Détection de contours et lissage d'image par deux algorithmes déterministes de relaxation. Mise en œuvre sur la machine à connexions CM2

Edge detection and image smoothing using two deterministic relaxation algorithms. Implementation on the connection machine CM2

Josiane ZERUBIA
INRIA Sophia Antipolis,
2004 Route des Lucioles,
06560 Valbonne, France

Josiane Zerubia est actuellement chargée de recherche à l'INRIA. Diplomée de l'ENSIEG-INPG (1981), elle a travaillé comme ingénieur de recherche chez Hewlett-Packard jusqu'en 1984. Elle a préparé une thèse de docteur-ingénieur (1986) et une nouvelle thèse (1988) au LASSY - Université de Nice - URA CNRS no 1376. Puis elle a passé une année post-doctorale au « Signal and Image Processing Institute » de USC à Los Angeles en 1989.

Principaux thèmes de recherche : traitement d'image, modélisation par champs de Markov, estimation de paramètres, connexionisme.

Florimond PLOYETTE
INRIA Sophia Antipolis,
2004 Route des Lucioles,
06560 Valbonne, France

Florimond Ployette est ingénieur de recherche à l'INRIA. Il a obtenu une thèse de docteur ingénieur en 1980 sur la compilation parallèle d'un langage de programmation sur un système distribué. Il s'est intéressé d'abord à la traduction des langages algorithmiques, puis à l'expression du parallélisme à grain fin dans les langages. Il a participé à la conception et la mise en œuvre de langages parallèles dans le cadre de travaux sur des systèmes distribués tolérant aux fautes.

Depuis fin 1989, il s'intéresse à la parallélisation d'algorithmes de vision et notamment à leur implantation sur une machine massivement parallèle.

RÉSUMÉ

Récemment, de nombreux algorithmes de minimisation de fonctions non convexes ont été proposés pour résoudre des problèmes de vision bas niveau. Il existe plusieurs méthodes de relaxation. Les techniques stochastiques, telles que le recuit simulé, convergent asymptotiquement, sous certaines conditions, vers le minimum global, mais sont très coûteuses en temps de calcul. Les méthodes de relaxation déterministes sont sous-optimales, mais donnent de bons résultats et sont plus rapides que les méthodes stochastiques.

Dans cet article, nous présentons la mise en œuvre parallèle de deux algorithmes déterministes de détection de contours et de lissage d'image : le GNC (« Graduated Non-Convexity ») proposé par Blake & Zisserman et le recuit par champs moyens (MFA) introduit par Geiger & Girosi et étendu aux champs de Markov composés anisotropes par Zerubia & Chellappa. Ces deux méthodes sont fondées sur le modèle de la membrane à contraintes de continuité lâches et sont séquentielles : à chaque pas est produit une image qui est utilisée au pas suivant. Pour le GNC, nous avons utilisé une méthode de minimisation de l'énergie appelée « successive over-relaxation (SOR) » et plus précisément une variante parallèle de cette technique. En ce qui concerne l'algorithme MFA, nous avons utilisé une méthode de descente de gradient conjugué à pas optimal.

MOTS CLÉS

Machine à Connexions, parallélisme de données, SIMD, Algorithmes déterministes de relaxation, détection de contours, lissage d'image.

ABSTRACT

Recently, a lot of algorithms minimizing a non-convex energy function have been proposed to solve low level vision problems. Different kinds of relaxation methods are available. The stochastic techniques, such as simulated annealing, asymptotically converge to the global minimum but require a high computational cost. Deterministic relaxation methods which are sub-optimal, give good results and are faster than the stochastic ones.

In this paper, we focus on the parallel implementation of two deterministic algorithms for edge detection and image smoothing : the graduated non-convexity (GNC) originally proposed by Blake & Zisserman and the mean field annealing (MFA) introduced by Geiger & Girosi and extended to anisotropic compound Gauss-Markov random fields by Zerubia & Chellappa. Both methods are based on a weak-membrane model and both algorithms are inherently serial : each step produces a pixel map which is taken as an input for the next step. For the GNC, we implement a checkerboard version of the successive over-relaxation (SOR) method to minimize the energy. For the MFA, we use an optimal step conjugate gradient descent.

KEY WORDS

Connection Machine, Data parallelism, SIMD, Deterministic relaxation algorithms, Edge detection, Image smoothing.

1. Introduction

L'utilisation des champs de Markov [13] en vision par ordinateur s'est développée depuis quelques années. Les principales applications de ce type de modélisation sont la détection de contours [3], [7], [25], [30], la restauration ou le lissage d'image [3], [8], [18], [25], [30], la reconstruction de surface [3], [7], la stéréoscopie [29], la synthèse et la classification de texture [5], [9], [23].

Le principal avantage des champs de Markov est de fournir un modèle mathématique simple permettant de prendre en compte les interactions locales au niveau du pixel. Un inconvénient majeur est le coût relatif à tout algorithme obtenu à partir d'une telle modélisation. C'est pourquoi le développement sur des machines parallèles d'algorithmes basés sur des champs de Markov est intéressant.

Dans cet article, nous présentons les résultats obtenus en détection de contours et lissage d'image grâce à deux méthodes déterministes de relaxation : le GNC (« Graduated Non Convexity ») [3], [25] et le recuit par champs moyens [7], [30]. Ces algorithmes itératifs ont été mis en œuvre sur la machine à connexions CM2 de l'INRIA (Institut National de Recherche en Informatique et Automatique).

2. Présentation de deux méthodes déterministes de relaxation

2.1. RAPPEL SUR LES CHAMPS DE MARKOV ET LES PROCESSUS DE LIGNE

Soit une image $\{u(s)\}$ définie sur un réseau fini Ω de sites s : $\Omega = \{s = (i,j) ; 1 \leq i,j \leq M\}$.

Considérons une famille arbitraire N de voisinages locaux de Ω : $N = \{N(s), s \in \Omega\}$.

A chaque famille de voisinages est associée une classe de distribution de probabilités, appelée champ de Markov et caractérisée par la propriété suivante :

$$P(u(s)\,|\,u(r), r \in \Omega - \{s\}) = P(u(s)\,|\,u(r), r \in N(s) - \{s\}), \quad (1)$$

ce qui signifie que la connaissance d'un voisinage local du pixel s est suffisante pour calculer la probabilité en chaque pixel s.

Une autre caractéristique importante vient du théorème de Hammersley-Clifford [1] qui montre qu'un champ aléatoire défini sur un réseau est un champ de Markov si et seulement si sa distribution de probabilité est une distribution de Gibbs, définie par :

$$P(u) = \frac{\exp - \frac{E(u)}{T}}{Z_{(T)}} \quad (2)$$

où $E(u)$ est une fonction énergie définie par :

$$E(u) = \sum_{c \in C} V_c(u) \quad (3)$$

où C est l'ensemble des cliques correspondant au voisinage choisi.

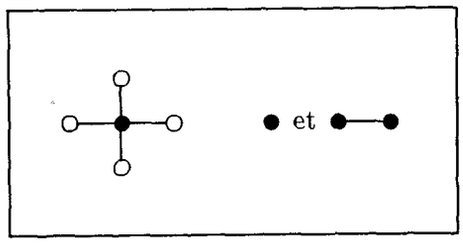


Figure 1. — Voisinage d'ordre 1 et cliques correspondantes.

$V_c(u)$ est une fonction potentiel permettant d'exprimer les contraintes et T est la température. $Z_{(T)}$ est appelée la

fonction de partition. C'est une constante définie par :

$$Z_{(T)} = \sum_{\text{ttes les configurations}} \exp - \frac{E(u)}{T} . \qquad (4)$$

Nous supposons que l'énergie observée est décrite par un champ d (par exemple : $d = u + n$ où n est un bruit blanc gaussien). En utilisant le théorème de Bayes, il vient :

$$P(u|d) = \frac{P(d|u)\,P(u)}{P(d)} = \frac{\exp - \frac{E(u|d)}{T}}{Z_{(T)}} . \qquad (5)$$

Dans cette expression, $P(d)$ est constant et $P(d|u)$ est connu puisque un modèle a priori a été choisi (i.e. bruit additif gaussien).

L'estimée $\hat{u}$ de u peut être obtenue par la maximisation de la probabilité a posteriori (MAP), ce qui revient à minimiser l'énergie $E(u|d)$.

Il s'avère intéressant de définir un champ booléen dual au processus intensité, appelé processus de ligne. Ce champ, introduit pour la première fois par Geman & Geman [8], représente explicitement la présence ou l'absence de discontinuités et vient ainsi rompre l'hypothèse de lissage faite par les méthodes classiques de régularisation [17].

Dans le modèle considéré, nous avons deux processus de ligne, l et m (horizontal et vertical, cf. *fig.* 2). L'énergie peut alors s'exprimer par :

$$E = \mathcal{D} + \mathcal{S} + \mathcal{P} \qquad (6)$$

avec $\mathcal{D} = \sum_{i,j} (u_{i,j} - d_{i,j})^2$ qui assure une bonne adéquation entre la solution trouvée et les données (compte tenu de l'équation (5)).

$$\mathcal{S} = \sum_{i,j} \lambda^2 [(u_{i,j} - u_{i-1,j})^2 (1 - l_{i,j}) + (u_{i,j} - u_{i,j+1})^2 (1 - m_{i,j})]$$

qui représente la contrainte de lissage classique (λ^2 * gradient^2) en tenant compte du processus de ligne (le choix de λ dépendant du rapport signal/bruit, cf. [7] section 4.1 pour une description des méthodes d'estimation).

$\mathcal{P} = \sum_{i,j} \alpha(l_{i,j} + m_{i,j})$, où $\alpha(\alpha > 0)$ est le coût à payer pour l'introduction d'une ligne (cf. [7] section 4.2).

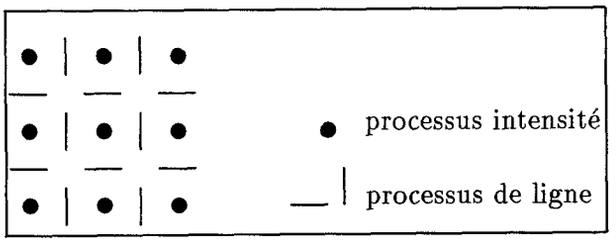


Figure 2. — Superposition des 2 réseaux.

Le modèle décrit par l'équation (6) est celui d'une membrane à contraintes de continuité lâches [3], [7], [8].

2.2. « GRADUATED NON CONVEXITY »

Pour minimiser l'énergie décrite par l'équation (6) qui est une fonction non-convexe, Blake et Zisserman [3] ont proposé un algorithme déterministe de relaxation, appelé GNC (« Graduated Non Convexity »).

Les techniques classiques de minimisation consistent soit en l'application d'une méthode déterministe, de type descente suivant la plus grande pente (auquel cas le système peut être bloqué dans un minimum local quelconque), soit en l'utilisation d'une technique stochastique de recuit simulé [8], [12], [14] (qui assure la convergence asymptotique vers le minimum global mais qui demande a priori un coût de calcul très élevé).

La méthode du GNC assure de trouver le minimum global dans des cas particuliers (par exemple si les discontinuités sont assez éloignées les unes des autres) et donne de bonnes solutions dans les autres cas (cf. Chap. 7 de [3] pour plus de détails). Nous en rappelons brièvement le principe ci-dessous.

L'équation (6) peut se mettre sous la forme :

$$E = \mathcal{D} + \sum_{i,j} h_{\alpha,\lambda}(u_{i,j} - u_{i-1,j}, l_{i,j}) + \sum_{i,j} h_{\alpha,\lambda}(u_{i,j} - u_{i,j+1}, m_{i,j}) \qquad (7)$$

avec

$$h_{\alpha,\lambda}(t, l) = \lambda^2 (t)^2 (1 - l) + \alpha l . \qquad (8)$$

La minimisation de E doit être faite par rapport à $u_{i,j}$, $l_{i,j}$ et $m_{i,j}$. Cependant, le terme $\mathcal{D}$ ne contient pas de processus de ligne. La minimisation par rapport à $l_{i,j}$ et $m_{i,j}$ peut être effectuée avant celle par rapport à $u_{i,j}$:

$$\min_{(u_{i,j})} E = \min_{(u_{i,j})} \left(\mathcal{D} + \sum_{i,j} g_{\alpha,\lambda}(u_{i,j} - u_{i-1,j}) + \sum_{i,j} g_{\alpha,\lambda}(u_{i,j} - u_{i,j+1}) \right) \qquad (9)$$

avec

$$g_{\alpha,\lambda}(t) = \min_{l \in \{0,1\}} h_{\alpha,\lambda}(t, l) . \qquad (10)$$

L'idée de base du GNC est la suivante : la première étape est la construction d'une approximation convexe E* de l'énergie E ; la minimisation de E* donne donc un minimum global. Puis une séquence d'énergies $E^{(p)}$ est construite telle que $E^1 = E^*$ et $E^0 = E$. Le GNC optimise toute la séquence d'énergies $E^{(p)}$ (par exemple, $p = 1, 1/2, 1/4, 1/8...$ pour une image) en utilisant comme conditions initiales :

- les données, pour la minimisation de E*,

• la solution trouvée lors de la minimisation de la fonction énergie précédente dans les autres cas.

Blake et Zisserman ont montré dans [3] que la fonction $g^{(p)}_{\alpha,\lambda}$ est la suivante (cf. *fig.* 3) :

$$g^{(p)}_{\alpha,\lambda}(t) = \begin{cases} \lambda^2(t)^2 & |t| < q \\ \alpha - c(|t| - r)^2/2 \quad q \leqslant & |t| < r \\ \alpha & |t| \geqslant r \end{cases}$$

avec

$$c = \frac{1}{4p}, \quad r^2 = \alpha\left(\frac{2}{c} + \frac{1}{\lambda^2}\right), \quad q = \frac{\alpha}{\lambda^2 r} \qquad (11)$$

où

$$t = (u_{i,j} - u_{i-1,j}) \quad \text{ou} \quad t = (u_{i,j} - u_{i,j+1})$$

suivant la direction considérée.

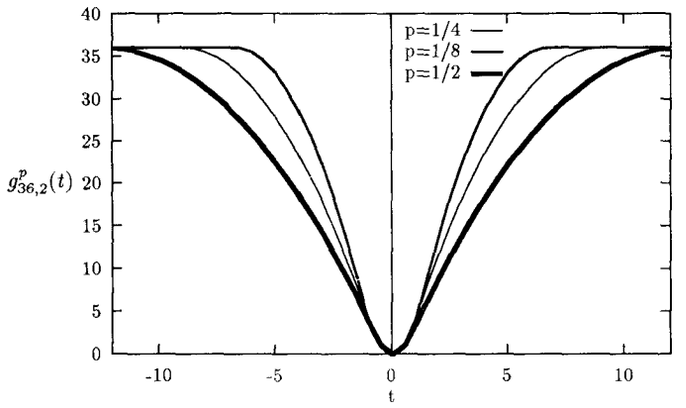


Figure 3. — La fonction $g^{(p)}_{\alpha,\lambda}(t)$.

Une fois la minimisation par rapport au champ intensité $u_{i,j}$ effectuée sur toute la série d'énergies $E^{(p)}$, les processus de ligne sont obtenus de la façon suivante :

$$l_{i,j} = \begin{cases} 1 & \text{si} \quad |u_{i,j} - u_{i-1,j}| > r \\ 0 & \text{si} \quad |u_{i,j} - u_{i-1,j}| < q \\ \text{ambigu} & \text{sinon} \end{cases}$$

Nous avons pris $l_{i,j} = 0$ pour le cas ambigu lors de la mise en œuvre de l'algorithme. Ce choix modifie l'algorithme du GNC mais c'est celui qui donne les meilleurs résultats pour l'image des contours. Pour obtenir $m_{i,j}$, il suffit de faire le même test en remplaçant $u_{i-1,j}$ par $u_{i,j+1}$ (la figure 4 donne l'algorithme du GNC).

Une extension de cette méthode au cas d'un modèle anisotrope de champs de Markov (caractérisé par θ_x, θ_y et une variance ν) a été réalisée par Simchony *et al.* [23]. Elle n'a pas été mise en œuvre sur la machine à connexions car le coût de calcul a été jugé trop important par rapport au gain de qualité obtenu pour la détection de contours et le lissage d'image (des tests ayant été effectués en séquentiel à USC).

Choisir λ et h_0.
$\alpha = h_0^2\lambda/2$.
$w = 2(1 - \frac{1}{\lambda\sqrt{2}})$
$p \in \{1, 1/2, 1/4, 1/8, 1/16, ..., 1/4\lambda\}$
Pour chaque p, **itérer** $n=1,2,...$
Pour $i = 1,...,N-1, j = 1,...,N-1$:

$$u^{(n+1)}_{i,j} = u^{(n)}_{i,j} - w\{2(u^{(n)}_{i,j} - d_{i,j}) + g^{(p)'}_{\alpha,\lambda}(u^{(n)}_{i,j} - u^{(n+1)}_{i-1,j}) + g^{(p)'}_{\alpha,\lambda}(u^{(n)}_{i,j} - u^{(n+1)}_{i,j-1}) + g^{(p)'}_{\alpha,\lambda}(u^{(n)}_{i,j} - u^{(n)}_{i+1,j}) + g^{(p)'}_{\alpha,\lambda}(u^{(n)}_{i,j} - u^{(n)}_{i,j+1})\}/(2 + 8\lambda^2)$$

Aux bords, les modifications suivantes sont à prendre en compte:
Pour $i=0, j=0$:

$$u^{(n+1)}_{0,0} = u^{(n)}_{0,0} - w\{2(u^{(n)}_{0,0} - d_{0,0}) + g^{(p)'}_{\alpha,\lambda}(u^{(n)}_{0,0} - u^{(n)}_{1,0}) + g^{(p)'}_{\alpha,\lambda}(u^{(n)}_{0,0} - u^{(n)}_{0,1})\}/(2 + 4\lambda^2)$$

et pour chaque coin de façon identique.

Pour $i=0, j=1,...,N-1$:

$$u^{(n+1)}_{0,j} = u^{(n)}_{0,j} - w\{2(u^{(n)}_{0,j} - d_{0,j}) + g^{(p)'}_{\alpha,\lambda}(u^{(n)}_{0,j} - u^{(n+1)}_{0,j-1}) + g^{(p)'}_{\alpha,\lambda}(u^{(n)}_{0,j} - u^{(n)}_{1,j}) + g^{(p)'}_{\alpha,\lambda}(u^{(n)}_{0,j} - u^{(n)}_{0,j+1})\}/(2 + 6\lambda^2)$$

et pour chaque coté de façon identique.

Figure 4. — Algorithme GNC (d'après figure 7.13 de [3]).

2.3. RECUIT PAR CHAMPS MOYENS

Une autre méthode déterministe de relaxation basée sur un recuit par champs moyens (« Mean Field Annealing ») a été introduite par Peterson et Anderson [20], [21] et appliquée à notre problème par Geiger et Girosi [6], [7].

La technique d'approximation par champs moyens est souvent utilisée en physique statistique [19], [22]. En effet, l'obtention de façon analytique de la valeur moyenne d'un champ comme fonction explicite des données et des paramètres n'est en général pas possible. La physique statistique utilise alors comme outil l'approximation par champs moyens afin d'obtenir une solution approchée donnée de façon implicite par un système d'équations non linéaires.

Tous les calculs reposent sur la fonction de partition $Z_{(T)}$ qui contient toute l'information du système physique considéré. Le calcul de la fonction de partition est équivalent à l'évaluation d'une intégrale multidimensionnelle qui ne peut pas être obtenue explicitement du fait de l'interaction entre les champs. L'approximation par champs moyens consiste à substituer l'interaction entre les champs de différents sites par l'interaction du champ de chaque site avec les valeurs moyennes des champs des sites voisins (voisinage défini suivant l'ordre du champ de Markov considéré). Alors la fonction de partition peut se factoriser en un produit de fonctions de partition relatives à un seul site et peut être calculée de façon implicite par un système d'équations non linéaires obtenues grâce à la dérivée du logarithme de la fonction de partition approximée.

initialiser $\beta = 2 * 10^{-4}$, G_0 et D_0 (cf. [22]),
puis répéter ($\beta = \beta * 4$), **tant que** $\beta \leq 1$
tant que $\Delta E < \epsilon$ **itérer** $k=0,1,\ldots$:
Mise à jour des processus de ligne suivant les équations (1 fois sur 5).
Mise à jour de l'intensité: $\forall(i,j)$,
$y_{i,j}^{k+1} = y_{i,j}^{k} + \alpha^k D_{i,j}^k$ où α^k minimise $E(y^k + \alpha D^k)$

$G_{i,j}^{k+1} = \frac{\partial E}{\partial y_{i,j}^{k+1}}$

$D_{i,j}^{k+1} = -G_{i,j}^{k+1} + \gamma_k D_{i,j}^k$

avec $\gamma^k = \frac{\sum_{i,j}(G_{i,j}^{k+1} - G_{i,j}^k)G_{i,j}^{k+1}}{\sum_{i,j}(G_{i,j}^k)^2}$

calcul du ΔE.
fin boucle
fin boucle

Figure 5. — Algorithme MFA avec Gradient Conjugué.

Cette technique a également été employée récemment pour résoudre des problèmes combinatoires comme par exemple le paritionnement de graphe [2], [10] ou le stockage d'exemples (« pattern storage ») [4] en utilisant des réseaux de neurones. Dans tous les cas, cette technique s'est avérée plus rapide par le recuit simulé (cf. [2] par exemple).

Par définition, le champ moyen $\bar{u}_{i,j}$ est donné par :

$$\bar{u}_{i,j} = \frac{\sum_{\text{ttes les config.}} u_{i,j} \exp - \frac{E(u,l,m)}{T}}{Z_{(T)}} \qquad (12)$$

$E(u, l, m)$ étant donnée par l'équation (6). Les champs moyens $\bar{l}_{i,j}$ et $\bar{m}_{i,j}$ sont définis de façon analogue.

En utilisant les équations (4), (6) et (12), Geiger et Girosi ont montré que [7] :

$$\bar{u}_{i,j} = d_{i,j} + \frac{T}{2}\frac{\partial \ln Z_{(T)}}{\partial d_{i,j}} \qquad (13)$$

et

$$\bar{l}_{i,j} = 1 - T\frac{\partial \ln Z_{(T)}}{\partial G_{i,j}^l} \qquad (14)$$

où

$$G_{i,j}^l = \alpha - \lambda^2(u_{i,j} - u_{i-1,j})^2 . \qquad (15)$$

Une équation similaire est définie pour $\bar{m}_{i,j}$.

Si nous voulons effectivement obtenir ces champs moyens, nous devons faire des approximations afin de calculer la fonction de partition $Z_{(T)}$ de façon explicite.

Considérons tout d'abord le calcul de $\bar{u}_{i,j}$. La fonction de partition $Z_{(T)}$ est définie par :

$$Z_{(T)} = \sum_{\text{ttes les config. u, l, m}} \exp - \frac{E(u,l,m)}{T} . \qquad (16)$$

La première étape consiste à éliminer les processus de ligne dans la fonction énergie E de telle sorte que :

$$E(u, l, m) = \mathcal{D}(u) + E^{\text{eff}}(u) . \qquad (17)$$

Ceci peut être calculé par analogie à la physique statistique en supposant qu'il n'existe pas d'interaction entre voisins pour les processus de ligne (cf. [6] et [7] pour plus de détails) mais cela présente certains désavantages (risque de formation de contours en escalier par exemple).

La seconde étape est d'approximer $Z_{(T)}$ (approximation du point de selle, cf. [7] pour plus de détails) :

$$Z_{(T)} = \sum_{\text{ttes les config. } u} \exp - \frac{\mathcal{D}(u) + E^{\text{eff}}(u)}{T} \qquad (18)$$

$$Z_{(T)} \approx \max_{(u)} \left(\exp - \frac{\sum_{i,j}(u_{i,j} - d_{i,j})^2 + E_{i,j}^{\text{eff}}(u)}{T} \right) \qquad (19)$$

$$Z_{(T)} \approx \exp - \frac{\sum_{i,j}(\bar{u}_{i,j} - d_{i,j})^2 + E_{i,j}^{\text{eff}}(\bar{u})}{T} . \qquad (20)$$

En pratique, cela revient à dire qu'au lieu de raisonner sur l'influence stochastique des champs des pixels voisins sur le champ du pixel (i,j), il suffit de raisonner sur l'influence exercée par les valeurs moyennes des champs des pixels voisins. Cela revient à « figer » l'environnement du pixel (i,j) dans son état moyen.

Pour obtenir $\bar{u}_{i,j}$, il suffit donc de minimiser l'énergie en faisant :

$$\frac{\partial(\mathcal{D}(u) + E^{\text{eff}}(u))}{\partial u_{i,j}} = 0 \qquad (21)$$

ce qui donne :

$$\begin{aligned}\bar{u}_{i,j} = d_{i,j} &- \lambda^2(\bar{u}_{i,j} - \bar{u}_{i,j+1})(1 - \bar{m}_{i,j}) \\ &+ \lambda^2(\bar{u}_{i,j-1} - \bar{u}_{i,j})(1 - \bar{m}_{i,j-1}) \\ &- \lambda^2(\bar{u}_{i,j} - \bar{u}_{i-1,j})(1 - \bar{l}_{i,j}) \\ &+ \lambda^2(\bar{u}_{i+1,j} - \bar{u}_{i,j})(1 - \bar{l}_{i+1,j}) . \end{aligned} \qquad (22)$$

En partant de l'équation (14) et en utilisant l'approximation par champs moyens décrite ci-dessus, il vient, après quelques calculs :

$$\bar{l}_{i,j} = \sigma_\beta(\lambda^2(\bar{u}_{i,j} - \bar{u}_{i-1,j})^2 - \alpha) \qquad (23)$$

$$\bar{m}_{i,j} = \sigma_\beta(\lambda^2(\bar{u}_{i,j} - \bar{u}_{i,j+1})^2 - \alpha) \qquad (24)$$

avec $\beta = \frac{1}{T}$ et $\sigma_\beta(x) = \frac{1}{1 + \exp - \beta x}$ (cf. *fig.* 6).

Il faut remarquer que, contrairement à la méthode du GNC, les processus de lignes ne sont plus des champs booléens mais des champs continus à valeur dans [0, 1]. Ce n'est que lorsque la température T tend vers 0 (i.e. $\beta \to \infty$) que la sigmoïde devient un échelon rendant ainsi les processus de ligne booléens.

Une extension de cette méthode au cas d'un modèle anisotrope de champs de Markov a été réalisée par Zerubia et Chellappa [30], [31]. Elle n'a pas été mise en œuvre sur la machine à connexions pour les mêmes raisons que celles invoquées au chapitre 2.2 pour le GNC.

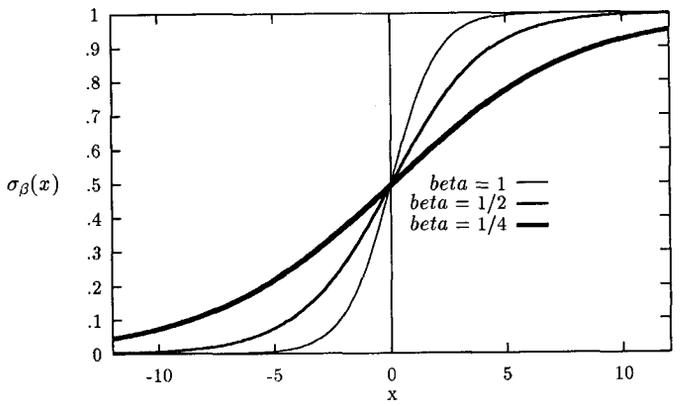


Figure 6. — La fonction sigmoïde.

3. La Machine à Connexions

Dans ce paragraphe, nous donnons une description succincte de cette machine dont on trouvera une documentation plus complète en [11], [26], [27]. La machine à connexions est une machine SIMD (« Single Instruction Multiple Data ») ayant de 8 K à 64 K processeurs. Chaque processeur est un processeur 1 bit disposant de 32 Koctets de mémoire locale et une horloge à 8 MHz. Les macro-instructions provenant du calculateur frontal sont traitées par un micro-contrôleur qui diffuse des nano-instructions à tous les processeurs.

L'organisation physique de l'architecture est la suivante (cf. *fig.* 7) :

- Une puce regroupe 16 processeurs.
- Une section regroupe 2 puces de 16 processeurs, la mémoire locale de ces processeurs et une autre puce contenant l'unité de calcul flottant.
- Les puces processeurs (contenant chacune 16 processeurs) sont interconnectées en un hypercube à 12 dimensions (il y a de 1 K à 4 K puces processeurs). Chaque processeur a une adresse codée sur 16 bits et deux processeurs sont directement reliés si leur adresse diffère de 1 bit.

Pour chaque application, l'utilisateur peut dynamiquement définir une géométrie particulière pour l'ensemble des processeurs qu'il utilise. Cette géométrie décrit l'organisation logique de ces processeurs : le nombre de dimensions de la structure et le nombre de processeurs par dimension. Il peut, en outre, s'affranchir de la limite imposée par le nombre de processeurs physiques grâce au mécanisme des processeurs virtuels. La machine à connexions est donc une machine à parallélisme de données. Une même opération est effectuée en parallèle sur une grande masse de données. Ce type de parallélisme est bien adapté au traitement d'image bas niveau puisque les données (l'image) sont de taille importante et que les traitements sont locaux et identiques pour chaque pixel [15].

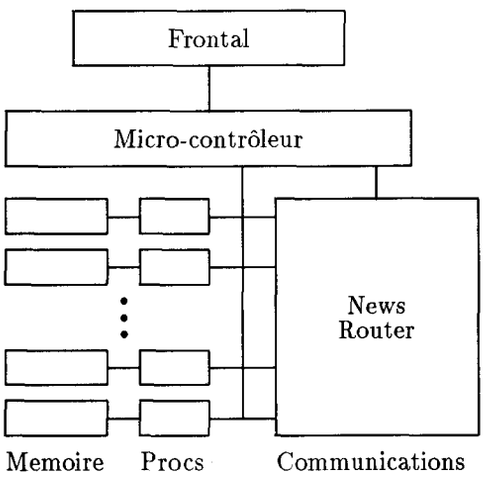


Figure 7. — Architecture de la CM2.

4. Mise en œuvre des algorithmes de relaxation sur CM2

Le GNC (« Graduated Non Convexity ») et le recuit par champs moyens (« Mean Field Annealing ») ont été implémentés sur la machine à connexions. Cette mise en œuvre a été réalisée en *Lisp [26] qui est une extension de Common Lisp pour permettre la programmation parallèle sur la machine à connexions. Dans ce paragraphe, nous présentons d'abord les primitives que nous employons pour le traitement d'image en général, puis nous décrivons la mise en œuvre des deux algorithmes.

4.1. PRIMITIVES DE TRAITEMENT D'IMAGE SUR LA CM2

4.1.1. Entrée-Sortie d'image

Pour effectuer un traitement sur une image il convient d'abord de la charger dans la mémoire de la machine. Dans un premier temps, il faut configurer la machine suivant une géométrie adaptée aux dimensions de l'image (cf. *fig.* 8). En général, la géométrie retenue est une grille à deux dimensions dont la taille est égale à la puissance de

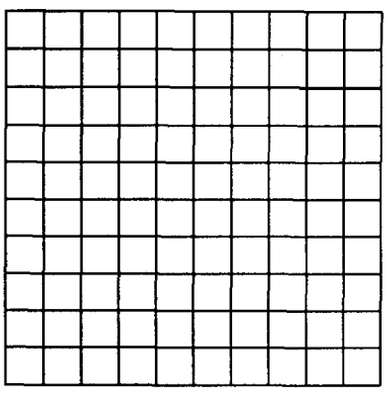


Figure 8. — Géométrie 2D pour une image, 1 pixel par processeur virtuel.

deux immédiatement supérieure au nombre de pixels. On obtient donc un nombre de processeurs virtuels supérieur ou égal au nombre de pixels de l'image. Le fichier image qui a été lu sur le calculateur frontal est un tableau dont chaque élément (pixel) est chargé sur un processeur de la géométrie. Ce transfer utilise un bus global qui relie le frontal à la mémoire des processeurs. La représentation de l'image dans la machine est alors une variable parallèle (pvar en *Lisp) dont chaque élément est un pixel. L'opération d'écriture d'une image depuis sa représentation dans la CM dans un fichier image est l'opération inverse.

Une méthode plus rapide consiste à stocker l'image sous forme d'un fichier parallèle dans le système de fichiers CM sur le « Data Vault » [26] ; la lecture et l'écriture de l'image sont alors plus rapides.

4.1.2. Opérations arithmétiques et logiques sur les variables parallèles

Toutes les opérations arithmétiques et logiques classiques sont disponibles pour les variables parallèles (pvar). Ces opérations prennent en général deux variables parallèles et délivrent en résultat une troisième variable parallèle. Ce sont les opérations de base de la CM qui sont effectuées en parallèle sur tous les processeurs.

Soit la variable parallèle image!! *représentant les intensités en niveau de gris d'une image (de 0 à 255), l'expression *Lisp suivante:*
(<!! image!! (!! 200))
délivre une pvar qui correspond à l'image seuillée avec le seuil 200. La comparaison est effectuée pour chacun des pixels en parallèle.

Figure 9. — Exemple d'opération arithmétique et logique parallèle.

4.1.3. Communications inter-processeurs

Il existe plusieurs types de communications entre les processeurs. La plus générale étant l'envoi en parallèle d'une valeur de chaque processeur vers n'importe quel autre. Par exemple l'expression *Lisp suivante :

(*pset pvar1 pvar2 dest-address-pvar)

où pvar1, pvar2 sont des variables parallèles et dest-address-pvar est également une variable parallèle contenant des adresses de processeurs. L'effet de cette instruction est d'envoyer chaque valeur de pvar1 dans l'élément de la variable parallèle pvar2 dont l'adresse processeur est donnée dans dest-address-pvar. L'ensemble des valeurs de pvar1 subit donc une permutation si toutes les adresses destination sont distinctes. Si certaines adresses destination sont identiques, il y a collision et il est possible d'associer des opérations courantes (+, max, min, ou etc...) de manière à combiner les différentes valeurs destinées à un processeur en une seule valeur. Nous utilisons ce type de communication générale pour dilater ou réduire une image.

Le type de communication que nous utilisons le plus en traitement d'image bas niveau est la communication locale

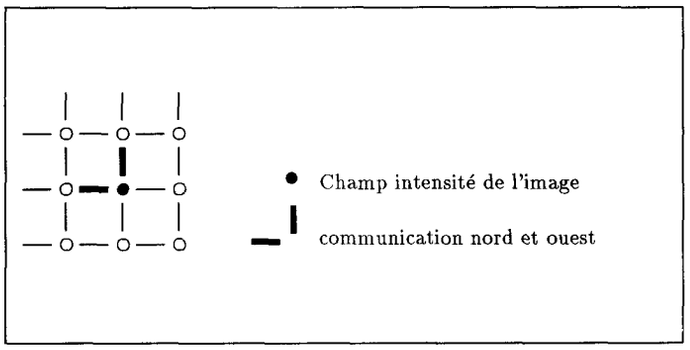


Figure 10. — Exemple de communication NEWS.

et relative à la géométrie choisie. Cette communication appelée NEWS (pour « North, East, West, South ») est très rapide car, dans la plupart des cas, le transfert d'information se fait entre des processeurs situés sur la même puce et par un circuit spécialisé différent du circuit de routage général.

4.1.4. Opérations globales de réduction

Ces opérations consistent à effectuer une opération élémentaire (+, −, max, min, ou, et, ...) successivement sur tous les éléments d'une variable parallèle et à délivrer une seule valeur qui est transmise au frontal par le bus global. Ces opérations sont très efficaces car leur mise en œuvre est parallélisée.

L'expression suivante calcule la moyenne des intensités d'une image
(/ (*sum* image!!) nb-pixels)
image!! *est la pvar qui représente l'image.*
*L'opération *sum est une opération globale de réduction .*

Figure 11. — Exemple d'opération de réduction.

4.2. MISE EN OEUVRE DES ALGORITHMES GNC ET MFA

D'un point de vue mise en œuvre, ces deux algorithmes sont assez semblables puisqu'il s'agit de deux algorithmes de relaxation itératifs. A l'issue de chaque itération, une nouvelle image lissée est calculée et est utilisée à l'itération suivante. La valeur de la fonction énergie est calculée à chaque itération et sert dans le test d'arrêt de la boucle (à la k-ième itération, arrêt si $\Delta E = E_k - E_{k-1} < \varepsilon$). Le calcul de cette valeur nécessite une opération globale de réduction sur toute l'image (une somme globale).

Dans l'algorithme GNC, le calcul du gradient se fait en deux passes successives. On considère l'image comme un échiquier, le calcul du gradient est d'abord effectué sur les cases blanches de cet échiquier, puis, à la seconde passe, sur les cases noires. Ces deux passes sont séquentielles car, à la seconde passe on utilise le résultat du gradient calculé à la première passe. En effet, chaque processeur communique avec ses quatre voisins cardinaux. La minimisation de l'énergie se fait par une méthode de type SOR (« Successive Over-Relaxation » cf. *fig* 4).

Dans l'algorithme MFA on utilise une méthode de descente optimale par gradient conjugué [16], [24] qui nécessite un nombre plus élevé de calcul de l'énergie (cf. *fig.* 5).

La table 1 montre le nombre d'instructions de chaque type pour chacune des itérations. Toutes les instructions sont flottantes sur 32 bits. Dans ce tableau ne figurent pas les instructions de gestion de la pile et de contrôle. Il est intéressant de noter que le rapport entre le nombre d'instructions de communication et le nombre d'instructions flottantes est très faible et que ces communications sont locales.

Table 1. — Nombre d'instructions CM par type pour chaque itération.

	Op.Arith.et Compar.	NEWS	Op.Global
GNC	117	10	1
MFA	123	24	4

L'expression suivante qui représente le calcul d'un gradient:

$$\sum_{i,j}((y_{i,j} - y_{i-1,j})^2 + (y_{i,j} - y_{i,j+1})^2)$$

*peut s'exprimer en *Lisp de la façon suivante:*

```
(*sum (+!!  (expt!! (-!! y!! (news-north!!  y!!))(!! 2))
            (expt!! (-!! y!! (news-east!!   y!!)) (!! 2))))
```

où:

- *y!! est une pvar représentant le champ intensité de l'image,*
- *+!!, −!! et expt!! sont des opérations arithmétiques parallèles,*
- *news-north!! est une macro instruction qui délivre une image dont chaque pixel est la valeur de l'intensité de son voisin nord,*
- *(!! 2) est une opération de diffusion globale qui envoie la valeur 2 à tous les processeurs actifs,*
- **sum est l'opération de réduction qui rend une valeur entière égale à la somme des éléments de la pvar.*

Figure 12. — Exemple de programmation parallèle en *Lisp.

5. Performance des différents algorithmes

Nous avons testé ces algorithmes sur des images fournies par l'ENST dans le cadre du Groupe de Recherche 134, Traitement du Signal et Images (base de données « Segmentation des Images », cf. Annexes n° 2 et 3) ainsi que sur des images provenant de USC (cf. Annexe n° 1) et de l'INRIA (cf. résultats dans les tables).

Dans un premier temps, nous présentons le choix des paramètres effectué pour chaque algorithme. Puis, nous nous intéressons au temps de calcul des différents algorithmes ; nous comparons les algorithmes entre eux, puis nous comparons l'approche parallèle avec l'approche séquentielle [32]. Enfin nous comparons ces algorithmes du point de vue de la qualité des résultats.

5.1. CHOIX DES PARAMÈTRES

Pour la méthode du GNC, les deux paramètres à choisir sont le facteur de lissage λ qui est également un facteur d'échelle (cf. [3]) et le seuil $h = \sqrt{\frac{2\alpha}{\lambda}}$ qui fixe la limite de détection des contours. Le test d'arrêt de l'algorithme est basé sur un seuillage de la variation d'énergie ΔE calculée après chaque itération. La table 2 donne les valeurs des paramètres utilisés pour chaque image (A1 = images contenues dans l'Annexe n° 1... etc).

Pour la méthode du recuit par champs moyens (MFA), le choix de la loi de variation de température est très important. En effet, lorsque les images sont peu contrastées ou fortement bruitées, une variation trop rapide de la température donne des images lissées de mauvaise qualité

Table 2. — Paramètres pour le GNC.

Image	A1	A2	A3
λ	4	3	4
h	16	10	16
ΔE	200	200	150

ainsi qu'une détection de contours moyenne alors que le test de convergence de l'algorithme (basé sur la variation d'énergie ΔE) est correct. Après avoir testé différentes lois de variation de température proposées dans la littérature, nous avons choisi une loi empirique qui donne de bons résultats sur toutes les images testées :

- $\beta = 0.0002$ à l'initialisation.
- $\beta = \beta * 4$ tant que $\beta \leqslant 1$.

Les paramètres à fixer sont le facteur de lissage λ et le coût α à payer pour créer une discontinuité (ce qui revient à choisir h). Pour les images présentées en annexe, ces paramètres sont donnés dans la table 3.

Table 3. — Paramètres pour le MFA.

Image	A1	A2	A3
λ^2	6	6	6
α	180	180	300
ΔE	100	100	100

5.2. COMPARAISON DES TEMPS DE CALCUL

Pour les deux algorithmes présentés dans cet article nous donnons :

- le VPR, c'est-à-dire le nombre de processeurs virtuels par processeurs réels ou le nombre de pixels par processeurs.
- Le temps CM, c'est le temps d'exécution des instructions CM.
- Le temps total.
- Le nombre d'itérations de l'algorithme pour le GNC et le MFA.
- Le temps CM par itération.

Les tables 4, 5 et 6 montrent ces résultats sur trois images de taille différente : une image 128 × 128, une image 512 × 512 et une image 1 024 × 1 024 (cf. [32] pour une présentation de ces images et des résultats obtenus avec le GNC et le MFA).

Il faut noter que les deux algorithmes de relaxation GNC et MFA sont sensiblement équivalents au point de vue temps de calcul, avec toutefois un petit avantage pour le GNC qui comporte moins d'instructions par itération. La figure 13 résume bien ce comportement. On constate en effet que, pour chacun des deux algorithmes, le temps par itération est proportionnel au nombre de pixels par processeur. Toutefois, lorsque le nombre de pixels croît, le rapport temps de calcul par itération/nombre de pixels par processeur a tendance à baisser. Ce phénomène est dû au fait que plus le VPR est important, plus les communications sont rapides puisqu'un pourcentage important de ces communications s'effectue localement à la puce. Cependant, ce phénomène se voit très peu sur la figure 13 du fait du plus grand partage de l'unité de calcul flottant lorsque le VPR augmente.

Table 4. — Image aérienne bruitée 128 × 128.

	VPR	Tps.CM	Tps.Total	Nb.Iter	Tps.CM par It
GNC	2	21s	64.5s	567	0.03s
MFA	2	10.2s	19.13s	174	0.05s

Table 5. — Image d'intérieur non bruitée 512 × 512.

	VPR	Tps.CM	Tps.Total	Nb.Iter	Tps.CM par It
GNC	32	91.5s	97.9s	187	0.49s
MFA	32	166.8s	203s	278	0.60s

Table 6. — Image IGN non bruitée 1 024 × 1 024.

	VPR	Tps.CM	Tps.Total	Nb.Iter	Tps.CM par It
GNC	128	302.13s	320.8s	156	1.93s
MFA	128	947.64s	1048.7s	411	2.3s

Table 7. — Image aérienne 128 × 128 Fortran-Sun4 et *Lisp-CM.

	Fortran-Sun4		*Lisp-CM	
	Tps.Total	Tps.par it.	Tps.Total	Tps.par it.
GNC	15mn.	1.69s	64.5s	0.11s
MFA	25mn.	7.97s	19.13s	0.109s

Table 8. — Image d'intérieur 512 × 512 Fortran-Sun4 et *Lisp-CM.

	Fortran-Sun4		*Lisp-CM	
	Tps.Total	Tps.par it.	Tps.Total	Tps.par it.
GNC	50mn	19.86s	97.9s	0.52s
MFA	8H40mn	1mn52s	203s	0.73s

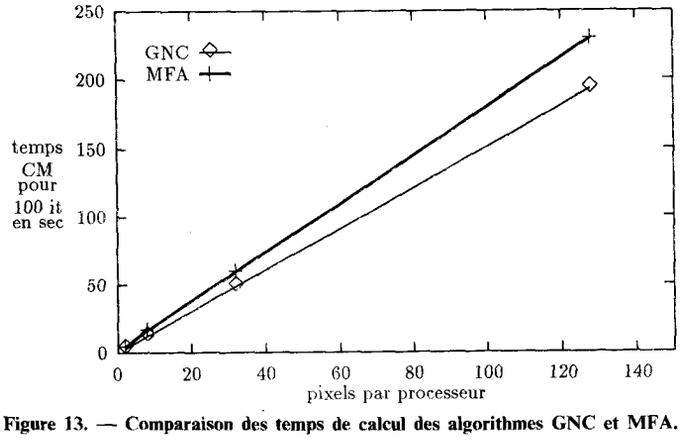


Figure 13. — Comparaison des temps de calcul des algorithmes GNC et MFA.

5.3. COMPARAISON DE LA QUALITÉ DES RÉSULTATS

Avant de commenter les résultats obtenus sur les images, il faut souligner que :

- aucun traitement complémentaire relatif à l'affinage ou la fermeture des contours n'a été appliqué avec les algorithmes déterministes de relaxation GNC et MFA,
- globalement, le GNC et le recuit par champs moyens sans interaction entre les processus de ligne donnent des résultats analogues tant en détection de contours qu'en lissage d'image.

Dand le cas d'images non bruitées (cf. Annexe n° 3), les algorithmes déterministes de relaxation donnent des contours fermés mais présentent des inconvénients :

- pas d'affinage de contours (cf. Annexe n° 3)
- quelques problèmes de détection dans le cas de texture ou de parcellaire (cf. [32]).

En ce qui concerne les images bruitées (cf. Annexe n° 1) ou de mauvaise qualité (cf. Annexe n° 2), les algorithmes déterministes de relaxation donnent des résultats acceptables en détection de contours même lorsqu'il y a un faible rapport signal sur bruit (tests effectués jusqu'à 0 dB). De plus, il n'y a pas de coût supplémentaire à payer pour obtenir une image lissée. Il faut noter que pour des images difficiles telles que celle présentée dans l'Annexe n° 2, la qualité du lissage avec le GNC ou le recuit par champs moyens n'est pas très bonne. Cependant, il peut être utile notamment pour la détection de bâtiments ou de routes. En ce qui concerne la détection de contours, on observe un phénomène de saturation, typique des algorithmes de relaxation (au niveau de piétons et des véhicules dans l'Annexe 2).

6. Conclusion

Dans cet article, nous avons présenté les résultats obtenus en détection de contours et lissage d'image par deux algorithmes déterministes de relaxation basés sur une modélisation Markovienne de l'image. Les principaux avantages d'une telle approche sont :

- la prise en compte des interactions locales au niveau du pixel,
- l'utilisation des processus de ligne pour la détection des contours,
- l'incorporation de contraintes dans l'énergie,

• l'obtention d'une solution déterministe qui peut être implantée à l'aide d'un algorithme parallèle itératif.

D'un point de vue méthodologique, nous sommes partis de la version séquentielle des algorithmes que nous avons transformée en une version parallèle. Dans les deux cas, cette transformation a été assez aisée puisqu'il ne nous a pas été nécessaire de repenser l'algorithme. Nous avons conservé sa structure en transformant chaque séquence d'opérations séquentielles sur l'ensemble des pixels par une opération parallèle équivalente. L'utilisation du langage *Lisp nous a permis d'exprimer ces algorithmes de façon plus concise et plus élégante que leur équivalent séquentiel.

Bien que l'avantage que nous ayons tiré de cette expérience soit essentiellement un gain de temps de calcul important, nous pensons que le modèle de parallélisme de données à grain fin est intéressant et apporte une nouvelle façon d'appréhender des problèmes réels. Si ce modèle est particulièrement bien adapté à la vision bas niveau, il semble prometteur également pour les problèmes de plus haut niveau où des expériences ont déjà été réalisées, comme par exemple en reconnaissance d'objets [28]. Les techniques sont alors différentes, mais la machine dispose de primitives (telles que le « *segmented scan* ») et d'une grande flexibilité de reconfiguration qui permettent d'envisager des solutions à la fois originales et « naturelles ».

Manuscrit reçu le 3 septembre 1990. Version révisée le 23 avril 1991.

REMERCIEMENTS

Les auteurs remercient David Ray, Harris Voorhees et James-Patrick Massar de Thinking Machine Corporation pour leurs conseils. Le premier auteur souhaite également remercier Rama Chellappa, Directeur du « Signal and Image Processing Institute » de USC (LA, USA) pour son accueil. Les algorithmes séquentiels dont les résultats sont présentés dans cet article ont été développés lorsque le premier auteur était en année post-doctorale à USC. Enfin, les auteurs remercient les experts du comité de lecture pour leurs remarques constructives.

ACKNOWLEDGEMENTS

The authors would like to thank David Ray, Harris Voorhees and James-Patrick Massar of Thinking Machine Corporation for many useful discussions. The first author also thanks Rama Chellappa, Director of the Signal and Image Processing Institute of USC (LA, USA) for having received her in his laboratory. The sequential algorithms whose results are presented in this paper have been implemented by the first author during her post-doctoral year at USC. Finally, the authors thanks the reviewers for their useful remarks.

BIBLIOGRAPHIE

[1] J. Besag, « Spatial interaction and the statistical analysis of lattice systems », *Jl of Royal Statistical Society* # 2, pp. 192-236, 1974.

[2] G. Bilbro, R. Mann, T. Miller, W. Snyder, D. Van den Bout et M. White, « Optimization by mean field annealing », *Advances in Neural Information Processing Systems,* Touretzky Ed., Vol. 1, pp. 91-98, 1988.

[3] A. Blake et A. Zisserman, « Visual reconstruction », *MIT Press, Cambridge - MA,* 1987.

[4] C. Campbell, D. Sherington et K. Y. M. Wong, « Statistical mechanics and neural networks », *Neural Computing Architecture, MIT Press, Cambridge - MA,* I. Aleksander Ed., pp. 239-257, 1989.

[5] R. Chellappa, S. Chatterjee, et R. Bagdazian, « Texture Synthesis and compression using Gaussian-Markov random field models », *IEEE Trans. Systems, Man and Cybern.,* Vol. SMC-15, pp. 298-303, mars/avril 1985.

[6] D. Geiger et F. Girosi, « Parallel and deterministic algorithms for MRFs : surface reconstruction and integration », *MIT AI Memo 1114,* juin 1989.

[7] D. Geiger et F. Girosi, « Parallel and deterministic algorithms for MRFs : surface reconstruction and integration », *Proc. ECCV,* Antibes, avril 1990.

[8] S. Geman et D. Geman, « Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images », *IEEE Trans. Pattern Analysis, Machine Intel.,* Vol. PAMI-6, pp ; 721-741, novembre 1984.

[9] S. Geman et C. Graffigne, « Markov random fields image models and their application to computer vision », *Proc. ICM86, Ed. A. M. Gleason, Amer. Math. Soc. - Providence,* 1987.

[10] L. Hérault et J. Niez, « Neural Networks and graph *k*-partitioning », *Complex Systems,* Vol. 3, pp. 531-575, 1989.

[11] W. D. Hillis, « The Connection Machine », *MIT Press, Cambridge - MA,* 1985.

[12] F. C. Jeng et J. M. Woods, « Image estimation by stochastic relaxation in the compound Gaussian case », *Proc. ICASSP,* New-York, avril 1988.

[13] R. Kindermann et J. L. Snell, « Markov random fields and their applications », *Amer. Math. Soc.,* Vol. 1, pp. 1-142, 1980.

[14] S. Kirkpatrick, C. Gelatt et M. Vecchi, « Optimization by simulated annealing », *Science 220,* pp. 671-680, 1983.

[15] J. Little, G. Belloch et T. Cass, « Algorithmic techniques for computer vision on a fine-grain parallel machine », *IEEE Trans. Pattern Analysis, Machine Intel.,* Vol. PAMI-11, pp. 244-257, mars 1989.

[16] D. Luenberger, « Linear and nonlinear programming », *Addison Wesley, 2nd Edition,* 1984.

[17] J. Marroquin, S. Mitter et T. Poggio, « Probabilistic solution of ill-posed problems in computational vision », *Proc. Image Understanding Workshop, Scient. App ; Int. Corp.,* août 1985.

[18] J. Marroquin, « Deterministic Bayesian estimation of Markovian random fields with applications to computational vision », *Proc. ICCV,* London, juin 1987.

[19] G. Parisi, « Statistical Field Theory », *Adisson Wesley,* 1988.

[20] C. Peterson J. Anderson, « A mean field learning algorithm for neural networks », *Complex Systems,* Vol. 1, pp. 995-1019, 1987.

[21] C. Peterson J. Anderson, « Neural networks and NP-complete optimization problems ; a performance study on the graph bisection problem », *Complex Systems,* Vol. 2, pp. 58-59, 1988.

[22] M. Plischke et B. Bergersen, « Equilibrium Statistical Physics », *Prentice Hall, Englewood Cliffs - NJ,* 1989.

[23] T. Simchony et R. Chellappa, « Stochastic and deterministic algorithms for MAP texture segmentation », *Proc. ICASSP, New-York,* avril 1988.

[24] T. Simchon, R. Chellappa et Z. Lichtenstein, « Pyramid implementation of optimal step conjugate search algorithms for some low level vision problems », *Proc. Conf. on Computer Vision,* Miami Beach, décembre 1988.

[25] T. simchony, R. Chellappa et Z. Lichtenstein, « The Graduated Non Convexity algorithm for image estimation using Compound Gauss-Markov Field models », *Proc. ICASSP,* Glasgow, mai 1989.

[26] Thinking Machine Corporation, « Connection Machine, Model CM2 Technical Summary », *TMC, Cambridge - MA,* mai 1989.

[27] L. W. Tucker et G. G. Robertson, « Architecture and applications of the Connection Machine », *IEEE Computer,* pp. 26-38, août 1988.

[28] H. Vorrhees, D. Fritzsche et L. Tucker, « Exploiting data parallelism in Vision on the Connection Machine system », *Proc. 10th ICPR,* Atlantic City, juin 1990.

[29] A. L. Yuille, D. Geiger et H. Bulthoff, « Stereo integration, mean field theory and psychophysics », *Proc. ECCV,* Antibes, avril 1990.

[30] J. Zerubia et R. Chellappa, « Mean field approximation using Compound Gauss-Markov Random Field for edge detection and image restoration », *Proc. ICASSP, Albuquerque,* avril 1990.

[31] J. Zerubia et R. Chellappa, « Mean field annealing for edge detection and image restoration », *Proc. EUSIPCO,* Barcelona, septembre 1990.

[32] J. Zerubia et F. Ployette, « Détection de contours et restauration d'image par des algorithmes déterministes de relaxation. Mise en œuvre sur la machine à connexions CM2 », *rapport de recherche INRIA no 1291,* octobre 1990.

A Annexe 1

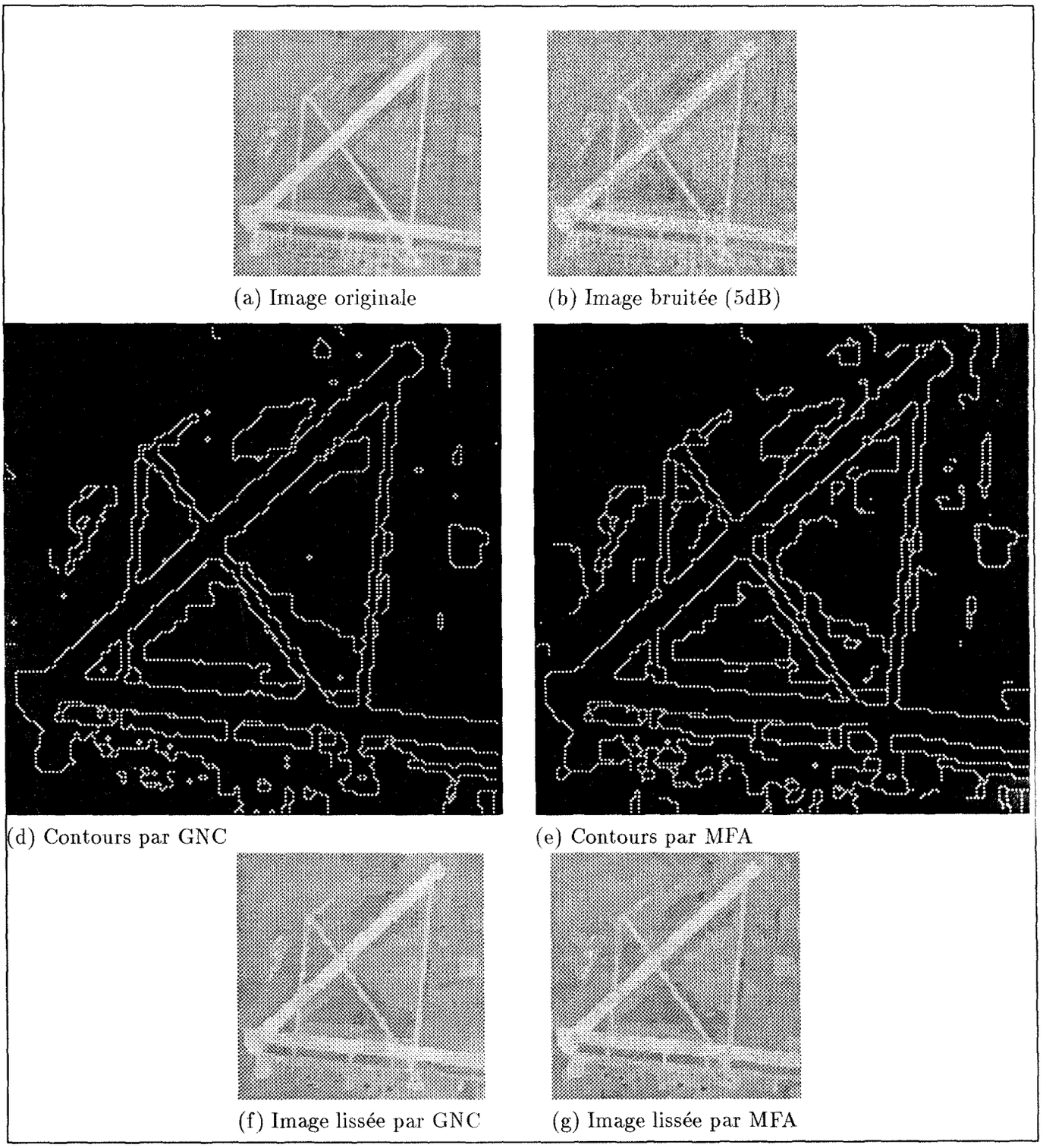


Figure 14. — Image aérienne 128 x 128.

B Annexe 2

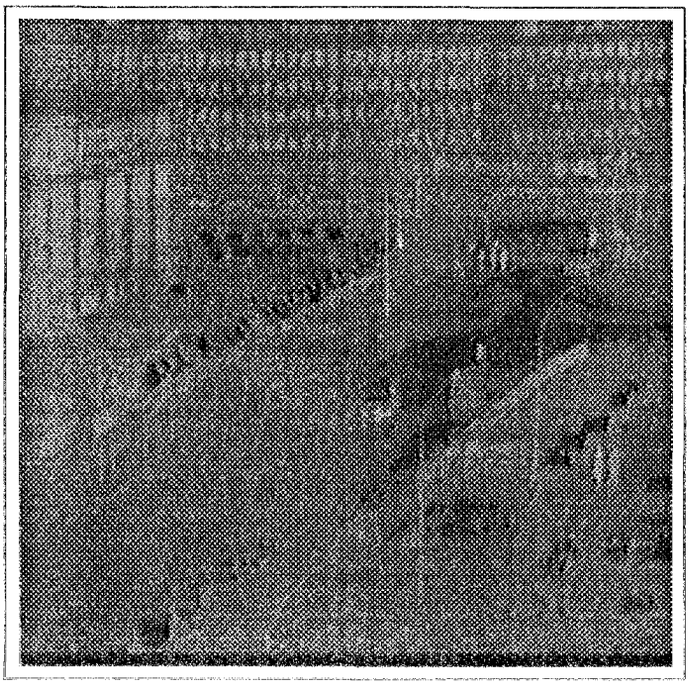
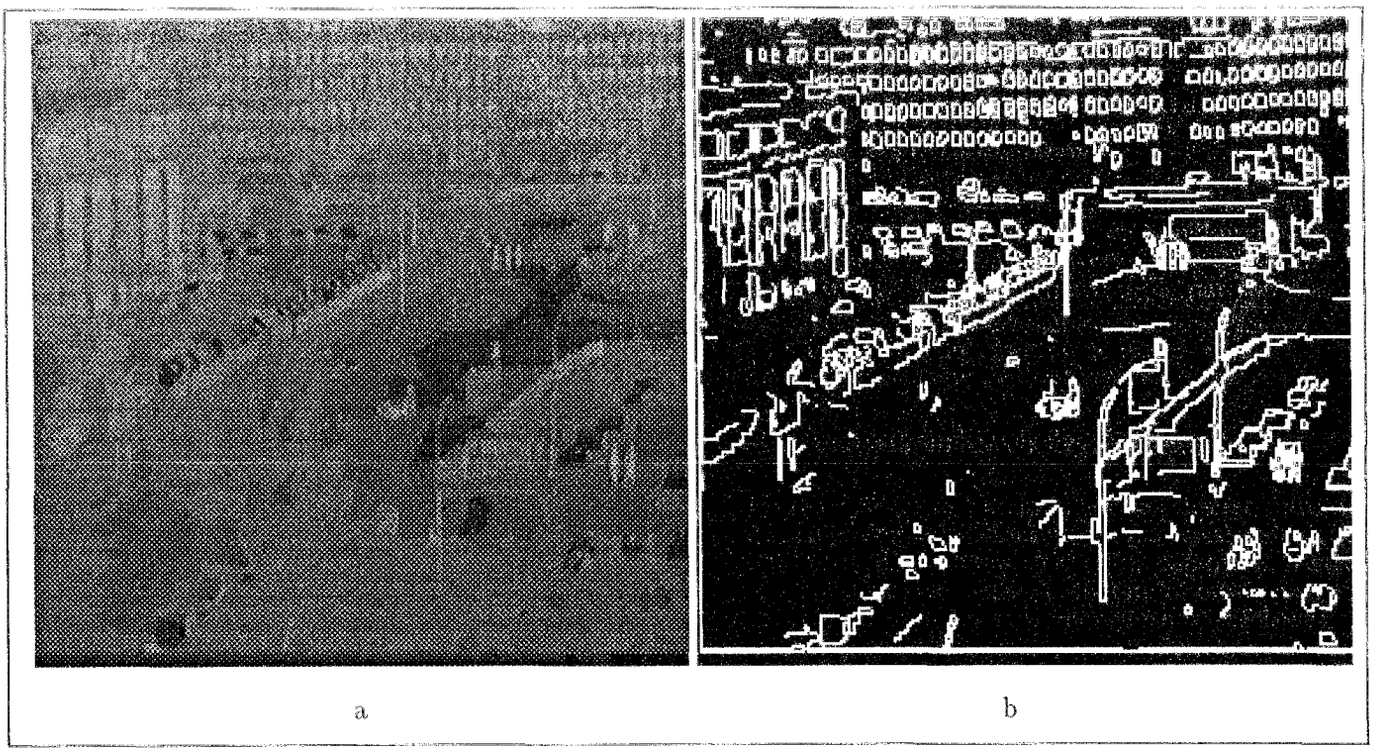


Figure 15. — Image Carrefour (infra-rouge) 256 x 256 : Image originale.



a b

Figure 16. — Algorithme GNC, Image carrefour 256 x 256 : a) Image lissée, b) Image des contours.

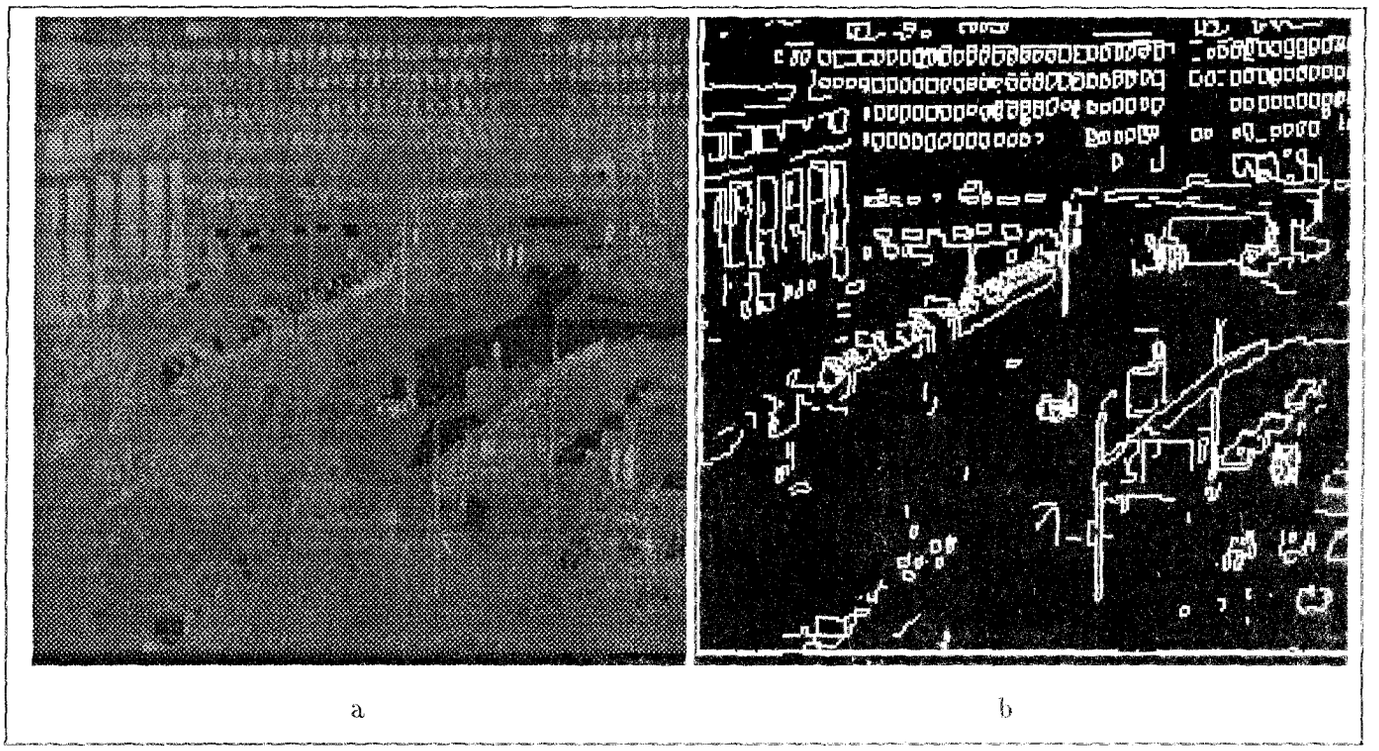


Figure 17. — Algorithme MFA, Image carrefour 256 x 256 : a) Image lissée, b) Image des contours.

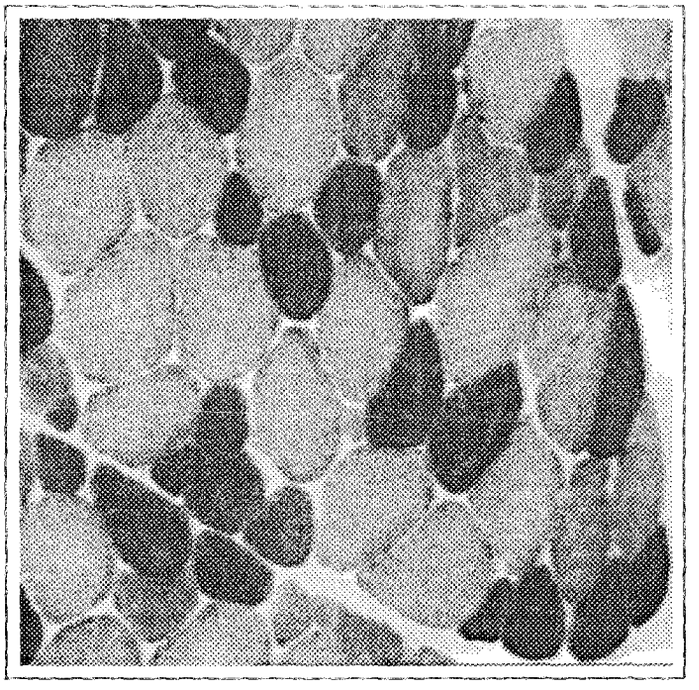


Figure 18. — Image de muscle 256 x 256 : Image originale.

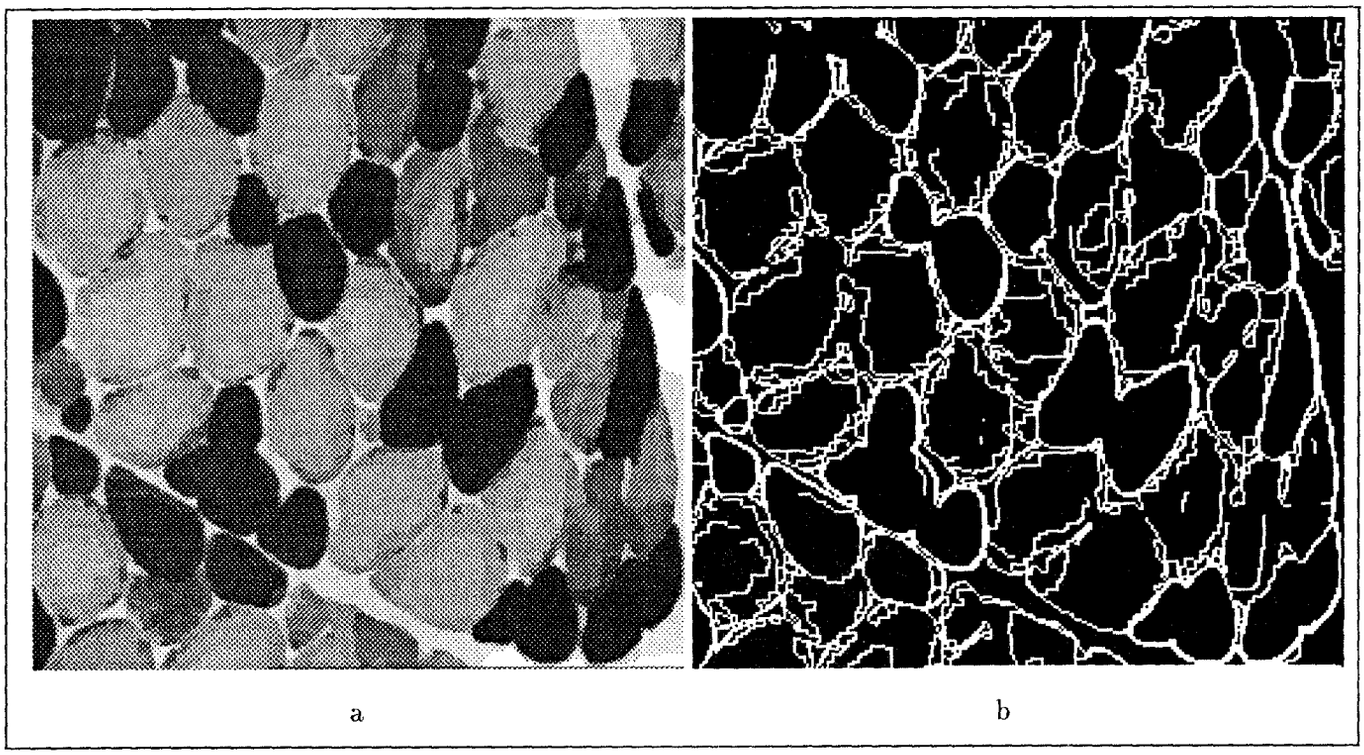


Figure 19. — Algorithme GNC, Image de muscle 256 x 256 : a) Image lissée, b) Image des contours.

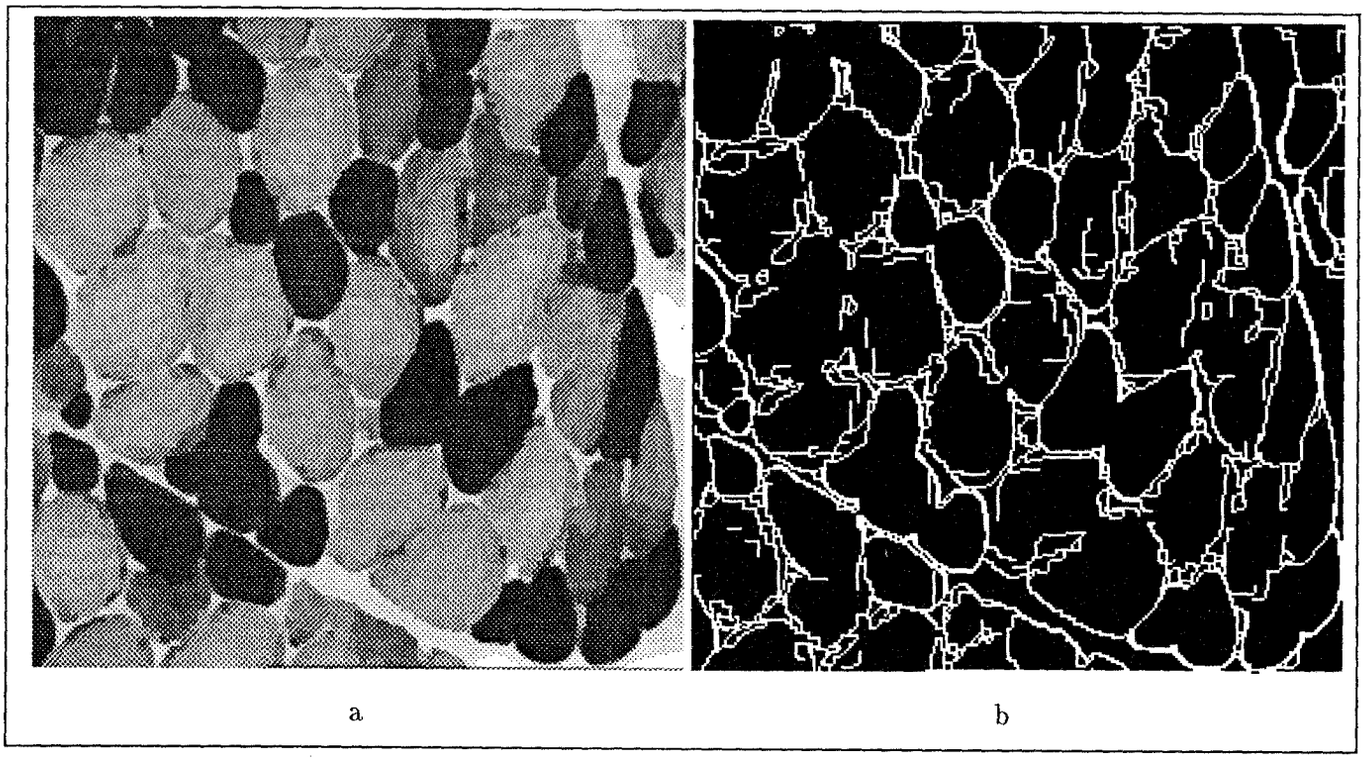


Figure 20. — Algorithme MFA, Image de muscle 256 x 256 : a) Image lissée, b) Image des contours.