Simulation de processus objets : Etude de faisabilité pour une application à la segmentation d'image

Mikael Imberty, Xavier Descombes

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Simulation de processus objets : Etude de faisabilité pour une application à la segmentation d'image

Mikael Imberty, Xavier Descombes

N° 3881

Février 2000

THÈME 3

ISSN 0249-6399 ISRN INRIA/RR--3881--FR

INRIA
SOPHIA ANTIPOLIS

Simulation de processus objets : Etude de faisabilité pour une application à la segmentation d'image

Mikael Imberty, Xavier Descombes

Thème 3 — Interaction homme-machine,
images, données, connaissances
Projet ARIANA

Rapport de recherche n° 3881 — Février 2000 — 50 pages

Résumé : Dans cette étude, nous comparons l'efficacité de deux techniques de simulation par chaînes de Markov (MCMC) de processus aléatoires sur des ensembles d'objets géométriques : l'algorithme de naissance-mort et celui de Metropolis-Hastings-Green. Les comparaisons sont effectuées sur différents modèles de processus objets de type attractif présentant un intérêt en traitement d'image. Nous appliquons ensuite ces méthodes de simulation à la segmentation d'image. Pour cela, nous nous plaçons dans le cadre bayésien : nous définissons donc un modèle a priori attractif simple sur des objets rectangulaires, ainsi qu'un terme d'attache aux données garantissant l'adéquation des objets à l'image. Nous utilisons ensuite un recuit simulé pour extraire les différentes zones de l'image. Des tests sont effectués sur des images synthétiques.

Mots-clés : Processus Markov Objets, Méthodes MCMC, Segmentation d'image

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles, B.P. 93, 06902 Sophia Antipolis Cedex (France)
Téléphone : 04 92 38 77 77 - International : +33 4 92 38 77 77 — Fax : 04 92 38 77 65 - International : +33 4 92 38 77 65

Object Processes simulation : Study for image segmentation purpose

Abstract: In this study, we compare the efficiency of two algorithms using Monte Carlo Markov chains methods in order to simulate random processes of geometrical objects sets : the algorithm of birth and death and the dynamics of Metropolis-Hastings-Green. The comparisons are carried out on various object models for clustered patterns, which could be of interest in image processing. Then we apply these methods of simulation to image segmentation, using the bayesian approach : thus we define a simple prior model on rectangular objects, as well as a posterior probability guaranteeing the adequacy of the objects to the data. We finally use a stochastic annealing to extract the various zones of the image. Some tests are performed on synthetic data

Key-words: Object Markov Process, MCMC methods, image segmentation

Table des matières

1 Introduction

Dans le cadre du traitement probabiliste des images, l'approche la plus classique consiste à modéliser une image numérique comme la réalisation d'un processus aléatoire : chaque point, ou pixel, de l'image est ainsi représenté par une variable aléatoire dont la valeur correspond à son intensité. La modélisation par *champs de Markov* consiste par exemple à considérer que chaque pixel suit une loi de probabilité dont la loi conditionnelle au reste de l'image ne dépend que des pixels voisins. Cette approche est couramment utilisée pour les problèmes de restauration ou de segmentation d'image - voir notamment [4, 9].

La segmentation d'une image consiste à attribuer à chaque pixel un label indiquant à quelle zone ou quel objet ce pixel appartient - cf par exemple [7]. Pour les images satellitaires et aériennes, on souhaite par exemple distinguer les zones urbaines des zones agricoles, extraire les routes ou différencier des champs de cultures différentes. Les approches pixéliques possèdent deux caractéristiques qui limitent leurs performances. En premier lieu, la vraisemblance (c'est-à-dire l'attache aux données) est calculée à une échelle pixélique. Dès lors, la robustesse pour de forts niveaux de bruit est limitée malgré le pouvoir régularisateur du modèle a priori. Une attache aux données calculée à l'échelle d'un objet semble alors préférable. En second lieu, il est difficile d'injecter des contraintes de type géométrique à la solution avec des interactions au niveau pixélique. Le but de la segmentation est de partitionner l'image en un certain nombre d'objets. Plutôt que de considérer l'image d'un point de vue pixélique, il semble donc intéressant de la modéliser comme un ensemble d'objets correspondants aux différentes zones que l'on veut récupérer. C'est cette nouvelle approche, introduite dans [2], que nous utilisons dans cette étude.

Comme dans le cas pixélique, la répartition des objets dans une image est traitée de façon stochastique : les objets, ou les zones de l'image, sont ainsi la réalisation d'un processus aléatoire, dit *processus objet*, ou *processus ponctuel marqué*. La simulation de tels processus est essentielle, car elle permet de tester la validité des modèles utilisés pour la représentation des images. De plus, la résolution de la plupart des problèmes inverses en imagerie - tels que segmentation, restauration, extraction de contours - prennent dans le cadre de

l'approche bayésienne la forme de la maximisation d'une loi de probabilité. La résolution de ce problème peut se faire par l'algorithme de *recuit simulé*, qui nécessite de pouvoir simuler la loi considérée.

Dans une première partie, nous donnons différents exemples de processus objets utilisés dans la suite du travail. Nous comparons ensuite sur différents exemples deux algorithmes de simulation de tels processus : l'algorithme de *Metropolis-Hastings-Green* (cf [12]) et l'algorithme de *naissance-mort* (cf [16]). La troisième partie teste la validité de cette approche en appliquant les algorithmes à la segmentation d'images simples.

2 Processus objets

Ce paragraphe introduit le vocabulaire et les définitions utilisés par la suite. Ces notations sont essentiellement reprises de l'article [13], puisque celui-ci s'intéresse au problème très similaire de la reconnaissance d'objets. Nous donnons ensuite quelques exemples de processus objets, qui seront étudiés numériquement dans les parties suivantes.

2.1 Notations

On s'intéresse dans cet article à la simulation de distributions d'objets dans un espace donné T, appelé *espace image*. Dans le cadre qui nous intéresse, cet espace est une image numérique, donc une grille finie de pixels. On suppose que les objets sont représentables par un nombre fini de paramètres, qui indiquent par exemple la taille et la position des objets. On note U l'espace de ces paramètres, appelé *espace objet*, et pour chaque objet $u \in U$ on note $R(u) \subseteq T$ son support dans l'image. Dans le cas continu, on suppose que U est un borélien de $\mathbb{R}^d$, muni de la tribu borélienne $\mathcal{A}$ et de la mesure de Lebesgue λ. On suppose de plus que $0 < \lambda(U) < +\infty$. Dans le cas discret, U est un ensemble fini de points de $\mathbb{R}^d$, que l'on munit de la tribu $\mathcal{A} = \mathcal{P}(U)$ et de la mesure de comptage λ.

Une configuration d'objets x est alors un ensemble non ordonné d'objets :

$$x = \{u_1,...,u_n\},\ u_i \in U$$

Si on note Ω_n l'ensemble des configurations de n objets, l'espace total des configurations est $\Omega = \bigcup_{i=0}^{+\infty} \Omega_n$. On peut munir cet espace d'une σ-algèbre $\mathcal{B}$ et d'une mesure μ. Les définitions sont données dans [1], et nous les rappelons ici brièvement.

Notons $\omega_n : U^n \to \Omega_n$ la projection de n-uplets ordonnés dans les configurations à n objets, et $\mathcal{A}_n$ la tribu produit sur U^n. Alors on définit $\mathcal{B}$ comme étant la tribu engendrée par $\omega_0(\mathcal{A}_0)$, $\omega_1(\mathcal{A}_1)$, $\omega_2(\mathcal{A}_2)$, ... et la mesure μ par :

$$\mu(B) = e^{-\lambda(U)} \left\{ 1[\emptyset \in B] + \sum_{n=1}^{\infty} \frac{\mu_n(B)}{n!} \right\} \tag{1}$$

$$\text{où} \qquad \mu_n(B) = \int ... \int 1[u_1,...,u_n \in B] \; \lambda(du_1)...\lambda(du_n) \tag{2}$$

pour $B \in \mathcal{B}$. La mesure μ est en réalité la distribution d'un processus ponctuel de Poisson sur U de mesure d'intensité λ, ce qui signifie que l'espérance du nombre de points d'un tel processus est $\lambda(U)$.

2.2 Processus ponctuels, processus objets

Un processus ponctuel X est donc une application mesurable définie sur un certain espace de probabilité et à valeurs dans $(\Omega,\mathcal{B})$. On note par la suite f la densité de ce processus par rapport à la mesure μ.

Dans ce cadre général on peut distinguer deux types de processus :

- les processus ponctuels spatiaux, pour lesquels $U = T$, c'est-à-dire pour lesquels les objets sont réduits à des points de l'espace image ;
- les processus objets, appelés également processus ponctuels marqués, pour lesquels $U = T \times M$ où M est l'espace des marques attachées aux objets, c'est-à-dire des paramètres autres que les coordonnées spatiales.

C'est cette dernière catégorie qui nous intéresse plus particulièrement, puisque nous nous intéressons à des processus sur des objets tels que des segments, des disques, des ellipses ou d'autres objets géométriques. Cependant, le cadre

décrit ci-dessus englobe également les processus ponctuels classiques, et les algorithmes décrits dans le paragraphe 4.2 s'appliquent en particulier aux deux types de processus.

3 Quelques modèles

Les densités des processus objets couramment utilisés sont de la forme :

$$f(x) \propto \alpha(n(x))\, g(x) \tag{3}$$

où $n(x)$ est le nombre d'objets dans la configuration x, $\alpha()$ et $g()$ deux fonctions mesurables positives. Dans le cas d'un processus de Poisson, $\alpha(n) = \beta^n$ et $g(x) = 1$. La fonction $\alpha()$ représente donc schématiquement l'intensité du processus, i.e. le nombre moyens d'objets de x, et $g()$ modélise les interactions entre objets.

3.1 Processus avec interactions par paires d'objets

Un premier type de processus objets est celui où la fonction g peut se mettre sous la forme :

$$g(x) = \prod_{\{u,v\} \subset x} \phi(u,v) \tag{4}$$

où $\phi : \Omega_2 \rightarrow [0,\infty)$. On suppose de plus que la densité f est héréditaire, i.e. :

$$f(x) > 0 \;\Rightarrow\; f(y) > 0 \quad \forall\, y \subseteq x \tag{5}$$

ce qui revient à dire ici que $\alpha(n(x)) > 0 \;\Rightarrow\; \alpha(n(y)) > 0$. Dans ce type de processus, les interactions n'existent qu'entre paires d'objets, d'où le nom - *pairwise interaction point processes* en anglais.

L'exemple le plus simple est sans doute le processus de Strauss, pour lequel :

$$\phi(u,v) = \gamma^{1[u \sim v]} \tag{6}$$

où $\sim$ est une relation symétrique sur U. Cette relation est a priori quelconque, mais on la définit généralement par

$$u \sim v \quad \Leftrightarrow \quad d(u,v) < R \tag{7}$$

où $d(.,.)$ est la distance entre deux objets. La densité f s'exprime alors par :

$$f(x) \propto \beta^{n(x)} \gamma^{p(x)} \tag{8}$$

où $p(x)$ désigne le nombre de paires d'objets qui sont à une distance inférieure à R l'un de l'autre.

Un autre exemple consiste à définir la relation par $u \sim v \Leftrightarrow R(u) \cap R(v) \neq \emptyset$: deux objets interagissent si leurs supports se coupent.

3.2 Processus de Markov

Le processus de Strauss possède en fait une particularité par rapport à la définition générale des processus avec interaction par paires d'objets : dans le processus de Strauss, deux objets u et v n'intéragissent que s'ils vérifient une relation de voisinage $u \sim v$. Ripley et Kelly ont introduit dans [18] les *processus ponctuels de Markov* qui sont une généralisation de cette idée. Leur définition équivaut en effet à dire qu'un processus X est de Markov si et seulement si sa densité est de la forme :

$$f(x) = \prod_{y \subset x} \Phi(y) \tag{9}$$

où Φ est une fonction positive telle que $\Phi(y) = 1$ s'il existe u, v dans y tels que $u \not\sim v$. Autrement dit, la densité f dépend d'une fonction d'interaction Φ qui agit sur les ensembles d'objets voisins. Cette définition est très similaire à celle des champs de Markov dans le cas discret.

3.3 Processus Markov objets aux plus proches voisins

Nous donnons maintenant la définition plus générale des *processus Markov objets aux plus proches voisins - nearest-neighbour Markov object process* en

anglais - introduite par Baddeley et Møller dans [1]. Pour ce type de processus, les objets n'intéragissent également qu'avec leurs voisins, mais la relation de voisinage peut dépendre de la configuration x.

On suppose donc que pour tout $x \in \Omega$ il existe une relation symétrique $\sim_x$ définie sur l'ensemble x. Deux points u et v tels que $u \sim_x v$ sont alors dits *x-voisins* et on appelle *x-voisinage* d'un sous-ensemble $y \subset x$ l'ensemble :

$$N(y \mid x) = \{u \in x \;/\; \exists\, v \in y \quad u \sim_x v\} \tag{10}$$

De plus, un sous-ensemble $y \subset x$ est une *clique* de x si $\forall\, u,v \in y \quad u \sim_x v$. Définissons enfin, pour un processus objets X de densité f, le rapport

$$\lambda^*(x,u) = \begin{cases} \frac{f(x \cup \{u\})}{f(x)} & \text{si } f(x) > 0 \\ 0 & \text{sinon} \end{cases} \tag{11}$$

est appelé *densité de Papangelou.*

Dans ce contexte, un processus Markov objets aux plus proches voisins est un processus objets X sur U dont la densité f vérifie :

1. $f(x) > 0 \quad \Rightarrow \quad f(y) > 0 \quad \forall\, y \subset x$
2. $\forall\, x \in \Omega$ tel que $f(x) > 0$ et $\forall\, u \notin x$, le rapport $\lambda^*(x,u)$ ne dépend que de u, $N(\{u\} \mid x \cup \{u\})$ et des restrictions de $\sim_x$ et $\sim_{x \cup \{u\}}$ à $N(\{u\} \mid x \cup \{u\})$.

En général, la relation $\sim_x$ est définie par l'intermédiaire d'une relation symétrique auxiliaire $\sim$ de telle sorte que

$$u \sim_x v \quad \Leftrightarrow \quad \exists\, u_1,u_2,...,u_n \in x \quad u = u_1 \sim u_2 \sim ... \sim u_m = v \tag{12}$$

En d'autres termes, $u \sim_x v$ si u et v sont dans la même composante connexe pour la relation $\sim$. Dans ce dernier cas, un résultat établi dans [3] montre que la définition d'un processus Markov objets aux plus proches voisins est équivalente au fait que la densité du processus soit de la forme :

$$f(x) \propto \prod_{y \in C(x)} \Phi(y) \tag{13}$$

où $C(x)$ est l'ensemble des cliques maximales de la configuration x.

Exemples Deux exemples de tels processus objets sont importants pour la modélisation de phénomènes attractifs, et seront étudiés numériquement dans la suite.

Le premier processus, nommé *continuum random-cluster model* en anglais, a pour fonction d'interaction $\Phi(y) = \beta^{n(y)}/\gamma$, soit la densité

$$f(x) \propto \beta^{n(x)}\gamma^{-c(x)} \tag{14}$$

où $c(x)$ est le nombre de cliques maximales dans x pour la relation $\sim_x$ (ou de façon équivalente le nombre de composantes connexes dans x pour la relation $\sim$). Ici, la relation $\sim$ est a priori quelconque ; il peut s'agir d'une relation de voisinage spatial ou de recouvrement des objets par exemple.

Le second processus, nommé *area-interaction model*, est défini par :

$$\Phi(y) = \beta^{n(y)}\gamma^{-\lambda(R_y)}$$

avec $R_y = \bigcup_{u\in y} R(u)$. La relation $\sim$ est dans ce cas définie par $u \sim v \Leftrightarrow R(u) \cap R(v) \neq \emptyset$. Sa densité est donc

$$f(x) \propto \beta^{n(x)}\gamma^{-\lambda(R_x)} \tag{15}$$

Il faut noter que ce processus est également un processus de Markov classique au sens de Ripley et Kelly.

Ces deux processus sont étudiés dans [8], dans le cas particulier où les objets sont des disques de rayon R fixé. Nous en donnerons des simulations dans le paragraphe 5.

3.4 Stabilité

Nous avons donné ci-dessus des exemples de densités pour la mesure μ sur l'espace des configurations d'objets sans nous soucier de l'intégrabilité de ces densités dans le cas continu. Geyer [11] donne deux conditions de *stabilité* qui impliquent toutes deux l'intégrabilité, et qui de plus sont utiles pour les preuves de convergence des algorithmes de simulation. Même si les simulations se font dans le cas discret, des problèmes numériques sont susceptibles d'être observés

si ces conditions ne sont pas vérifiées : c'est le cas notamment du processus de Strauss pour $\gamma > 1$ (voir paragraphe 5.2).

Ces deux conditions sont :

- Condition 1 : *Un processus ponctuel avec une densité non-normalisée par rapport à la mesure μ est stable dans le sens de Ruelle s'il existe $M \in [1,\infty)$ tel que :*

$$f(x) \leq M^{n(x)}, \quad \forall x \in \Omega \tag{16}$$

- Condition 2 : *Un processus ponctuel avec une densité non-normalisée par rapport à la mesure μ est stable s'il existe $M \in R$ tel que :*

$$f(x \cup u) \leq M f(x), \quad \forall x \in \Omega \quad \forall u \in U \tag{17}$$

  i.e., si la densité de Papangelou est bornée.

Cette dernière condition est la plus simple à vérifier, même si elle est plus contraignante que la première.

4 Méthodes MCMC

La difficulté des simulations de processus objets vient de ce que l'on ne connaît pas en général la constante de normalisation de la densité du processus. Comme l'espace des objets est très grand, l'estimation de cette constante n'est pas envisageable.

Les méthodes les plus fréquemment employées pour obtenir des réalisations de tels processus sont alors de type Monte Carlo, où l'on simule une chaîne de Markov convergente vers la distribution recherchée - ces méthodes sont dites MCMC (“ Markov Chain Monte Carlo” en anglais). On recherche donc des chaînes telles que pour tout état initial, la distribution de la chaîne converge vers la distribution π :

$$\lim_{t \to +\infty} \| P^t(x,A) - \pi(A) \| = 0 \qquad \forall A \in \mathcal{B}$$

où $P^t(x,A) = P(X_t \in A \mid X_0 = x)$.

Avant de présenter les algorithmes de simulation, nous rappelons brièvement quelques définitions et résultats sur la convergence des chaînes de Markov.

4.1 Chaînes de Markov et convergence

Dans la suite, X_t désigne une chaîne de Markov sur l'espace $(\Omega,\mathcal{B},\mu)$, dont le noyau de transition est $P(x,A)$ pour $x \in \Omega$ et $A \in \mathcal{B}$.

Invariance : Une loi π est invariante pour la chaîne de Markov si :

$$\pi(A) = \int P(x,A)\,\pi(dx) \qquad \forall A \in \mathcal{B} \tag{18}$$

Cette condition est nécessaire pour obtenir la convergence de la chaîne vers π.

Réversibilité : La chaîne est réversible pour π si le noyau de transition vérifie :

$$\int_B P(x,A)\pi(dx) = \int_A P(x,B)\pi(dx) \qquad \forall A,B \in \mathcal{B} \tag{19}$$

Cette condition implique l'invariance pour π, et signifie que sous la distribution stationnaire π la probabilité de passer de A à B est la même que de passer de B à A. La plupart des algorithmes de simulation sont en réalité construits pour produire des chaînes de Markov réversibles.

Irréductibilité : La chaîne est dite π-irréductible si $\forall\, x \in \Omega$ et $\forall\, A \in \mathcal{B}$:

$$\pi(A) > 0 \quad \Rightarrow \quad \exists\, t \quad P^t(x,A) > 0 \tag{20}$$

Cela signifie que la chaîne a une probabilité non nulle d'atteindre en temps fini tout ensemble π-probable. Cette condition est clairement nécessaire pour que la chaîne converge en distribution vers π avec n'importe quelle condition initiale.

Apériodicité : L'apériodicité assure que les déplacements entre états n'ont pas trop de contraintes déterministes. Formellement, la chaîne est apériodique s'il n'existe pas d'ensembles $A_0,...,A_{d-1} \in \mathcal{B}$ avec $d \geq 2$ tels que :

$$P(x,A_{i+1\,[d]}) = 1 \quad \forall\, x \in A_i, \quad i = 0,...,d-1 \tag{21}$$

Une condition suffisante (mais non nécessaire) pour qu'il y ait apériodicité est :

$$\forall\, x \in \Omega \quad P(x,x) > 0 \tag{22}$$

Dans ce cas, la chaîne est dite fortement apériodique. Dans le cas discret, il suffit de vérifier cette propriété pour un seul $x \in \Omega$.

Lorsque la chaîne est π-irréductible et apériodique, alors elle converge en distribution vers π pour presque toute condition initiale :

$$\parallel P^t(x,.) - \pi(.) \parallel \to 0 \quad \text{pour presque tout } x \tag{23}$$

Récurrence au sens de Harris : Cette notion permet d'assurer une convergence pour toute condition initiale. C'est une notion plus forte que l'irréductibilité, qui assure que $\forall\, x \in \Omega$ et $\forall\, A \in \mathcal{B}$:

$$\pi(A) > 0 \quad \Rightarrow \quad P(\exists\, t\; X_t \in A \,|\, X_0 = x) = 1 \tag{24}$$

Pour vérifier que les algorithmes MCMC fonctionnent, on vérifie donc en général que la chaîne de Markov produite admet la loi π comme probabilité invariante, qu'elle est apériodique et π-irréductible, c'est-à-dire qu'elle est ergodique.

4.2 Algorithmes

Nous décrivons maintenant les deux algorithmes utilisés pour simuler des processus objets. Le premier, dit algorithme de *naissance-mort* est propre aux processus ponctuels, tandis que le second, l'algorithme de *Metropolis-Hastings-Green* est plus général et peut être utilisé pour une large catégorie de processus

aléatoires.

Comme nous l'avons souligné plus haut, les processus que l'on souhaite simuler ne sont connus que par leur densité non normalisée. La construction de la chaîne de Markov ne doit donc pas faire intervenir la constante de normalisation.

4.2.1 Naissance-mort

L'algorithme de *naissance-mort* consiste à n'autoriser à chaque étape que deux types de transitions : l'ajout ou la suppression d'un objet à la configuration. La distribution utilisée pour le choix de l'objet à ajouter ou supprimer dépend explicitement de la loi π, ceci afin d'assurer la convergence de la chaîne vers la bonne loi.

Plus précisément, on définit sur l'espace U une mesure de naissance $B(x,.)$ qui dépend de la configuration x, et pour chaque objet u de la configuration un taux de mort $d(x,u)$. On calcule ensuite les taux de naissance et mort globaux :

$$D(x) = \sum_{u_i \in x} d(x,u_i) \tag{25}$$

$$B(x) = B(x,U) \tag{26}$$

L'algorithme de naissance-mort se décrit alors de la façon suivante :

1. étant donnée la configuration x_t à l'étape t, la modification effectuée est l'ajout d'un point avec une probabilité $B(x_t)/(D(x_t)+B(x_t))$ et la suppression d'un point de x_t avec la probabilité $D(x_t)/(D(x_t)+B(x_t))$
2. si la modification est une naissance, on choisit alors un point u de U selon la distribution $B(x_t,.)/B(x_t)$ et on pose $x_{t+1} = x_t \cup \{u\}$
3. si la modification est une mort, on choisit le point v de x_t avec la probabilité $d(x_t,v)/D(x_t)$ et on pose $x_{t+1} = x_t \setminus \{v\}$

On se place généralement dans le cas où la mesure $B(x,.)$ possède une densité pour la mesure μ, densité que l'on note $b(x,.)$.

Pour assurer la convergence de la chaîne vers la loi π, il faut bien sûr utiliser les taux de naissance-mort $B(x,.)$ et $d(x,.)$ adéquats. Dans le cas où la densité f est héréditaire (cf équation 5) et lorsque la condition de stabilité (cf 17) est vérifiée, un résultat démontré dans [13] assure que pour les taux de naissance-mort suivants :

$$b(x,u) = \begin{cases} \left(\frac{f(x\cup\{u\})}{f(x)}\right)^k & \text{si } f(x) > 0 \\ 0 & \text{si } f(x) = 0 \end{cases} \tag{27}$$

$$d(x,u) = \begin{cases} \left(\frac{f(x\setminus\{u\})}{f(x)}\right)^{1-k} & \text{si } f(x) > 0 \\ 1/n(x) & \text{si } f(x) = 0 \end{cases} \tag{28}$$

où $k \in [0,1]$, la chaîne de Markov produite converge en loi vers π pour toute condition initiale. De plus, le calcul des taux est aisé lorsque la densité de Papangelou est connue.

L'inconvénient de cet algorithme est que le calcul des taux de naissance-mort doit être effectué à chaque itération, ce qui peut être très coûteux lorsque l'espace U est grand. Pour supprimer le calcul des taux de morts, on utilise souvent le cas $k = 1$: les objets à supprimer sont tirés de façon uniforme sur x, de telle sorte que que $D(x) = n(x)$. Le cas $k = 0$ pourrait sembler intéressant dans la mesure où il évite le calcul des taux de naissance, qui est en général bien plus coûteux que celui des taux de mort lorsque l'espace U est grand. Cependant, ce cas n'est pas efficace car l'algorithme a tendance à rajouter beaucoup d'objets pour ensuite les effacer.

On peut limiter le nombre de calcul à chaque itération puisqu'en général certains taux de naissance ne sont pas modifiés par l'ajout ou la suppression d'un objet. Cependant, les taux de naissances qui doivent être mis à jour à chaque étape varient selon le modèle, et peuvent être également en très grand nombre. De plus, la place mémoire nécessaire pour l'algorithme est proportionnelle à la taille de l'espace U.

4.2.2 Metropolis-Hastings

Ce deuxième algorithme est beaucoup plus général que le premier, et il peut s'appliquer à d'autres processus que les processus ponctuels. Il consiste à proposer à chaque étape des transitions de l'état courant x_t vers un état y a priori quelconque. Les "mouvements" autorisés ne sont donc pas nécessairement une naissance ou une mort, et le choix de l'état de destination y ne dépend pas forcément de la densité f à simuler. La transition proposée de x_t à y est ensuite acceptée avec une certaine probabilité α. Ce taux d'acceptation α fait intervenir généralement la densité f, ce qui assure la convergence de l'algorithme vers la bonne loi.

Plus précisément, l'algorithme utilise une densité de transition auxiliaire $q(x,.)$ que l'on suppose calculable explicitement et simulable facilement. L'algorithme se décrit alors de la facon suivante :

1. étant donnée la configuration x_t à l'étape t, on simule une configuration y selon la densité $q(x_t,.)$
2. on calcule alors le taux

$$r = \frac{f(y)\, q(y,x_t)}{f(x_t)\, q(x_t,y)} \tag{29}$$

3. avec la probabilité $\alpha = \min(1,r)$ on pose $x_{t+1} = y$ et avec la probabilité $1-\alpha$ on pose $x_{t+1} = x_t$.

Comme dans le cas de l'algorithme de naissance-mort, la constante de normalisation n'intervient pas dans les calculs. Les formules ne sont définies que lorsque $f(x_t)$ est non nul. Si on initialise l'algorithme avec une configuration x_0 telle que $f(x_0) > 0$, alors la chaîne de Markov ne passera que par des états de probabilité non nulle, et le calcul du taux d'acceptation sera toujours défini et strictement positif.

La densité de transition auxiliaire $q(.,.)$ est a priori quelconque, mais il faut lui imposer quelques restrictions pour assurer la convergence de l'algorithme vers la densité f. Dans tous les cas, la loi f est invariante pour la chaîne de Markov produite par l'algorithme - voir par exemple [19]. Pour qu'il y ait

ergodicité, il faut vérifier l'apériodicité et la f-irréductibilité. L'apériodicité est vraie si $P(x,x) > 0$ pour tout x, c'est-à-dire si à chaque étape l'événement $x_{t+1} = x_t$ se produit avec une probabilité non nulle. Cette condition est vérifiée assez facilement dans la plupart des cas.

Dans le cas discret, l'irréductibilité signifie que tout état de probabilité non nulle peut être atteint en un nombre fini de transitions avec une probabilité non nulle. Il est donc important de vérifier que les mouvements autorisés permettent d'atteindre toutes les configurations possibles.

Si l'espace U est discret, l'irréductibilité suffit pour assurer la convergence de l'algorithme pour toute condition initiale. Dans le cas continu, il faut vérifier en plus la récurrence au sens de Harris. Des conditions assez faibles pour que cette propriété soit vraie sont données dans [21, 5].

D'autre part, une remarque importante sur la densité $q(.,.)$ est que pour toute transition possible d'un état x vers l'état y, la transition inverse de y vers x doit être possible. En effet, dans le cas contraire le rapport $q(y,x)/q(x,y)$ est nul, donc le mouvement de x à y sera toujours rejeté.

L'algorithme de Metropolis-Hastings nécessite donc moins de calculs à chaque itération que l'algorithme de naissance-mort. En revanche, lorsque la densité $q(.,.)$ ne dépend pas de f, le choix des transitions est indépendant de cette loi et l'algorithme propose beaucoup de mouvements qui sont immédiatement rejetés. Dans ce cas, le nombre d'itérations à effectuer avant de parvenir à la convergence est plus important que pour l'algorithme de naissance-mort. Notons aussi que la définition des mouvements est primordiale pour l'efficacité de l'algorithme.

4.2.3 Metropolis-Hastings-Green

Une contrainte essentielle sur la densité auxiliaire $q(.,.)$ est qu'elle soit simulable facilement. Dans le cas des processus ponctuels, les transitions que l'on autorise à chaque itération sont donc relativement simples. Ce sont par exemple l'ajout ou la mort d'un objet de la configuration, le déplacement d'un ou plusieurs objets, la dilatation d'un ou plusieurs objets, etc... Lorsque l'espace U est discret, l'algorithme précédent et l'équation 29 restent valables, à

condition de remplacer la densité $q(.,.)$ par une matrice de transition $Q(.,.)$. Lorsque U est continu en revanche, le fait de ne considérer que des noyaux de transition ayant une densité pour la mesure μ de l'espace Ω est trop restrictif. Notamment, le noyau de transition n'autorisant que les naissances et les morts n'admet pas une telle densité.

Green a ainsi introduit une autre version de l'algorithme de Metropolis-Hastings, qui est sensiblement identique à la précédente mais où les transitions sont définies par des mesures de probabilité et non des densités.

La densité non-normalisée f est donc remplacée par la mesure π non-normalisée sur l'espace des états Ω. La densité auxiliaire est remplacée par le noyau $Q(x,A)$, $x \in \Omega$, $A \in \mathcal{B}$. Nous supposons également l'existence d'une mesure symétrique ξ sur $\Omega \times \Omega$ dominant $\pi(dx)Q(x,dy)$, de sorte que la dérivée de Radon-Nikodym existe:

$$h(x,y) = \frac{\pi(dx)Q(x,dy)}{\xi(dx,dy)} \tag{30}$$

qui remplace $f(x)q(x,y)$ dans l'échantillonneur de Metropolis-Hastings. Le taux d'acceptation devient alors :

$$\alpha = \frac{h(y,x)}{h(x,y)} \tag{31}$$

L'algorithme est donc le suivant :

1. Simuler $y \sim Q(x,\cdot)$
2. Calculer le rapport $r = h(y,x)/h(x,y)$
3. Accepter le nouvel état y avec la probabilité $\min(1,r)$.

Cette formulation permet de démontrer la convergence de l'algorithme pour les processus ponctuels sur des espaces continus. On se reportera notamment à [10] pour les démonstrations.

Dans la suite, nous nous plaçons dans le cas de simulations numériques finies : l'espace image T est une image numérique discrétisée, et l'espace objet U est également fini. Le calcul des taux d'acceptation se fera donc avec la formule 29.

5 Simulations de processus objets

Le but de ce paragraphe est de comparer l'efficacité des 2 algorithmes en terme de temps de calcul sur plusieurs types de processus objets, et par la même occasion d'étudier les propriétés de quelques processus attractifs.

Toutes les simulations ont été programmées dans le langage C++.

5.1 Implantation

5.1.1 Metropolis-Hastings-Green (MHG)

Pour comparer les performances des 2 algorithmes, nous avons utilisé une version simplifiée de l'algorithme MHG. Les seuls mouvements autorisés à chaque itération sont comme pour l'algorithme de naissance-mort l'ajout ou la suppression d'un objet. Les objets sont dans un premier temps uniquement des cercles de rayon fixe R. À chaque itération, on effectue donc les opérations suivantes :

1. choisir si l'on va effectuer une naissance avec une probabilité P_1 ou une mort avec une probabilité $P_2 = 1 - P_1$;
2. dans le cas d'une naissance :
   - choisir un objet u uniformément dans $U \setminus x$;
   - calculer :
   $$r = \lambda^*(x,u)\,\frac{Q(x \cup \{u\},x)}{Q(x,x \cup \{u\})} = \lambda^*(x,u)\,\frac{P_2}{P_1}\,\frac{|U| - n(x)}{n(x)+1} \qquad (32)$$
   - avec la probabilité $\min(1,r)$, poser $x_{t+1} = x \cup \{u\}$.
3. dans le cas d'une mort :
   - choisir un objet v uniformément dans x ;
   - calculer :
   $$r = \frac{1}{\lambda^*(x \setminus \{v\},v)}\,\frac{Q(x \setminus \{v\},x)}{Q(x,x \setminus \{v\})} = \frac{1}{\lambda^*(x \setminus \{v\},v)}\,\frac{P_1}{P_2}\,\frac{n(x)}{|U| - n(x) + 1} \qquad (33)$$

– avec la probabilité min(1,r), poser $x_{t+1} = x \setminus \{v\}$.

Les deux mouvements autorisés, l'ajout et la suppression d'un objet, sont suffisants pour garantir l'irréductibilité pour toute distribution f strictement positive sur Ω. En effet, dans ce cas les taux d'acceptation pour les deux mouvements ne sont jamais nuls. Il est en particulier possible de rejoindre la configuration vide en un temps fini à partir de n'importe quelle configuration x, puisque la probabilité d'effectuer $n(x)$ suppressions successives est non nulle. Il est de même possible de rejoindre à partir de la configuration vide toute configuration de Ω. L'irréductibilité en découle immédiatement.

5.1.2 Naissance-mort

L'algorithme implanté est celui décrit dans le paragraphe 4.2.1, dans le cas $k = 1$. Les taux de naissance sont enregistrés dans un tableau, et le point délicat consiste à mettre à jour ces mêmes taux. Pour cela, il faut considérer chaque modèle au cas par cas et essayer de n'effectuer que les mises-à-jour nécessaires, c'est-à-dire de ne recalculer que les taux susceptibles d'avoir changé.

5.1.3 Convergence

Le problème de détecter quand la chaîne de Markov a atteint la convergence est assez difficile. C. Robert donne par exemple dans [19] quelques indicateurs de convergence, mais ceux-ci ne sont pas toujours calculables dans notre cas particulier. Nous préférons donc des méthodes graphiques, qui pour les modèles simples suivants suffisent à déterminer lorsque la chaîne s'est stabilisée. Nous traçons donc plusieurs statistiques significatives $h_i(X)$ du modèle ainsi que leurs moyennes empiriques :

$$\bar{h_i}(T) = \frac{1}{T} \sum_{t=1}^{T} h_i(x_t) \tag{34}$$

Les suites $(h_i(T))_T$, qui convergent lorsque la chaîne est stationnaire, donnent un indicateur visuel de la convergence de l'algorithme. Pour le processus de Strauss, les statistiques utilisées sont par exemple le nombre d'objets $n(x)$ et le nombre de paires d'objets s'intersectant $p(x)$.

5.2 Processus de Strauss

La densité de ce processus est

$$f(x) \propto \beta^{n(x)}\gamma^{p(x)} \tag{35}$$

où $p(x)$ est le nombre de paires d'objets qui s'intersectent. La densité de Papangelou s'exprime alors simplement par

$$\lambda^*(x,u) = \beta\,\gamma^{s(x,u)} \tag{36}$$

$$\text{où} \qquad s(x,u) = \sum_{v\in x} 1[R(v)\cap R(u)\neq\emptyset]$$

Ce processus a été introduit par Strauss dans [20] afin de modéliser la croissance de jeunes pousses de séquoia autour de troncs plus anciens. Strauss s'est intéressé plus particulièrement au cas attractif, c'est-à-dire au cas où $\gamma > 1$. Plusieurs études, par exemple [14], ont cependant souligné que la simulation du processus de Strauss dans le cas attractif est délicat car très sensible aux valeurs des paramètres. En effet les deux algorithmes donnent pour de grandes valeurs de γ une configuration contenant un grand nombre d'objets agglutinés en un seul cluster. Ceci s'explique par le fait que la densité 35 est fortement concentrée autour de la configuration contenant un seul amas d'objets et remplissant tout l'espace. Dans le cas continu, lorsque $\Omega = I\!R^d$, cette densité n'est d'ailleurs pas intégrable - voir [17]. Geyer explique dans [11] pourquoi cette densité ne constitue pas une modélisation intéressante d'objets attractifs. Dans le cas où le nombre d'objets est fixé, il montre qu'un tel processus est en quelque sorte bimodal, puisque ses réalisations apparaissent soit comme un seul bloc d'objets amassés dans une sphère de rayon R (correspondant à $\gamma = +\infty$), soit comme la réalisation d'un processus de Poisson (correspondant à $\gamma = 1$). Cette dualité a effectivement été constatée en pratique : à partir d'une certaine valeur du rapport β/γ, les objets "envahissent" l'image, et en-dessous de cette même valeur, les interactions sont très faibles.

Le cas répulsif, lorsque $0 \leq \gamma < 1$, fournit en revanche un modèle intéressant pour les motifs réguliers. La figure 1 donne un exemple de réalisation de ce processus, comparé avec un processus de Poisson.

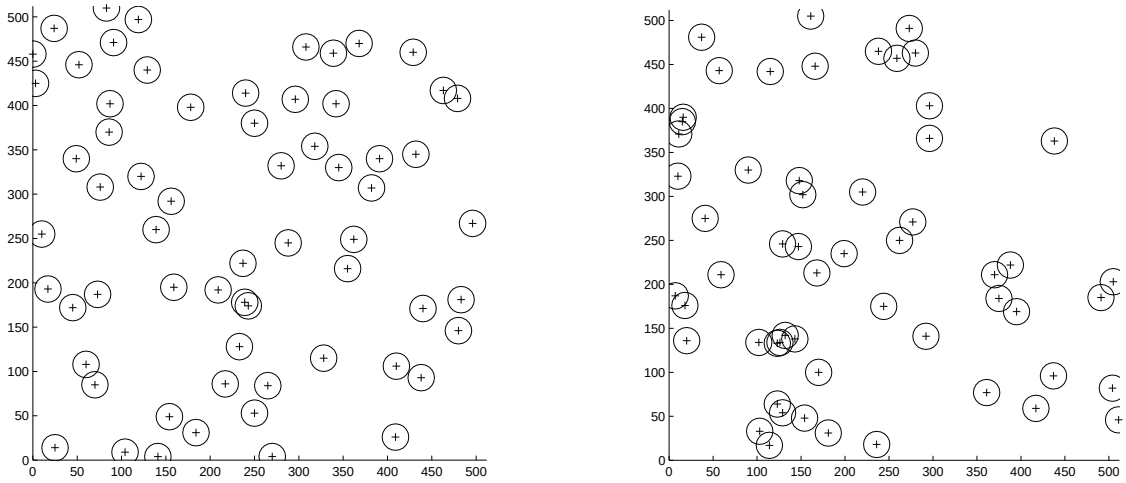

FIG. 1 – *À gauche, simulation d'un processus de Strauss répulsif de paramètres* $\beta = 0.0005$ *et* $\gamma = 0.1$. *Les disques ont un diamètre de 30 dans une grille de 512x512 pixels. À droite, simulation d'un processus de Poisson ayant le même nombre moyen d'objets.*

Bien que le modèle ne soit pas très intéressant en lui-même (du moins pour simuler des interactions attractives), la comparaison des performances des 2 algorithmes est assez instructive. En effet, le modèle est des plus simples, et les taux de naissance qui varient sont faciles à trouver et à mettre à jour. En effet :

- après la naissance d'un objet u, $b(x \cup \{u\},v) = \gamma\, b(x,v)$ pour tous les objets v tels que $u \sim v$. Pour les autres objets, le taux de naissance est inchangé ;
- après la mort d'un objet u, $b(x \setminus \{u\},v) = b(x,v)/\gamma$ pour les objets v tels que $u \sim v$. Pour les autres objets, le taux est inchangé.

Comme de plus les objets ont une taille fixe, le nombre d'objets tels que $v \sim u$ pour u fixé est constant, et ces objets sont aisés à déterminer. Par conséquent, le modèle est a priori un cas très favorable pour l'algorithme de naissance-mort, et les calculs de mise-à-jour des taux semblent peu pénalisants par rapport à l'algorithme MHG.

Or les simulations numériques montrent que malgré cela l'algorithme de Metropolis est plus rapide dans la plupart des cas. Le nombre d'itérations requis par ce dernier pour atteindre la convergence est effectivement plus élevé, mais le temps total de calcul reste généralement plus faible. Plus exactement, l'algorithme de naissance-mort est plus rapide lorsque les objets ont une taille très faible, mais lorsque R augmente, le nombre de taux à mettre à jour augmente : dans le cas de cercles de rayon R, le nombre d'objets $v \in U$ tels que $v \sim u$ à u fixé est en effet de l'ordre de $O(R^2)$. Lorsque R est grand, l'algorithme de MHG devient plus intéressant. Il a fallu ainsi un peu plus d'une minute et environ 700 itérations pour obtenir la simulation de la figure 1 avec l'algorithme de naissance-mort, contre moins de 10 secondes mais 2500 itérations avec MHG. D'autre part, soulignons encore une fois que la version de l'algorithme MHG que nous avons implantée n'autorise que des mouvements de type naissance-mort et que les naissances sont choisies de façon uniforme sur $U \setminus x$. En envisageant par exemple des naissances non uniformes, il est possible de diminuer le nombre d'itérations de l'algorithme MHG et d'améliorer ses performances.

5.3 Processus de regroupement aléatoire (*Random-cluster model*)

Nous nous intéressons maintenant à la simulation du modèle attractif de densité

$$f(x) \; \alpha \; \beta^{n(x)} \gamma^{-c(x)}$$

$c(x)$ désignant le nombre de composantes connexes du graphe de x pour la relation $u \sim v \;\Leftrightarrow R(u) \cap R(v) \neq \emptyset$. Le paramètre γ est strictement supérieur à 1. La densité de Papangelou vaut ici :

$$\lambda^*(x,u) = \beta \, \gamma^{c(x,u)-1} \tag{37}$$

où $c(x,u)$ est le nombre de composantes connexes de x qu'intersecte l'objet u.

5.3.1 Implantation

Afin de pouvoir calculer efficacement le rapport $\lambda^*(x,u)$, il faut mémoriser le graphe de x pour la relation $\sim$. Ceci se fait de façon classique, en utilisant une table de hachage qui à chaque objet u associe la liste des objets v de x coupant u. D'autre part, les objets sont étiquetés par un entier en fonction de la composante connexe à laquelle ils appartiennent : ainsi deux objets sont dans la même composante si et seulement si leurs étiquettes sont égales.

Le calcul des taux de naissance qui doit être effectué dans l'algorithme de naissance-mort est beaucoup plus pénalisant que dans le modèle précédent. Pour pouvoir effectuer une comparaison "honnête" entre les deux algorithmes, il faut donc s'efforcer d'implanter le calcul de ces taux de la manière la plus efficace possible. Ce calcul consiste à évaluer les grandeurs $c(x,u)$, où $u \in U \setminus x$. Pour cela, nous avons essentiellement deux possibilités :

1. Commencer par chercher les objets $v \in U$ tels que $u \sim v$: il y a $O(R^2)$ tels objets. Parmi ceux-ci, chercher les objets qui sont dans la configuration x et compter le nombre d'étiquettes différentes. Ce nombre est précisément le nombre de composantes connexes qu'intersecte u. Comme nous utilisons une table de hachage (supposée parfaite), la recherche d'un élément dans la configuration prend un temps constant $O(1)$. Le temps total du calcul de $c(x,u)$ par cette méthode est donc en $O(R^2)$. Cette

méthode doit par conséquent être employée lorsque les objets sont de petite taille.

2. Parcourir directement toute la configuration x et trouver tous les voisins de u dans x, puis compter de la même manière le nombre d'étiquettes différentes parmi ces objets. Ce calcul prend un temps total $O(n(x))$, donc cette méthode doit être utilisée lorsque les objets sont peu nombreux.

Les opérations de mise à jour du graphe et des étiquettes à chaque suppression ou insertion d'un objet dans x connaissent la même alternative d'implantation, mais l'enjeu est dans ce cas moins crucial : comme nous avons pu le constater, le facteur limitant dans l'algorithme de naissance-mort est bien le calcul des taux de naissance. À titre d'exemple, les temps de calcul pour les diverses opérations de cet algorithme ont été mesurés avec les paramètres de la figure 5 (donc avec des objets relativement gros, mais finalement peu nombreux). Il s'est ainsi avéré qu'avec notre implantation 97% du temps de calcul est consacré à la mise-à-jour des taux de naissance-mort, 2% aux choix des objets à ajouter ou à supprimer et moins de 1% à la mise-à-jour du graphe et des étiquettes.

5.3.2 Résultats numériques

Pour ce processus, l'algorithme de naissance-mort est particulièrement inefficace lorsque les objets sont relativement nombreux ou qu'ils ont une taille trop grande. En effet les opérations de mise-à-jour des taux de naissance et de mort sont dans ce cas particulièrement coûteuses. Lorsque par exemple on ajoute un point u à la configuration x, les objets $v \in U \backslash x$ dont il faut recalculer le taux de naissance $b(x \cup \{u\},u)$ sont ceux tels que

$$v \sim_{x \cup \{u,v\}} u$$

Pour tous ces objets, il est en effet possible que $c(x \cup \{u\},v)$ diffère de $c(x,u)$ (ce n'est pas toujours le cas, mais on peut trouver des configurations telles que cette assertion soit vraie). Lorsque les objets sont de grande taille et nombreux, le nombre de tels objets devient très important, et le coût des mises-à-jour des taux de naissance-mort est donc élevé.

Il est possible d'envisager d'autres méthodes que celles proposées dans le paragraphe précédent pour améliorer le temps de calcul des grandeurs $c(x,u)$. On pourrait par exemple mémoriser pour chaque $u \in U$ la liste de ses objets voisins dans la configuration x (au lieu de ne mémoriser ce graphe que sur les objets $u \in x$). Le calcul de $c(x,u)$ serait ainsi linéaire en le nombre de voisins de u dans x (au lieu de $O(n(x))$ ou $O(R^2)$). Cette méthode est envisageable dans notre exemple où l'espace U est de la même taille que l'image, mais lorsque les objets ont plus de paramètres, l'espace mémoire nécessaire sera très grand. De plus, le nombre de taux à mettre à jour n'est pas diminué, ce qui laisse penser que l'avantage restera à l'algorithme de Metropolis-Hastings-Green.

Les figures suivantes 2 et 5 présentent les résultats obtenus par les deux algorithmes. Pour deux simulations, nous donnons l'évolution dans le temps des 2 grandeurs significatives de la chaîne de Markov, $n(X)$ et $c(X)$, ainsi que celle de leurs moyennes empiriques.

Au vu des simulations, nous pouvons dire que ce modèle donne des résultats bien plus probants que le processus de Strauss. Les objets sont effectivement plus agglomérés que dans un processus de Poisson, et les simulations sont moins sensibles aux paramètres que le processus de Strauss : la dichotomie relevée dans le paragraphe 5.2 n'apparaît pas dans ce cas.

5.4 Processus d'interaction d'aires

Ce processus constitue comme le précédent un modèle intéressant d'interactions attractives, puisqu'il prend en compte la surface totale occupée par les objets dans l'image. La densité de probabilité décroit exponentiellement selon cette surface :

$$f(x) \propto \beta^{n(x)}\, \gamma^{-\lambda(R_x)} \tag{38}$$

où $R_x = \bigcup_{u \in x} R(u)$ et $\gamma > 1$. La densité de Papangelou s'exprime par :

$$\lambda^*(x,u) = \beta\, \gamma^{\lambda(R_x \cup R(u)) - \lambda(R(u))} = \beta\, \gamma^{\lambda(R_x \cup R(u)) - \pi R^2} \tag{39}$$

Contrairement aux deux processus précédents, cette densité est fastidieuse à calculer. Une méthode indirecte pour simuler cette densité est donnée dans

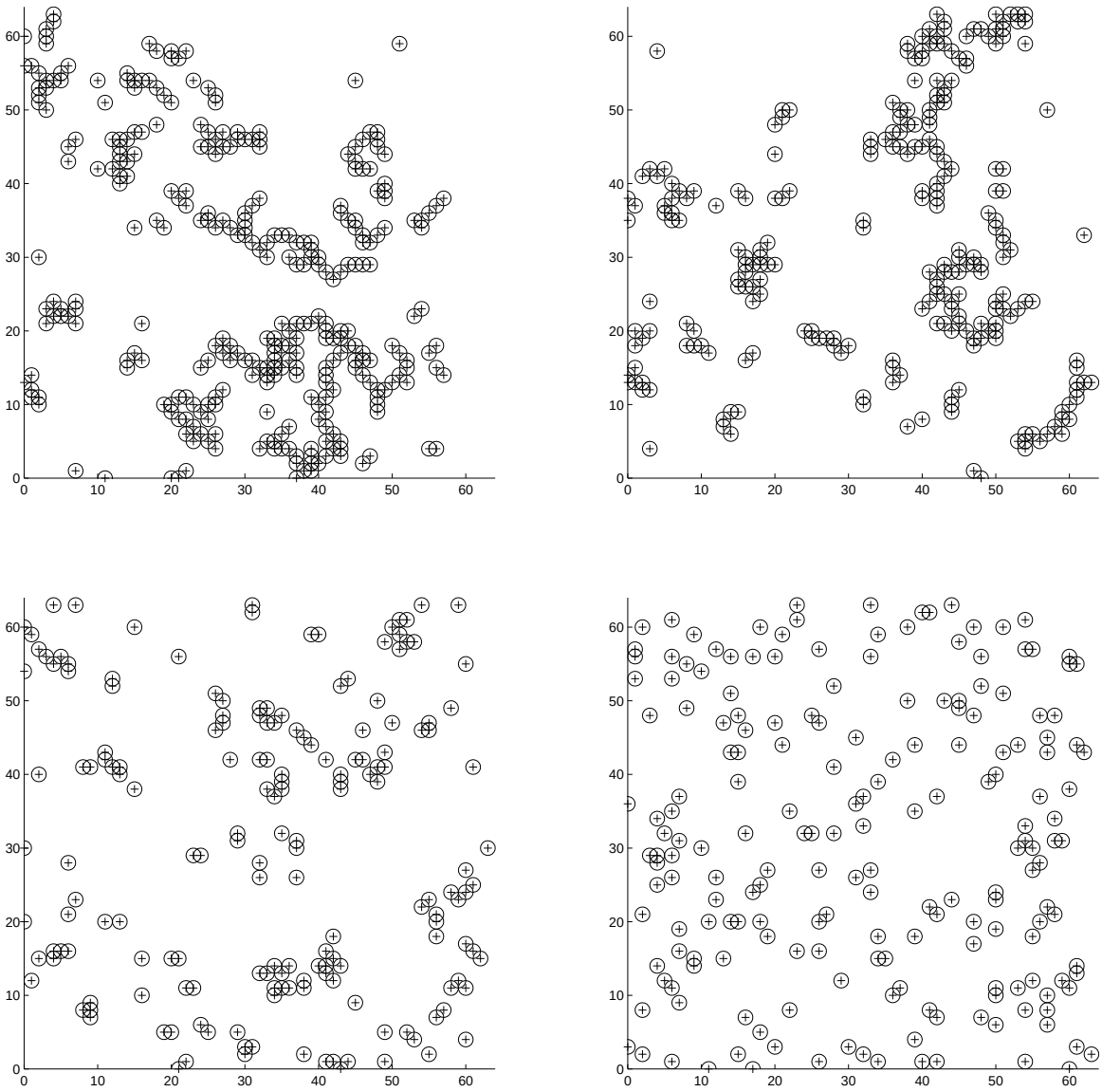

FIG. 2 – *Simulations d'un modèle de clustering. Les objets sont des disques de diamètre 2 sur une grille de* 64×64 *pixels. En haut à gauche, les paramètres sont* $\beta = 0.15$, $\gamma = 30$, *en haut à droite* $\beta = 0.12$, $\gamma = 50$, *en bas à gauche* $\beta = 0.1$, $\gamma = 5$ *et en bas à droite un processus de Poisson.*

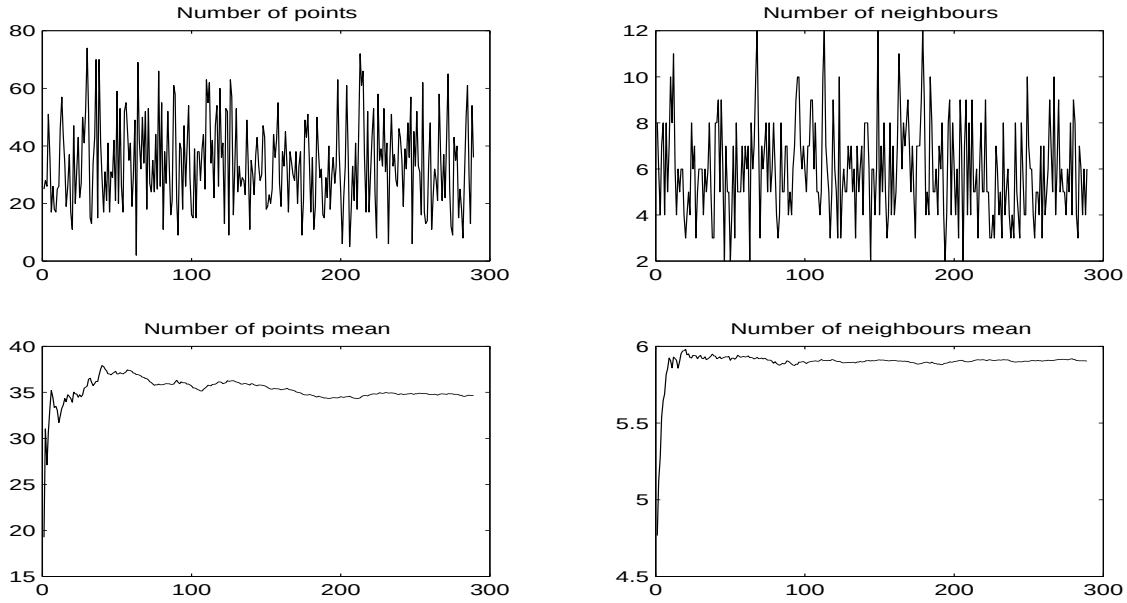

FIG. 3 – *Evolution du nombre de points et de clusters lors de la simulation par un algorithme de type Metropolis-Hastings d'un modèle attractif, avec* $\beta = 0.018$, $\gamma = 30$, $R = 2.5$. *Le temps de calcul est de 10 min, mais la stationnarité est atteinte plus rapidement.*

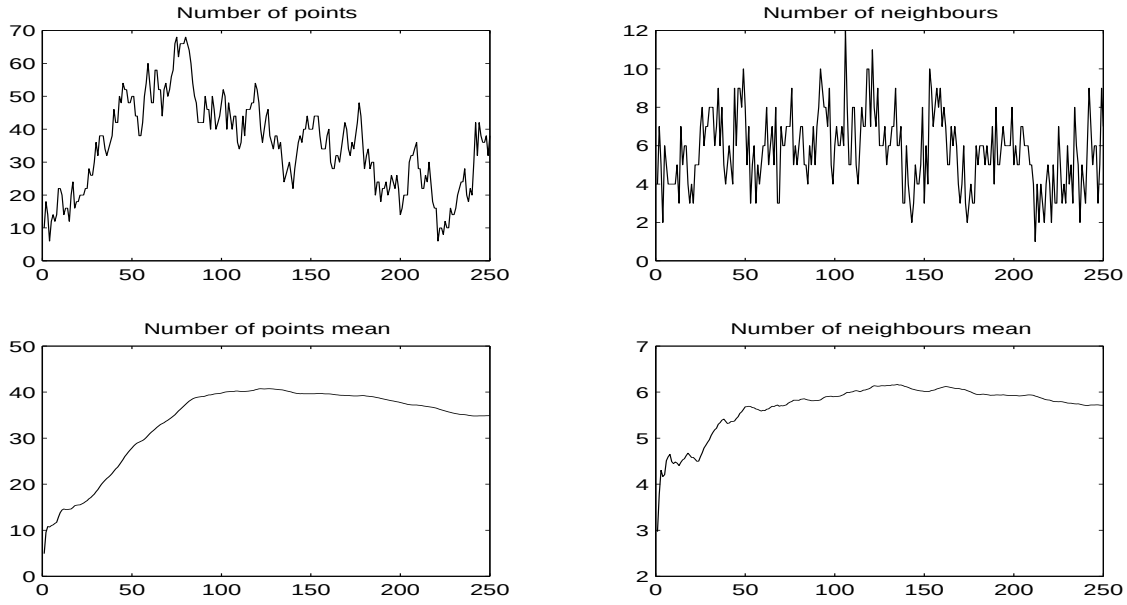

FIG. 4 – *Simulation par l'algorithme de naissance-mort du même processus après 10 min de calcul également. Cette fois-ci, l'algorithme de naissance-mort n'a pas atteint la stationnarité.*

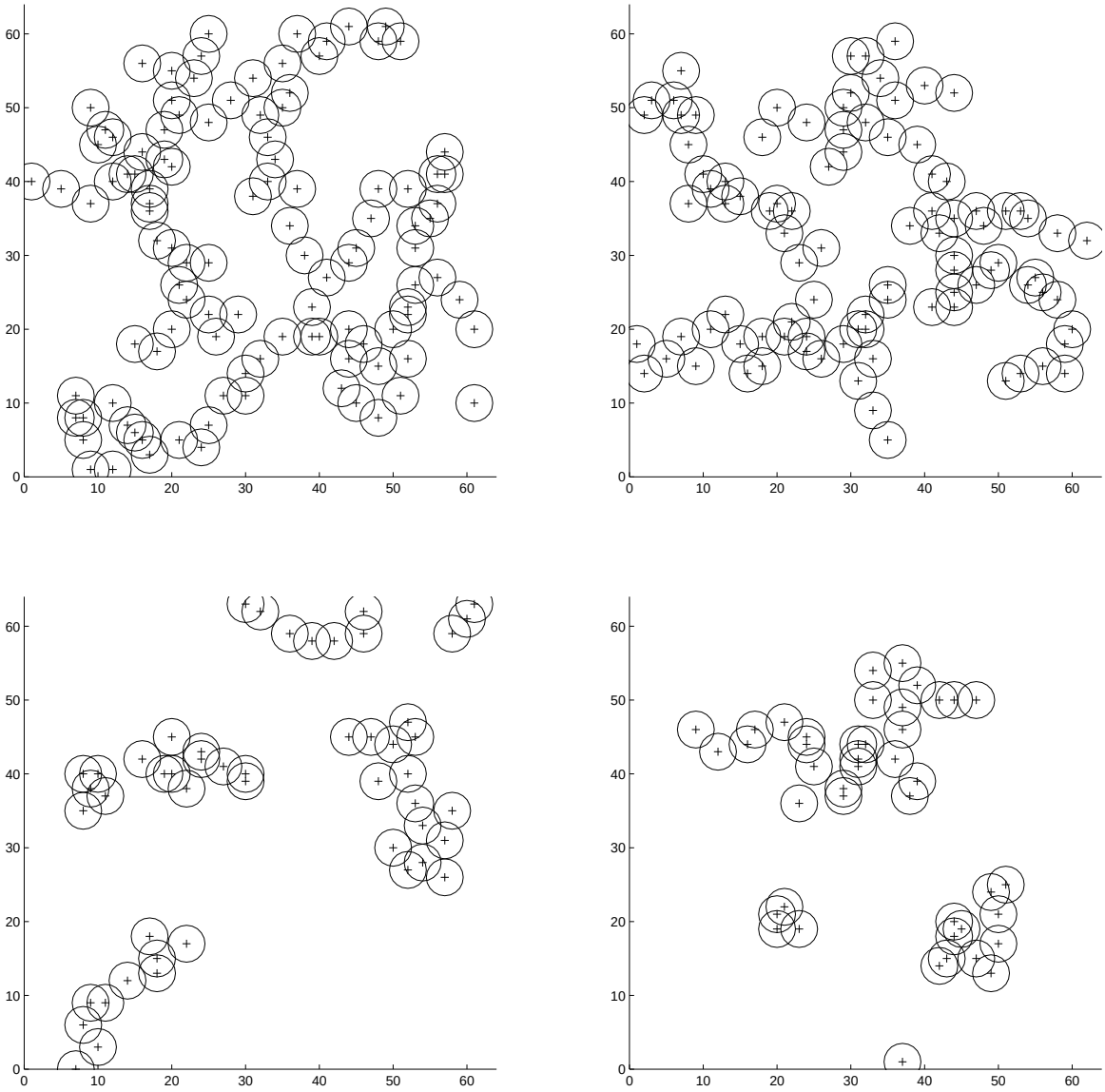

FIG. 5 – *Simulations d'un modèle de clustering. Les objets sont des disques de diamètre 5 sur une grille de* 64×64 *pixels. En haut, les paramètres sont* $\beta = 0.02$, $\gamma = 30$, *et en bas* $\beta = 0.018$, $\gamma = 30$.

[22] et reprise dans [8]. L'idée consiste à simuler deux processus de Poisson distincts dans le même espace, en imposant que deux objets appartenant à 2 processus différents ne peuvent se couper. Les deux processus se repoussent ainsi mutuellement, obligeant les objets d'un même processus à se chevaucher. Ainsi, si l'on considère le processus sur l'espace $\Omega \times \Omega$ de densité :

$$f(x,y) \propto \beta_1^{n(x)}\, \beta_2^{n(y)}\, 1[R_x \cap R_y = \emptyset] \tag{40}$$

on peut montrer que la densité marginale de x est :

$$f(x) \propto \beta_1^{n(x)}\, \exp(\beta_2\, \lambda(R_x)) \tag{41}$$

c'est-à-dire un processus d'interaction d'aires de paramètres (β_1, e^{β_2}).

Pour simuler ce processus sur $\Omega \times \Omega$, nous avons choisi de n'utiliser que l'algorithme de MHG, plus simple d'emploi dans ce cas. Voici la description des calculs effectués à chaque itération :

1. Choisir de façon équiprobable un des 4 mouvements suivants : ajout d'un objet à la configuration x, ajout d'un objet à la configuration y, suppression d'un point de x, suppression d'un point de y.
2. dans le cas de l'ajout d'un point à x :
   - choisir un objet u uniformément dans $U \setminus x$;
   - calculer :

$$r = \beta_1\, 1[R(u) \cap R_y = \emptyset]\, \frac{|U| - n(x)}{n(x)+1} \tag{42}$$

   - avec la probabilité $\min(1,r)$, poser $x_{t+1} = x \cup \{u\}$.
3. dans le cas de la suppression d'un objet de x :
   - choisir un objet v uniformément dans x ;
   - calculer :

$$r = \frac{1}{\beta_1}\, \frac{n(x)}{|U| - n(x) + 1} \tag{43}$$

   - avec la probabilité $\min(1,r)$, poser $x_{t+1} = x \setminus \{v\}$.

4. L'ajout et la suppression d'un objet de y se traitent de la même façon.

Comme prévu, les simulations montrent que les objets se superposent beaucoup plus qu'avec le modèle de clustering précédent. D'autre part, nous constatons que le modèle est cette fois-ci très sensible aux paramètres : si par exemple le rapport β_2/β_1 est trop grand, le processus y a tendance à étouffer le processus x, dont le nombre d'objets à l'équilibre est quasiment nul. Quelques simulations sont présentées dans la figure 6.

Il serait également intéressant de simuler le cas répulsif $\gamma < 1$: au lieu de se chevaucher, les objets auraient tendance à occuper le maximum d'espace, ce qui peut être intéressant pour la segmentation. Malheureusement, la méthode de simulation proposée ci-dessus ne fonctionne pas pour le cas répulsif, et la méthode directe est probablement fastidieuse à implanter dans le cas général.

5.5 Conclusion

Sur les quelques modèles présentés ici, l'algorithme de Metroplis-Hastings-Green semble donc plus performant : plus simple à implanter, il est également dans bien des cas plus rapide. L'algorithme de naissance-mort n'est en fait intéressant que lorsque l'espace des objets U est suffisamment petit et que les objets n'ont des interactions qu'à faible distance. Son avantage réside dans le fait qu'à travers le calcul des taux de naissance, il permet de rejoindre plus rapidement les configurations de plus grande vraisemblance, mais ces calculs sont cependant très pénalisants pour la vitesse totale de l'algorithme.

Nous avons comparé les deux algorithmes uniquement du point de vue du temps de calcul CPU. Pour une étude plus mathématique, concernant notamment les propriété de mélange de la chaîne de Markov produite, le lecteur pourra se référer à [1] ou [6].

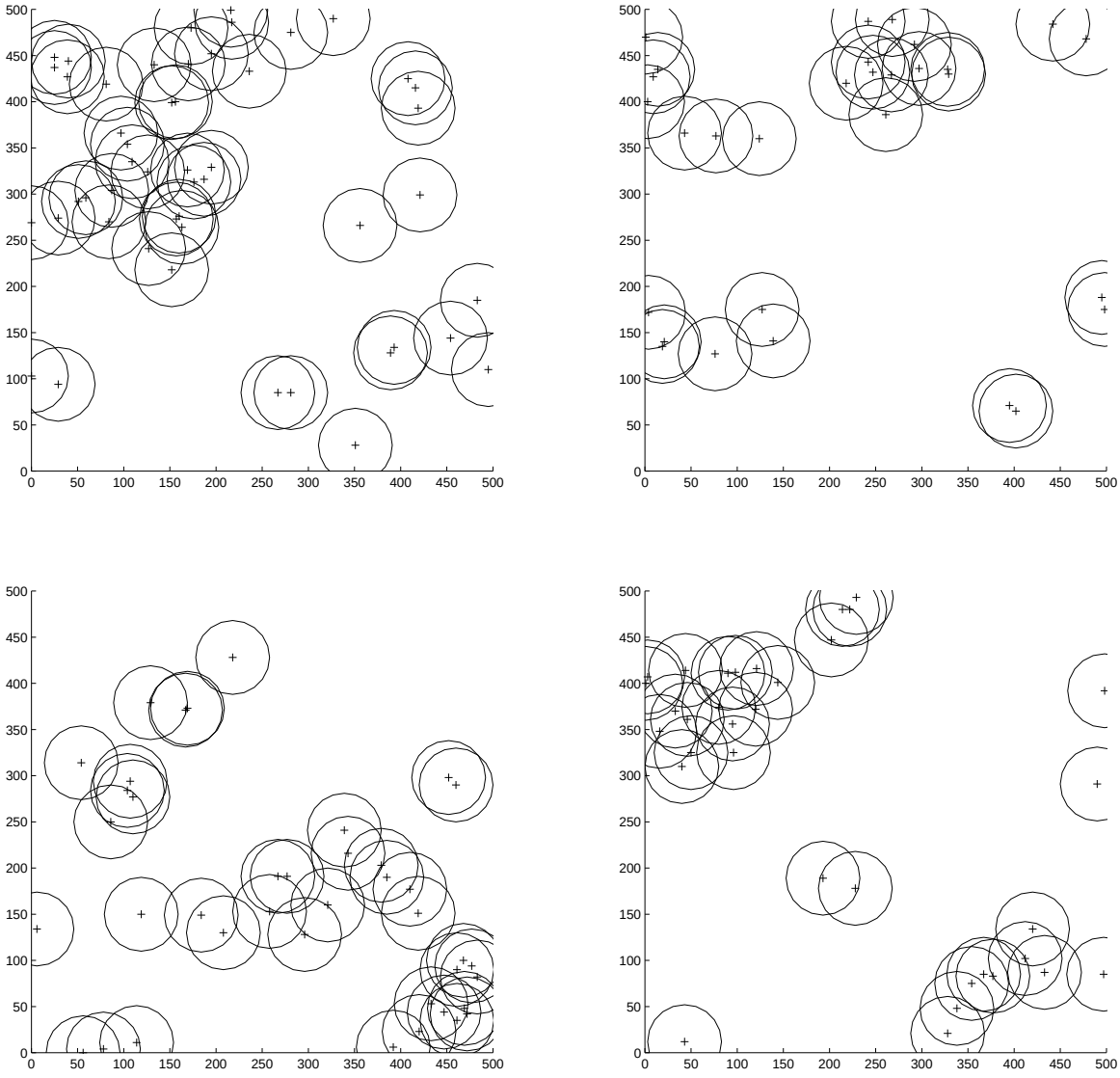

FIG. 6 – *Simulations d'un modèle d'interaction d'aire. Les disques ont un diamètre 65 dans une grille de 512x512 pixels, et les paramètres sont* $\beta = 0.0007$, $\gamma = \exp(.0008)$

6 Applications à la segmentation

6.1 Cadre bayésien

Une approche classique pour l'extraction d'information ou la restauration d'images est l'approche bayésienne. On dispose ainsi d'une image observée y qui est la "déformation" d'une image idéale x que l'on souhaite récupérer. On suppose également que l'image x a été générée selon une *distribution a priori* de densité $p(x)$ et que la déformation de l'image suit une loi de probabilité de densité $f(y \mid x)$. La méthode bayésienne consiste à optimiser au sens d'un certain critère la loi a posteriori $P(x \mid y)$. La probabilité conditionnelle de x connaissant y est :

$$p(x \mid y) = \frac{f(y \mid x)\, p(x)}{p(y)} \propto f(y \mid x)\, p(x) \tag{44}$$

Le critère du Maximum A Posteriori (MAP) est donné par :

$$\bar{x} = \arg\max_{x} f(y \mid x)\, p(x) \tag{45}$$

Nous nous intéressons ici au problème de la segmentation : on veut ainsi extraire de l'image les zones appartenant à des entités différentes. Pour des images satellitaires ou aériennes, le but est de partitionner l'image en distinguant par exemple les zones urbaines des zones rurales, ou les champs des routes, etc...

La nouvelle approche que nous proposons ici consiste à ne plus considérer l'image idéale x comme un champ de pixels markovien, mais comme la réalisation d'un processus objets. On espère par cette méthode pouvoir représenter directement chaque zone de l'image par un ou un nombre restreint d'objets de forme géométrique simple.

6.2 Recuit simulé

L'intérêt de simuler rapidement des processus aléatoires est de pouvoir ensuite utiliser l'algorithme du *recuit simulé* pour résoudre le problème de la

recherche du maximum $\bar{x}$. Cette méthode consiste à simuler successivement les lois

$$f_T(x) \propto (f(y \mid x)\, p(x))^{\frac{1}{T}} \tag{46}$$

où $T > 0$ est un réel appelé *température*. Lorsque T tend vers 0, la loi $f_T(x)$ tend vers une distribution uniforme sur l'ensemble des solutions maximales $\bar{x} = \arg\max_x f(y \mid x)\, p(x)$ (donc un Dirac en la solution maximale si celle-ci est unique). L'algorithme de recuit-simulé effectue plusieurs simulations des lois $f_T(x)$ en faisant baisser progressivement la température vers 0. Le résultat de ces simulations tend alors vers la solution optimale recherchée avec une forte probabilité. La décroissance de la température doit néanmoins être "suffisamment" lente, car dans le cas contraire les simulations peuvent rester bloquées sur des maxima locaux.

6.3 But de l'étude

Dans la suite de cette étude, nous allons construire un modèle simple permettant de tester l'efficacité de cette approche géométrique pour la segmentation d'image. Les images tests seront des images de synthèse comme celle de la figure 11, constituées de zones rectangulaires parallèles aux axes. Le modèle peut sembler simpliste, mais le but de l'étude est plutôt de tester la méthode, d'en appréhender les avantages ou les difficultés que d'obtenir dès maintenant des résultats sur des cas réels.

Les simulations se feront donc avec des rectangles de tailles variables, mais d'orientation fixe. Nous cherchons ainsi à récupérer la configuration idéale x des rectangles ayant produit l'image observée y.

6.4 Modèle a priori

Les objets sont des rectangles paramétrés par les coordonnées de leur coin inférieur gauche, leur longueur et leur largeur. Les longueurs des côtés sont comprises entre une valeur minimale et une valeur maximale. On cherche une distribution sur ces rectangles qui modélise un pavage aléatoire de l'image. Cette distribution doit donc favoriser les rapprochements entre rectangles et

pénaliser les chevauchements. Il semble dans ce cas plus approprié d'utiliser un modèle avec interactions par paires d'objets. Nous définissons donc un potentiel g sur $U \times U$ tel que :

$$p(x) \propto \beta^{n(x)} \prod_{\{u_i,u_j\}\subset x} g(u_i,u_j) \tag{47}$$

Le potentiel $g(u,v)$ comporte deux parties :

1. Une partie répulsive : lorsque $|R(u) \cap R(v)| \neq 0$,

$$g(u,v) = \exp\left(-\gamma_1 \frac{|R(u) \cap R(v)|}{\min(|R(u)|,|R(v)|)}\right) \tag{48}$$

   où $\gamma_1 > 0$ et où $|.|$ désigne la mesure de Lebesgue de $\mathbb{R}^2$.

2. Une partie attractive : lorsque $|R(u) \cap R(v)| = 0$,

$$g(u,v) = 1 + \gamma_3\, l(u,v)\, \exp\left(-\gamma_2\, d(u,v)\right) \tag{49}$$

   où $\gamma_2 > 0$, $\gamma_3 > 0$, $d(u,v)$ est une distance entre les objets u et v, $l(u,v)$ la longueur des côtés en regard (voir figure 7).
   Ce potentiel est maximal lorsque les deux rectangles ont un côté en commun.

En utilisant des valeurs suffisamment élevées des paramètres γ_1, γ_2 et γ_3, c'est-à-dire à basse température, les simulations présentent l'aspect recherché d'un pavage aléatoire de l'image (voir figure 8). Les paramètres β et γ_3 permettent de faire varier le nombre d'objets moyen. Le pavage obtenu est satisfaisant. Néanmoins, nous pouvons remarquer que les rectangles ont tendance à être de petite taille. Pour pallier cet inconvénient, on peut rajouter un terme favorisant les gros rectangles dans le modèle a priori. Ce terme doit être une fonction croissante h de l'aire d'un rectangle, qui doit de plus vérifier $h(a_1+a_2) > h(a_1)h(a_2)$, afin que la réunion de 2 rectangles soit "meilleure" que ces 2 rectangles séparés. On peut par exemple prendre pour h une exponentielle quadratique. Le modèle a priori devient alors :

$$p(x) \propto \beta^{n(x)} \prod_{u_i\in x} \exp(\delta\, v^2(u_i)) \prod_{\{u_i,u_j\}\subset x} g(u_i,u_j) \tag{50}$$

où $v(u)$ désigne l'aire de u et où $\delta > 0$. Une simulation du modèle a priori contenant un terme favorisant les grands objets donne le résultat de la figure 9.

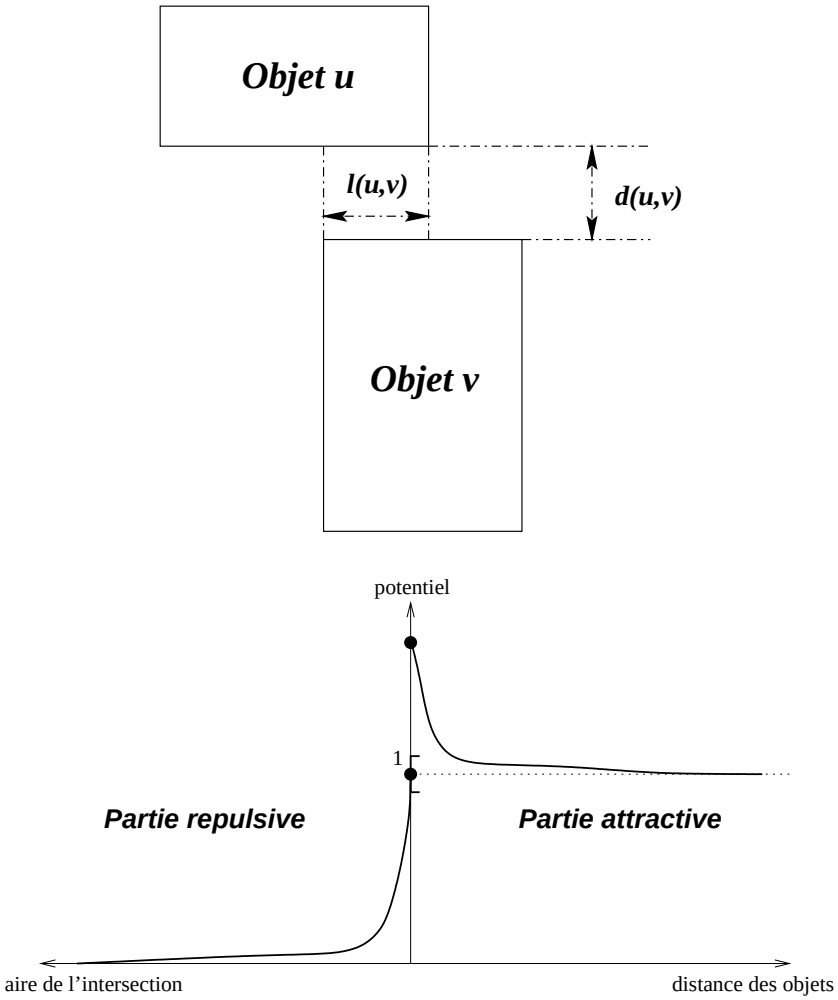

FIG. 7 – *Distances utilisées entre les rectangles (en haut) et représentation schématique du potentiel (en bas).*

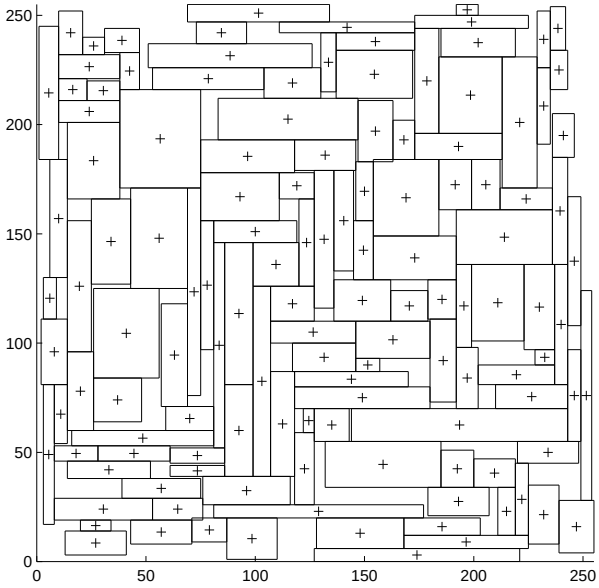

FIG. 8 – *Simulation du modèle a priori*

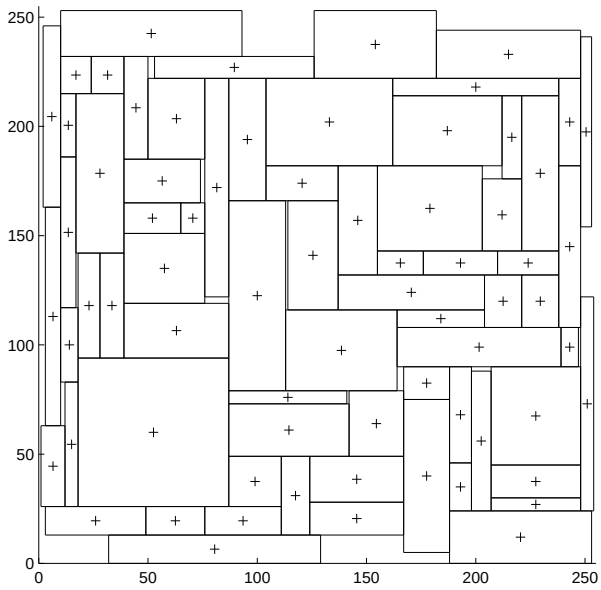

FIG. 9 – *Simulation du modèle a priori après rajout d'un terme favorisant les objets de grande taille.*

6.5 Attache aux données

Le terme d'attache aux données correspond à la loi $p(y \mid x)$, qui modélise le passage de la configuration idéale x à l'image réelle observée y. Baddeley et Van Lieshout [2] proposent un modèle consistant en une déformation déterministe suivie d'un bruit aléatoire. Ils écrivent alors la probabilité conditionnelle $p(y \mid x)$ comme le produit :

$$p(y \mid x) \propto \prod_{t \in T} g(y(t) \mid \theta_x(t)) \tag{51}$$

où θ_x représente l'image idéale résultant de la configuration idéale x, et où $g(.|\theta)$ est une distribution modélisant le bruit. Cette représentation est intéressante pour la reconnaissance d'objets identiques dans une image, mais elle est difficilement extensible au cas où les données de l'image observée y, associées aux objets de la configuration x, ont des distributions différentes. Ce modèle suppose de plus que la distribution g est connue.

Une autre façon de représenter l'attache aux données est de définir un potentiel q sur U et d'écrire ainsi :

$$p(y \mid x) \propto \prod_{u_i \in x} q(u_i) \tag{52}$$

On souhaite que $q(u)$ soit maximal lorsque le support $R(u)$ de l'objet dans l'image est une zone homogène, c'est-à-dire une zone qui correspond à un même objet physique (par exemple un champ ou une ville). Nous pourrions supposer connues les lois régissant la "synthèse" de l'image y à partir d'une configuration d'objets, et ainsi comparer la distribution des pixels à l'intérieur d'un rectangle u à ces différentes lois. Nous préférons cependant utiliser une méthode qui évite ce type d'hypothèse, car dans les images réelles les paramètres ne sont pas connus. Nous cherchons donc simplement un test d'homogénéité d'une zone de l'image, sans émettre d'hypothèse sur la nature des distributions des pixels de l'image.

L'homogénéité est une notion qui peut sembler intuitive, mais qui n'est pas aisée à formaliser. Si l'on pense à la façon dont nous percevons visuellement

l'homogénéité d'une image, il peut sembler naturel de dire qu'une zone est homogène si toutes ses sous-parties présentent le même aspect. Quand nous parlons de sous-partie, nous pensons à une sous-partie connexe, à un ensemble de pixels voisins et non disjoints. La nécessité de procéder à une subdivision de la zone pour tester son homogénéité apparaît sur la figure 10 : en haut la zone est considérée comme inhomogène car ses deux moitiés ont des radiométries différentes, en revanche, si l'on mélange aléatoirement les pixels, la zone apparaît homogène alors que la distribution globale est inchangée. Nous proposons donc un test simple : diviser le rectangle en $N > 2$ colonnes de même largeur et comparer les distributions de ces colonnes, puis procéder de même avec des subdivisions horizontales.

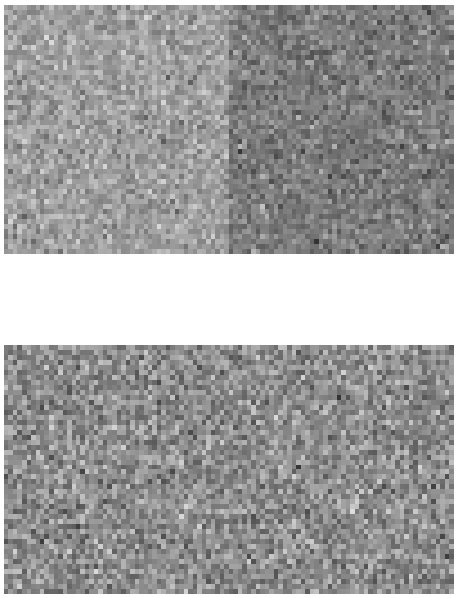

FIG. 10 – *La figure du bas a été obtenue en mélangeant aléatoirement les pixels de la figure du haut.*

Test de Kolmogorov-Smirnoff Ce test permet de mesurer l'écart entre deux distributions - voir par exemple [15], pages 490-494. Étant donnés deux

échantillons (x_i) et (y_i) sur $\mathbb{R}$ de cardinaux respectifs n_1 et n_2, on définit les deux distributions $S_{n_1}(t)$ et $S_{n_2}(t)$: $S_n(t)$ représente la fraction d'échantillons inférieurs à t. On calcule alors la statistique :

$$D = \max_{-\infty<t<\infty} | S_{n_1}(t) - S_{n_2}(t) | \tag{53}$$

Asymptotiquement en n_1 et n_2, la distance D entre les deux distributions suit une loi calculable explicitement. En effet, en posant :

$$Q_{KS}(\lambda) = 2 \sum_{j=1}^{\infty} (-1)^{j-1} e^{-2j^2\lambda^2} \tag{54}$$

alors dans le cas de l'hypothèse nulle où les deux distributions sont les mêmes, on a :

$$\text{Probabilité}\,(D > \alpha) = Q_{KS}(\sqrt{\frac{n_1 n_2}{n_1 + n_2}}\,\alpha) \tag{55}$$

Si l'expression 55 est inférieure à un seuil donné a (par exemple $a = 0.05$), les deux échantillons ont une probabilité inférieure à a de suivre la même distribution.

Le calcul du potentiel $q(u)$ consiste donc à effectuer $N-1$ tests de Kolmogoroff-Smirnoff sur N subdivisions verticales du rectangle u et autant de tests sur N subdivisions horizontales. Nous définissons alors le potentiel $q(u)$ comme la plus petite valeur résultant de ces tests. Plus le nombre de subdivisions N est élevé et plus le test est fiable géométriquement. En revanche, plus les sous régions sont grandes, plus les statistiques sont robustes. En pratique $N = 4$ représente un bon compromis.

Pour inciter les objets à se placer sur les contours des zones de l'image, il est intéressant de rajouter au potentiel $q(u)$ un terme favorisant les rectangles u dont les bords sont situés sur des limites de zones. Pour cela, on compare la distribution des pixels dans une étroite bande d'un côté et de l'autre d'un bord du rectangle u. Lorsque ces distributions sont différentes, c'est à dire lorsque le test de Kolmogoroff-Smirnoff renvoie une valeur inférieure à un seuil donné,

on augmente le potentiel $q(u)$ d'une quantité par exemple proportionnelle à la longueur du bord considéré. Ce terme supplémentaire améliore les résultats en pratique, et le positionnement des rectangles sur les contours est plus précis.

6.6 Algorithme de recuit

Pour résoudre l'équation 45, nous allons simuler la loi

$$f_T(x) \propto \left(\beta^{n(x)} \prod_{u_i \in x} q(u_i) \prod_{\{u_i,u_j\} \subset x} g(u_i,u_j) \right)^{\frac{1}{T}} \tag{56}$$

en abaissant progressivement la température T vers 0.

Pour effectuer les simulations, nous choisissons l'algorithme de Metropolis-Hastings-Green, mais cette fois-ci en ajoutant la possibilité de mouvements autres que des ajouts et suppressions d'objets à chaque étape. L'intéret de rajouter des mouvements est d'accélérer la convergence des simulations, mais également d'améliorer la fiabilité du recuit-simulé en limitant les risques que les simulations ne se bloquent dans des "puits" de la distribution dont on cherche l'optimum.

Puisque le taux d'acceptation du mouvement $x \to y$ fait intervenir le rapport :

$$\frac{Q(y,x)}{Q(x,y)} \tag{57}$$

il faut que pour chaque modification autorisée, la modification contraire soit également possible, sinon ce mouvement sera toujours rejeté. Nous donnons maintenant la liste des mouvements que nous avons utilisés et l'expression des taux d'acceptation. La longueur et la largeur des rectangles est comprise entre les valeurs $lmin$ et $lmax$, l'image est une grille de $X \times Y$ pixels. Pour plus de simplicité dans les formules des ratios, nous négligerons les problèmes pouvant survenir aux bords de l'image, bien que ceux-ci aient été pris en compte dans

l'implantation.

1. **Ajout d'un rectangle** (probabilité P_a) : puisque le but est de segmenter l'image, on souhaite que les objets recouvrent entièrement cette image. Afin d'accélérer le recouvrement, les naissances ne sont plus tirées de manière aléatoire, mais de sorte que les rectangles dont le centre est un point non encore recouvert soient privilégiés. Pour que ceci se fasse de manière efficace, on met à jour au cours de l'algorithme un tableau de la taille de l'image indiquant quels pixels sont couverts. Si on effectue le tirage de telle sorte qu'un point non recouvert soit choisi avec une probabilité a fois plus importante qu'un point recouvert, la probabilité de choisir un tel point est $a/(XY + (a-1)n_c(x))$, où $n_c(x)$ est le nombre de pixels non couverts dans l'image. Si le rectangle u choisi a un centre libre, le taux d'acceptation est donc :

$$r = \lambda^*(x,u)\,\frac{P_s}{P_a}\,\frac{1}{a}\,\frac{(XY + (a-1)n_c(x))(lmax - lmin)^2}{n(x)+1} \qquad (58)$$

   Si le rectangle a un centre recouvert, le taux doit être multiplié par a. Cette méthode permet effectivement de choisir plus rapidement des rectangles dans les zones non couvertes, mais il faut cependant noter que lorsque $n_c(x)$ est faible, le ratio est plus petit d'un facteur a que le taux correspondant à un tirage uniforme. La méthode est donc réellement intéressante lorsque les taux d'acceptation sont supérieurs à 1.
2. **Suppression d'un rectangle** (probabilité P_s) : les rectangles à supprimer sont toujours choisis de manière uniforme. Le ratio est :

$$r = \frac{1}{\lambda^*(x \setminus \{u\},u)}\,\frac{P_a}{P_s}\,\frac{a\,n(x)}{(XY + (a-1)n_c(x \setminus \{u\})(lmax - lmin)^2} \qquad (59)$$

   lorsque le centre du rectangle u n'est couvert par aucun autre objet que u (ce qui est pratiquement toujours le cas lorsque le potentiel a priori est suffisamment répulsif).
3. **Déplacement d'un rectangle** (probabilité P_m) : ce mouvement consiste à choisir une direction de déplacement (verticale ou horizontale) puis à

déplacer le rectangle d'une certaine distance, choisie de manière aléatoire. Le but de ce mouvement étant surtout d'ajuster le placement des objets sur les limites de zones et favoriser les regroupements d'objets, nous limitons la distance de déplacement à une valeur $mmax$, relativement faible. Le rapport 57 est ici égal à 1. Lorsque l'objet u est translaté en l'objet v, le taux d'acceptation est donc :

$$r = \frac{p(x \cup \{v\} \setminus \{u\})}{p(x)} = \frac{q(v)}{q(u)} \prod_{u_i \in x,\, u_i \neq u} \frac{g(v,u_i)}{g(u,u_i)} \tag{60}$$

4. **Dilatation d'un rectangle** (probabilité P_d) : pour ce mouvement, on choisit aléatoirement un côté d'un rectangle et, tout en maintenant les autres fixes, on le déplace perpendiculairement à lui-même d'une distance choisie aléatoirement entre $-dmax$ et $dmax$. Comme le précédent, ce mouvement permet d'accélérer les regroupements d'objets en colmatant d'éventuels interstices entre les rectangles. Le taux d'acceptation a également l'expression 60.
5. **Déplacement d'une frontière commune** (probabilité P_f) : si le potentiel g est trop attractif, deux objets collés risquent de ne plus pouvoir être déplacés au cours de l'algorithme de simulation, notamment à basse température. Il est donc intéressant d'autoriser le déplacement d'un bord commun à plusieurs objets, afin de rectifier d'éventuelles imprécisions sur le placement des objets. Nous avons implanté ce mouvement pour une frontière commune à deux objets. Les deux rectangles ayant un bord en commun et la distance de déplacement sont choisies de manière aléatoire et "uniforme". Le rapport 57 est toujours égal à 1 et le taux d'acceptation est donc de la forme :

$$r = \frac{p(x \cup \{v1,v2\} \setminus \{u1,u2\})}{p(x)} \tag{61}$$

6.7 Résultats et commentaires

Les résultats obtenus sont présentés sur la figure 11 pour des rapports signal à bruit de $14.7dB$ et $1.6dB$. A titre de comparaison, les résultats obtenus avec un modèle de Potts sont montrés sur la figure 12. Pour un bruit faible, le terme

de régularisation peut être réduit. Dès lors, le modèle de Potts donne d'excelents résultats, meilleur même que le processus Markov objets. En revanche, lorsque le bruit augmente, le terme de régularisation devient prépondérant. Suivant le choix des paramètres, le modèle de Potts a alors tendance, soit à sous-régulariser (présence de bruit résiduel), soit à sur-régulariser (déformation des contours). Le processus Markov objet est, quant à lui, beaucoup plus robuste et les résultats sont de même qualité pour les deux images. Le plan est bien pavé. Nous obtenons une légère sur-segmentation qui pourrait aisément être affiner par un post-traitement.

7 Conclusion

Les processus objets sont un outil très prometteur pour la segmentation car ils permettent une approche à la fois géométrique et stochastique de l'image. Ils constituent ainsi une modélisation plus globale que les méthodes classiques par champs de Markov. Nous avons vu que cette technique fonctionne correctement sur le modèle simple étudié dans ce rapport. Les premiers résultats semblent ainsi indiquer une bonne robustesse au bruit.

Il convient maintenant d'étendre le modèle proposé aux cas des images réelles. Les objets utilisés devront notamment permettre de décrire des formes géométriques plus complexes, telles que les formes des champs observées sur les images aériennes.

En outre, le choix des différents mouvements de l'algorithme d'optimisation afin de maximiser le taux d'acceptation reste un problème tout comme l'estimation des paramètres.

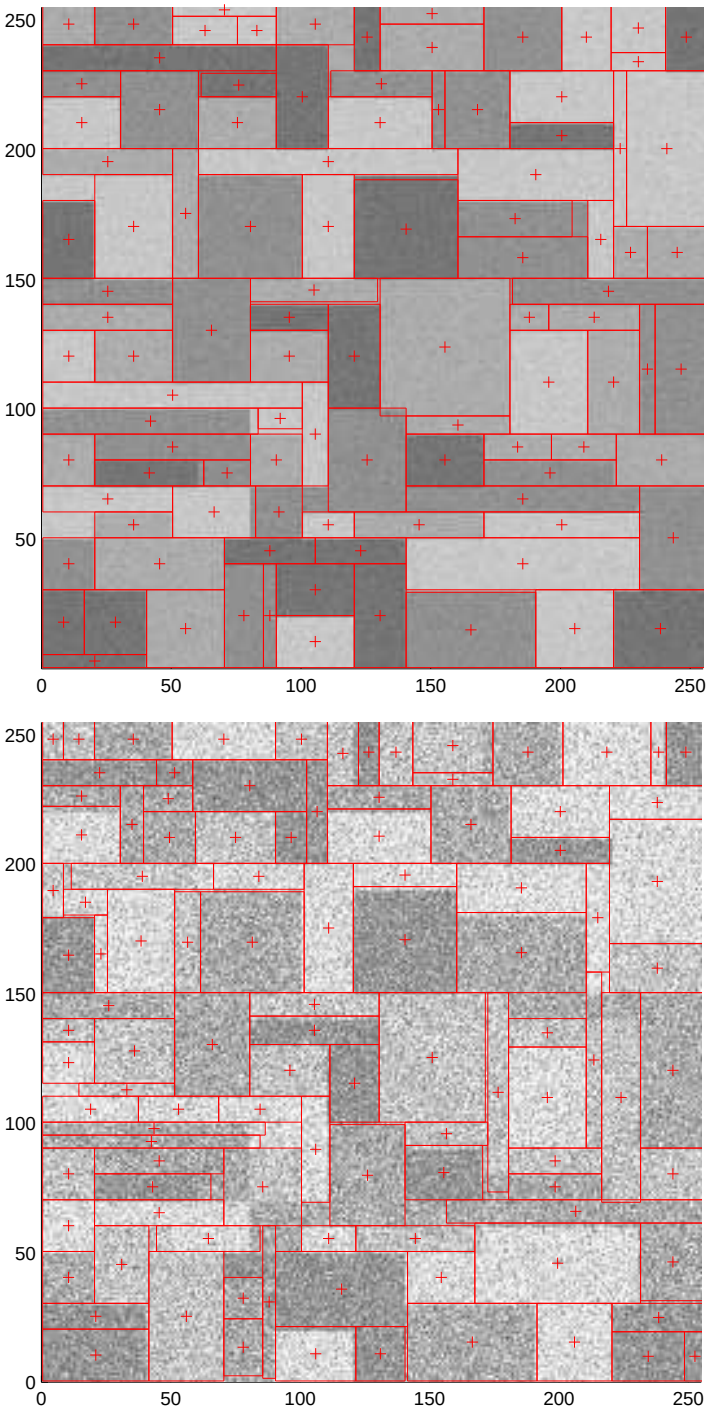

FIG. 11 – *Résultats avec le Processus Markov Objets défini. L'image du bas est plus bruitée ($SNR = 1.6dB$) que celle du haut ($SNR = 14.7dB$). Pour obtenir ces résultats, nous avons fait tourner le recuit-simulé 30 minutes pour la figure du haut et 1 heure pour celle du bas, sur une station UltraSparc 1 à 167MHz.*

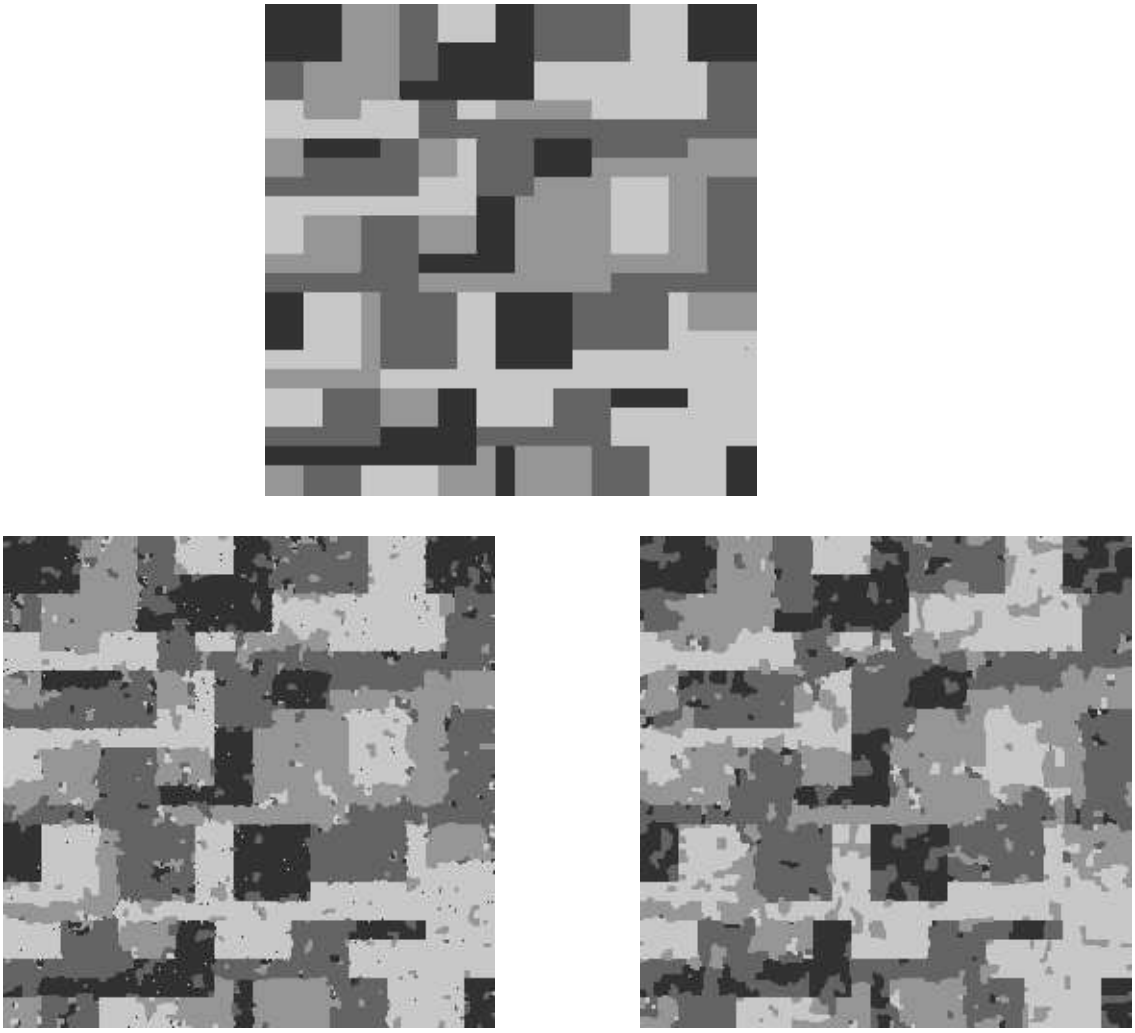

FIG. 12 – *Résultats obtenus avec un recuit simulé utilisant un modèle de Potts. L'énergie du modèle est :*

$$U(x) = \beta \sum_{<s,s'>} \delta_{x_s \neq x_{s'}} + \sum_s \sum_l \left(\frac{(y_s - \mu_l)^2}{2\sigma_l^2} + \frac{1}{2} log(2\pi\sigma_l^2) \right) \delta_{x_s = l}$$

où $< s,s' >$ désignent deux pixels s et s' voisins, x_s est le label proposé au pixel s. Les μ_l et σ_l sont les paramètres des lois gaussiennes associées à chacun des labels l, paramètres que l'on suppose donc ici connus.

Références

[1] Baddeley, A.J. and Møller, J. (1989). Nearest-neighbour Markov point processes and random sets, *International Statistic Review* **57**, 90-121.

[2] Baddeley, A.J. and van Lieshout, M.N.M (1992) Object recognition using Markov spatial processes. In: *Proceedings 11th IAPR International Conference on Pattern Recognition*, Volume B, 136-139, IEEE Computer Society Press, Los Alamitos.

[3] Baddeley, A.J., van Lieshout, M.N.M and Møller, J. (1996) Markov properties of cluster processes. *Advances in Applied Probability (SGSA)*, **28**, 346-355.

[4] Besag, J. (1974). Spatial interaction and the statistical analysis of lattice systems. *J. Royal Statist. Soc. (series B)*, **36**, 192-326.

[5] Chan, K.S. and Geyer, C.J. (1994) Discussion of the paper by Tierney. *Annals of Statistics*, **22**, 1747-1758.

[6] Clifford, P. and Nicholls, G. (1994) Comparison of birth-and-death and Metropolis-Hastings Markov chain Monte Carlo for the Strauss process, not published but available at : `http://matu1.math.auckland.ac.nz/~nicholls/linkfiles/papers.html`

[7] Cocquerez, J.-P. et Philipp, S. (1995). Analyse d'images : filtrage et segmentation. Editions Masson.

[8] Häggström, O., van Lieshout, M.N.M. and Møller, J. (November, 1996) Characterisation results and Markov chain Monte Carlo algorithms including exact simulation for some spatial point processes. Aalborg University, Denmark. 60D05, 60K35.

[9] Geman, S. and Geman, D. (1984) Stochastic relaxation, Gibbs distribution, and the bayesian restauration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI-6**(6), 721-741.

[10] Geyer, C.J and Møller, J. (1994) Simulation and likelihood inference for spatial point processes. *Scandinavian Journal of Statistics*, **21**, 359-373.

[11] Geyer, C. (1996), Likelihood Inference for Spatial Point Processes. *Research Report*, University of Minnesota, USA.

[12] Green, P.J. (1995) Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, **82**, 711-732.

[13] van Lieshout, M.N.M. (1993) Stochastic annealing for nearest-neighbour point processes with application to object recognition, CWI Report BS-R9306.

[14] Møller, J. (1998) Markov Chain Monte Carlo and spatial point processes. In: *Proceedings Séminaire Européen de Statistique, "Stochastic geometry, likelihood, and computation"*, London, Chapman and Hall.

[15] Press W., Flannery B., Teukolsky S., Vetterling W. (1988), Numerical Recipes in C, Cambridge University Press.

[16] Preston, C.J. (1977) Spatial birth-and-death processes. *Bulletin of the International Statistical Institute*, **46**, 371-391.

[17] Kelly, F.P and Ripley, B.D. (1976) On Strauss's model for clustering, *Biometrika*, **63**, 357-360.

[18] Ripley, B.D. and Kelly, F.P. (1977) Markov point processes. *J. Lond. Math. Soc.* **15**, 188-192.

[19] Robert, C. Méthodes de Monte Carlo par chaînes de Markov, *Economica*.

[20] Strauss, D.J. (1975) A model for clustering,*Biometrika*, **63**, 467-475.

[21] Tierney, L. (1994) Markov Chains for exploring posterior distributions (with discussion). *Annals of Statistics*, **22**, 1701-1762.

[22] Widom, B. and Rowlinson, J.S. (1970) A new model for the study of liquid-vapor phase transitions. *Journal of Chemical Physics*, **52**, 1670-1684.

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles - B.P. 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Lorraine : Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - B.P. 101 - 54602 Villers lès Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot St Martin (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 Le Chesnay Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, B.P. 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399