image à l'aide d'une modélisation markovienne. Pour ce faire, on cherche à maximiser la probabilité a posteriori. Le maximum correspond alors à la meilleure segmentation possible.

On considère que l'on veut segmenter notre image en n classes gaussiennes. On note k une de ces classes, μ_k sa moyenne et σ_k^2 sa variance. La probabilité a posteriori que le pixel $x \in X$ appartienne à la classe $k \in [1,n]$ s'écrit, sous l'hypothèse d'indépendance des données conditionnellement à la carte de segmentation et en se basant sur le modèle de Potts pour définir un a priori qui régularise le résultat :

$$p(K=k|X=x) \propto exp(-U(k)) = exp\Big(-\sum_{s\in S}\big(\frac{1}{2}ln\sigma_{k_s}^2 + \frac{(x_s-\mu_{k_s})^2}{2\sigma_{k_s}^2}\big) + \beta \sum_{s,t\in C_2}\big(1-\delta(k_s,k_t)\big)\Big) \tag{1.1}$$

Maximiser la probabilité précédente équivaut à minimiser l'énergie globale $U(k)$. Etant donné qu'il n'est pas possible de tester toutes les combinaisons, différents algorithmes basés sur un emphrecuit simulé sont utilisés afin d'atteindre ce minimum : l'algorithme de Métropolis et l'échantillonneur de Gibbs. L'image originale testée dans de nombreux cas est l'image de la Figure 1.1

1.1 Segmentation en classes prédéfinies

Dans un premier temps, nous considérons que les classes sont prédéfinies : chaque classe est caractérisée au préalable par sa moyenne. Seuls les paramètres de la méthode de segmentation restent à déterminer afin de minimiser notre énergie U :

— T_0, la température initiale du recuit

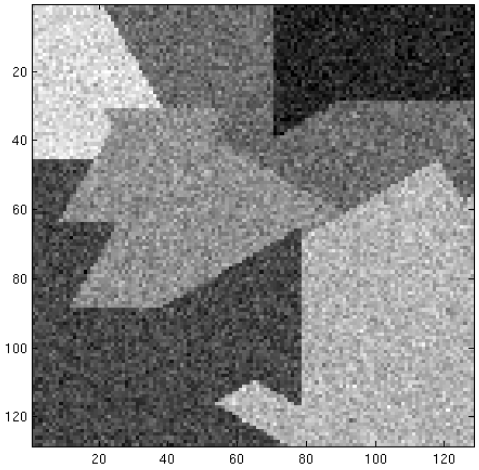


FIGURE 1.1 Image originale

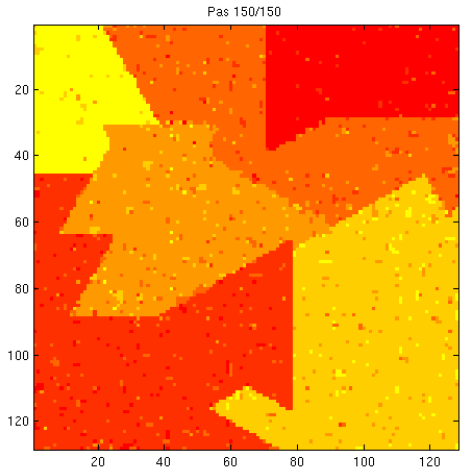
— q_{max}, le pas d'itération
— α, le coefficient caractérisant la rapidité avec laquelle on diminue la température du recuit
— β, l' "hyper-paramètre" du modèle. Plus β est grand plus on favorise l'uniformité. En effet, si β est élevé, deux voisins auront de plus grandes chances d'appartenir à la même classe.

1.1.1 Dynamique de Métropolis

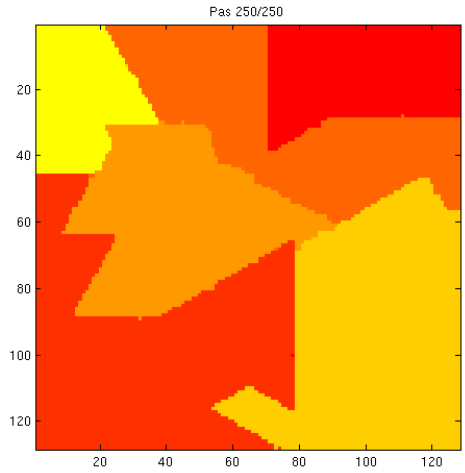
Dans un premier temps, notre recuit suit une dynamique de Métropolis. Ainsi, on attribut de manière aléatoire à chaque pixel une nouvelle classe. Si l'*énergie nouvelle* est inférieure à l'*énergie courante* alors on modifie la classe du pixel. Autrement, on modifie cette classe avec une probabilité :

$$p = exp(-(U_{Nouv} - U_{Cour})/T) \tag{1.2}$$

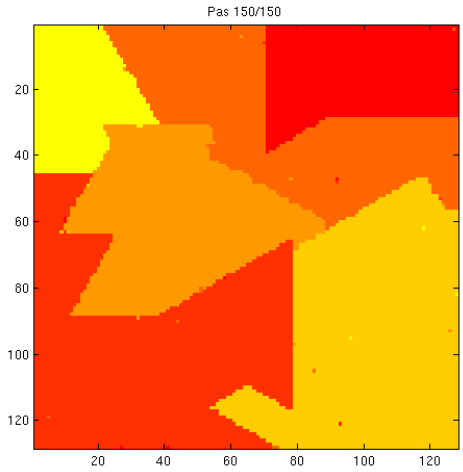
Deux voisinages ont été testés : un modèle de Potts 4-connexe (basé sur 4 clics) et un modèle de Potts 8-connexe (basé sur 8 clics).



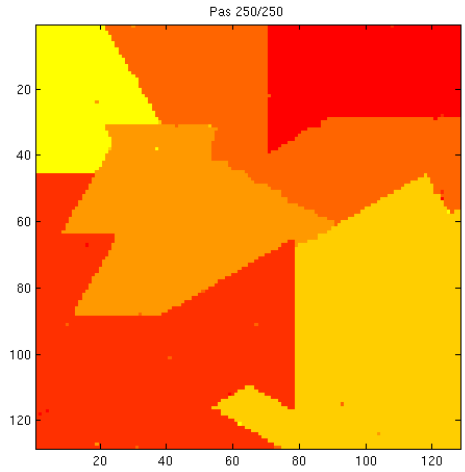
(a) Cas 1, 93.44%
$T_0 = 1.0$ $q_{max} = 150$ $\alpha = 0.99$ $\beta = 1.0$



(b) Cas 2, 99.58%
$T_0 = 2.0$ $q_{max} = 250$ $\alpha = 0.99$ $\beta = 5.0$



(c) Cas 3, 99.58%
$T_0 = 1.0$ $q_{max} = 150$ $\alpha = 0.9$ $\beta = 2.0$



(d) Cas 4, 99.60%
$T_0 = 1.0$ $q_{max} = 250$ $\alpha = 0.995$ $\beta = 2.0$

FIGURE 1.2 Comparaison des résultats de l'algorithme de Métropolis pour divers configurations

Modèle de Potts 4-connexe

La première étape, une fois l'algorithme implémenté, est d'obtenir la meilleure combinaison des paramètres $(T_0, q_{max}, \alpha, \beta)$. L'algorithme a été testé dans les quatre cas suivants : $(1.0, 150, 0.99, 1.0)$, $(2.0, 250, 0.99, 5.0)$, $(1.0, 150, 0.9, 2.0)$, $(1.0, 250, 0.995, 2.0)$. Les résultats sont représentés Figure 1.2.

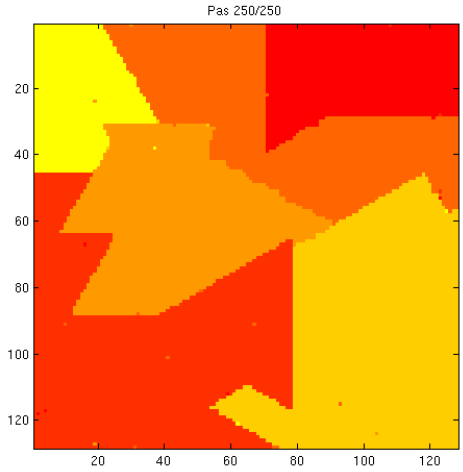
Dans le premier cas, les pixels sont classés correctement à 93.44%. Ceci résultat plutôt faible pouvant être lié à une valeur d'uniformisation β trop faible, un nombre d'itérations q_max trop faible, ou une température T_0 trop faible, une deuxième configuration à température, nombre d'itérations et paramètre β plus élevés a été testée. Les résultats sont effectivement meilleurs. En revanche, la valeur de β étant très élevée, il est possible que les contours soient trop pénalisés. De plus, une température et un nombre d'itérations trop élevés vont avoir tendance à ralentir l'algorithme. Afin de palier à ces problèmes et de trouver une configuration rapide de notre algorithme, la température, le nombre d'itérations et α ont été à nouveau abaissés. les résultats sont toujours aussi bons. Le risque dans ce cas 3 peut-être que l'algorithme ne converge pas vers le minimum global du fait de l'abaissement rapide de la température. Enfin, la configuration 4 semble apporter la meilleure segmentation de notre image. C'est donc ce compromis entre rapidité de la convergence et qualité de la segmentation qui a été choisi pour la suite.

Modèle de Potts 8-connexe

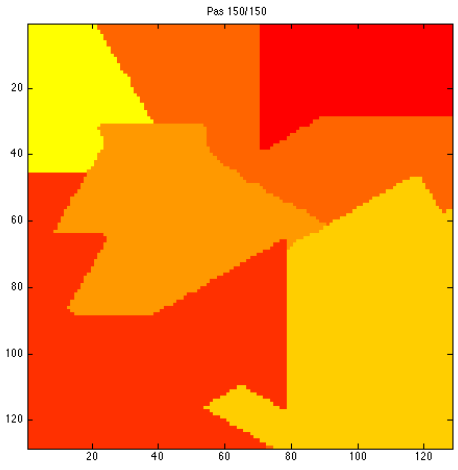
Afin d'étudier l'impact du choix du voisinage sur la segmentation de notre image, un modèle à huit clics a été testé. Là encore, le cas 4 apporte la meilleure segmentation. Le résultat est légèrement meilleur que celui obtenu dans le cas du modèle de Potts à quatre voisins (cf. Figure 1.3).

Diminution du nombre de classes

L'image précédente a été segmentée en quatre classes et non plus six. On remarque que les classes dont on connaît toujours les bonnes valeurs de σ_k^2 et de μ_k sont segmentées correctement. Ainsi, les classes en haut à droite et en bas à gauche sont correctement segmentées. En revanche, les classes dont on ne connait plus la variance et la moyenne sont affectées à une mauvaise classe, Figure 1.4.

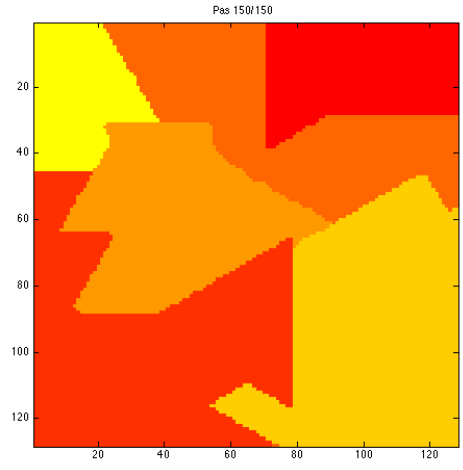


(a) Résultat Potts 4-connexe, 99.60%
$T_0 = 1.0$ $q_{max} = 250$ $\alpha = 0.995$ $\beta = 2.0$

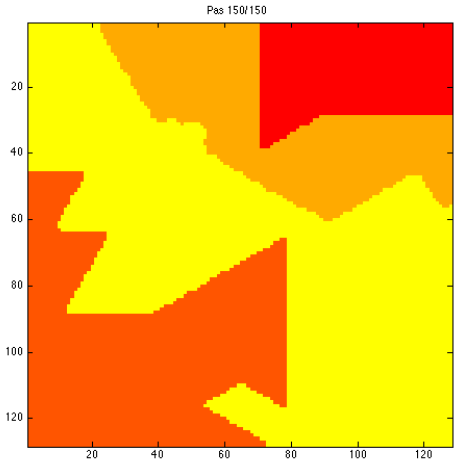


(b) Résultat Potts 8-connexe, 99.71%
$T_0 = 1.0$ $q_{max} = 150$ $\alpha = 0.995$ $\beta = 2.0$

FIGURE 1.3 Comparaison des modèles de Potts



(a) Résultat 6 classes



(b) Résultat 4 classes (68.7%)

FIGURE 1.4 Comparaison de la segmentation lors de la diminution du nombre de classes, $T_0 = 1.0$ $q_{max} = 150$ $\alpha = 0.995$ $\beta = 2.0$

Les performances

Les performances obtenues sont les suivantes :

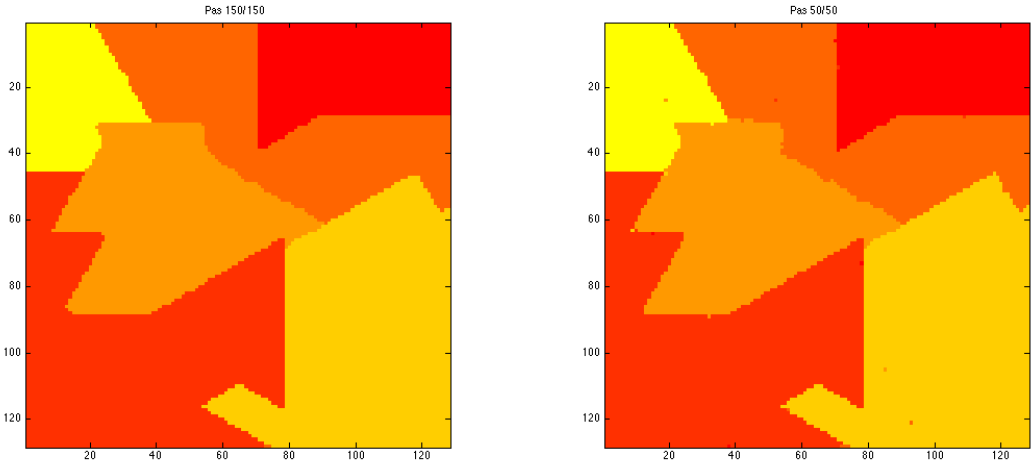
Algorithme	Paramètres				Performances	
	T_0	q_{max}	α	β	%	Temps (s)
Metropolis 4c	1.0	250	0.995	2.0	99.60	17.35
Metropolis 8c	1.0	200	0.995	2.0	99.66	15.77
Gibbs 8c	1.0	10	0.99	1.0	99.60	7.52

TABLE 1.1 Comparaison des performances des différents algortihmes

1.1.2 Echantillonneur de Gibbs

L'algorithme de Métropolis est maintenant remplacé par l'échantillonneur de Gibbs. La nouvelle classe de notre pixel courant n'est plus choisie de manière aléatoire avec une probabilité égale pour l'ensemble de nos classes. Pour chaque classe, on associe au pixel courant une probabilité calculée à l'aide de la Formule *(1.1)*. En normalisant ces probabilités par la somme des probabilités des différentes classes, on obtient un segment $[0,1]$ divisé en n parties. Un nombre aléatoire compris entre 0 et 1 est alors tiré. Selon le segment auquel il appartient, on choisit notre nouvelle classe.

A chaque itération, l'échantillonneur de Gibbs est donc plus coûteux que la dynamique de Métropolis, en revanche, il nécessite moins d'itérations car la nouvelle classe est choisie plus judicieusement. En effet, l'algorithme converge en 50 itérations vers une segmentation correcte à 99.66%, Figure 1.5.



(a) Dynamique de Métropolis : 99.71%
$T_0 = 1.0$ $q_{max} = 250$ $\alpha = 0.995$ $\beta = 2.0$

(b) Echantillonneur de Gibbs : 99.66%
$T_0 = 1.0$ $q_{max} = 50$ $\alpha = 0.99$ $\beta = 1.0$

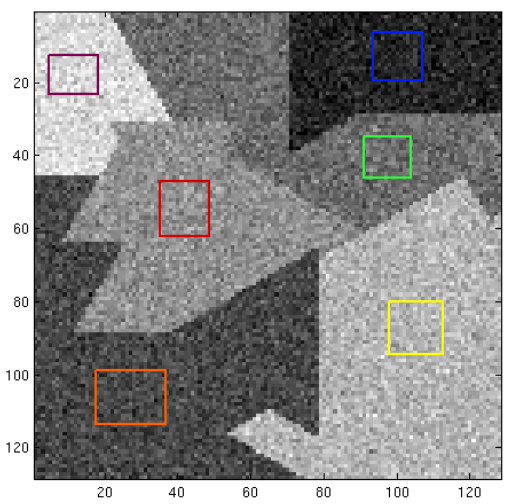
FIGURE 1.5 Comparaison des dynamiques

1.2 Segmentation par classification supervisée

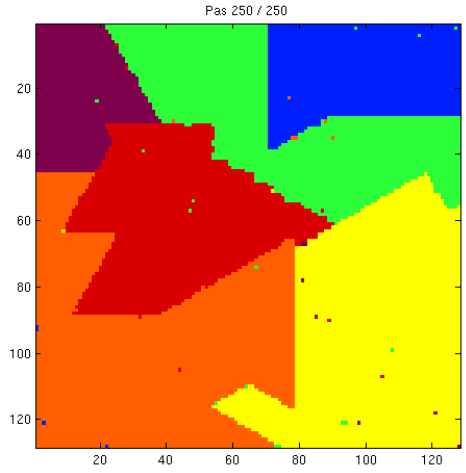
Le principe de l'algorithme est similaire à l'algorithme de Métropolis appliqué dans la première partie. La différence réside dans la manière d'estimer les paramètres des différentes classes k à savoir la moyenne μ_k et la variance σ_k^2. Les classes ne sont plus prédéfinies. On en estime les paramètres cités précédemment en sélectionnant des échantillons de chaque classe dans l'image à segmenter. A partir de ces échantillons, la moyenne et variance sont estimées. On procède donc à une classification supervisée : des informations a priori sur les échantillons sont récoltées.

1.2.1 Application à notre image

L'estimation des paramètres à l'aide des échantillons de la Figure 1.6a et la dynamique de Métropolis nous ont permis d'obtenir la segmentation représentée Figure 1.8b. Avec les paramètres $T_0 = 1.0$ $q_{max} = 250$ $\alpha = 0.995$ $\beta = 2.0$, 99.55% des pixels de l'image ont été correctement classifiés. Le résultat est donc tout aussi bon que dans le cas où les classes sont prédéfinies.



(a) Echantillons sélectionnés



(b) Résultat de la segmentation 99.55%, $T_0 = 1.0$ $q_{max} = 250$ $\alpha = 0.995$ $\beta = 2.0$

FIGURE 1.6 Segmentation par classification supervisée

1.2.2 Influence des échantillons

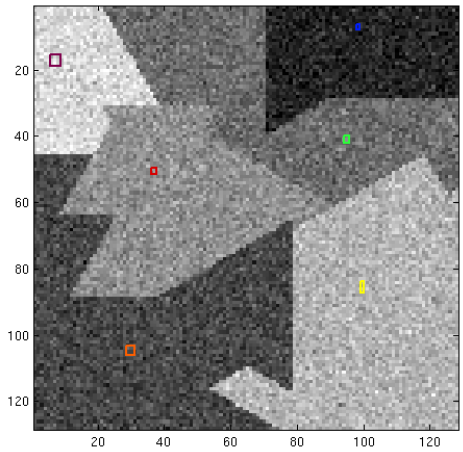
Afin de connaître l'influence des échantillons sur le résultat de la segmentation, nous avons considéré l'impact de la taille des échantillons ainsi que l'influence d'une mauvaise sélection d'un échantillon.

Influence de sa taille

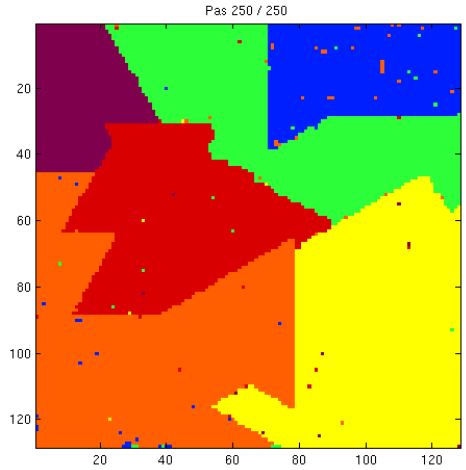
Des échantillons de petite et de grande taille ont été sélectionnés afin d'estimer les moyennes et les variances de chaque classe. Les résultats sont présentés Figure 1.7 et Figure 1.8. Lorsque les échantillons sont trop petits, les moyennes et variances obtenues ne sont plus représentatives de l'échantillon global. Ceci va donc impacter la classification. C'est pourquoi la classification est un peu moins bonne dans le cas de petits échantillons que dans le cas précédent. La sélection des grands échantillons ne modifie pas considérablement le pourcentage de pixel bien classés.

Mauvaise sélection d'un échantillon

Une mauvaise sélection d'échantillon conduit à une mauvaise segmentation. Comme on aurait pu s'y attendre, la classe non sélectionnée est réparti dans les classes d'intensités et de variances voisines. Ainsi, la classe centrale est affectée à la classe verte et à la case jaune. L'échantillon qui a été sélectionné deux fois est divisé entre les deux couleurs correspondantes,

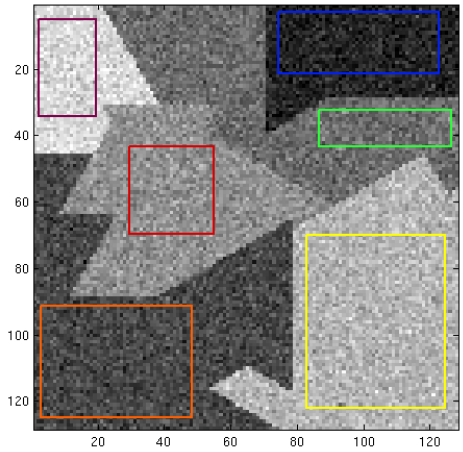


(a) Petits échantillons

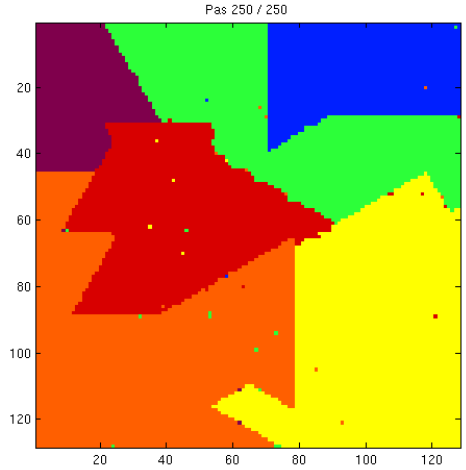


(b) Résultat de la segmentation 99.15%, $T_0 = 1.0$ $q_{max} = 250$ $\alpha = 0.995$ $\beta = 2.0$

FIGURE 1.7 Segmentation par classification supervisée avec de petits échantillons



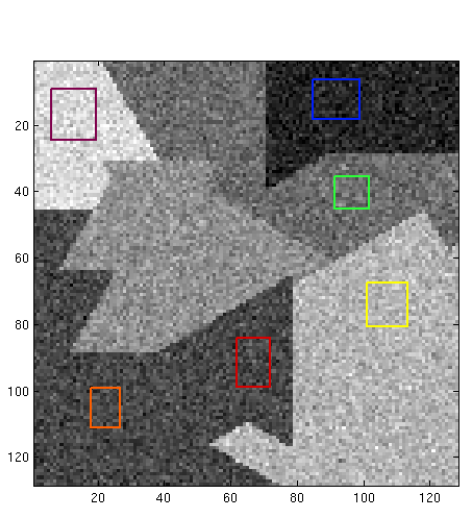
(a) Grands échantillons



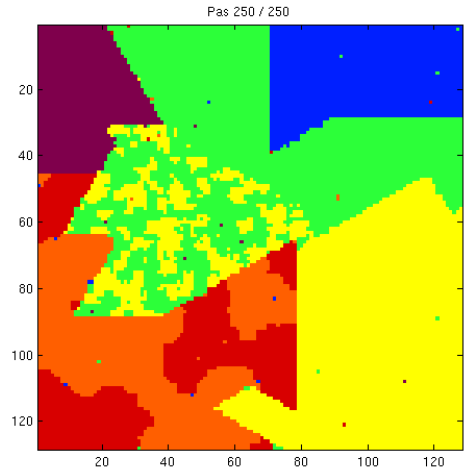
(b) Résultat de la segmentation 99.49%, $T_0 = 1.0$ $q_{max} = 250$ $\alpha = 0.995$ $\beta = 2.0$

FIGURE 1.8 Segmentation par classification supervisée avec de grands échantillons

Figure 1.9.



(a) Mauvais échantillons

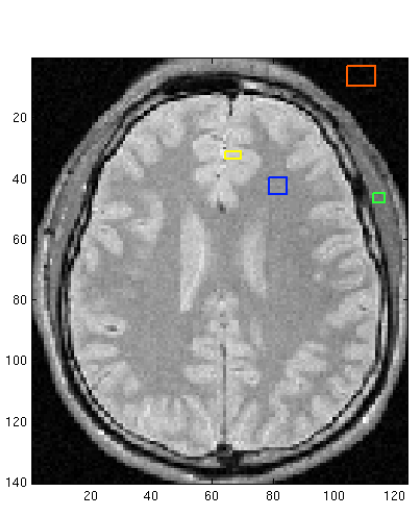


(b) Résultat de la segmentation 62.03%, $T_0 = 1.0$ $q_{max} = 250$ $\alpha = 0.995$ $\beta = 2.0$

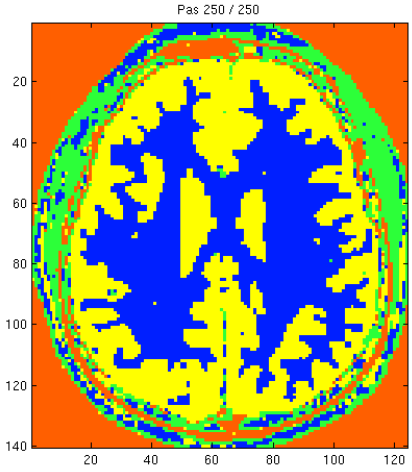
FIGURE 1.9 Segmentation par classification supervisée avec un échantillon mal sélectionné

1.2.3 Application à un cerveau

L'avantage de cette nouvelle méthode est qu'elle est transférable à d'autres images en niveaux de gris. En effet, il n'est pas nécessaire de connaître les moyennes et les variances des classes puisqu'elles sont obtenues directement lors de la phase de sélection des échantillons. Nous l'avons donc appliquée à une image scanner d'un cerveau. En sélectionnant quatre classes, à savoir la matière grise, la matière blanche, la boîte crânienne et la partie extérieure du cerveau, on obtient la segmentation de la Figure 1.10.



(a) Selection de 4 classes d'échantillons



(b) Résultat de la segmentation, $T_0 = 1.0$ $q_{max} = 250$ $\alpha = 0.995$ $\beta = 2.0$

FIGURE 1.10 Segmentation d'une image de cerveau

1.3 Débruitage

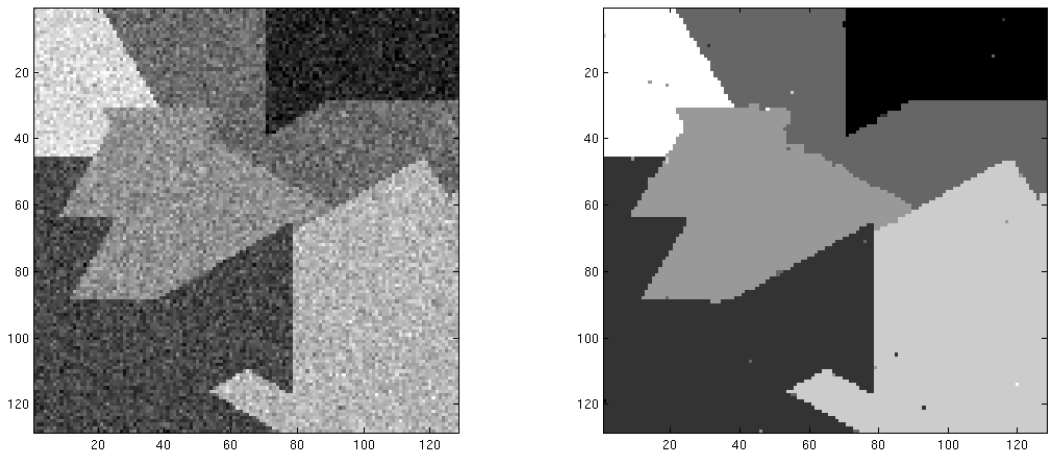
Les principes utilisés lors de la segmentation peuvent aussi servir à débruiter une image. En effet, si l'on considère que la variance au sein de chaque classe résulte d'un bruit qu'il convient de supprimer, le principe de segmentation peut permettre d'enlever ce bruit.

1.3.1 Une première méthode

La première méthode utilisée consiste à affecter à chaque pixel d'une même classe la valeur moyenne de cette classe. Le résultat de cette méthode est présenté Figure 1.11. Cette méthode est efficace quand l'image peut être clairement caractérisée par un nombre fini de niveaux de gris dont on connaît la moyenne et la variance. Les zones critiques sont naturellement les contours qui sont plus difficiles à classer correctement du fait de leur voisinage qui est composé de pixels appartenant deux classes distinctes.

1.3.2 Débruitage basé sur un modèle markovien gaussien

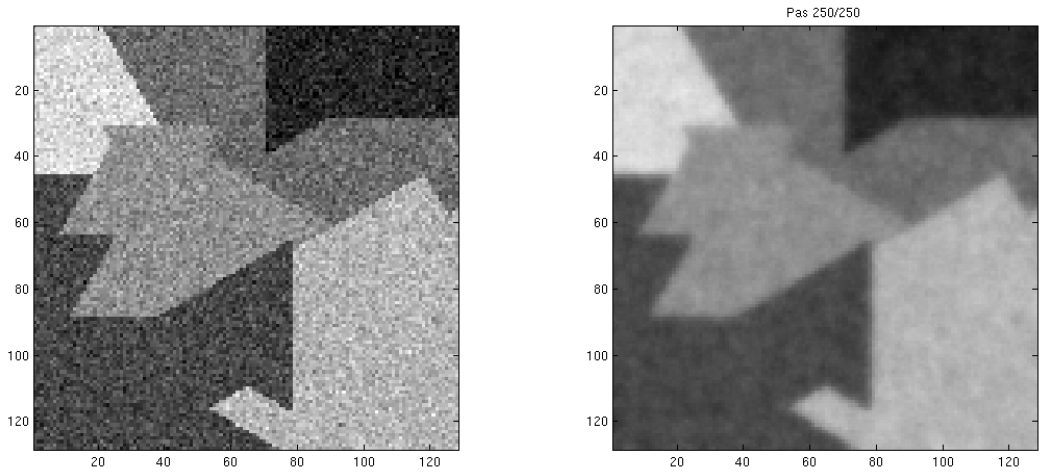
Le second débruitage peut s'appliquer à toute image sans connaissance *a priori* sur les classes de l'image. Il vise à débruiter une image en utilisant un modèle markovien gaussien. Le résultat est présenté Figure 1.12. On constate que cette fois, nous n'avons pas de pixels très mal classifiés au centre d'une classe comme c'était le cas lorsqu'on appliquait la première



(a) Original (b) Après filtrage

FIGURE 1.11 Débruitage

méthode. En revanche, l'image apparaît floutée et ne semble pas entièrement débarrassée de son bruit. Les contours semblent moins nettes.



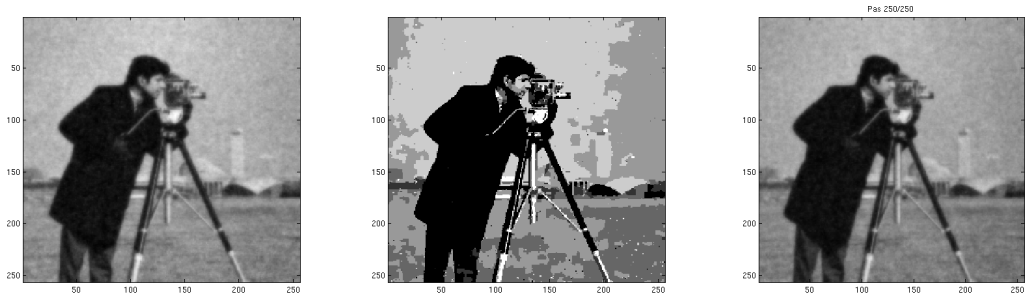
(a) Original (b) Image débruitée

FIGURE 1.12 Débruitage à l'aide d'un modèle markovien gaussien

1.3.3 Application à une autre image

Dans le cas du cameraman, il est possible de filtrer une image préalablement bruitée à l'aide des deux algorithmes précédents. Les résultats sont représentés Figure 1.13. Comme

on pouvait s'y attendre, les classes ne correspondants pas aux classes du cameraman pour la première méthode, l'image est altérée. Le ciel et la mer sont mal débruités. Par contre, le cameraman ressort nettement. La deuxième méthode semble moins efficace pour extraire le cameraman, mais elle n'introduit pas de mauvaise classification au niveau du fond.



(a) Image originale bruitée (bruit gaussien)

(b) Image débruitée (méthode 1)

(c) Image débruitée (méthode 2)

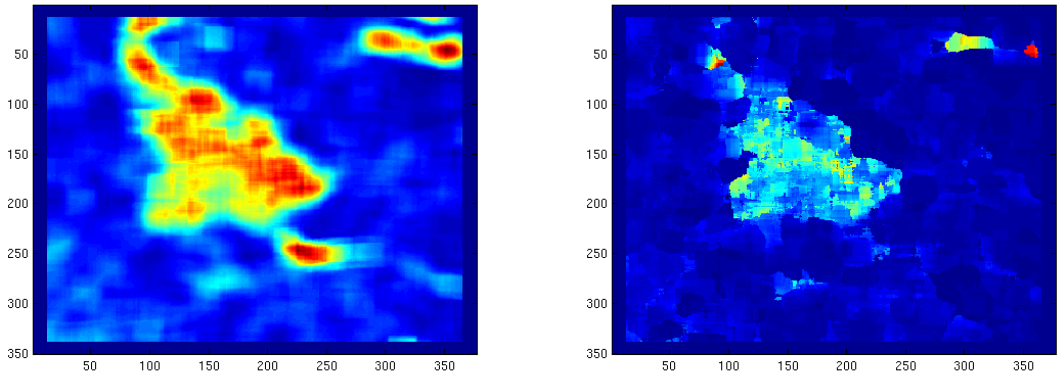
FIGURE 1.13 Débruitage du cameraman à l'aide de la première méthode et du modèle markovien gaussien

1.4 Prise en compte de la texture dans la segmentation

Lors d'un précédent BE, nous avions cherché à extraire une zone urbaine d'une image d'un village. Le niveau de gris était insuffisant pour effectuer cette extraction. Nous allons donc extraire un paramètre de texture et appliquer les algorithmes de segmentation précédents à cette image de texture.

1.4.1 Obtention des estimations

Deux estimateurs sont utilisés pour extraire les informations de texture de l'image. Pour ce faire, on modélise notre image par un champ markovien gaussien 4-connexe isotrope. La première étape consiste à estimer pour chaque pixel la valeur de la moyenne d'intensité sur son voisinage. Chaque pixel est donc caractérisé par son intensité et la moyenne des intensités sur son voisinage.



(a) Première estimation (b) Deuxième estimation

FIGURE 1.14 Obtention des estimations

Dans un second temps, on se place sur une fenêtre glissante W_s centrée en s et on calcule la probabilité conditionnelle :

$$c_{ij} = P(x_t = i | m_t = j) = \frac{\#_{ij}}{\#_i} \tag{1.3}$$

On cherche donc la probabilité qu'un pixel soit d'intensité i sachant que sa moyenne est d'intensité est j, avec :

$$\#_{ij} = card t \in W_s : x_t = i \; et \; m_t = j \tag{1.4}$$

et

$$\#_i = card t \in W_s : m_t = j \tag{1.5}$$

En calculant :

$$\hat{\sigma}^2(j) = \left(\sum_i c_{ij} i^2\right) - \left(\sum_i c_{ij} i\right)^2 \tag{1.6}$$

on obtient les estimateurs

$$E_1 = \frac{\sum_j \#_j \hat{\sigma}^2(j)}{\sum_j \#_j} \tag{1.7}$$

et

$$E_2 = \hat{\sigma}^2(argmax_j \#_j) \tag{1.8}$$

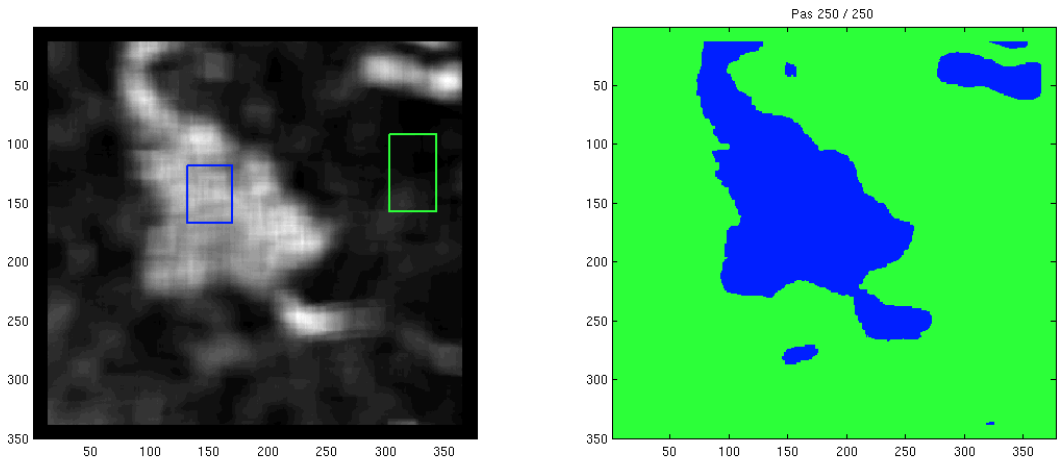
L'application des estimateurs à l'image contenant la zone urbaine nous permet d'obtenir les images de texture Figure 1.14.

1.4.2 Segmentation à partir des estimations

Il s'agit maintenant de segmenter les images de texture obtenues.

Le résultat par simple seuillage

La première méthode, la plus simple consiste à segmenter l'image de l'estimateur 1 par simple seuillage. Le résultat pour chaque estimateur est très correct, Figure 1.15.

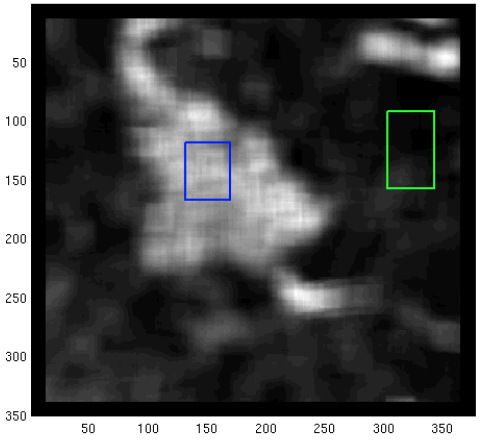


(a) Seuillage de l'estimation 1 (b) Seuillage de l'estimation 2

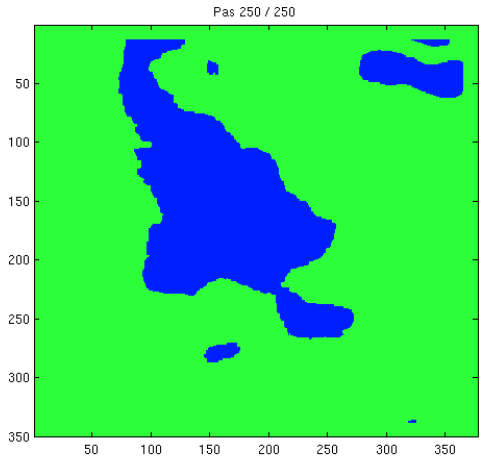
FIGURE 1.15 Segmentation par seuillage des deux estimations

Estimation 1

En appliquant le programme de segmentation par classification supervisée et en sélectionnant deux classes : une classe pour la zone urbaine et une autre pour les champs à l'estimateur 1, on obtient le résultat Figure 1.16. Le résultat est comparable au seuillage.



(a) Échantillon des deux classes

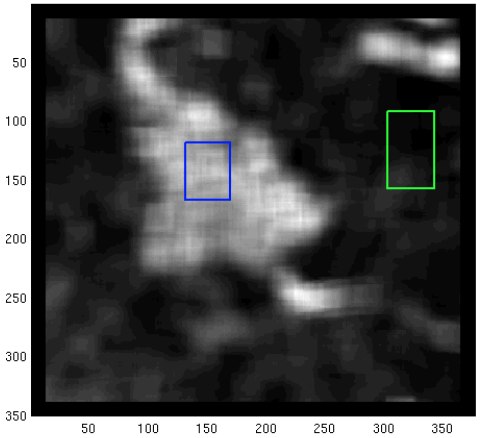


(b) Résultat de la segmentation, $T_0 = 1.0$ $q_{max} = 250$ $\alpha = 0.995$ $\beta = 2.0$

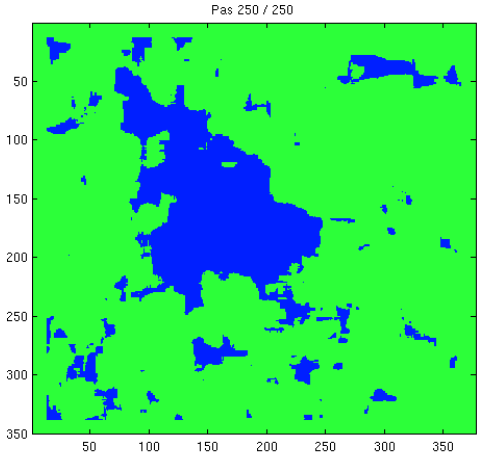
FIGURE 1.16 Segmentation à l'aide de la première estimation

Estimation 2

En appliquant le programme de segmentation par classification supervisée et en sélectionnant deux classes : une classe pour la zone urbaine et une autre pour les champs à l'estimateur 1, on obtient le résultat Figure 1.16. Le résultat est moins bon que les précédents : il semble parasité au niveau des zones de fort contour, notamment aux frontières entre les champs.



(a) Échantillon des deux classes



(b) Résultat de la segmentation, $T_0 = 1.0$ $q_{max} = 250$ $\alpha = 0.995$ $\beta = 2.0$

FIGURE 1.17 Segmentation à l'aide de la première estimation