Université de Montréal

Détection non-supervisée de contours et localisation de formes à l'aide de modèles statistiques

par

François Destrempes

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Mémoire présenté à la Faculté des études supérieures
en vue de l'obtention du grade de
Maître ès sciences (M.Sc.) en Informatique

Avril 2002

Université de Montréal

Faculté des études supérieures

Ce mémoire intitulé

Détection non-supervisée de contours et localisation de formes à l'aide de modèles statistiques

présenté par :

François Destrempes

a été évalué par un jury composé des personnes suivantes:

(président-rapporteur)

Jean Meunier

(directeur de recherche)

Max Mignotte

(membre du jury)

Yoshua Bengio

Mémoire accepté le : 30 août 2002

Sommaire

Dans ce mémoire, nous présentons un modèle statistique du champ vectoriel du gradient des niveaux de gris, basé sur des résultats empiriques. Nous intégrons ce modèle statistique dans la définition de la vraisemblance d'un modèle markovien des contours. Notre modèle est adapté à une procédure ECI pour l'estimation des paramètres. La détection de contours est alors formulée comme la minimisation du champ de Gibbs de l'*a posteriori* d'un modèle simplifié et s'effectue à l'aide de l'algorithme de Viterbi.

Nous présentons également un modèle markovien des chemins dans une image basé sur le modèle statistique du gradient. On obtient alors des fonctions de coût qui permettent d'utiliser l'algorithme de Dijkstra, tout comme pour l'algorithme du live-wire, ou bien l'algorithme du jetstream, pour l'extraction semi-automatique de contours.

Finalement, on présente un modèle markovien des déformations de formes reposant toujours sur le modèle statistique du gradient, mais aussi sur un modèle probabiliste pour la réduction de la dimension. La localisation d'une forme dans une image peut alors être formulée comme la minimisation du champs de Gibbs de la posteriori. Nous présentons en détails l'algorithme d'optimisation stochastique utilisé pour effectuer cette tâche.

Mots-clés : traitement d'images, détection de contours, localisation de formes, estimation de paramètres, segmentation non-supervisée, champs de Markov, program-

mation dynamique, optimisation stochastique.

Abstract

In this Master Thesis, we present a statistical model for the gradient vector field of the gray level in images based on empirical results. Moreover, we present a global Markov model for contours in images that uses this statistical model for the likelihood. Our model is suitable for an ICE procedure for the estimation of the parameters. We then formulate the detection of contours as the minimization of the a posteriori Gibbs field of a simplified model ; we resort to the Viterbi algorithm to solve that problem.

Moreover, we present a Markov model for paths in images also based on the distribution law of the gradient vector field of the gray level. This yields cost functions that allow the use of Dijkstra algorithm as in the live-wire algorithm, or the jetstream algorithm, for the semi-automatic extraction of contours.

Finally, we present a Markov model for deformations of shapes based on the statistical model of the gradient, as well as a probabilistic model for reduction of dimension. The localization of a shape in an image is then viewed as the minimizing of the a posteriori Gibbs field. We present in details the stochastic algorithm used for solving this problem.

Keywords : image processing, detection of contours, localization of shapes, parameters estimation, unsupervised segmentation, Markov Random Field model, dynamic programming, stochastic optimization.

Table des matières

Liste des figures

Liste des tableaux

Liste des abréviations

ACPP	Analyse en composantes principales probabiliste
ASM	Active Shape Model
ECI	Estimation Conditionnelle Itérative
EM	Expectation Maximization
E/S	Exploration/Sélection
ICE	Iterative Conditional Estimation
ICM	Iterated Conditional Modes
IS	Importance Sampling
MAP	Maximum a Posteriori
MV	Maximum de Vraisemblance
RS	Recuit simulé
RSG	Recuit simulé généralisé
SEM	Stochastic Expectation Maximization
SVD	Singular Value Decomposition

Remerciements

Je ne saurais trop exprimer ma gratitude envers le département d'informatique et de recherche opérationnelle de l'Université de Montréal pour m'avoir permis un retour aux études dans ce domaine passionnant et exigeant qu'est l'informatique. En particulier, merci à M. Guy Lapalme pour m'avoir admis au programme du D.E.S.S., ainsi qu'à M. Jean Meunier pour m'avoir encouragé à poursuivre mes études au programme de M.Sc.

J'aimerais souligner le soutien constant de mon directeur de recherche, M. Max Mignotte, grâce à qui j'ai pu bénéficié de nombreuses discussions sur le traitement d'images. Je remercie également tous mes autres professeurs pour leur dévouement pédagogique et scientifique. En particulier, je tiens à remercier M. Yoshua Bengio pour m'avoir mentionner l'existence du modèle probabiliste de réduction de la dimension de Tipping et Bishop.

J'aimerais également remercier mon collègue d'études M. Mathieu Miller pour avoir clarifié la structure de données à utiliser pour l'algorithme de Dijkstra dans l'extraction semi-automatique de contours, ainsi que M. Jean-François Laliberté pour des discussions utiles sur la programmation.

Ce mémoire est dédié à mes parents, mes frères et soeurs, ainsi qu'à mes neveux et nièces.

Chapitre 1

Introduction

Un des buts du traitement d'image est de pouvoir détecter et localiser des objets dans une image afin de procéder ensuite à l'étape de classification.

La donnée brute de l'image, c'est-à-dire des niveaux de gris en chacun des pixels, constitue une somme formidable d'information souvent trop complexe pour être utilisée directement aux fins d'analyse. Une première tâche du traitement d'images consiste donc à organiser l'information en regroupant les pixels de façon cohérente. Il s'agit là de la segmentation de l'image en primitives autres que les pixels, c'est-à-dire des contours ou encore des régions. Les deux types de segmentation peuvent d'ailleurs être effectuées en parallèle.

Dans ce mémoire, nous traitons principalement de la segmentation d'une image en contours et en tirons quelques applications. On peut identifier les problèmes suivants reliés à la segmentation en contours.

Un premier problème porte sur la détection non-supervisée des contours. Il s'agit de produire tous les points de contours significatifs de l'image sans *a priori* sur les formes cherchées. Il est impératif que l'algorithme ne nécessite pas l'ajustement supervisé de paramètres internes.

Le deuxième problème porte sur l'extraction semi-automatique de contours dans

l'image. Une première solution consiste à produire une courbe de contour en ne spécifiant que les positions initiale et finale. Une deuxième solution consiste à prolonger une courbe de contour à partir de sa position et direction initiales.

Un troisième problème porte sur la localisation non-supervisée d'un contour dans l'image ressemblant le plus à une forme donnée, sans qu'il y ait interaction avec l'usager. En outre, on ne peut fournir une solution initiale près de la forme cherchée. Par contre, on sait d'avance que la forme se retrouve dans l'image.

Un quatrième problème porte sur la détection automatique d'une forme donnée dans l'image. Dans ce cas-ci, la forme cherchée peut ne pas se retrouver dans l'image et l'algorithme doit décider si c'est le cas ou non.

Nous proposons dans ce travail une solution aux trois premiers de ces problèmes. Notre solution repose sur un modèle statistique empirique de la distribution du champ vectoriel du gradient des niveaux de gris de l'image. Ce modèle nous permet de construire la vraisemblance d'un modèle Markovien des contours et des chemins dans une image. Par contre, les modèles *a priori* sont heuristiques (sauf pour la localisation de formes), mais toutefois basés sur des critères géométriques relativement simples. On peut ensuite formuler chacun des problèmes ci-dessus comme un problème de minimisation du champ de Gibbs de la distribution *a posteriori*.

Dans le chapitre 2, nous présentons une revue de littérature suffisamment détaillée pour illustrer la contribution des travaux précédents dans l'élaboration de notre projet, ainsi que l'originalité de notre propre contribution.

Au chapitre 3, nous présentons quelques algorithmes fondamentaux pour le traitement d'images. Le chapitre 4 présente deux algorithmes de programmation dynamique. Dans le chapitre 5, on présente l'algorithme d'optimisation stochastique de O. François, dont on se sert pour la localisation de formes. Au chapitre 6, on présente le modèle probabiliste de réduction de la dimension de Tipping et Bishop. Ces quatre chapitres sont de nature didactique.

Dans le chapitre 7, nous présentons un nouveau modèle markovien des contours,

basé sur un modèle statistique de la distribution du gradient des niveaux de gris d'une image, ainsi qu'une version simplifiée qui ignore les points de jonction. On en déduit une méthode de détection non-supervisée des contours, qui utilise l'algorithme de Viterbi pour optimiser la version simplifiée de notre modèle.

Dans le chapitre 8, nous présentons un nouveau modèle statistique des chemins discrets dans une image qui nous permet d'utiliser l'algorithme de Dijkstra dans une méthode d'extraction semi-automatique de contours à partir des positions initiales et finales. De plus, nous présentons un nouveau modèle pour les chemins continus et nous appliquons ce modèle à l'extraction semi-automatique de contours à partir des positions et directions initiales.

Dans le chapitre 9, nous présentons une nouvelle méthode de localisation non-supervisée de formes basée sur un modèle statistique des déformations d'un patron de forme dans une image.

Ces trois derniers chapitres constituent une contribution originale de l'auteur de ce mémoire au domaine du traitement d'images. Notre modèle présenté au chapitre 7 est un nouveau modèle statistique pour les contours ; notre modèle des chemins discrets et continus est également un nouveau modèle statistique présenté dans ce contexte ; finalement, notre modèle pour la localisation de formes est un nouveau modèle statistique utilisé pour résoudre ce problème. L'utilisation de modèles statistiques plutôt que de modèles heuristiques offre un avantage majeur, du moins en principe, puisque l'estimation de paramètres statistiques permet d'éviter l'ajustement manuel de paramètres heuristiques. Nos idées originales ont été exposées dans les actes de conférences internationales [21], [20] et [22], tous déjà acceptés pour fin de publication.

Finalement, nous terminons par une brève discussion et une conclusion au chapitre 10.

Les logiciels développés pour élaborer ce projet ont été programmés en C++ et comportent plus de 17,000 lignes de code.

Chapitre 2

Revue de littérature

Nous présentons dans ce chapitre les méthodes de détection de contours, d'extraction semi-automatique de contours, et de localisation de formes, antérieures à nos travaux. Comme pourra le constater le lecteur, les modèles présentés sont heuristiques, et comportent donc plusieurs ajustements manuels de paramètres. Dès lors, nous n'utilisons aucune de ces méthodes, mise à part l'algorithme d'alignement de formes de Cootes et al. Par contre, nous nous sommes inspirés de plusieurs idées générales sous-jacentes à ces méthodes.

2.1 Détection de contours

2.1.1 Détecteur de contours de Canny (1986)

Canny [7] a formulé un modèle des contours (contour en escalier) et de bruit (bruit gaussien blanc) dans le cas d'un signal unidimensionnel ou bi-dimensionnel (une image), ainsi que des critères mathématiques d'une bonne segmentation au sens des contours : bonne détection, bonne localisation, contrainte de l'unicité locale du point de contour détecté. Il démontre l'unicité d'un filtre linéaire optimal (le détecteur

de contours de Canny) au sens de ces critères. Cependant, on n'en connaît pas de forme analytique. Toutefois, Canny propose d'approximer ce filtre par la dérivée d'un filtre gaussien dont la variance dépend du bruit. Dans le cas bi-dimensionnel, une bonne approximation de ce filtre de détection consiste à appliquer le gradient sur l'image obtenue après lissage par un filtre gaussien. Ensuite, il s'agit de distinguer à l'aide de seuils les véritables points de contours du simple bruit grâce à un algorithme d'hystérésis.

Des variantes ont ensuite été proposées par Deriche [19], ainsi que Castan [8] et Shen [50], basées sur des critères légèrement différents. Toutefois, pour chacun de ces algorithmes, il faut ajuster judicieusement des paramètres internes selon l'image à traiter.

Nous reproduisons ci-dessous la version algorithmique bien connue du détecteur de contours de Canny.

2.1.2 Modèle markovien de Geman et al (1990)

Le modèle suivant pour la segmentation d'une image au sens des contours a été proposé par Geman et al [31].

Étant donné une image de taille $M \times N$, on considère le graphe $G = (V_G, E_G)$ formé des pixels de l'image et muni du système de voisinage habituel tel qu'illustré à la figure 2.1. On pose $G_B = (V_B, E_B)$ le graphe des bords formé du réseau des $M \times (N-1) + (M-1) \times N$ noeuds intercalés entre les sites de G.

Si s est un site de G_B (c'est-à-dire un bord), Y_s est la variable aléatoire qui représente le niveau de gris au pixel s^* adjacent situé au coin supérieur gauche, tandis que X_s représente l'étiquette en ce site et prend ses valeurs dans l'ensemble $\{0, 1\}$ ("hors-contours" ou "contours").

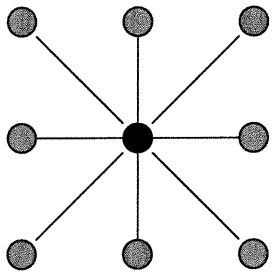


FIG. 2.1 – *Un pixel et ses 8 voisins.*

Algorithme de Canny

But de l'algorithme : détecter les courbes de contours d'une image.

1. **Calcul des descripteurs de contours** : Appliquer un filtre gaussien de variance σ^2 sur l'image. Calculer la norme y_s du gradient des niveaux de gris en chaque site s de l'image. Approximer la direction du gradient au multiple de $\pi/4$ le plus près.
2. **Suppression des non-maximaux (pré-segmentation)** : Ne considérer comme points de contours potentiels que les points pour lesquels la norme du gradient est un maximum local dans la direction du gradient.
3. **Hystérésis** : Fixer deux seuils $\tau_l \leq \tau_h$. Pour chaque site s de l'image pour lequel $y_s \geq \tau_h$, ajouter tous les points t obtenus à l'étape de la pré-segmentation qui sont connectés à s le long d'un chemin normal au champ vectoriel du gradient et pour lesquels $y_t \geq \tau_l$.

La vraisemblance d'un étiquetage s'exprime alors sous la forme

$$P_{Y \mid X}(y \mid x) = \frac{1}{Z} \exp\big(-U(x, y)\big)$$

où

$$U(x,y) = \sum_{<s,t>} (1 - x_s x_t)\phi(\Delta_{s,t}(y)).$$

La fonction ϕ est donnée par

$$\phi(x) = \begin{cases} -\left(\frac{x-d^*}{d^*}\right)^2 & 0 \leq x \leq d^* \\ +\left(\frac{x-d^*}{\alpha-d^*}\right)^2 & d^* < x \end{cases}$$

où d^* et α doivent être choisis judicieusement. De plus, on pose

$$\Delta_{s,t}(y) = \frac{|y_{s^*} - y_{t^*}|}{\gamma + \sum |y_{s_i^*} - y_{t_i^*}|}$$

avec γ déterminé empiriquement, et les $< s_i, t_i >$ pris dans le voisinage immédiat de $< s, t >$.

On considère également une fonction $V(x)$ qui donne une pénalité artificielle aux diverses configurations des bords ; par exemple, on privilégie des contours rectilignes. La distribution *a priori* s'exprime alors sous la forme

$$P_X(x) = \delta_{\Omega^*}(x)$$

où Ω^* est l'ensemble des réalisations x du champs des étiquettes qui minimisent la fonction V.

La distribution a posteriori que l'on obtient n'est plus markovienne (certaines probabilités s'annulent). Cependant, le recuit simulé et l'échantilloneur de Gibbs ont été développés rigoureusement dans ce contexte [30]. On parle de modèle "markovien sous contraintes" car la maximisation de la fonction de densité est équivalente à la minimisation du champ de Gibbs U sous la contrainte que la fonction V soit minimale. Par abus de langage, on dira simplement "modèle markovien".

2.2 Extraction semi-automatique de contours

2.2.1 Live-wire de Mortenson et al (1992)

L'algorithme suivant est tiré de Mortenson et al [46] et [45].

Étant donné une image, on considère le graphe non-orienté $G = (V_G, E_G)$ formé des pixels avec les voisinages formés des 8-voisins habituels. Si s est un site de G, y_s dénote la norme du gradient des niveaux de gris par rapport à la position (x, y) en coordonnées cartésiennes au pixel correspondant.

Soit $c = (s_0, s_1, ..., s_n)$ un chemin dans le graphe G dont le point de départ s_0 est fixé par l'usager. On considère une fonction de coût de la forme

$$l(s_1, s_2, ..., s_n) = \sum_{i=1}^{n} \lambda(s_{i-1}, s_i),$$

où

$$\lambda(s, t) = w_1(1 - \frac{y_t}{m}) + w_2\theta_{s,t}$$

avec $\theta_{s,t}$ la moyenne de la valeur absolue des angles (en radians) entre le gradient aux points s et t, et la normale au segment reliant s avec t (voir figure 2.2). La constante m est la valeur maximale de y_s sur l'image, tandis que w_1 et w_2 sont des poids à établir empiriquement.

Si t est un site arbitraire de l'image, on cherche le chemin de moindre coût reliant s_0 à t. L'algorithme de Dijkstra [23] peut être utilisé directement pour résoudre ce problème. Voir les sections 4.1 et 8.1.

2.2.2 Jetstream de Pérez et al (2001)

Pérez et al [47] présentent un nouveau point de vue pour l'extraction semi-automatique de contours.

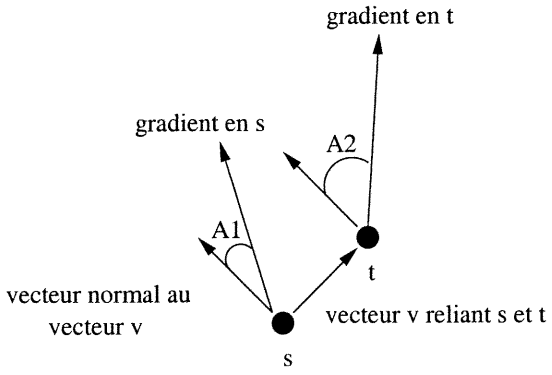


FIG. 2.2 – *Illustration de l'angle $\theta_{s,t}$: on pose $\theta_{s,t} = (|A_1| + |A_2|)/2$.*

Étant donné une image de taille $M \times N$, on considère le graphe infini indénombrable $G = (\Lambda, E)$, où $\Lambda = [0, M] \times [0, N]$ et $(x_1, x_2) \in E$ si et seulement si $\| x_1 - x_2 \| = 1$. Si $x \in \Lambda$, le gradient des niveaux de gris en x est obtenue par interpolation bilinéaire du gradient aux points de Λ à coordonnées entières, c'est-à-dire correspondant aux pixels de l'image. Soit $x_{0:n} = (x_0, x_1, ..., x_n)$ un chemin dans le graphe G dont le point de départ x_0 est fixé par l'usager.

Si $s \notin x_{0:n}$, y_s représente la norme du gradient des niveaux de gris en s. Si $s = x_i$, y_s représente la racine carrée de la norme du gradient (des niveaux de gris) en s multipliée par l'angle ψ entre ce gradient et le vecteur normal au segment (x_{i-1}, x_i). Notez que d'utiliser pour la définition de y_s, la norme du gradient dans un cas et l'angle ψ dans l'autre cas, est une faiblesse de ce modèle. On modélise la distribution p_{off} de la norme du gradient η par une loi exponentielle

$$p_{\text{off}}(\eta) = \frac{1}{\lambda} \exp(-\frac{\eta}{\lambda})$$

qui est estimée au sens du maximum de vraisemblance par la moyenne de la norme du gradient sur l'image. Par ailleurs, on modélise la distribution p_{on} de l'angle ψ par une loi gaussienne

$$p_{\text{on}}(\psi \mid x_{0:n}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{\psi^2}{2\sigma^2})$$

où σ est fixé à 1. Notez qu'on ne devrait pas modéliser la distribution d'un angle appartenant à un intervalle borné, par une loi gaussienne. On considère alors la fonction

de vraisemblance donnée par

$$\begin{aligned} P(y \mid x_{0:n}) &= \prod_{s \notin x_{0:n}} p_{\text{off}}(y_s) \prod_{i=0}^{n} p_{\text{on}}(y_{x_i} \mid x_{0:n}) \\ &\propto \prod_{i=0}^{n} \frac{p_{\text{on}}(y_{x_i} \mid x_{0:n})}{p_{\text{off}}(y_{x_i})}. \end{aligned}$$

On considère également une distribution *a priori* pour les chemins *continus* de la forme

$$p(x_{0:n}) = p(x_{0:1}) \prod_{i=2}^{n} q(x_i; x_{i-2:i-1}).$$

On pose $x_i = x_{i-1} + R(\theta_i)(x_{i-1} - x_{i-2})$ où $R(\theta_i)$ dénote la rotation par l'angle θ_i obéissant à une fonction de densité donnée par

$$\rho(\theta_i) = \frac{\nu}{\pi} + (1 - \nu) N(\theta_i; 0, d\sigma_\theta^2).$$

Rappelons que x_i est un élément de l'ensemble $\Lambda = [0, M] \times [0, N]$. On prend $\nu = 0.01$, $d = 1$ et $\sigma_\theta = 0.1$. Notez encore une fois que de modéliser la distribution d'un angle par une loi gaussienne est une faiblesse de ce modèle.

On obtient ainsi une distribution *a posteriori* qui satisfait la formule récursive

$$p_{i+1}(x_{0:i+1} \mid y) \propto p_i(x_{0:i} \mid y) q(x_{i+1} \mid x_{i-1:i}) l(y_{x_{i+1}} \mid x_{0:n+1})$$

où $l = \frac{p_{\text{on}}}{p_{\text{off}}}$.

On peut alors utiliser une méthode de simulation pour étendre le germe s_0 à un contour vraisemblable passant par ce germe. Il s'agit de l'algorithme du jetstream de Pérez et al. Cet algorithme s'inspire du principe de l'IS que l'on rappelle maintenant. Soient $g_1(x), g_2(x)$ deux distributions ayant même support, et $h(x)$ une fonction mesurable. Soient $x_1, x_2, ..., x_m$ un échantillon de variables aléatoires $X_1, ..., X_m$ i.i.d. de loi $g_2(x)$. Alors,

$$\int h(x) g_1(x)\, dx \approx \frac{1}{m} \sum_{i=1}^{m} h(x_i) \frac{g_1(x_i)}{g_2(x_i)}.$$

Dans le cas du jetstream, les auteurs suggèrent de ré-échantillonner $x_1, ..., x_m$ selon les poids $g_1(x_1)/g_2(x_1), ..., g_1(x_m)/g_2(x_m)$, afin d'obtenir un échantillon de loi $g_1(x)$. Dans notre cas, on considère

$$\begin{aligned} g_1(x_{0:n+1}) &= p_{n+1}(x_{0:n+1} \mid y), \\ g_2(x_{0:n+1}) &= p_n(x_{0:n} \mid y) f(x_{n+1} \mid x_{n-1:n}), \end{aligned}$$

où f est une distribution positive quelconque. Les auteurs proposent de prendre pour $f(x_{n+1} \mid x_{n-1:n})$ une légère modification de $q(x_{n+1} \mid x_{n-1:n})$ qui tienne compte des points de jonction. On peut obtenir un échantillon de la distribution p_{n+1} à partir d'un échantillon $x_{0:n}^{(1)}, x_{0:n}^{(2)}, ..., x_{0:n}^{(m)}$ de loi p_n, en prolongeant chaque chemin d'un point supplémentaire par une simulation de $f(x_{n+1} \mid x_{n-1:n}^{(i)})$, pour $i = 1, ..., m$. Ensuite, il suffit de ré-échantillonner l'ensemble des m chemins obtenus selon les poids

$$\pi_{n+1}^{(i)} = \frac{p_{n+1}(x_{0:n+1}^{(i)} \mid y)}{p_n(x_{0:n}^{(i)} \mid y) f(x_{n+1}^{(i)} \mid x_{n-1:n}^{(i)})} \propto \frac{q(x_{n+1}^{(i)} \mid x_{n-1:n}^{(i)}) l(y_{x_{n+1}}^{(i)} \mid x_{0:n+1}^{(i)})}{f(x_{n+1}^{(i)} \mid x_{n-1:n}^{(i)})}.$$

On obtient ainsi de façon itérative un échantillon de chemins de longueur n à partir d'un échantillon de chemins de longueur 0. Il suffit ensuite de prendre le chemin de probabilité maximale ou encore le chemin moyen.

2.3 Localisation de formes

2.3.1 Modèle de Jain et al (1996)

L'approche suivante pour la localisation de formes dans une image a été formulée par Jain et al [38]. Elle a été reprise par la suite avec quelques modifications par Jain [39], Dubuisson [24], et Mignotte [43].

Soit γ_0 un patron de forme. On considère des déformations γ_θ de γ_0, où θ prend ses valeurs dans un espace compact. On associe à cet espace de déformations la variable aléatoire Θ avec fonction de densité P_Θ. Cette fonction de densité devrait

être estimée préalablement dans une phase d'apprentissage, mais les auteurs en donne une définition *ad hoc*.

Algorithme Jetstream

But de l'algorithme : extraire une courbe de contours en spécifiant les position et direction initiales.

1. **Initialisation** : $k = 0$. On pose $x_{0:0}^{(i)} = x$, pour $i = 1, 2, ..., m$, x étant la position initiale.
2. **Simulation** : Pour chaque $i = 1, 2, ...m$, prolonger $x_{0:k}^{(i)}$ d'un point en simulant $f(x_{k+1} \mid x_{k-1:k}^{(i)})$. Si $k = 1$, utiliser la direction initiale v.
3. **Calcul des poids** : Pour chaque $i = 1, 2, ...m$, poser

$$\pi_{k+1}^{(i)} = \frac{q(x_{n+1}^{(i)} \mid x_{n-1:n}^{(i)}) l(y_{x_{n+1}}^{(i)} \mid x_{0:n+1}^{(i)})}{f(x_{n+1}^{(i)} \mid x_{n-1:n}^{(i)})}$$

   et normaliser afin d'obtenir $\sum_{i=1}^{m} \pi_{k+1}^{(i)} = 1$.
4. **Échantillonnage** : Ré-échantillonner $x_{0:k+1}^{(1)}, ..., x_{0:k+1}^{(m)}$ selon les poids $\pi_{k+1}^{(i)}$.
5. Augmenter k de 1, et retour à 2 tant que k est plus petit que la longueur désirée.

On considère la variable aléatoire Y dont les valeurs sont des données observables y reliées à l'image. Il n'est pas nécessaire de spécifier la fonction de densité P_Y puisque l'image est fixée. On considère également une fonction de vraisemblance $P_{Y \mid \Theta}(y \mid \theta) = \frac{1}{Z} \exp(-\epsilon_l(\theta, y))$ qui tient compte des contours, textures, etc., et qui est définie de façon heuristique. On obtient alors la fonction de densité a posteriori

$$P_{\Theta \mid Y}(\theta \mid y) \propto P_{Y \mid \Theta}(y \mid \theta) P_{\Theta}(\theta)$$

qui peut s'écrire sous la forme

$$\frac{1}{Z}\exp(-\epsilon(\theta, y)).$$

Il s'agit ensuite de maximiser cette fonction de densité, ce qui revient à minimiser la fonction d'énergie correspondante.

2.3.1.1 Déformations et distribution a priori

On pose $\theta = (\tau_x, \tau_y, s, \psi, \xi)$ avec (τ_x, τ_y) le vecteur de translation, s le facteur d'homothétie, ψ l'angle de rotation, et $\xi = (\xi^x_{mn}, \xi^y_{mn};\ 1 \leq m \leq M,\ 1 \leq n \leq N)$, le vecteur des déformations non-linéaires.

On pose

$$D_\xi(x, y) = \sum_{m=1}^{M}\sum_{m=1}^{N} \frac{\xi^x_{mn} e^x_{mn}(x, y) + \xi^y_{mn} e^y_{mn}(x, y)}{\lambda_{mn}}$$

où $\lambda_{mn} = \pi^2(m^2 + n^2)$, avec

$$e^x_{mn}(x, y) = (2\sin(\pi n x)\cos(\pi m y), 0)$$

$$e^y_{mn}(x, y) = (0, 2\cos(\pi n x)\sin(\pi m y)).$$

L'espace des transformations $D(x, y)$ telles que $(x, y) \mapsto (x, y) + D(x, y)$ est une transformation lisse de $[0, 1]^2$ dans lui-même, a pour base orthogonale $\{e^x_{mn}, e^y_{mn}; m, n \geq 1\}$.

Une courbe est représentée par une suite

$$\gamma = (x_1, y_1, ..., x_d, y_d)$$

où les d points $(x_1, y_1), ..., (x_d, y_d)$ sont équidistants. Si γ_0 est un patron de forme (c'est-à-dire, une courbe), on définit γ_θ par

$$\gamma_\theta = T_{\tau_x, \tau_y, s, \psi}((Id + D_\xi)(\gamma_0))$$

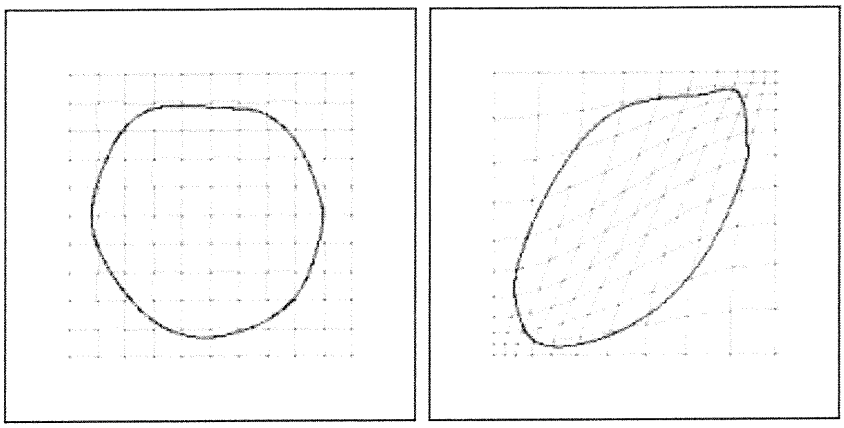


FIG. 2.3 – *Une forme ainsi que sa déformation pour les valeurs* $M = N = 1$, $\xi^x_{1,1} = \xi^y_{1,1} = 2$.

où

$$T_{\tau_x,\tau_y,s,\psi}(x,y) = s(x\cos(\psi) - y\sin(\psi) + \tau_x, x\sin(\psi) + y\cos(\psi)y + \tau_y).$$

Les auteurs donnent en fait une formulation légèrement différente pour γ_θ, mais qui est équivalente à toute fin pratique.

On suppose $\tau_x, \tau_y, s, \psi, \xi^x_{mn}, \xi^y_{mn}$ indépendants, avec τ_x, τ_y, s, ψ uniformes, et ξ^x_{mn}, ξ^y_{mn} gaussiens. D'où

$$P_\Theta(\theta) = \frac{1}{Z}\exp(\frac{-\parallel \xi \parallel^2}{2\sigma^2})$$

où Z est une constante de normalisation et σ^2 est à déterminer par essais et erreurs pour chaque image traitée.

2.3.1.2 Vraisemblance

On pose de façon heuristique

$$P_{Y\mid\Theta}(y \mid \theta) = \frac{1}{Z}\exp(-\epsilon_l(\theta, y))$$

avec

$$\epsilon_l(\theta, y) = \frac{1}{N_{\gamma_\theta}}\sum_{z\in\gamma_\theta}(1 - \exp(-c \parallel \delta(z) \parallel)|\cos\beta(z)|)$$

où $\delta(z)$ est la distance de z au point de contour de l'image y le plus près, et $\beta(z)$ est l'angle entre le vecteur tangent de γ_θ en z et le vecteur tangent au point de contour de y le plus près (voir Figure 2.4), et c un facteur de pondération ajusté manuellement pour chaque image traitée. Ici, le contour est obtenu à l'aide du détecteur de Canny, ce qui requiert l'ajustement manuel de seuils. Comme on le constate, la méthode de Jain et al est loin d'être automatique.

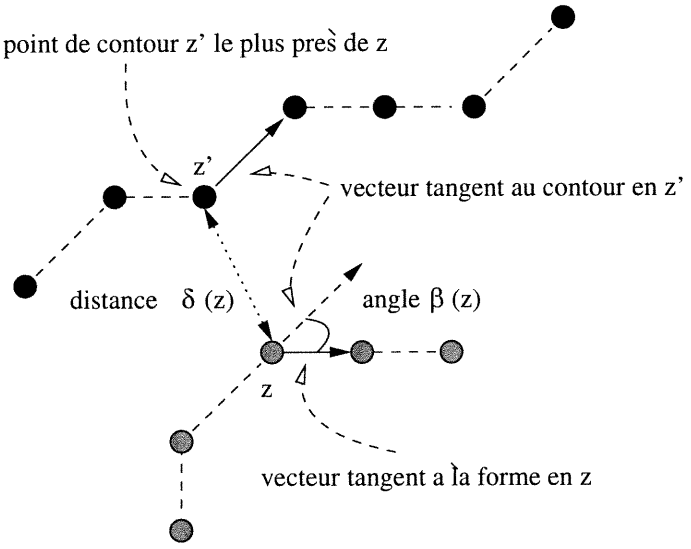


FIG. 2.4 – *Illustration de la distance $\delta(z)$ et de l'angle $\beta(z)$.*

2.3.1.3 Optimisation

Les auteurs utilisent une procédure de multi-résolution en trois étapes avec un algorithme de descente du gradient déterministe à chaque étape. En pratique, il faut fournir une initialisation de la solution suffisamment près de la solution cherchée.

2.3.2 Mignotte et al (2001)

Des modèles pour la localisation de formes basés sur les textures plutôt que les contours ont été proposés par Jain [56], Liu [40] et Mignotte et al [44]. Nous ne

présentons que celui de Mignotte et al, car il est le seul à comporter une estimation des paramètres statistiques. Il s'agit là d'un avantage majeur, du moins en principe, puisque l'estimation de paramètres statistiques permet d'éviter l'ajustement manuel de paramètres heuristiques.

2.3.2.1 Déformations et distribution a priori

Il s'agit du même modèle que celui présenté à la section 2.3.1.1.

2.3.2.2 Vraisemblance

Dans le cas de la recherche du contour endocardial dans une séquence d'images échocardiographique, on considère deux classes e_1 et e_2 correspondant aux textures "sang" et "muscle du coeur". Étant donné un patron de forme γ_θ, la région intérieure est dénotée par $\gamma_\theta^\bullet$ et la région extérieure par γ_θ°. On cherche le contour pour lequel la région intérieure coïncide avec la texture "sang" et la région extérieure avec la texture "muscle".

On considère la fonction de vraisemblance

$$P_{Y \mid \Theta}(y \mid \theta) = \frac{1}{Z} \exp(-\epsilon_l(\theta, y))$$

où

$$\epsilon_l(\theta, y) = -\frac{1}{N_{\gamma_\theta^\bullet}} \sum_{s \in \gamma_\theta^\bullet} \ln P_{Y_s \mid X_s}(y_s \mid e_1) - \frac{1}{N_{\gamma_\theta^\circ}} \sum_{s \in \gamma_\theta^\circ} \ln P_{Y_s \mid X_s}(y_s \mid e_2)$$

où s est un site de l'image, y_s est le niveau de gris du pixel au site s, et x_s est la classe correspondante. Notez qu'on ne devrait pas normaliser ainsi le log vraisemblance.

On utilise dans ce cas-ci le modèle de Rayleigh

$$P_{Y_s \mid X_s}(y_s \mid e_i) = \mathcal{R}(y_s; \min_i, \alpha_i) = \frac{(y_s - \min_i)}{\alpha_i^2} \exp(-\frac{(y_s - \min_i)^2}{2\alpha_i^2})$$

avec $y_s > \min_i$ et $\alpha_i > 0$, où le vecteur de paramètres $(\min_i, \alpha_i)$ dépend de la classe e_i.

2.3.2.3 Estimation des paramètres

Si $(y_1, ..., y_l)$ est un échantillon d'une Raleigh, on utilise les estimateurs non-biaisés du maximum de vraisemblance

$$\hat{\min} = \min_i y_i - 1$$

$$\hat{\alpha}^2 = \frac{1}{2l} \sum_{i=1}^{l} (y_i - \hat{\min}).$$

Notez que les valeurs estimées ne seront utilisées que pour l'image considérée, et non pour d'autres images. Donc, on est assuré que $y_s > \hat{\min}$.

Pour la distribution *a priori* d'un étiquetage x de l'image, on considère le modèle isotropique de Potts avec 8-connectivité

$$P_X(x) = \frac{1}{Z} \exp\Big(- \sum_{<s,t>} \beta_{st}(1 - \delta(x_s, x_t))\Big)$$

où $< s, t >$ parcourt l'ensemble des quatre paires de voisins selon les deux diagonales et les droites horizontale et verticale, et Z est une constante de normalisation. On prend $\beta_{st} = 1$ afin de favoriser des régions homogènes sans privilégier de directions.

On obtient alors la fonction de densité a posteriori

$$\begin{aligned} P_{X \mid Y,\Phi}(x \mid y, \Phi) &= P_{Y \mid X,\Phi}(y \mid x, \Phi) P_X(x) / P_Y(y) \\ &\propto \Big(\prod_{j=1}^{2} \prod_{s:x_s=e_j} \mathcal{R}(y_s; \min_j, \alpha_j)\Big) P_X(x). \end{aligned}$$

Pour chaque classe, le vecteur des paramètres $\Phi = (\min_i, \alpha_i : i = 1, 2)$ est estimé grâce à une procédure de type ECI [6], [48] et [49] que l'on rappelle au chapitre 3.

2.3.2.4 Optimisation

Les auteurs proposent un algorithme génétique avec stratégie de préservation de l'élite ou avec stratégie hybride.

2.3.3 Modèle de Cootes et al (2000)

Le modèle de Cootes [15] et [16] constitue une première tentative de modélisation statistique des déformations d'une forme pouvant être utilisée pour définir la distribution *a priori*.

2.3.3.1 Déformations et distribution a priori

Soient (τ_x, τ_y) un vecteur de translation, s un facteur d'homothétie, et ψ un angle de rotation. On définit

$$T_{\tau_x,\tau_y,s,\psi}(x,y) = s(x\cos(\psi) - y\sin(\psi) + \tau_x, x\sin(\psi) + y\cos(\psi)y + \tau_y).$$

On suppose donné un échantillon de formes (c'est-à-dire de courbes) $\{\gamma_1, ..., \gamma_l\}$ obtenu de façon semi-automatique. Une courbe est représentée par une suite

$$\gamma = (x_1, y_1, ..., x_d, y_d)$$

où les d points $(x_1, y_1), ..., (x_d, y_d)$ sont équidistants. On aligne ces formes sur un patron de forme moyen γ_0.

On applique ensuite l'analyse en composantes principales à l'échantillon des formes alignées dans l'espace vectoriel de dimension $2d$. Soit U_q la matrice des vecteurs propres de la matrice de covariance correspondants aux q plus grandes valeurs propres. On définit

$$T_\theta(\gamma_0) = T_{\tau_x,\tau_y,s,\psi}(U_q\xi + \gamma_0)$$

où $\theta = (\tau_x, \tau_y, s, \phi, \xi)$, ξ étant un vecteur dans l'espace vectoriel de dimension q. On suppose $\tau_x, \tau_y, s, \phi, \xi$ indépendants et τ_x, τ_y, s, ϕ uniformes. Donc, $P_\Theta(\theta) \propto P_\Xi(\xi)$. La fonction de densité de Ξ est estimée à l'aide d'un mélange de noyaux gaussiens. Cette méthode n'est pas optimale au sens du maximum de vraisemblance. Il faut plutôt utiliser le modèle probabiliste de réduction de la dimension de Tipping et al [51].

2.3.3.2 Vraisemblance

On considère la fonction heuristique

$$\epsilon_l(\theta, y) = \sum_{z \in \gamma_\theta} \delta(z)^2$$

où $\delta(z)$ est la distance de z au point de contour de l'image le plus près.

2.3.3.3 Distribution a posteriori

Les auteurs ne travaillent pas directement avec $P_{\Theta/Y}$. Leur approche est équivalente à considérer la fonction

$$\epsilon(\theta, y) = \epsilon_l(\theta, y) + \epsilon_a(\theta)$$

où

$$\epsilon_a(\theta) = \begin{cases} \infty & \text{si } P_\Xi(\xi) < P_0 \\ 0 & \text{sinon} \end{cases}$$

avec P_0 un seuil fixé.

2.3.3.4 Optimisation

L'algorithme proposé (ASM) nécessite une bonne initialisation de θ et est résumé ci-dessous.

Algorithme pour obtenir un patron de forme moyen

$\gamma_1, \gamma_2, ...\gamma_l$	échantillon de courbes comportant un même nombre de points 2D
γ_0	forme moyenne à l'itération précédente
$\bar{\gamma}$	forme moyenne à l'itération courante

But de l'algorithme : Aligner les courbes $\gamma_1, \gamma_2, ...\gamma_l$ sur une forme moyenne $\bar{\gamma}$.

1. **Initialisation** : Ajuster chaque forme γ_i pour que le centre géométrique soit $(0,0)$. Poser $\bar{\gamma} = \gamma_1/|\gamma_1|$, et $\gamma_0 = \bar{\gamma}$.
2. **Calcul récursif** : répéter jusqu'à convergence de $\bar{\gamma}$.
   (a) Aligner chaque γ_i avec γ_0 ; c'est-à-dire, trouver les valeurs de s et ψ qui minimisent $|T_{0,0,s,\psi}(\gamma_i) - \gamma_0|$ (voir l'algorithme d'alignement), et remplacer γ_i par $T_{0,0,s,\psi}(\gamma_i)$. Normaliser chaque γ_i ; c'est-à-dire, remplacer γ_i par $\gamma_i/|\gamma_i|$.
   (b) Calculer le patron de forme moyen $\bar{\gamma}$ des γ_i obtenus.
   (c) Aligner $\bar{\gamma}$ avec γ_0. Normaliser $\bar{\gamma}$. Poser $\gamma_0 = \bar{\gamma}$.

Algorithme d'alignement

x, x'	deux ensembles comportant un même nombre de points 2D dont le centre géométrique est en $(0, 0)$
s	facteur d'homothétie
ψ	angle de rotation (en radians)

But de l'algorithme : Trouver les valeurs de s et ψ qui minimisent $|T_{0,0,s,\psi}(x) - x'|$.

1. Calculer

$$a = (\sum_{i=1}^{d}(x_i x'_i + y_i y'_i))/|x|^2$$

$$b = (\sum_{i=1}^{d}(x_i y'_i - y_i x'_i))/|x|^2$$

où

$$x = (x_1, y_1, ..., x_d, y_d)$$

$$x' = (x'_1, y'_1, ..., x'_d, y'_d).$$

2. Poser

$$s = \sqrt{a^2 + b^2}$$

$$\psi = \tan^{-1}(b/a).$$

Algorithme ASM

γ_0	patron de forme moyen
k	itération
$\theta^{[k]}$	vecteur des paramètres de déformation à l'itération k

But de l'algorithme : Trouver la valeur de θ qui minimise $\epsilon(\theta, y)$.

1. **Initialisation** : $k = 0$. $\theta^{[0]}$ est une initialisation de θ.
2. **Calcul récursif** : répéter jusqu'à convergence de $\theta^{[k]}$:
   (a) Soit $\gamma_{\theta^{[k]}}$ la déformation courante du patron de forme. Trouver pour chaque point z de $\gamma_{\theta^{[k]}}$ le point de contour de l'image y le plus près. On obtient une courbe γ'.
   (b) Trouver $\theta^{[k+1]}$ qui minimise $|T_{\theta^{[k+1]}}(\gamma_0) - \gamma'|$ (voir l'algorithme d'ajustement).
   (c) $k = k + 1$.

Algorithme d'ajustement

γ_0	patron de forme moyen
γ'	courbe quelconque dans le plan
k	itération
$\tau_x^{[k]}, \tau_y^{[k]}, s^{[k]}, \psi^{[k]}, \xi^{[k]}$	valeurs des paramètres de déformation à l'itération k

But de l'algorithme : Trouver les valeurs de τ_x, τ_y, s, ψ et ξ qui minimisent $|T_{\tau_x,\tau_y,s,\psi}(U_q\xi + \gamma_0) - \gamma'|$.

1. **Initialisation** : $k = 0$; poser $\tau_x^{[0]}, \tau_y^{[0]}, s^{[0]}, \psi^{[0]}, \xi^{[0]} = 0$.
2. **Calcul récursif** : répéter jusqu'à convergence de $\tau_x^{[k]}, \tau_y^{[k]}, s^{[k]}, \psi^{[k]}, \xi^{[k]}$:
   (a) Poser $\gamma = U_q\xi^{[k]} + \gamma_0$.
   (b) Trouver $\tau_x^{[k]}, \tau_y^{[k]}, s^{[k]}, \phi^{[k]}$ qui minimisent $|T_{\tau_x^{[k]},\tau_y^{[k]},s^{[k]},\phi^{[k]}}(\gamma) - \gamma'|$ (voir l'algorithme d'alignement).
   (c) Remplacer γ' par $T^{-1}_{\tau_x^{[k]},\tau_y^{[k]},s^{[k]},\phi^{[k]}}(\gamma')$.
   (d) Normaliser γ' (c'est-à-dire, diviser par la norme euclidienne).
   (e) Poser $\xi^{[k+1]} = U_q^T(\gamma' - \gamma_0)$.
   (f) Si $P_\Xi(\xi^{[k+1]}) < P_0$, ajuster $\xi^{[k+1]}$ pour que $P_\Xi(\xi^{[k+1]}) \geq P_0$ en remontant le long du gradient.
   (g) $k = k + 1$.

2.4 Synthèse des idées

Comme l'a constaté le lecteur, chacun des modèles mentionnés ci-dessus, présente la faille majeure de requérir l'ajustement manuel de seuils. De plus, aucun de ces modèles n'est cohérent d'un point de vue statistique. Toutefois, on y retrouve plusieurs

idées générales intéressantes que nous avons su exploiter dans nos travaux.

Tout comme pour le détecteur de contours de Canny [7], il est naturel de considérer le champs vectoriel du gradient des niveaux de gris. Cependant, pour éviter une supervision dans le choix des seuils, il faut procéder comme pour le Jetstream de Pérez et al [47], c'est-à-dire modéliser la distribution de ce champs vectoriel. Par contre, plutôt que d'estimer une distribution moyenne sur un ensemble d'images, il est préférable d'effectuer une estimation pour chaque image, comme c'est le cas dans les travaux de Mignotte et al [44]. De plus, contrairement au modèle de Pérez et al [47], on évite d'utiliser une variable aléatoire pour les points hors contours et une toute autre pour les points de contours.

Il s'agit ensuite d'intégrer les distributions empiriques du gradient dans la définition de la vraisemblance d'un modèle markovien, qui n'est donc plus basé sur des fonctions heuristiques, assez arbitraires, comme c'est le cas dans l'article de Geman et al [31]. De plus, pour l'optimisation, il est contre-indiqué d'utiliser le recuit simulé sous contraintes, dont la convergence est beaucoup trop lente. En fait, par construction, une version simplifiée et satisfaisante de notre modèle, qui ignore les points de jonction, permet d'utiliser l'algorithme de Viterbi [25].

Pour l'extraction semi-automatique de contours, nous avons recours à l'algorithme de Dijkstra [23] pour trouver le chemin de moindre coût, tout comme dans l'algorithme de Mortenson [46] et [45], mais non plus avec des fonctions heuristiques, mais bien des fonctions découlant directement d'un modèle statistique cohérent. On peut appliquer le même modèle à la méthode du Jetstream de Pérez et al [47].

Pour la localisation de formes, nous nous inspirons du modèle de Jain [38], tout comme dans [39], [24] et [43], mais avec une vraisemblance qui comporte un véritable sens statistique, et une distribution *a priori* basée sur le modèle probabiliste de réduction de dimension de Tipping et Bishop [51]. Du modèle de Cootes et al [15], on ne retiendra que l'algorithme d'alignement des formes sur un patron de forme moyen. Finalement, la minimisation du champ de Gibbs de la distribution *a poste-*

riori s'effectue à l'aide de l'algorithme d'optimisation stochastique de O. François [27], qui possède de bonnes propriétés de convergence asymptotique et dont tous les paramètres internes peuvent être ajustés de façon automatique.

Les algorithmes en traitement d'images utilisés (ICM, RS, l'échantilloneur de Gibbs, ECI) sont expliqués au chapitre 3. L'algorithme de Dijkstra et celui de Viterbi sont rappelés au chapitre 4. Nous consacrons le chapitre 5 à l'algorithme d'optimisation de O. François. Le modèle de Tipping et Bishop est résumé en détails au chapitre 6.

Chapitre 3

Segmentation et estimation

Dans ce chapitre, nous présentons quelques algorithmes utiles en traitement d'images pour la segmentation, ainsi que pour l'estimation de paramètres. Certains d'entre eux peuvent être formulés dans un contexte plus général. Nous avons préféré une formulation plus simple, suffisante pour nos applications. Pour nos travaux présentés dans ce mémoire, nous n'utiliserons que les sections 3.3 et 3.4.

3.1 L'algorithme ICM

Le but de l'algorithme ICM est de trouver une segmentation optimale au sens du MAP d'un modèle markovien caché. La solution trouvée par l'algorithme ICM est en fait sous-optimale, mais toutefois très satisfaisante dans le cas de plusieurs modèles en traitement d'images.

Soit $G = (V_G, E_G)$ un graphe non-orienté. On considère un couple de champs aléatoires $Z = (X, Y)$, où $Y = \{Y_s, s \in V_G\}$ représente le champs des observations aux sites s, et $X = \{X_s, s \in V_G\}$ représente le champs des étiquettes. Chaque X_s prend ses valeurs dans un ensemble $\{e_1, e_2, ..., e_K\}$.

On suppose une fonction de vraisemblance de la forme

$$P_{Y \mid X}(y/x) = \prod_{s \in V_G} P_{Y_s \mid X_s}(y_s \mid x_s)$$

ainsi qu'un modèle *a priori* markovien

$$P_X(x) = \exp(-U(x))$$

avec champs de Gibbs d'ordre 2, c'est-à-dire de la forme

$$U(x) = \sum_{<s,t> \in E_G} U_{<s,t>}(x_s, x_t).$$

Une segmentation x est optimale au sens du MAP si elle maximise la distribution *a posteriori*

$$P_{X \mid Y}(x \mid y) \propto P_{Y \mid X}(y \mid x)\, P_X(x).$$

3.2 L'algorithme du RS

On conserve les hypothèses de la section 3.1. L'algorithme du recuit simulé (RS) permet de trouver une segmentation optimale au sens du MAP. L'algorithme dépend d'un paramètre appelé *température* qui décroît à chaque itération. La convergence asymptotique de l'algorithme est assurée pourvu que la température décroisse suffisamment lentement vers 0 (en fait, exponentiellement lentement). Malheureusement, ce taux de décroissance en température dépend de la fonction à optimiser. De plus, le nombre d'itérations nécessaire à l'obtention d'un résultat acceptable peut être très grand. En pratique, on préfère donc l'ICM. Notons que l'ICM est un RS avec température figée à 0.

Algorithme ICM

But de l'algorithme : obtenir une segmentation x sous-optimale (mais acceptable) au sens du MAP.

1. **Initialisation au sens du MV** : Pour chaque site $s \in V_G$, poser :

$$x_s = \arg\max_i P_{Y_s \mid X_s}(y_s \mid e_i).$$

2. **Maximisation locale au sens du MAP** : Balayer les sites de G (selon un ordre fixe) et à chaque site visité, poser :

$$x_s = \arg\max_i \exp\{\log P_{Y_s \mid X_s}(y_s \mid e_i) - \sum_t U_{<s,t>}(e_i, x_t)\}$$

où t parcourt l'ensemble des voisins de s.

3. Si x a été modifié, retourner à 2.

3.3 L'échantillonneur de Gibbs

Il est parfois important de simuler une segmentation selon la distribution *a posteriori*. On utilise alors l'échantillonneur de Gibbs qui permet d'obtenir des échantillons selon la distribution *a posteriori* $P_{X \mid Y}$. Il s'agit d'un RS avec température fixée à 1. Notez qu'en principe il est important d'attendre la convergence en loi de l'échantillonneur de Gibbs avant de colliger les échantillons. Toutefois, dans le cas de nos applications, un seul balayage de l'image suffit amplement pour obtenir des résultats acceptables.

Algorithme RS

But de l'algorithme : obtenir une segmentation x optimale au sens du MAP.

1. **Initialisation** : Pour chaque site $s \in V_G$, choisir x_s de façon aléatoire.
2. **Simulation** : Balayer les sites de G (selon un ordre fixe) et à chaque site visité, choisir $x_s = e_i$ avec probabilité

$$\frac{\exp\{[\log P_{Y_s \mid X_s}(y_s \mid e_i) - \sum_t U_{<s,t>}(e_i, x_t)]/T\}}{\sum_{j=1}^{K} \exp\{[\log P_{Y_s \mid X_s}(y_s \mid e_j) - \sum_t U_{<s,t>}(e_j, x_t)]/T\}}$$

   où t parcourt l'ensemble des voisins de s.
3. Décroître T et retourner à 2 jusqu'à un critère d'arrêt.

L'échantillonneur de Gibbs

But de l'algorithme : obtenir des échantillons selon la distribution *a posteriori* $P_{X \mid Y}$.

1. **Initialisation** : Pour chaque site $s \in V_G$, choisir x_s de façon aléatoire.
2. **Simulation** : Balayer les sites de G (selon un ordre fixe) et à chaque site visité, choisir $x_s = e_i$ avec probabilité

$$\frac{\exp\{\log P_{Y_s \mid X_s}(y_s \mid (e_i) - \sum_t U_{<s,t>}(e_i, x_t)\}}{\sum_{j=1}^{K} \exp\{\log P_{Y_s \mid X_s}(y_s \mid e_j) - \sum_t U_{<s,t>}(e_j, x_t)\}}$$

   où t parcourt l'ensemble des voisins de s.
3. Retourner à 2 jusqu'à un critère d'arrêt.

3.4 L'algorithme ECI

On se place toujours dans le contexte de la section 3.1, en supposant de plus que chaque distribution $P_{Y_s \mid X_s}(y_s \mid x_s)$ dépend d'un vecteur de paramètres Φ. On suppose donné un estimateur $\hat{\Phi}(x, y)$ du vecteur des paramètres au sens du MV pour les données complètes (x, y). Le but de l'algorithme ECI est d'estimer le vecteur de paramètres Φ au sens des moindres carrés pour les données incomplètes y ; c'est-à-dire qu'on cherche un vecteur Φ pour lequel

$$\Phi = E_{\Phi}(\hat{\Phi}(x, y))$$

où E_{Φ} dénote l'espérance selon la distribution $P_{X \mid Y,\Phi}(x \mid y, \Phi)$.

Pour obtenir ce point fixe, on procède itérativement à partir d'une estimation initiale $\Phi^{[0]}$, en posant $\Phi^{[k+1]} = E_{\Phi^{[k]}}(\hat{\Phi}(x, y))$. Comme le calcul de $E_{\Phi^{[k]}}(\hat{\Phi}(x, y))$ est souvent impossible, on utilise l'approximation habituelle

$$E_{\Phi^{[k]}}(\hat{\Phi}(x, y)) \approx \frac{1}{n} [\hat{\Phi}(x_{(1)}, y) + \cdots + \hat{\Phi}(x_{(n)}, y)],$$

où $x_{(i)}$, $i = 1, \ldots, n$ sont des réalisations de X selon la distribution *a posteriori* $P_{X \mid Y,\Phi}(x \mid y, \Phi^{[k]})$ obtenues à l'aide de l'échantillonneur de Gibbs. Pour plusieurs applications en traitement d'images, on obtient des résultats satisfaisants en prenant $n = 1$. De plus, dans nos applications, un seul balayage de l'image pour l'échantillonneur de Gibbs suffit amplement pour obtenir des résultats acceptables. Nous utilisons cette version rapide au chapitre 7.

Pour initialiser l'ECI, on regroupe l'ensemble des données observables $\{y_s : s \in V_G\}$ en K classes grâce à l'algorithme des K-moyennes. En utilisant un critère adapté au problème, on peut ensuite assigné une étiquette x_s à chaque site s du graphe G, ce qui donne une réalisation initiale $x^{[0]}$ du champs aléatoire X. Par exemple, dans le cas de notre modèle des contours présenté au chapitre 7, nous considérons deux classes $(K = 2)$: la classes e_1 des points hors contours, et la classe e_2 des points de contours. De plus, la donnée observée y_s représente la norme du gradient des niveaux de gris

par rapport à la position du pixel s. L'algorithme des K-moyennes ($K = 2$) permet de classifier les pixels en deux classes. La classe pour laquelle la valeur moyenne de y_s est la plus grande est étiquetée e_2. Ici, le critère utilisé est physique : les points de contours correspondent aux sites pour lesquels la variation des niveaux de gris est la plus accentuée.

Algorithme ECI

But de l'algorithme : obtenir une estimation des paramètres sous-optimale (mais acceptable) au sens des moindres carrés.

1. **Initialisation** : $k = 0$. Initialisation de $\Phi^{[0]}$.
2. **Calcul récursif** : Afin d'estimer $\Phi^{[k+1]} = E_{\Phi^{[k]}}(\hat{\Phi}(x, y))$, poser

$$\Phi^{[k+1]} = \frac{1}{n} [\hat{\Phi}(x_{(1)}, y) + \cdots + \hat{\Phi}(x_{(n)}, y)],$$

   où $x_{(i)}$, $i = 1, \ldots, n$ sont des réalisations de X selon la distribution *a posteriori* $P_{X \mid Y,\Phi}(x \mid y, \Phi^{[k]})$, obtenues à l'aide de l'échantillonneur de Gibbs.
3. Si Φ n'a pas convergé, retourner à 2.

Algorithme ECI (Version rapide)

But de l'algorithme : obtenir une estimation des paramètres sous-optimale (mais acceptable) au sens des moindres carrés. Il s'agit de l'ECI avec $n = 1$.

1. **Initialisation** : $k = 0$. On utilise l'algorithme des K-moyennes pour obtenir une première réalisation $x^{[0]}$ du champs caché X. On peut alors estimer $\Phi^{[0]}$ au sens du maximum de vraisemblance.
2. **Simulation** : Soit $\Phi^{[k]}$ l'estimation courante des paramètres à l'itération k. On obtient une nouvelle réalisation $x^{[k+1]}$ de X selon la distribution $P_{X \mid Y,\Phi}(x \mid y, \Phi^{[k]})$, à l'aide de l'échantillonneur de Gibbs.
3. **Estimation** : On estime Φ au sens du MV à partir des données complètes : on pose $\Phi^{[k+1]} = \hat{\Phi}(x^{[k+1]}, y)$.
4. Si Φ n'a pas convergé, retourner à 2.

Chapitre 4

Programmation dynamique

4.1 Algorithme de Dijkstra

Soit $G = (V_G, E_G)$ un graphe fini non-orienté et s un site de G. On considère une fonction positive λ définie sur les paires de sites adjacents (s, t) de G. Étant donné un site initial s_0, on cherche pour tout site t de G un chemin $c = (s_0, s_1, ..., s_n)$, avec $s_n = t$, qui minimise la fonction

$$l(s_1, ..., s_n) = \sum_{i=1}^{n} \lambda(s_{i-1}, s_i).$$

L'algorithme de Dijkstra permet d'effectuer cette tâche à l'aide de la programmation dynamique. Nous utilisons cet algorithme pour l'extraction semi-automatique de contours à partir de positions initiale et finale (voir les sections 2.2.1 et 8.1).

En fait, nous utilisons une version [46] légèrement modifiée de l'algorithme, qui a le mérite de trouver rapidement une solution approchée (mais acceptable) de la solution optimale. Dans la version standard de l'algorithme, on utilise une liste de sites triée selon le coût total ; il faut donc parcourir cette liste pour insérer de nouveaux éléments. Par contre, dans la version modifiée, on utilise un vecteur de listes de sites, pour lequel l'indice donne le coût total des sites compris dans la liste correspondante ; on peut

donc insérer un nouvel élément en temps constant. Par contre, il faut approximer le coût total d'un site par sa partie entière afin d'obtenir un indice entier. On perd donc en précision. Ceci dit, on peut augmenter la précision arbitrairement en remplaçant la fonction λ par la fonction $C\lambda$, où $C >> 0$. Dans notre application présentée à la section 8.1, on a pris $C = 10$, ce qui donne une approximation du coût total au dixième près.

4.2 Algorithme de Viterbi

Soient n et N deux entiers positifs et des fonctions de coût $\lambda_0 : \{1, ..., N\} \rightarrow R$, et $\lambda_t : \{1, ..., N\} \times \{1, ..., N\} \rightarrow R$, $t = 1, ..., n$. On cherche une segmentation $(i_0, i_1, ..., i_n) \in \{1, ..., N\}^n$ qui minimise la fonction

$$l(i_0, i_1, ..., i_n) = \lambda_0(i_0) + \sum_{t=1}^{n} \lambda_t(i_{t-1}, i_t).$$

L'algorithme de Viterbi permet de résoudre ce problème à l'aide de la programmation dynamique. Nous utilisons cet algorithme avec $N = 2$ pour la segmentation au sens des contours. En fait, cet algorithme sera appliqué sur chaque chemin appartenant à un ensemble de chemins simples, non-vides, et disjoints. Voir le chapitre 7.

Algorithme de Dijkstra

s_0	position initiale
$\lambda(s,t)$	coût de s à t (on doit avoir $\lambda(s,t) \geq 0$)
L	liste des pixels triés selon le coût total
$N(s)$	ensemble des 8-voisins du site s
$e(s)$	booléen indiquant si le site s a été traité
$\delta(s)$	coût du germe au site s
p	pointeur indiquant le chemin minimal

But de l'algorithme : trouver un chemin de coût minimal allant de s_0 à t, pour chaque pixel t de l'image.

1. **Initialisation** : poser $\delta(s_0) = 0$; ajouter s_0 à L.
2. **Calcul récursif** : répéter tant que L n'est pas vide :
   - enlever l'élément minimal s de L
   - poser $e(s) =$ vrai ;
   - pour chaque $t \in N(s)$ tel que $e(t) =$ faux faire :
     - $\delta_{tmp} = \delta(s) + \lambda(s,t)$;
     - si $t \in L$ et $\delta_{tmp} < \delta(t)$: enlever t de L ;
     - si $t \notin L$: poser $\delta(t) = \delta_{tmp}$, $p(t) = s$, ajouter t à L.

Algorithme de Dijkstra (version approchée et rapide)

s_0	position initiale (germe)
$\lambda(s,t)$	coût de s à t (on doit avoir $\lambda(s,t) \geq 0$)
L	vecteur de listes des pixels triés selon le coût total :
	$L(i)$ est la liste des pixels dont le coût total l satisfait $\lfloor l \rfloor = i$
$N(s)$	ensemble des 8-voisins du site s
$e(s)$	booléen indiquant si le site s a été traité
$\delta(s)$	coût du germe au site s
p	pointeur indiquant le chemin minimal
i_*	indice i minimal pour lequel la liste $L(i)$ est non-vide

But de l'algorithme : trouver un chemin de coût approximatif minimal allant de s_0 à t, pour chaque pixel t de l'image.

1. **Initialisation** : poser $\delta(s_0) = 0$; ajouter s_0 à $L(0)$; poser $i_* = 0$.
2. **Calcul récursif** : répéter tant que L n'est pas vide :
   - enlever s de $L(i_*)$, mettre à jour i_* (en parcourant si nécessaire le vecteur L à partir de i_* jusqu'à ce qu'une liste de la forme $L(i)$ soit non-vide)
   - poser $e(s) =$ vrai ;
   - pour chaque $t \in N(s)$ tel que $e(t) =$ faux faire :
     - $\delta_{tmp} = \delta(s) + \lambda(s,t)$;
     - si $t \in L$ et $\delta_{tmp} < \delta(t)$: enlever t de $L(\lfloor \delta(t) \rfloor)$, et mettre à jour i_* ;
     - si $t \notin L$: poser $\delta(t) = \delta_{tmp}$, $p(t) = s$, ajouter t à $L(\lfloor \delta(t) \rfloor)$, et mettre à jour i_* (en remplaçant i_* par $\lfloor \delta(t) \rfloor$ si nécessaire).

Algorithme de Viterbi

$\lambda_0(i)$	coût initial pour i
$\lambda_t(i,j)$	coût de i à j de l'itération $t-1$ à l'itération t
$\delta_t(j)$	coût minimal de 0 à l'itération t se terminant par j
$p_t(j)$	pointeur indiquant l'argument optimal à l'itération précédente

But de l'algorithme : trouver une solution $(i_0, i_1, ..., i_n) \in \{1, ..., N\}^n$ qui minimise la fonction $\lambda_0(i_0) + \sum_{t=1}^{n} \lambda_t(i_{t-1}, i_t)$.

- **Initialisation** : pour $j = 1, ..., N$, poser :

$$\begin{aligned} \delta_0(j) &= \lambda_0(j); \\ p_0(j) &= 1. \end{aligned}$$

- **Calcul récursif** : pour $1 \leq t \leq n$, et $j = 1, .., N$, calculer :

$$\begin{aligned} \delta_t(j) &= \min_{i=1,...,N} \{ \delta_{t-1}(i) + \lambda_t(i,j) \}; \\ p_t(j) &= \arg \min_{i=1,...,N} \{ \delta_{t-1}(i) + \lambda_t(i,j) \}. \end{aligned}$$

- **Fin** :

$$\begin{aligned} \delta_* &= \min_{i=1,...,N} \delta_n(i); \\ i_n &= \arg \min_{i=1,...,N} \delta_n(i). \end{aligned}$$

- **Segmentation optimale** : pour $t = n, n-1, ..., 1$:

$$i_{t-1} = p_t(i_t).$$

Chapitre 5

Optimisation stochastique

5.1 Introduction

Divers algorithmes d'optimisation stochastique ont été étudiés dans les deux dernières décennies.

Un premier résultat sur la convergence du recuit simulé ([1]) a été démontré par Geman et Geman [33]. Une analyse plus approfondie a ensuite été établie par Hajek [35] et Catoni [9].

Hwang et Sheu [37] ont introduit le recuit simulé généralisé (RSG) comme un analogue dans le cas discret du cadre théorique de Freidlin et Wentzell [29]. Une analyse de convergence approfondie du RSG a été établie par Trouvé [55], ainsi qu'une étude des cycles [54]. Une contribution importante à cette théorie se retrouve dans les travaux de Catoni [9], Chang et Chow [13], [14], ainsi que Hwang et Sheu [37], [36].

Le recuit simulé en parallèle ([2]) s'inscrit dans le cadre du RSG, ainsi que l'a montré Trouvé [52] et [53].

Une première modélisation de l'algorithme génétique ([34]) par des chaînes de Markov a été présentée par Davies et Principe [17]. Ultérieurement, un nouvel algo-

rithme génétique a été proposé par Cerf [11] et [12] dans le cadre de la théorie de Freidlin et Wentzell et du RSG.

Plus récemment, un nouvel algorithme évolutionaire ([3]) a été proposé et étudié par O. François [26], [27] et [28], dans le cadre du RSG.

Ainsi, les trois types d'algorithmes d'optimisation stochastique (recuit simulé, algorithmes génétiques, algorithmes évolutionaires) s'inscrivent maintenant dans le cadre général du RSG pour lequel une théorie de la convergence a été développée.

Notre choix a porté sur l'algorithme d'O. François, car parmi ces algorithmes, c'est le seul pour lequel l'ajustement des paramètres internes ne dépend pas de la fonction à minimiser. En fait, il suffit de connaître le diamètre du graphe d'exploration.

5.2 Présentation de l'algorithme E/S

5.2.1 Version standard de l'algorithme

L'algorithme suivant a été introduit par O. François [28].

Soit E un ensemble fini. On considère une fonction à valeurs réelles f de domaine E dont on cherche un minimum global, c'est-à-dire un élément de l'ensemble $E_* = \{a_* \in E \, : \, f(a_*) \leq f(a), \, \forall a \in E\}$.

Pour $n \geq 2$, on considère E^n. Si $x = (x_1, ..., x_n) \in E^n$, on défini $\hat{x}$ comme l'unique élément $x_i \in E_x = \{x_1, ..., x_n\}$ tel que $f(x_j) > f(x_i)$, pour $1 \leq j < i$, et $f(x_j) \geq f(x_i)$, pour $i < j \leq n$.

On considère un graphe $\mathcal{G}$ sur E appelé *graphe d'exploration*, que l'on suppose connexe et symétrique. Pour chaque $a \in E$, $N(a)$ dénote le voisinage de a dans le graphe $\mathcal{G}$. Pour chaque $b \in E$, on fixe une distribution positive a_b sur $N(b)$.

L'algorithme dans sa forme standard peut être énoncé comme suit.

Algorithme E/S (Version standard)

But de l'algorithme : minimiser la fonction f sur l'ensemble fini E.

1. **Initialisation** : Initialisation aléatoire de $x = (x_1, ..., x_n) \in E^n$.
2. **Exploration/Sélection** : Répéter jusqu'à un critère d'arrêt :
   - déterminer $\hat{x}$.
   - pour chaque $i = 1, ..., n$, remplacer, avec probabilité p, x_i par $y_i \in N(x_i) \backslash \{\hat{x}\}$ selon la distribution a_{x_i} ; sinon, remplacer x_i par $\hat{x}$ (avec probabilité $1 - p$).
   - décroître p.

Dans [28], la distribution a_{x_i} est fixée, mais on peut choisir une distribution positive quelconque.

La probabilité p est appelée *probabilité d'exploration* et dépend d'un paramètre $T > 0$, appelé *température*. On prend $p_T = \exp(-1/T)$. On suppose que T décroît vers 0 suffisamment lentement et que n est suffisamment grand. L'ajustement de ces paramètres ne dépend que du diamètre D du graphe d'exploration.

5.2.2 Version rapide de l'algorithme

Dans [27], la variante suivante de l'algorithme E/S a été étudiée. Ici, a_{x_i} est une distribution positive quelconque sur $N(x_i) \setminus \{\hat{x}\}$. On suppose que f est injective dans [27], mais dans [28], cette restriction a été levée. Cette version de l'algorithme est un peu plus rapide que la version standard, puisqu'on n'a besoin que d'une copie de $\hat{x}$.

Algorithme E/S (Version rapide)

But de l'algorithme : minimiser la fonction f sur l'ensemble fini E.

1. **Initialisation** : Initialisation aléatoire de $x = (x_1, ..., x_n) \in E^n$.
2. **Exploration/Sélection** : Répéter jusqu'à un critère d'arrêt :
   - déterminer $\hat{x}$.
   - tirer N selon une distribution binômiale $b(n, p)$. Pour $i \leq N$, remplacer x_i par $y_i \in N(x_i) \setminus \{\hat{x}\}$ selon la distribution a_{x_i}. Pour $i > N$, remplacer x_i par $\hat{x}$.
   - décroître p.

5.2.3 Un cas particulier

On présente maintenant un cas particulier de l'algorithme E/S. Soit E un produit cartésien d'intervalles compacts $\prod_{j=1}^{L} I_j$, où $I_j = [a_j, b_j]$. (E est en fait un sous-ensemble discret du produit.)

Fixons $1 > r > 0$. On défini un graphe d'exploration par les voisinages $N(a) = \cup_{j=1}^{L} N_{j,r}(a)$ où

$$N_{j,r}(a) = \{b \ : \ |b_j - a_j| \leq r|I_j|, \ b_i = a_i, \ i \neq j\}. \tag{5.1}$$

Ce graphe symétrique correspond à la modification simple des composantes. L'étape d'exploration de l'algorithme peut s'énoncer comme suit.

- choisir au hasard une composante x_i^j de x_i et la remplacer par un nombre dans l'intervalle $[x_i^j - r|I_j|, x_i^j + r|I_j|] \cap I_j$ selon une distribution uniforme.

Le diamètre du graphe est donné par $D = L/r$.

On peut également prendre les voisinages $N(a) = B_r(a)$ où

$$B_r(a) = \{b : |b_j - a_j| \leq r|I_j|, 1 \leq j \leq L\}. \tag{5.2}$$

Cette version correspond à une modification multiple des composantes. Nous fixons la probabilité de modifier chaque composante à $1/L$. On a $D = 1/r$.

Au chapitre 9, nous utilisons la version rapide avec modification d'une composante à la fois. De plus, on prend $D = 32$ et $n = 100$.

5.3 Convergence de l'algorithme E/S

On rappelle que la distance $d(a, b)$ entre deux sommets a, b d'un graphe $\mathcal{G}$ est la longueur (nombre de segments) d'un chemin minimal reliant a et b; si a et b sont disconnectés dans le graphe, on pose $d(a, b) = \infty$. Le diamètre d'un graphe est défini comme la distance maximale D entre deux sommets du graphe. En outre, si $\mathcal{G}$ comporte un nombre fini de sommets, on a $D < \infty$ si et seulement si $\mathcal{G}$ est connexe.

L'algorithme E/S simule une chaîne de Markov non homogène sur l'ensemble E^n, puisque la température dépend de l'itération $t \geq 0$. X_t^T dénotera l'état du vecteur x à l'itération t, où $T = T(t)$; ainsi, la variable X_t^T prend ses valeurs dans l'ensemble E^n. Rappelons que la probabilité d'exploration est donnée par $p_T = \exp(-1/T)$. La constante n est parfois appelée *taille de la population.*

La convergence asymptotique de l'algorithme E/S a été démontrée par O. François [28]. Dans ce qui suit, un élément de la forme $(a) \in E^n$ est identifié avec l'élément correspondant $a \in E$. Rappelons que E_* dénote le sous-ensemble de E constitué des minima de la fonction f.

Théorème de O. François. *Supposons que $\mathcal{G}$ est connexe et symétrique. Soit $n > n_* = \max_{a \notin E_*, b \in E_*} d(a, b)$. Pour toute suite décroissante $(T(t))_{t \geq 0}$ convergeant vers 0, on a*

$$\lim_{t \to \infty} \sup_x P(X_t^{T(t)} \notin E_* \mid X_0^{T(0)} = x) = 0$$

si et seulement si

$$\sum_{t=1}^{\infty} \exp(-H_1/T(t)) = \infty$$

où

$$H_1 = \max_{A\,:\,A\cap E_*=\emptyset} \min_{a\in A, b\notin A\,:\,f(b)\le f(a)} d(a,b).$$

Notez qu'on a $n_* \le D$, ainsi que $H_1 \le D$, où D est le diamètre du graphe d'exploration $\mathcal{G}$. On déduit le résultat suivant qui est suffisant pour l'application de l'algorithme E/S.

Corollaire. *Supposons que $\mathcal{G}$ est connexe et symétrique, de diamètre D. Soit $n > D$ et $p_t = (t+2)^{-1/D}$ (c'est-à-dire, par définition, $T(t) = \frac{D}{\ln(t+2)}$). Alors,*

$$\lim_{t\to\infty} \sup_x P(X_t^{T(t)} \notin E_* \mid X_0^{T(0)} = x) = 0.$$

Preuve : On a $\exp(-1/T(t)) = p_t = (t+2)^{-1/D}$; d'où, $\sum_{t=1}^{\infty} \exp(-H_1/T(t)) = \sum_{t=1}^{\infty} (t+2)^{-H_1/D} \ge \sum_{t=3}^{\infty} t^{-1} = \infty$, car $0 \le H_1 \le D$.

Chapitre 6

Réduction de dimension

Le modèle suivant a été introduit par Tipping et Bishop [51]. Il s'agit de l'analyse en composantes principales probabiliste (ACPP). On considère deux variables aléatoires T et Ξ reliées par l'identité

$$T = W\Xi + \nu + \varepsilon$$

où $\varepsilon \sim \mathcal{N}(0, \sigma^2 I_d)$, $\Xi \sim \mathcal{N}(0, I_q)$, et W est une matrice $d \times q$ (I_r dénote la matrice identité $r \times r$). Il s'agit d'un cas particulier de [5] tel que présenté dans [51]. La variable T représente les données complètes, tandis que Ξ représente les données réduites. Habituellement, la dimension q de Ξ est choisie sensiblement plus petite que la dimension d de T.

On peut montrer que, si $\sigma^2 > 0$, T suit une distribution gaussienne exprimée par

$$p_T(t) = \mathcal{N}(t; \nu, \sigma^2 I_d + WW^t).$$

De plus, la matrice inverse de $\sigma^2 I_d + WW^t$ est donnée par

$$\sigma^{-2}\Big(I_d - W(W^tW + \sigma^2 I_q)^{-1}W^t\Big).$$

Soit maintenant $t_1, ..., t_l$ un échantillon d'observations i.i.d. issues de T. On cherche les valeurs optimales de W, ν, σ^2 au sens du MV, c'est-à-dire qui maximisent la fonction

de vraisemblance

$$\mathcal{L}(W, \nu, \sigma^2) = \prod_{i=1}^{l} p_T(t_i).$$

Le théorème suivant est prouvé dans [51].

Théorème. ([51]) *Soit $t_1, t_2, ..., t_l$ un échantillon d'observations i.i.d. d'une variable aléatoire T. Soient $\Xi \sim \mathcal{N}(0, I_d)$ et $T = W\Xi + \nu + \varepsilon$, où W est une matrice $d \times q$ $(q \leq d)$ et $\varepsilon \sim N(0, \sigma^2 I_d)$. Alors, toute solution optimale au sens du MV est de la forme*

$$\begin{aligned} \nu &= \bar{t} = \frac{1}{l}\sum_{i=1}^{l} t_i \\ \sigma^2 &= \frac{1}{d-q}\sum_{i=q+1}^{d} \lambda_i \\ W &= U_q(\Lambda_q - \sigma^2 I_q)^{1/2} R \end{aligned}$$

où $\lambda_1, ..., \lambda_d$ sont les valeurs propres en ordre décroissant de la matrice de covariance empirique, Λ_q est la matrice diagonale avec entrées $\lambda_1, ..., \lambda_q$, les colonnes de U_q engendrent le sous-espace principal de la matrice de covariance empirique, et R est une matrice orthogonale $q \times q$ quelconque.

Rappelons ici que la matrice de covariance empirique $S = (s_{ij})$ est définie par

$$s_{ij} = \frac{1}{n}\sum_{k=1}^{l}(t_k(i) - \bar{t}(i))(t_k(j) - \bar{t}(j))$$

où $v(j)$ représente la j-ième composante d'un vecteur v.

Puisque la distribution de T conditionnelle à une valeur donnée de Ξ est donnée [51] par

$$P_{T \mid \Xi}(t \mid \xi) = (2\pi\sigma^2)^{-q/2} \exp\Big(-\frac{1}{2\sigma^2}||t - U_q(\Lambda_q - \sigma^2 I_q)^{1/2}\xi - \bar{t}||^2\Big),$$

la fonction de reconstruction définie par

$$\phi(\xi) \;=\; U_q(\Lambda_q - \sigma^2 I_q)^{1/2}\xi + \bar{t}$$

est optimale au sens du MV. Dans ce cas, la fonction de réduction doit être définie par

$$\psi(t) \;=\; (\Lambda_q - \sigma^2 I_q)^{-1/2} U_q^T (t - \bar{t})$$

afin de minimiser l'erreur de reconstruction

$$\varepsilon \;=\; \sum_{i=1}^{l} ||t_i - \phi(\psi(t_i))||^2.$$

On a alors

$$\varepsilon \;=\; \sum_{i=1}^{l} ||t_i - U_q U_q^T (t_i - \bar{t}) - \bar{t}||^2$$

comme il se doit.

Le lemme suivant est utilisé au chapitre 9, et se vérifie par un simple calcul.

Lemme. *Soit* $T = U_q(\Lambda_1 - \sigma^2 I_q)^{1/2}\Xi + \bar{t} + \varepsilon$, *où* $\varepsilon \sim \mathcal{N}(0, \sigma^2 I_d)$ *et* $\Xi \sim \mathcal{N}(0, I_q)$. *Alors,*

$$p_T(U_q(\Lambda_1 - \sigma^2 I_q)^{1/2}\xi + \bar{t}) \propto \exp\Big(-\frac{1}{2}\xi^t(I_q - \sigma^2\Lambda_q^{-1})\xi\Big).$$

En pratique, nous procédons de la manière suivante pour calculer l'ACPP.

Algorithme pour le calcul de l'ACPP

l	taille de l'échantillon $t_1, t_2, ..., t_l$
d	dimension de l'espace vectoriel des données complètes
q	dimension de l'espace vectoriel des données réduites
$t_i(j)$	j-ième composante du vecteur t_i
α	seuil de tolérance pour l'erreur moyenne de reconstruction

But de l'algorithme : obtenir une ACPP optimale au sens du MV à partir de l'échantillon $t_1, t_2, ..., t_l$.

1. Calculer la moyenne empirique $\bar{t} = \frac{1}{l}\sum_{i=1}^{l} t_i$.
2. Calculer la matrice $d \times n$, $A = \frac{1}{\sqrt{l}}(t_i(j) - \bar{t}(j))$; la matrice de covariance empirique S est donnée par $A^T A$.
3. Effectuer la décomposition SVD de la matrice A : $A = UDV^T$, où U est une matrice de dimension $n \times n$ dont les colonnes sont des vecteurs unitaires mutuellement orthogonaux, tout comme pour la matrice V de dimension $d \times d$, et où la matrice $n \times d$ $D = (d_{ij})$ est diagonale, avec $d_{11} \geq d_{22} \geq ....$
4. Poser $\lambda_i = d_{ii}^2$, pour $1 \leq i \leq \min(d, n)$, où $D = (d_{ij})$; poser $\lambda_i = 0$, pour $\min(d, n) < i \leq d$; les valeurs propres de la matrice S sont données par $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_d$.
5. Pour $m = 1, ..., d-1$:
   (a) La matrice U_m est formées des m premières colonnes de V.
   (b) Calculer l'erreur de reconstruction moyenne
   $$\varepsilon = \frac{1}{l}\sum_{i=1}^{l} ||t_i - U_m U_m^T(t_i - \bar{t}) - \bar{t}||^2.$$
   (c) Si $\varepsilon < \alpha$, poser $q = m$ et sortir de la boucle.
6. Poser $\nu = \bar{t}$, $\sigma^2 = \frac{1}{d-q}\sum_{i=q+1}^{d} \lambda_i$, et $W = U_q(\Lambda_q - \sigma^2 I_q)^{1/2}$.

Chapitre 7

Détection de contours

Dans ce chapitre, nous présentons un modèle statistique original pour la distribution du gradient des niveaux de gris dans une image [20], basé sur des résultats empiriques. De plus, nous présentons un nouveau modèle markovien pour les contours, qui utilise ce modèle statistique pour définir la fonction de vraisemblance. Notre modèle est adapté à une procédure ECI pour l'estimation des paramètres. Afin de faciliter notre tâche pour la segmentation de l'image au sens des contours, nous traitons une version simplifiée de notre modèle, qui ignore les points de jonction. Cette version simplifiée est adaptée à l'algorithme de Viterbi. On obtient ainsi une nouvelle méthode statistique de détection non-supervisée de contours. Nous avons également développé une version [21] de notre modèle, qui n'utilise que la distribution de la norme du gradient.

7.1 Modèle des contours

Étant donné une image de taille N, $G = (V_G, E_G)$ dénote le graphe formé des N pixels de l'image et des voisinages formés des 8-voisins habituels. On considère également le graphe $G' = (V_{G'}, E_{G'})$, obtenu en posant $V_{G'} = V_G \cup E_G$, et dans lequel

deux sites adjacents de G le sont encore, de même qu'un segment de G et ces deux sommets (donc, $E_{G'} = E_G \cup \{(s,u),(u,s),(t,u),(u,t) : u = (s,t) \in E_G\}$).

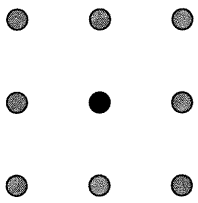


FIG. 7.1 – *Un pixel et ses 8 voisins (8 pixels) dans G.*

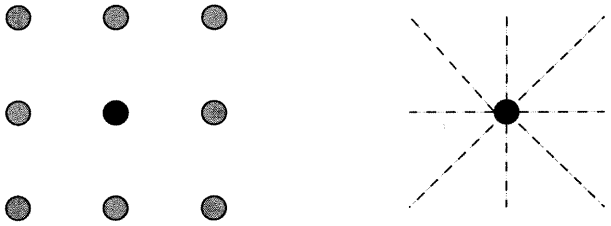


FIG. 7.2 – *Un pixel et ses 16 voisins (8 pixels et 8 segments) dans G'.*

FIG. 7.3 – *Un segment et ses 2 voisins (2 pixels) dans G'.*

Si $s \in V_G$, la variable aléatoire Y_s représente la norme du gradient des niveaux de gris en ce site. Physiquement, Y_s représente le contraste dans les niveaux de gris au point s. Nous postulons que la valeur de Y_s est élevée lorsque s est un point de contour.

Lorsque s n'est pas un point de contours, nous modélisons la distribution de Y_s par une loi de Weibull [41],

$$\mathcal{W}(y; \min, C, \alpha) = \frac{C}{\alpha}\left(\frac{y-\min}{\alpha}\right)^{C-1} \exp\left(-\frac{(y-\min)^C}{\alpha^C}\right),$$

où $y > \min$. Dans nos tests, la valeur de C varie entre 0.3 et 0.95. Notez que la loi exponentielle est le cas particulier d'une loi de Weibull pour laquelle $C = 1$.

Par contre, si s est un point de contours, nous modélisons la distribution de Y_s par un mélange de lois gaussiennes

$$\mathcal{M}(y; w_i, \mu_i, \sigma_i) = \sum_{i=1}^{2} w_i \mathcal{N}(y; \mu_i, \sigma_i).$$

Nos test suggèrent que ce modèle est suffisamment flexible pour tenir compte des variations de la distribution de Y_s d'une image à l'autre.

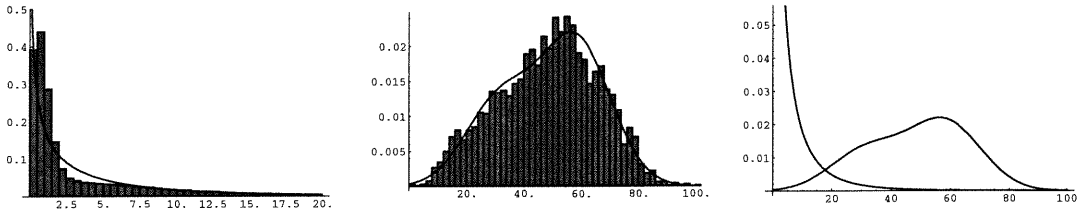


FIG. 7.4 – *Exemple de distributions pour la norme du gradient. De gauche à droite : norme du gradient pour les points hors contours; norme du gradient pour les points de contours; comparaison entre les deux distributions.*

Si $(s,t) \in E_G$, on considère également la distribution de la valeur absolue Y_{st} de l'angle entre la moyenne du gradient aux pixels s et t, et la normale au vecteur reliant s à t. Les angles sont normalisés entre $-\pi/2$ et $\pi/2$. Physiquement, Y_{st} représente la valeur absolue de l'angle entre la courbe de niveau des niveaux de gris et la courbe des contours passant par s et t. Si le segment (s,t) est situé sur une courbe de contours, nous postulons que la courbe de contours coïncide localement avec la courbe de niveau des niveaux de gris, c'est-à-dire que la valeur de Y_{st} est près de 0.

Dans le cas où (s,t) n'appartient pas à un contour, nous modélisons Y_{st} par une distribution uniforme sur l'intervalle $[0, \frac{\pi}{2}]$,

$$\mathcal{U}(y_{st}; 0, \frac{\pi}{2}) = \frac{2}{\pi}, \qquad 0 \leq y_{st} \leq \frac{\pi}{2}.$$

En effet, si un segment n'appartient pas à une véritable courbe de contours, l'angle qu'il forme avec le gradient peut être quelconque, sans aucune valeur privilégiée.

Nous avons observé que lorsque le segment (s,t) appartient à un contour, Y_{st} suit une loi exponentielle tronquée sur l'intervalle $[0, \frac{\pi}{2}]$

$$k_0 \mathcal{E}(y_{st}; \alpha_0) = k_0 \exp(-y_{st}/\alpha_0)/\alpha_0, \qquad 0 \leq y_{st} \leq \frac{\pi}{2}$$

où le facteur k_0 est donné par $\{\int_0^{\frac{\pi}{2}} \mathcal{E}(y; \alpha_0)\, dy\}^{-1} = \{1 - \exp(-\frac{\pi}{2\alpha_0})\}^{-1}$.

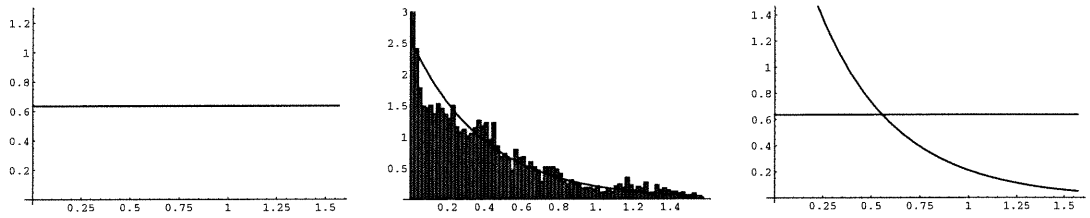


FIG. 7.5 – *Exemple de distributions pour l'angle du gradient. De gauche à droite : angle pour les points hors contours; angle pour les points de contours; comparaison entre les deux distributions.*

Notre choix de distributions est justifié par des résultats empiriques. Voir les figures 7.11, 7.14, 7.17, 7.20, et 7.23 pour des exemples de distributions empiriques et de distributions estimées. On constate que les distributions estimées correspondent assez bien aux distributions empiriques.

Les variables X_s, X_{st} prennent leurs valeurs dans l'ensemble $\{e_1 =$ "hors-contours", $e_2 =$ "contour"$\}$. On obtient donc un champ aléatoire $Z = (X, Y)$, en posant $Y = \{Y_s, Y_{st}, s \in V_G, < s,t > \in E_G\}$, et $X = \{X_s, X_{st}, s \in V_G, < s,t > \in E_G\}$. Y représente le champs des observations, et X représente le choix des étiquettes.

La distribution de vraisemblance est alors modélisée par l'équation

$$P_{Y \mid X}(y \mid x) \;\propto\; \prod_{s \in V_G} P_{Y_s \mid X_s}(y_s \mid x_s) \prod_{(s,t) \in E_G} P_{Y_{st} \mid X_{st}}(y_{st} \mid x_{st})$$

où

$$\begin{aligned} P_{Y_s \mid X_s}(y_s \mid e_1) &= \mathcal{W}(y_s; \min, C, \alpha) \\ P_{Y_s \mid X_s}(y_s \mid e_2) &= \mathcal{M}(y_s; w_j, \mu_j, \sigma_j) \\ P_{Y_{st} \mid X_{st}}(y_{st} \mid e_1) &= \mathcal{U}(y_{st}; 0, \frac{\pi}{2}) \\ P_{Y_{st} \mid X_{st}}(y_{st} \mid e_2) &= k_0 \mathcal{E}(y_{st}; \alpha_0). \end{aligned}$$

Soit maintenant T^+ l'ensemble des sites de l'image pour lesquels la norme du gradient est un maximum local dans la direction du gradient (approximée à un multiple entier de $\pi/4$). On obtient un ensemble T en enlevant les points isolés de T^+ ; un élément de l'ensemble T^+ est un point isolé si aucun de ses 8-voisins n'est dans l'ensemble T^+ (voir Figure 7.6). Les éléments de l'ensemble T sont les points de contours potentiels. On pose $T^* = \{(s,t) \mid s,t \in T \text{ et } (s,t) \in E\}$.

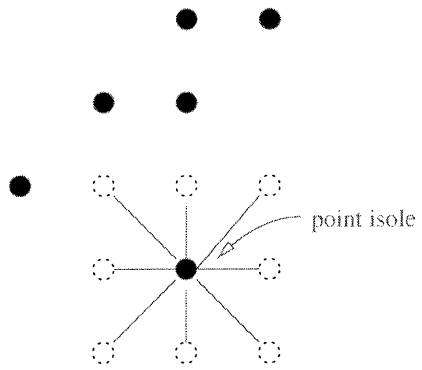


FIG. 7.6 – *Illustration d'un point isolé d'un ensemble.*

Pour le modèle *a priori*, nous adoptons le modèle correspondant à la fonction $\sum_{<s,t>:(s,t)\in T^*}\big(1-\delta(x_s,x_t)\big)$, afin de favoriser une segmentation homogène. Nous imposons également les contraintes suivantes : $x_s = e_1$ pour tout $s \notin T$ (seul un point de l'ensemble T peut être un point de contours) ; $x_{st} = e_1$ lorsque $(s,t) \notin T^*$ (seul un segment ayant ses deux extrémités dans T peut être sur une courbe de contours) ; $x_s = e_2$ et $x_t = e_2$, si et seulement si $x_{st} = e_2$ (un segment est sur une courbe de contours si et seulement si ses deux extrémités sont des points de contours). Étant

donné ce modèle *a priori* avec contraintes, la distribution $P_X(x)$ peut s'écrire sous la forme

$$P_X(x) = \frac{1}{Z}\exp\Big\{-\sum_{<s,t>:(s,t)\in T^*}\big(1-\delta(x_s,x_t)\big)\Big\} \prod_{s\notin T}\delta(x_s,e_1)\prod_{(s,t)\notin T^*}\delta(x_{st},e_1)\prod_{(s,t)\in T^*}\eta(x_s,x_t,x_{st})$$

où la somme se fait sur tous les segments de T^*, δ est le symbole de Kronecker, Z est une constante de normalisation, et $\eta(x_s,x_t,x_{st})=1$ si $x_s=x_t=x_{st}=e_2$, ou bien $x_{st}=e_1$ et, $x_s=e_1$ ou $x_t=e_1$, et sinon $\eta(x_s,x_t,x_{st})=0$.

Soit maintenant U le champs de Gibbs défini par

$$U(x,y) = \sum_{s\in S}-\ln P_{Y_s\mid X_s}(y_s\mid x_s)+\sum_{(s,t)\in E}-\ln P_{Y_{st}\mid X_{st}}(y_{st}\mid x_{st}) + \sum_{<s,t>:(s,t)\in T^*}\Big(1-\delta(x_s,x_t)\Big)$$

et V la fonction donnée par

$$V(x) = \sum_{s\notin T}\Big(1-\delta(x_s,e_1)\Big)+\sum_{(s,t)\notin T^*}\Big(1-\delta(x_{st},e_1)\Big) + \sum_{(s,t)\in T^*}\Big(1-\eta(x_s,x_t,x_{st})\Big).$$

La distribution *a posteriori* du modèle des contours s'exprime alors sous la forme

$$P_{X/Y}(x/y) = \frac{1}{Z}\exp\Big(-U(x,y)\Big)\,\chi(\{x\,:\,V(x)\text{ est minimal}\},x),$$

où $\chi(A,\cdot)$ dénote la fonction caractéristique d'un ensemble A, et Z est une constante de normalisation. Il s'agit donc d'un modèle markovien sous contrainte.

7.2 Estimation

Nous montrons maintenant comment adapter la procédure ECI pour l'estimation du vecteur des paramètres $\Phi = (\min, C, \alpha, w_1, \mu_1, \sigma_1^2, w_2, \mu_2, \sigma_2^2, k_0, \alpha_0)$ pour notre

modèle.

Pour la segmentation initiale, nous utilisons l'algorithme des K-moyennes, avec pour attribut la norme du gradient des niveaux de gris et $K = 2$. La classe avec moyenne minimale est étiquetée e_1.

On peut utiliser l'échantilloneur de Gibbs sous contrainte [30] pour simuler le champ X selon la distribution $P_{X/Y}(x/y)$. Cependant, nous imposons les contraintes directement en posant $x_s = e_1$ si $s \notin T$, $x_{st} = e_1$ lorsque $(s,t) \notin T^*$, et $x_{st} = e_2$ si et seulement si $x_s = e_2$ et $x_t = e_2$. On peut alors utiliser l'échantilloneur de Gibbs classique [33] sur l'ensemble $T \cup T^*$. Pour ce faire, les sites de $T \cup T^*$ sont visités ainsi. Si s est un site dans T, on pose $E(s) = \{t : t \in T, (s,t) \in E_G\}$ (les voisins du pixel s qui sont dans l'ensemble T). Il y a deux possibilités pour l'étiquette de s et chacune de ces possibilités détermine l'étiquette du segment (s,t), pour $t \in E(s)$, en utilisant la contrainte $\eta(x_s, x_t, x_{st}) = 1$. Pour chacune de ces possibilités, on pose $p(x_s)$ égal à

$$P_{Y_s \mid X_s}(y_s \mid x_s) \prod_{t \in E(s)} \{P_{Y_{st} \mid X_{st}}(y_{st} \mid x_{st}) \exp(-1 + \delta(x_s, x_t))\}$$

avec chaque x_{st} ajusté de telle sorte que $\eta(x_s, x_t, x_{st}) = 1$. On choisit alors pour valeur de x_s, e_1 avec probabilité $p(e_1)/(p(e_1) + p(e_2))$.

À l'étape de l'estimation, on ne peut calculer directement les estimateurs MV. Toutefois, si $Y = (Y_1, \ldots, Y_M)$ sont M variables aléatoires i.i.d. selon une loi de Weibull $\mathcal{W}_Y(y; \min, C, \alpha)$, et que $y = (y_1, \ldots, y_M)$ est une réalisation de Y, les estimateurs MV de $C_{\mathrm{ML}}, \alpha_{\mathrm{ML}}$ pour les données complètes sont donnés par [42],

$$\begin{aligned} \hat{C}_{\mathrm{ML}} &= F(\hat{C}_{\mathrm{ML}}), \\ \hat{\alpha}_{\mathrm{ML}} &= \left(\frac{1}{M} \sum_{i=1}^{M} \tilde{y}_i^{\hat{C}_{\mathrm{ML}}} \right)^{\frac{1}{\hat{C}_{\mathrm{ML}}}}, \end{aligned}$$

où $\tilde{y} = (y - \min)$ et

$$F(x) = \frac{M \sum_{i=1}^{M} \tilde{y}_i^x}{M \sum_{i=1}^{M} (\tilde{y}_i^x \ln \tilde{y}_i) - \sum_{i=1}^{M} \ln \tilde{y}_i^x \sum_{i=1}^{M} \tilde{y}_i^x}.$$

De plus, on pose tout simplement min $= -10^{-5}\nu$, où ν est le maximum de la norme du gradient sur l'image traitée. On pourrait poser min $= 0$, mais pour des raisons numériques, il nous a semblé préférable de prendre min légèrement inférieur à 0. Nous utilisons la méthode itérative [42] pour déterminer $\hat{C}_{ML}$.

Pour l'estimation d'un mélange de gaussiennes, l'utilisation de l'algorithme SEM [10] a produit de bons résultats lors de nos tests. Il s'agit d'une version stochastique de l'algorithme EM [18], et est en tout point semblable à l'algorithme ECI sauf pour la distribution *a priori*, qui est remplacée par la proportion de chaque classe.

L'estimateur MV d'une loi exponentielle tronquée ne peut être calculé directement. Nous avons choisi d'estimer d'abord la loi exponentielle sur l'intervalle $(0, \infty)$, puis de l'ajuster en calculant son intégrale de 0 à $\frac{\pi}{2}$ par la méthode de Simpson 1/3.

7.3 Segmentation

Pour le calcul d'une segmentation optimale au sens du MAP, notre modèle ne permet pas l'utilisation de l'algorithme de Viterbi. Afin de faciliter notre tâche, nous introduisons une version simplifiée de notre modèle. On considère une décomposition de T en une famille $\mathcal{P} = \{p_\alpha\}$ de chemins simples non-vides disjoints (dans G) avec la propriété que chaque chemin est maximal. Une telle décomposition n'est pas unique, mais nous en fixons une pour l'image traitée. On définit alors T' comme l'ensemble des segments (s,t) apparaissant dans les chemins de $\mathcal{P}$.

Notre modèle simplifié consiste à considérer la distribution *a priori*

$$\begin{aligned} P_X(x) &= \frac{1}{Z}\exp\Big\{-\sum_{<s,t>:(s,t)\in T'}\big(1-\delta(x_s,x_t)\big)\Big\} \\ &\quad \prod_{s\notin T}\delta(x_s,e_1)\prod_{(s,t)\notin T'}\delta(x_{st},e_1)\prod_{(s,t)\in T'}\eta(x_s,x_t,x_{st}). \end{aligned}$$

Notez qu'on a ainsi ignoré tous les points de jonction (voir Figure 7.7).

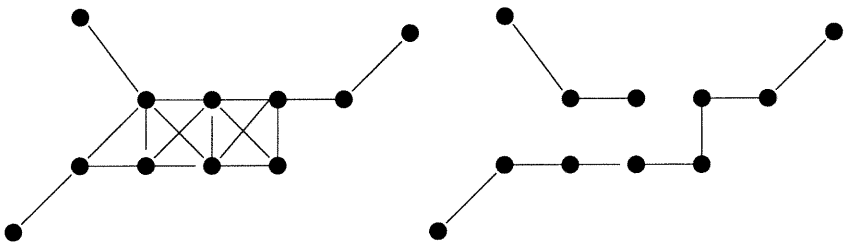


FIG. 7.7 – *À gauche : ensemble T^* de segments. À droite : un ensemble T' de segments possible parmi tant d'autres.*

Soit U' le champs de Gibbs défini par

$$\begin{aligned} U'(x,y) &= \sum_{s\in V_G} -\ln P_{Y_s \mid X_s}(y_s \mid x_s) + \sum_{(s,t)\in E_G} -\ln P_{Y_{st} \mid X_{st}}(y_{st} \mid x_{st}) \\ &+ \sum_{<s,t>:(s,t)\in T^*} \Big(1-\delta(x_s,x_t)\Big) \end{aligned}$$

et V' la fonction donnée par

$$\begin{aligned} V'(x) &= \sum_{s\notin T}\Big(1-\delta(x_s,e_1)\Big) + \sum_{(s,t)\notin T'}\Big(1-\delta(x_{st},e_1)\Big) \\ &+ \sum_{(s,t)\in T'}\Big(1-\eta(x_s,x_t,x_{st})\Big). \end{aligned}$$

La distribution *a posteriori* du modèle simplifié des contours s'exprime sous la forme

$$P_{X/Y}(x/y) = \frac{1}{Z}\exp\Big(-U'(x,y)\Big)\,\chi(\{x \,:\, V'(x) \text{ est minimal}\},x),$$

où Z est une constante de normalisation.

Selon ce modèle simplifié, la segmentation d'une image en contours est équivalente à la minimisation du champ de Gibbs U' sous la contrainte que la fonction V' soit minimale. Plutôt que d'utiliser le recuit simulé sous contrainte [30], nous utilisons pour le modèle simplifié l'algorithme de Viterbi [25] sur chaque chemin p de T, obtenu à l'étape de pré-segmentation, de la façon suivante.

Soit $p = (s_0, s_1, ..., s_n)$ un chemin dans $\mathcal{P}$. On considère la fonction

$$\begin{aligned} U_p(x,y) &= \sum_{t=0}^{n} -\ln P_{Y_{s_t}/X_{s_t}}(y_{s_t}/e_{i_t}) \\ &+ \sum_{t=1}^{n} -\ln P_{Y_{s_{t-1},s_t}/X_{s_{t-1},s_t}}(y_{s_{t-1},s_t}/e_{i_{t-1},i_t}) + \big(1 - \delta(e_{i_{t-1}}, e_{i_t})\big) \end{aligned}$$

avec chaque e_{i_{t-1},i_t} ajusté de telle sorte que $\eta(e_{i_{t-1}}, e_{i_t}, e_{i_{t-1},i_t}) = 1$. Notre problème de minimisation sous contrainte est alors équivalent à minimiser la fonction

$$\sum_{s \in V_G \setminus T} -\ln P_{Y_s/X_s}(y_s/e_1) + \sum_{(s,t) \in E_G \setminus T'} -\ln P_{Y_{st}/X_{st}}(y_{st}/e_1) + \sum_{p \in \mathcal{P}} U_p(x,y).$$

Ainsi, on est ramené à minimiser indépendemment chaque fonction U_p.

Pour ce faire, on considère pour chaque chemin simple p la fonction

$$l_p(i_0, ..., i_n) = -\ln P_{Y_{s_0} \mid X_{s_0}}(y_{s_0} \mid e_{i_0}) + \sum_{t=1}^{n} \lambda_t(i_{t-1}, i_t),$$

où le terme $\lambda_t(i,j)$ est défini par

$$-\ln P_{Y_{s_t} \mid X_{s_t}}(y_{s_t} \mid e_j) - \ln P_{Y_{s_{t-1},s_t} \mid X_{s_{t-1}s_t}}(y_{s_{t-1},s_t} \mid e_{i,j}) + 1 - \delta(e_i, e_j)$$

avec $e_{i,j}$ ajusté de telle sorte que $\eta(e_i, e_j, e_{i,j}) = 1$. Le problème de minimisation considéré (pour le modèle simplifié) est alors équivalent à la minimisation des fonctions l_p indépendamment sur chaque chemin. On utilise alors l'algorithme de Viterbi sur chaque chemin, ce qui donne une segmentation optimale au sens du MAP pour le modèle simplifié.

Notez que cette solution n'est pas optimale pour le modèle complet qui tient compte des points de jonction. Ceci dit, pour notre méthode de localisation de formes présentée au chapitre 9, nous n'utilisons pas la segmentation de l'image au sens des contours, mais plutôt l'estimation des paramètres.

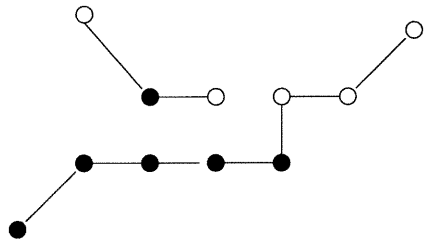


FIG. 7.8 – *Illustration d'une segmentation de deux chemins.*

7.4 Résultats empiriques

Dans notre implantation, nous normalisons les niveaux de gris entre 0 et 255, puis nous appliquons un masque gaussien 3×3. Finalement, nous normalisons la norme du gradient entre 0 et 100. Ainsi, min $= -0.001$. Nous fixons le nombre d'itérations à 10 pour la procédure ECI, à 100 pour chaque application de l'algorithme SEM et à au plus 30 pour chaque estimation de la loi de Weibull. La méthode prend au total 22sec, 37sec, 3sec, 64sec, 76sec, sur une station 1,2GHz pour les images présentées aux figures 7.9, 7.12, 7.15, 7.18, 7.21, respectivement. Le temps de calcul ne dépend que de la taille de l'image. Voir les tableaux 7.1, 7.2, 7.3, 7.4, 7.5, et les figures 7.11, 7.14, 7.17, 7.20, 7.23, pour les valeurs estimées des paramètres et une comparaison entre les distributions empiriques et estimées.

Nous présentons une comparaison de notre méthode avec celle de Canny, aux figures 7.10, 7.13, 7.16, 7.19, 7.22. Notez que dans le cas de l'algorithme de Canny, il faut spécifier les seuils τ_l et τ_h pour l'hystérisis (voir section 2.1.1). Pour trouver ces seuils, on peut considérer la fonction de répartition empirique $\hat{F}$ de la norme du gradient des niveaux de gris par rapport à la position des pixels. Si p_l et p_h sont deux nombres dans l'intervalle $[0, 1]$, on prend alors τ_h tel que $\hat{F}(\tau_h) = p_h$, et on pose $\tau_l = p_l \tau_h$. On peut penser qu'il s'agit là d'une façon de déterminer automatiquement les seuils τ_l et τ_h, mais encore faut-il choisir p_l et p_h. En définitive, la méthode de Canny est supervisée car elle ne repose pas sur un modèle statistique. Par contre, elle est beaucoup plus rapide que notre méthode (1sec sur une station 1,2GHz).

7.5 Conclusion

Notre méthode offre non seulement l'avantage d'effectuer une détection de contours non-supervisée (contrairement à la méthode de Canny), mais permet de plus d'attacher à chaque pixel de l'image une vraisemblance d'être un point hors-contours ou un point de contours. Il s'agit donc d'une détection de contours probabiliste. Ce dernier point jouera un rôle essentiel dans notre version originale de deux méthodes déjà existantes d'extraction semi-automatique de contours présentées au chapitre 8, ainsi que de notre méthode originale de localisation de formes présentée au chapitre 9. La qualité de détection de contours de notre méthode est comparable à celle offerte par la méthode de Canny. Par contre, notre méthode est plus lente que celle de Canny, car elle nécessite l'estimation de paramètres statistiques.

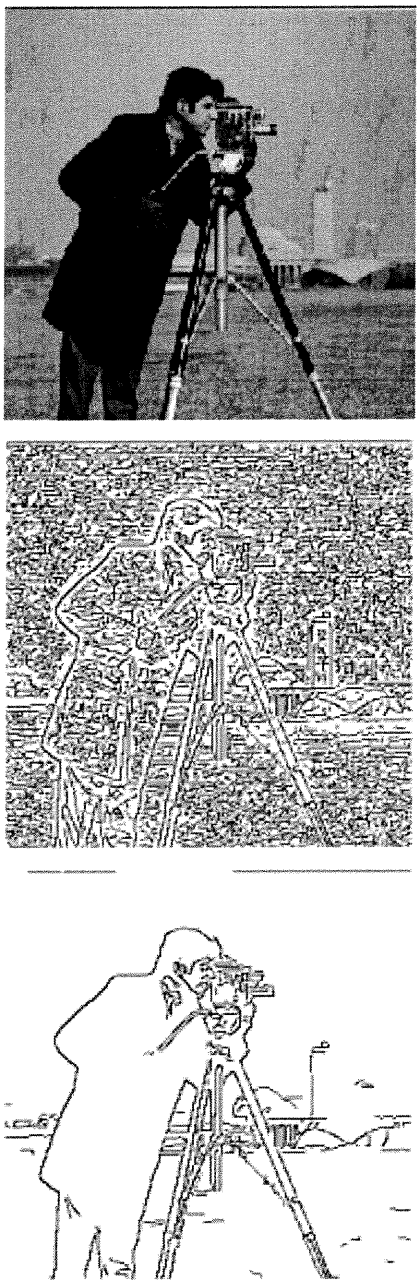


FIG. 7.9 – *Exemple d'une détection non-supervisée de contours à l'aide de l'algorithme de Viterbi basée sur les paramètres estimés par la procédure ECI. De haut en bas : image originale ; pré-segmentation basée sur le gradient ; contours détectés dans l'image.*

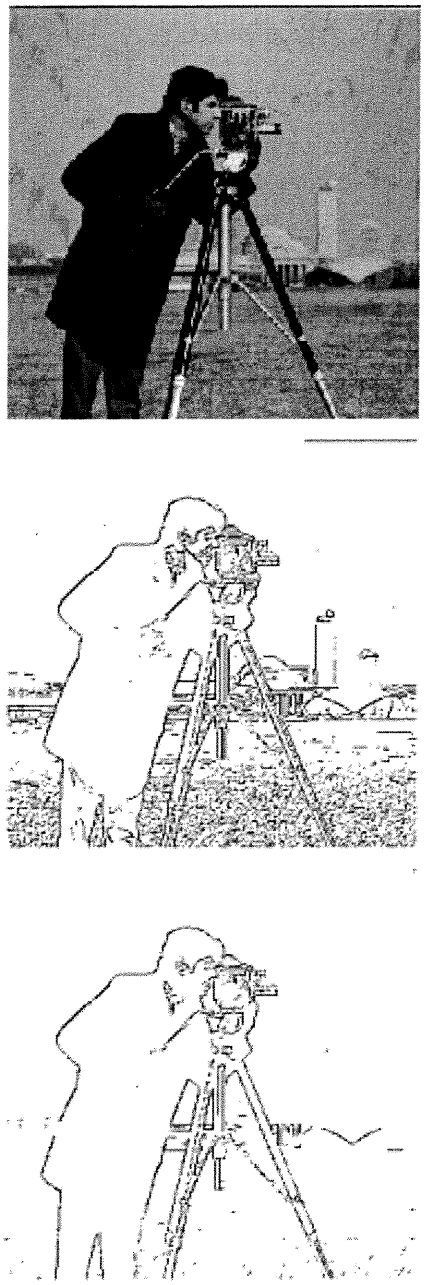


FIG. 7.10 – *Exemple de détection supervisée de contours à l'aide de l'algorithme de Canny. De haut en bas : image originale ; contours détectés avec* $\tau_l = 11.5$ *et* $\tau_h = 23$ *; contours détectés avec* $\tau_l = 38$ *et* $\tau_h = 76$.

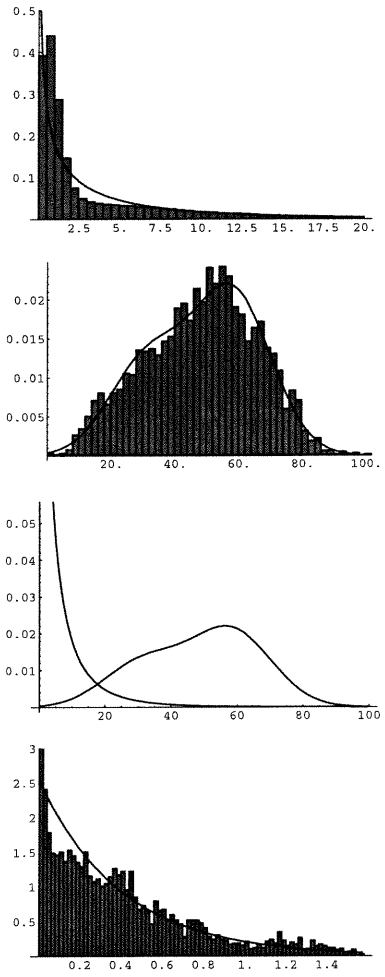


FIG. 7.11 – *Exemple des distributions empiriques et des distributions estimées par la procédure ECI pour l'image de la figure 7.9. De haut en bas : norme du gradient pour les points hors contours ; norme du gradient pour les points de contours ; comparaison entre les deux distributions ; angle entre le gradient et la normale à la courbe de contours.*

min	C	α
-0.001	0.639748	3.61411
w_1	μ_1	σ_1
0.634215	58.0508	147.16
w_2	μ_2	σ_2
0.365784	31.6062	140.601
α_0	k_0	ν
0.400958	1.02029	107.366

TAB. 7.1 – *Valeurs des paramètres estimés par la procédure ECI pour l'image de la figure 7.9.*

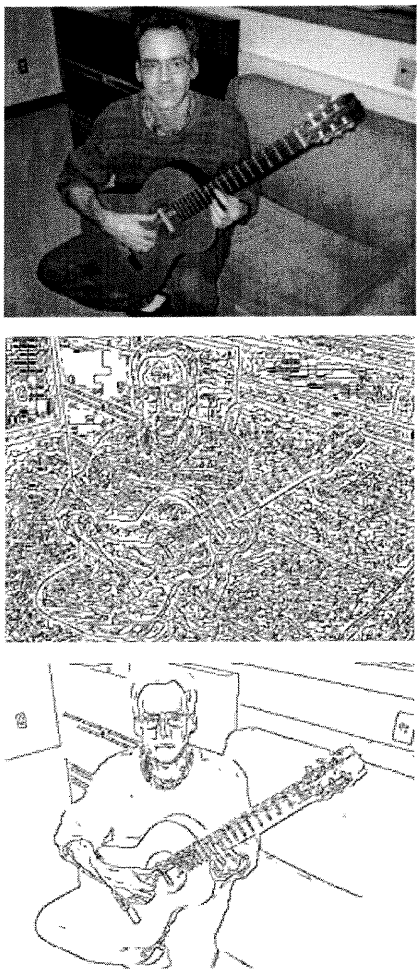


FIG. 7.12 – *Exemple d'une détection non-supervisée de contours à l'aide de l'algorithme de Viterbi basée sur les paramètres estimés par la procédure ECI. De haut en bas : image originale ; pré-segmentation basée sur le gradient ; contours détectés dans l'image.*

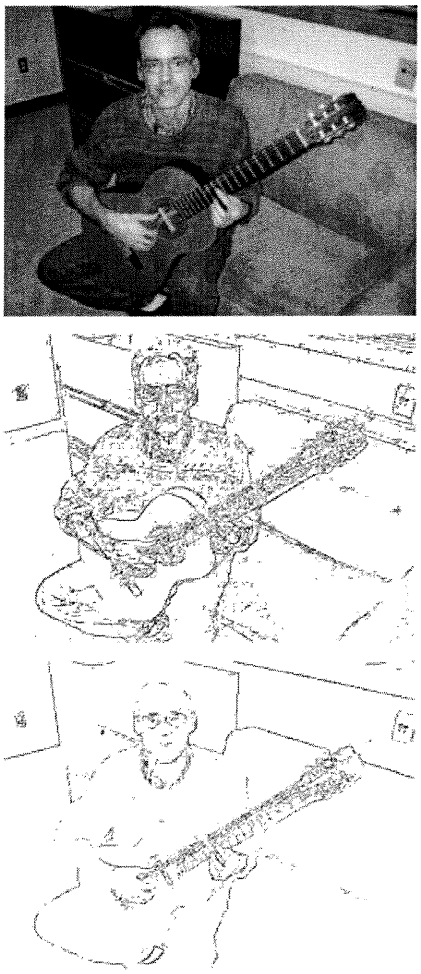


FIG. 7.13 – *Exemple de détection supervisée de contours à l'aide de l'algorithme de Canny. De haut en bas : image originale; contours détectés avec* $\tau_l = 7$ *et* $\tau_h = 14$*; contours détectés avec* $\tau_l = 22$ *et* $\tau_h = 44$.

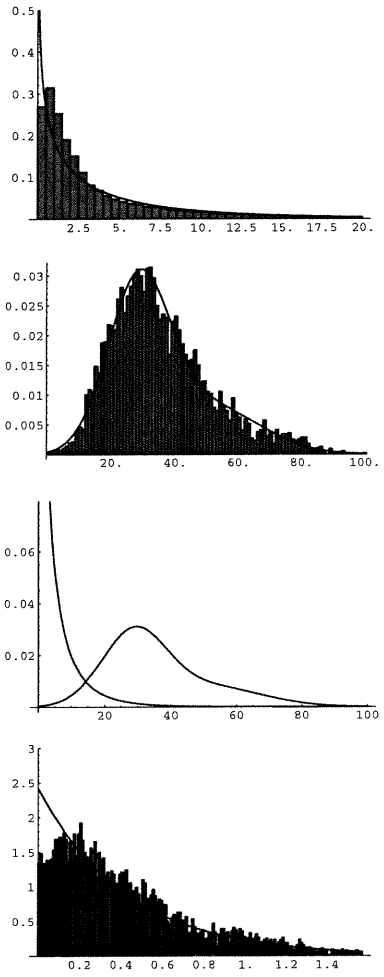


FIG. 7.14 – *Exemple des distributions empiriques et des distributions estimées par la procédure ECI pour l'image de la figure 7.12. De haut en bas : norme du gradient pour les points hors contours ; norme du gradient pour les points de contours ; comparaison entre les deux distributions ; angle entre le gradient et la normale à la courbe de contours.*

min	C	α
-0.001	0.715127	3.80892
w_1	μ_1	σ_1
0.331191	50.5294	268.267
w_2	μ_2	σ_2
0.668809	28.7663	93.2973
α_0	k_0	ν
0.422855	1.02497	96.6096

TAB. 7.2 – *Valeurs des paramètres estimés par la procédure ECI pour l'image de la figure 7.12.*

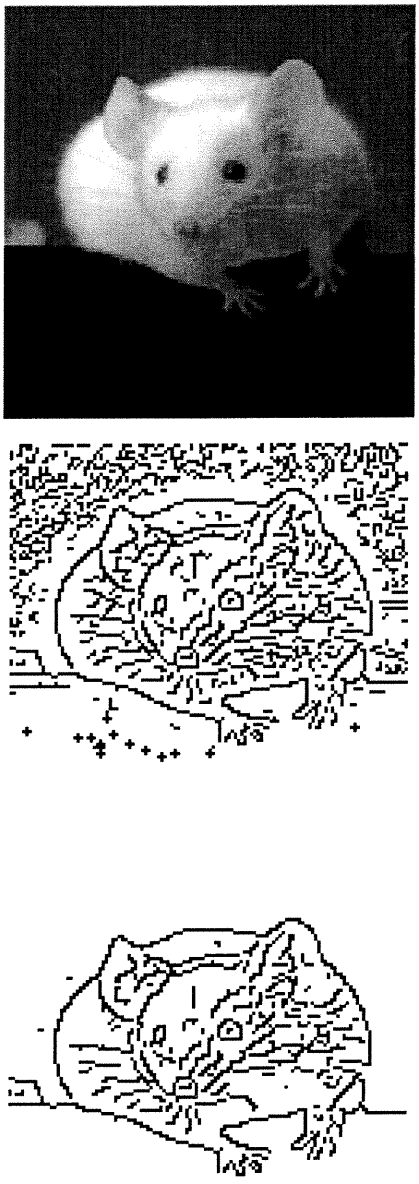


FIG. 7.15 – *Exemple d'une détection non-supervisée de contours à l'aide de l'algorithme de Viterbi basée sur les paramètres estimés par la procédure ECI. De haut en bas : image originale ; pré-segmentation basée sur le gradient ; contours détectés dans l'image.*

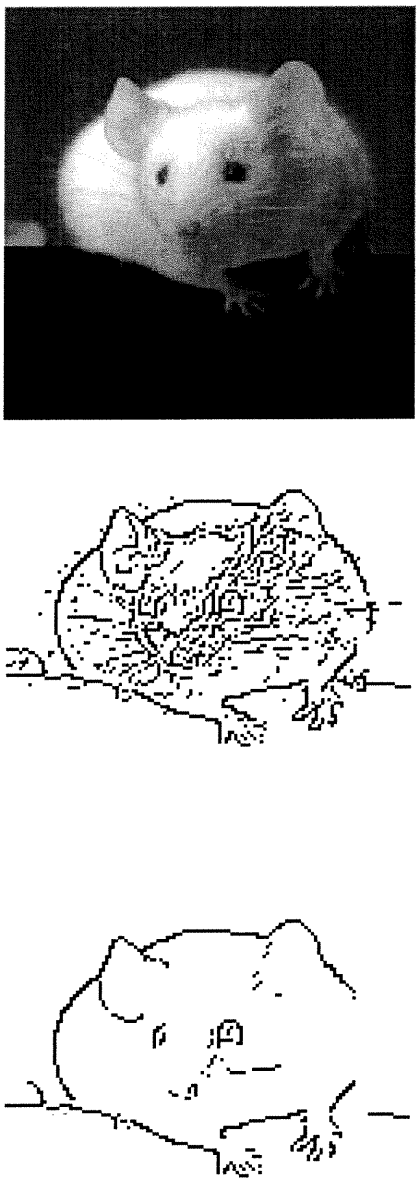


FIG. 7.16 – *Exemple de détection supervisée de contours à l'aide de l'algorithme de Canny. De haut en bas : image originale ; contours détectés avec* $\tau_l = 9.5$ *et* $\tau_h = 19$ *; contours détectés avec* $\tau_l = 27$ *et* $\tau_h = 54$.

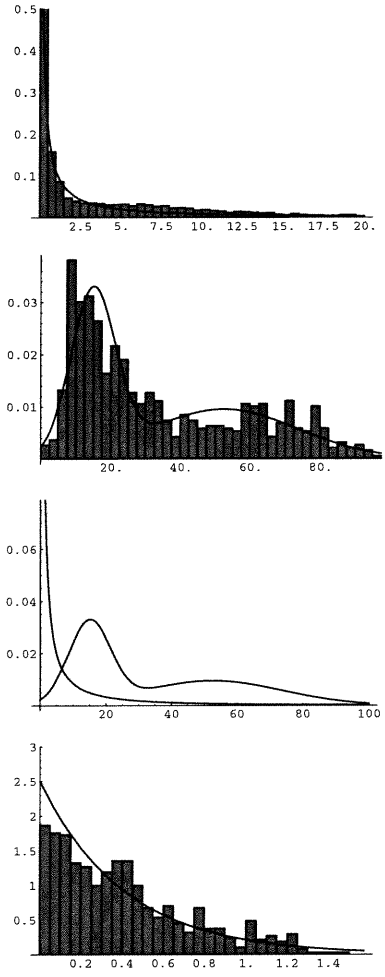


FIG. 7.17 – *Exemple des distributions empiriques et des distributions estimées par la procédure ECI pour l'image de la figure 7.15. De haut en bas : norme du gradient pour les points hors contours ; norme du gradient pour les points de contours ; comparaison entre les deux distributions ; angle entre le gradient et la normale à la courbe de contours.*

min	C	α
-0.001	0.30901	1.11153
w_1	μ_1	σ_1
0.495364	14.9943	40.3576
w_2	μ_2	σ_2
0.504636	52.4631	445.97
α_0	k_0	ν
0.407792	1.0217	67.6705

TAB. 7.3 – *Valeurs des paramètres estimés par la procédure ECI pour l'image de la figure 7.15.*

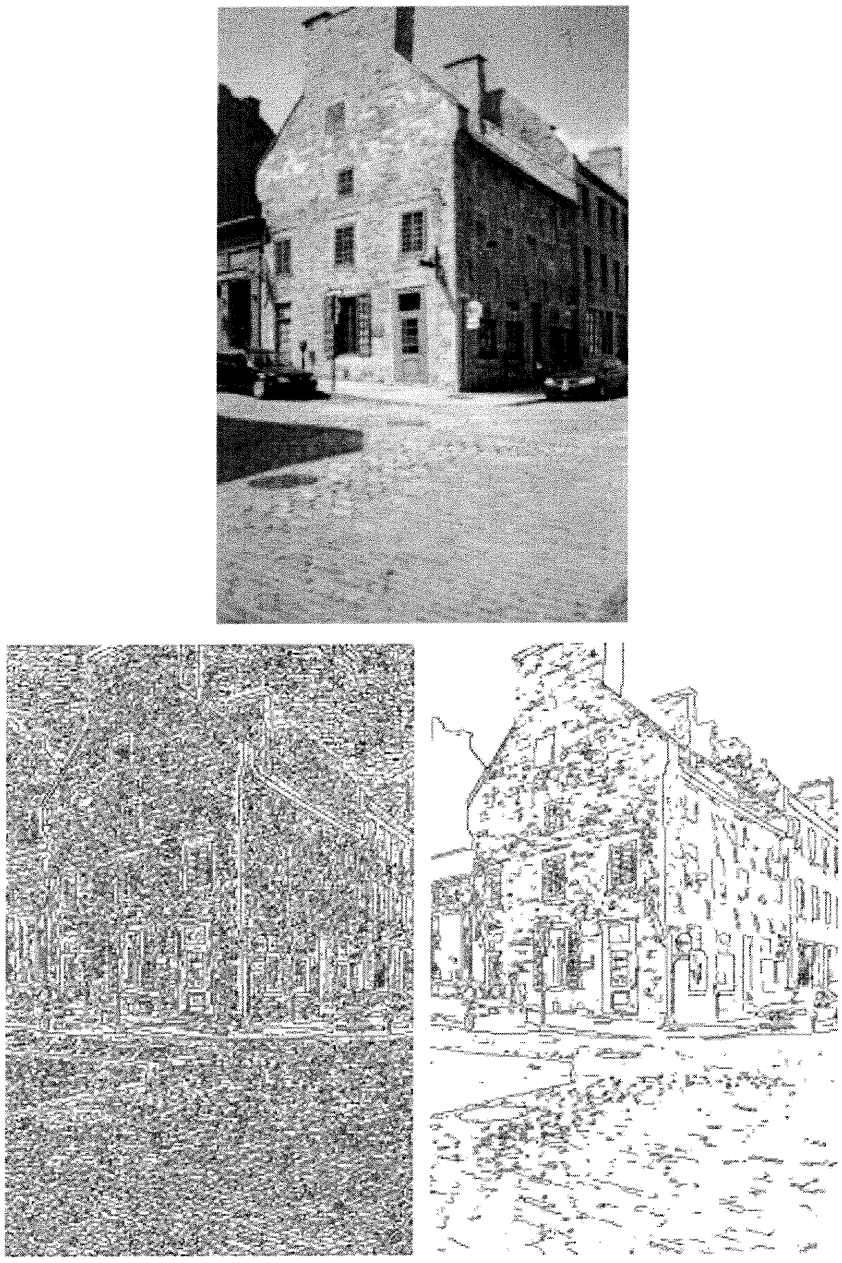


FIG. 7.18 – *Exemple d'une détection non-supervisée de contours à l'aide de l'algorithme de Viterbi basée sur les paramètres estimés par la procédure ECI. De haut en bas : image originale ; pré-segmentation basée sur le gradient ; contours détectés dans l'image.*

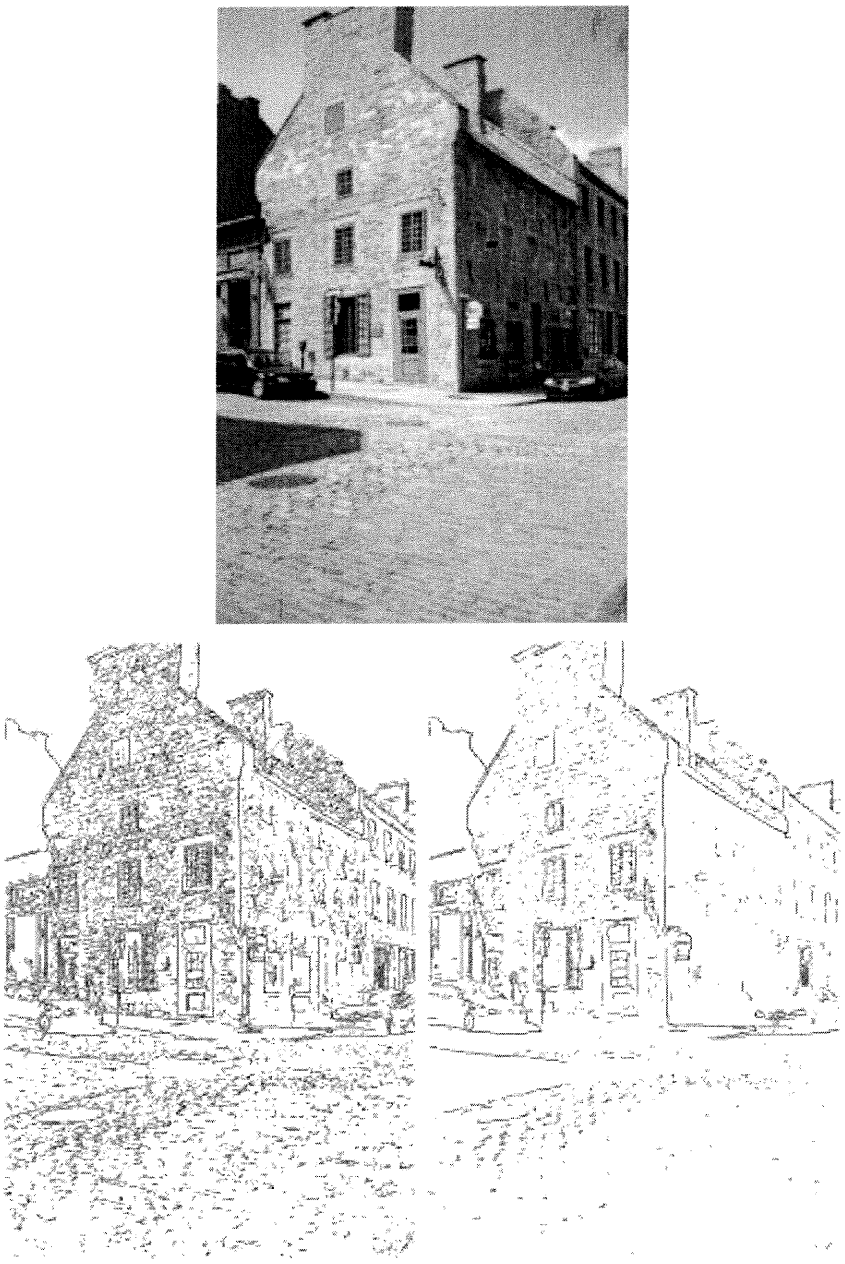


FIG. 7.19 – *Exemple de détection supervisée de contours à l'aide de l'algorithme de Canny. De haut en bas : image originale ; contours détectés avec* $\tau_l = 18.5$ *et* $\tau_h = 37$ *; contours détectés avec* $\tau_l = 40$ *et* $\tau_h = 80$.

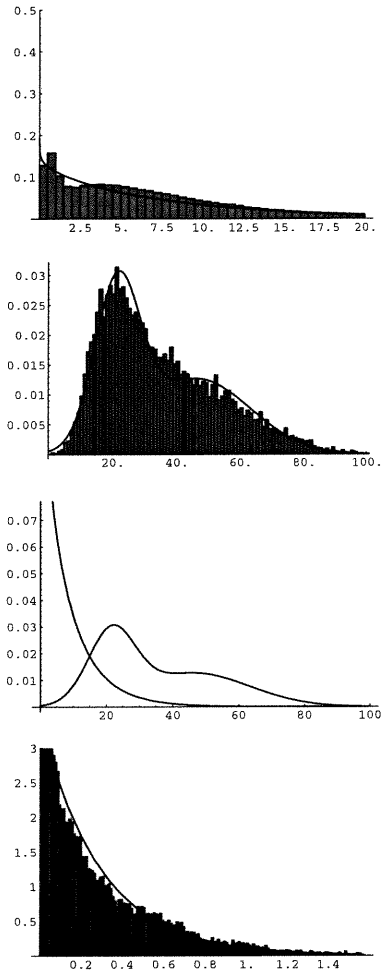


FIG. 7.20 – *Exemple des distributions empiriques et des distributions estimées par la procédure ECI pour l'image de la figure 7.18. De haut en bas : norme du gradient pour les points hors contours ; norme du gradient pour les points de contours ; comparaison entre les deux distributions ; angle entre le gradient et la normale à la courbe de contours.*

min	C	α
-0.001	0.94424	8.10021
w_1	μ_1	σ_1
0.465343	21.5961	49.4592
w_2	μ_2	σ_2
0.534657	46.4591	283.415
α_0	k_0	ν
0.294376	1.00484	88.9733

TAB. 7.4 – *Valeurs des paramètres estimés par la procédure ECI pour l'image de la figure 7.18.*

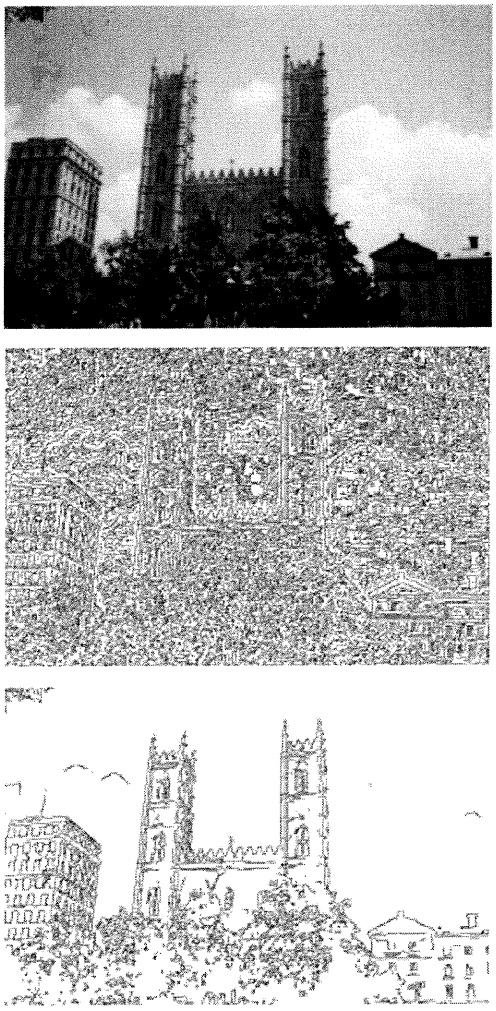


FIG. 7.21 – *Exemple d'une détection non-supervisée de contours à l'aide de l'algorithme de Viterbi basée sur les paramètres estimés par la procédure ECI. De haut en bas : image originale ; pré-segmentation basée sur le gradient ; contours détectés dans l'image.*

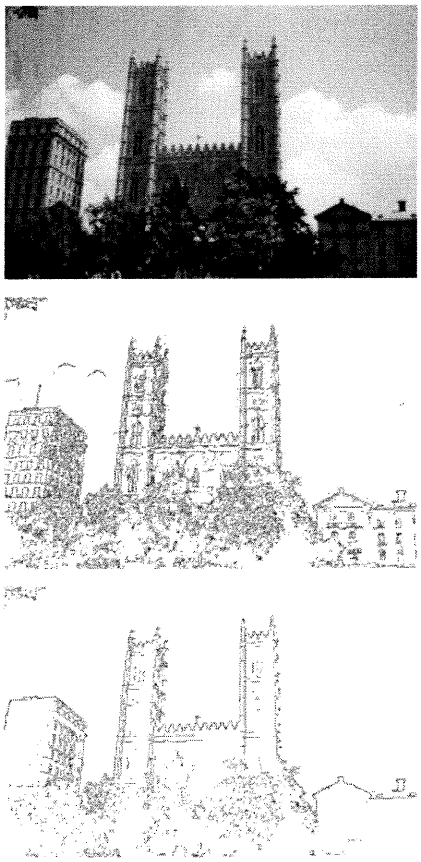


FIG. 7.22 – *Exemple de détection supervisée de contours à l'aide de l'algorithme de Canny. De haut en bas : image originale ; contours détectés avec* $\tau_l = 10.5$ *et* $\tau_h = 21$ *; contours détectés avec* $\tau_l = 30$ *et* $\tau_h = 61$.

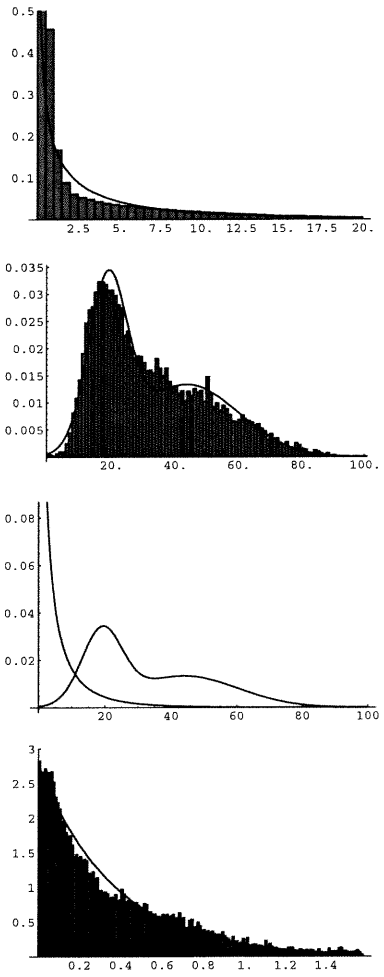


FIG. 7.23 – *Exemple des distributions empiriques et des distributions estimées par la procédure ECI pour l'image de la figure 7.21. De haut en bas : norme du gradient pour les points hors contours ; norme du gradient pour les points de contours ; comparaison entre les deux distributions ; angle entre le gradient et la normale à la courbe de contours.*

min	C	α
-0.001	0.607237	3.20307
w_1	μ_1	σ_1
0.455085	19.0735	35.9777
w_2	μ_2	σ_2
0.544915	44.0365	265.116
α_0	k_0	ν
0.36534	1.01376	93.1732

TAB. 7.5 – *Valeurs des paramètres estimés par la procédure ECI pour l'image de la figure 7.21.*

Chapitre 8

Extraction de contours

Dans ce chapitre, nous présentons un modèle markovien pour les chemins discrets dans une image [20], basé sur le modèle statistique des contours du chapitre 7. Nous montrons comment utiliser ce modèle pour l'extraction semi-automatique de contours à partir des positions initiale et finale de la courbe en posant un modèle *a priori* simple qui impose une restriction sur la longueur de la courbe. Il s'agit donc d'une variation de l'algorithme du live-wire de Mortensen et al [46] et [45]. Nous montrons également que notre modèle s'adapte bien à l'algorithme du jetstream de Pérez et al [47]. Il s'agit d'une méthode d'extraction de contours à partir des position et direction initiales de la courbe.

Dans le cas du live-wire, les positions initiale et finale sont fournies par l'usager à l'aide de la souris. Pour le jetstream, c'est également la souris qui permet à l'usager de spécifier les position et direction initiales de la courbe.

8.1 Algorithme du live-wire

8.1.1 Modèle des chemins discrets

Les variables aléatoires Y_s et Y_{st} sont définies comme à la section 7.1 et décrivent la distribution du gradient des niveaux de gris. Les variables X_s, X_{st} prennent leurs valeurs dans l'ensemble $\{e_1 = \text{“hors-contours”}, e_2 = \text{“contours”}\}$.

Étant donné un chemin $c = (s_0, ..., s_n)$ dans le graphe G de la section 7.1, nous considérons semblablement à [47] et [32], une fonction de vraisemblance $P_{Y \mid C}(y \mid c)$ donnée par

$$\begin{aligned}
&\textstyle\prod_{s\notin c} P_{Y_s \mid X_s}(y_s \mid e_1) \prod_{s\in c} P_{Y_s \mid X_s}(y_s \mid e_2) \\
&\textstyle\prod_{(s,t)\notin c} P_{Y_{st} \mid X_{st}}(y_{st} \mid e_1) \prod_{(s,t)\in c} P_{Y_{st} \mid X_{st}}(y_{st} \mid e_2) \\
= \quad & k(y) \textstyle\prod_{s\in c} \frac{P_{Y_s \mid X_s}(y_s \mid e_2)}{P_{Y_s \mid X_s}(y_s \mid e_1)} \prod_{(s,t)\in c} \frac{P_{Y_{st} \mid X_{st}}(y_{st} \mid e_2)}{P_{Y_{st} \mid X_{st}}(y_{st} \mid e_1)} \\
= \quad & k(y) \textstyle\prod_{i=0}^{n} \frac{\mathcal{M}(y_{s_i}; w_j, \mu_j, \sigma_j)}{\mathcal{W}(y_{s_i}; \min, C, \alpha)} \prod_{i=1}^{n} \frac{k_0\, \mathcal{E}(y_{s_{i-1}, s_i}; \alpha_0)}{\mathcal{U}(y_{s_{i-1}, s_i}; 0, \frac{\pi}{2})}
\end{aligned}$$

où

$$k(y) = \prod_{s\in V_G} P_{Y_s \mid X_s}(y_s \mid e_1) \prod_{(s,t)\in E_G} P_{Y_{st} \mid X_{st}}(y_{st} \mid e_1).$$

Notez que le facteur $k(y)$ ne dépend que de l'image et non de la courbe. De plus le facteur principal

$$\prod_{i=0}^{n} \frac{\mathcal{M}(y_{s_i}; w_j, \mu_j, \sigma_j)}{\mathcal{W}(y_{s_i}; \min, C, \alpha)} \prod_{i=1}^{n} \frac{k_0\, \mathcal{E}(y_{s_{i-1}, s_i}; \alpha_0)}{\mathcal{U}(y_{s_{i-1}, s_i}; 0, \frac{\pi}{2})}$$

est invariant sous transformations affines des niveaux de gris.

De plus, nous considérons un modèle *a priori* qui impose une restriction sur la longueur du chemin entre les positions initiale et finale ; c'est-à-dire

$$P_C(c) = \frac{1}{Z} \exp(-\rho\, |c|)$$

où $|c|$ est le nombre de segments dans le chemin c, et Z est un facteur de normalisation. Sans un tel modèle *a priori*, l'algorithme ne choisirait pas le plus court chemin de contour entre deux positions spécifiées, mais plutôt le plus long. Pour notre application, il faut donc imposer un tel modèle en prenant ρ suffisamment grand. On explique ci-dessous comment on choisit la valeur de ρ en pratique.

On obtient donc la distribution *a posteriori*

$$P_{C\,|\,Y}(c\,|\,y) \;\propto\; P_{Y\,|\,C}(y\,|\,c)P_C(c) \;\propto\; \exp(-l(c,y)).$$

Le champ de Gibbs correspondant s'exprime à une constante près, par

$$l(c,y) \;=\; l(s_1,...,s_n) \;=\; \sum_{i=1}^{n} \lambda(s_{i-1},s_i),$$

où $\lambda(s,t)$ est défini par

$$\log(\mathcal{W}(y_s;\min,C,\alpha)) - \log(\mathcal{M}(y_s;w_j,\mu_j,\sigma_j))$$
$$+\log(\mathcal{U}(y_{st};0,\tfrac{\pi}{2})) - \log(k_0\,\mathcal{E}(y_{st};\alpha_0)) + \rho.$$

Nous choisissons ρ minimal tel que $\lambda(s,t) \geq 0$. Plus précisément, on calcule $m_1 = \min_{s,t} f(s,t)$, où

$$\begin{aligned} f(s,t) \;=\; & \log(\mathcal{W}(y_s;\min,C,\alpha)) - \log(\mathcal{M}(y_s;w_j,\mu_j,\sigma_j)) \\ & +\log(\mathcal{U}(y_{st};0,\frac{\pi}{2})) - \log(k_0\,\mathcal{E}(y_{st};\alpha_0)), \end{aligned}$$

et on pose $\rho = -m_1$. Dès lors, nous pouvons utiliser l'algorithme de Dijkstra [23] (voir section 4.1) afin de trouver un chemin de coût minimal entre un pixel de départ s_0 et un pixel t quelconque de l'image, tout comme dans l'algorithme de Mortensen et al [46] et [45]. Toutefois les fonctions de coût ne sont plus définies de façon heuristique.

Dans notre implantation de l'algorithme, on normalise la fonction λ entre 0 et 1, en posant

$$\lambda(s,t) \;=\; (f(s,t)+\rho)/(m_2-m_1)$$

où $m_2 = \max_{s,t} f(s,t)$. Notez que ce changement d'échelle ne modifie pas en principe le chemin optimal, mais permet d'uniformiser la précision avec laquelle on travaille en pratique.

8.1.2 Résultats empiriques

Il est parfois nécessaire de figer manuellement une partie du contour. C'est également le cas dans la version originale de l'algorithme. Notre méthode semble fonctionner tout aussi bien que la version originale. Ainsi, cette application permet de valider notre modèle statistique. Voir la figure 8.1 pour un exemple.

FIG. 8.1 – *Exemple d'extraction semi-automatique de contours à l'aide de l'algorithme de Dijkstra et basée sur les paramètres estimés par la procédure ECI. On y indique les positions initiale et finale spécifiées par l'usager à l'aide de la souris.*

8.2 Algorithme du jetstream

8.2.1 Modèle des chemins continus

Étant donné une image de taille $M \times N$, on considère le graphe infini indénombrable $G = (\Lambda, E)$, où $\Lambda = [0, M] \times [0, N]$, et $(x_1, x_2) \in E$ si et seulement si $\| x_1 - x_2 \| = 1$. Si $x \in \Lambda$, le gradient des niveaux de gris en x est obtenu par interpolation bilinéaire du gradient aux points de Λ à coordonnées entières, c'est-à-dire correspondant aux pixels de l'image.

Si x est un site de G, la variable aléatoire Y_x représente la norme du gradient des niveaux de gris en x. Si x et x' sont deux sites adjacents de G, la variable aléatoire $Y_{xx'}$ représente la valeur absolue de l'angle entre la moyenne du gradient en x et x', et le vecteur allant de x à x'. Cet angle est normalisé entre $-\pi/2$ et $\pi/2$.

Si $x_{0:n} = (x_0, ..., x_n)$ est un chemin dans le graphe G, nous considérons comme à la section 8.1, la fonction de vraisemblance donnée à un facteur près par

$$P_{Y \mid X_{0:n}}(y \mid x_{0:n}) \quad \propto \quad \prod_{i=0}^{n} \frac{\mathcal{M}(y_{x_i}; w_j, \mu_j, \sigma_j)}{\mathcal{W}(y_{x_i}; \min, C, \alpha)} \prod_{i=1}^{n} \frac{k_0 \, \mathcal{E}(y_{x_{i-1}, x_i}; \alpha_0)}{\mathcal{U}(y_{x_{i-1}, x_i}; 0, \frac{\pi}{2})}.$$

De plus, comme en [47], nous posons comme distribution *a priori* d'un chemin

$$P_{X_{0:n}}(x_{0:n}) \quad \propto \quad \prod_{i=1}^{n} g(\psi_i; \sigma_\psi)$$

où ψ_i est l'angle entre les vecteurs (x_{i-1}, x_i) et (x_{i-2}, x_{i-1}) (normalisé entre $-\pi$ et π), si $i \geq 2$, ou l'angle entre (x_{i-1}, x_i) et une direction initiale v, si $i = 1$. Ici, la distribution $g(\psi_i; \sigma_\psi)$ est définie par

$$\mathcal{N}(\psi_i; 0, \sigma_\psi) + c\delta(\psi_i), \qquad -\pi/2 \leq \psi_i \leq \pi/2$$

où δ dénote la distribution delta de Dirac, et $c = 1 - \int_{[-\frac{\pi}{2}, \frac{\pi}{2}]} \mathcal{N}(y; 0, \sigma_\psi) \, dy$. Nous avons fixé la valeur de σ_ψ à 0.1. Si la valeur de σ_ψ est trop petite, l'algorithme préfère des contours rectilignes.

On obtient alors la distribution *a posteriori*

$$P_{X_{0:n} \mid Y}(x_{0:n} \mid y) \quad \propto \quad P_{Y \mid X_{0:n}}(y \mid x_{0:n})\, P_{X_{0:n}}(x_{0:n}).$$

8.2.2 Extraction de contours

Soient x un point de départ, v une direction initiale, et n une longueur donnée. On cherche parmi tous les chemins de longueur n avec condition initiales x et v, celui qui maximise la distribution *a posteriori* $P_{X_{0:n} \mid Y}(x_{0:n} \mid y)$. Nous adoptons la méthode du jetstream de Pérez et al [47] présentée au chapitre 2.

On pose donc

$$\begin{aligned} g_1(x_{0:n+1}) &= P_{X_{0:n+1} \mid Y}(x_{0:n+1} \mid y), \\ g_2(x_{0:n+1}) &= P_{X_{0:n} \mid Y}(x_{0:n} \mid y) f(x_{n+1} \mid x_{0:n}), \end{aligned}$$

en prenant pour f la distribution

$$f(x_{n+1} \mid x_{0:n}) = g(\psi_i; \sigma_\psi).$$

On obtient alors les poids

$$\pi_{n+1}^{(i)} \quad = \quad \frac{g_1(x_{0:n+1})}{g_2(x_{0:n+1})} \quad = \quad k_0 \frac{\mathcal{M}(y_{x_{n+1}^{(i)}}; w_j, \mu_j, \sigma_j)}{\mathcal{W}(y_{x_{n+1}^{(i)}}; \min, C, \alpha)} \frac{\mathcal{E}(y_{x_n^{(i)}, x_{n+1}^{(i)}}; \alpha_0)}{\mathcal{U}(y_{x_n^{(i)}, x_{n+1}^{(i)}}; 0, \frac{\pi}{2})}.$$

Notons que la distribution f est facile à simuler. Dans la version originale du jetstream [47], la fonction f est légèrement différente.

8.2.3 Résultats empiriques

Dans notre implantation, le calcul des probabilités $P_{X_{0:n} \mid Y}(x_{0:n}^{(i)} \mid y)$ est effectué récursivement. Il est important de normaliser ces probabilités entre 0 et 1 à chaque étape, afin d'éviter d'être à court de précision numérique ! De plus, tout comme pour l'algorithme original, il est parfois nécessaire de reprendre une partie du contour en

spécifiant à nouveau la position et la direction de départ. Nous présentons à la figure 8.2 un exemple d'extraction de contours à l'aide de notre version de l'algorithme jetstream. Nous avons choisi $m = n = 100$ pour nos tests.

Algorithme Jetstream (notre version)

But de l'algorithme : extraire une courbe de contours en spécifiant les position et direction initiales.

1. **Initialisation** : $k = 0$. On pose $x_{0:0}^{(i)} = x$, pour $i = 1, 2, ..., m$, x étant la position initiale.
2. **Simulation** : Pour chaque $i = 1, 2, ...m$, prolonger $x_{0:k}^{(i)}$ d'un point en simulant $f(x_{k+1} \mid x_{0:k}^{(i)})$. Si $k = 1$, utiliser la direction initiale v.
3. **Calcul des poids** : Pour chaque $i = 1, 2, ...m$, poser

$$\pi_{k+1}^{(i)} = k_0 \frac{\mathcal{M}(y_{x_{k+1}^{(i)}}; w_j, \mu_j, \sigma_j)}{\mathcal{W}(y_{x_{k+1}^{(i)}}; \min, C, \alpha)} \frac{\mathcal{E}(y_{x_k^{(i)}, x_{k+1}^{(i)}}; \alpha_0)}{\mathcal{U}(y_{x_k^{(i)}, x_{k+1}^{(i)}}; 0, \frac{\pi}{2})}.$$

4. **Échantillonnage** : Ré-échantillonner $x_{0:k+1}^{(1)}, ..., x_{0:k+1}^{(m)}$ selon les poids $\pi_{k+1}^{(i)}$.
5. Augmenter k de 1, et retour à 2 tant que k est plus petit que la longueur désirée.

8.3 Conclusion

Nous nous sommes intéressé au problème de l'extraction semi-automatique de contours, qu'en vue de valider notre modèle statistique des contours présenté au chapitre 7.

FIG. 8.2 – *Exemple d'une extraction semi-automatique de contours à l'aide de notre version de l'algorithme jetstream, basé sur les paramètres estimés par la procédure ECI. On y indique la position et la direction initiales spécifiées par l'usager à l'aide de la souris.*

Pour les deux algorithmes présentés dans ce chapitre, l'usager doit souvent scinder la courbe à extraire en plusieurs (petites) courbes. C'est là un fait inévitable, puisqu'une image présente plusieurs courbes de contours possibles, parmi lesquelles l'usager doit bien choisir, faute de modèle *a priori*. Notez qu'il existe toute une panoplie d'algorithmes pour faciliter l'interaction de l'usager avec le logiciel, que nous n'avons pas jugé pertinent d'étudier dans le cadre de ce mémoire.

En pratique, nous avons observé que notre version de l'algorithme du live-wire est plus efficace que notre version du jetstream ; il est nettement plus rapide et demande moins d'essais pour extraire un contour. De plus, dans le cas du jetstream, si la direction initiale est imprécise, l'algorithme ne pourra trouver la courbe de contours désirée.

Chapitre 9

Localisation de formes

Dans ce chapitre, nous présentons un modèle markovien original pour les déformations de formes [22] basé sur notre modèle statistique des chemins dans une image (voir chapitre 8), ainsi que sur l'analyse en composantes principales probabiliste de Tipping et Bishop (voir chapitre 6). On peut alors formuler le problème de la localisation d'une forme comme minimisation du champ de Gibbs de la distribution *a posteriori* des déformations de cette forme. Nous utilisons l'algorithme d'optimisation stochastique présenté au chapitre 5 pour effectuer cette tâche. On obtient ainsi une nouvelle méthode de localisation non-supervisée de formes.

9.1 Modèle des déformations d'une forme

9.1.1 Représentation des déformations d'une forme

On représente une courbe dans le plan par une suite de la forme

$$\gamma = (x_1, y_1, ..., x_d, y_d)$$

en prenant d points $(x_1, y_1), ..., (x_d, y_d)$ équidistants entre des points clés sur la courbe. Voir la figure 9.2 pour une illustration de points clés.

On considère un échantillon $\gamma_1, ..., \gamma_n$ de courbes représentant une même forme, et comportant un même nombre de points. Ces courbes sont obtenues à l'aide d'une version préliminaire de notre méthode d'extraction semi-automatique de contours présentée à la section 8.1, ainsi que d'une procédure automatique qui choisit un nombre fixé d'avance de points équidistants entre des points clés sur la courbe. Nous utilisons la méthode de Cootes [15] (voir section 2.3.3) pour aligner l'ensemble de ces courbes sur un patron de forme moyen.

On utilise ensuite la méthode présentée au chapitre 6 pour réduire la dimension de $2d$ à q. Pour déterminer la dimension réduite q, on majore l'erreur de reconstruction moyenne par une borne supérieure fixée d'avance. Considérant le vecteur aléatoire $T = (X_1, Y_1, ..., X_d, Y_d)$, on obtient ainsi un modèle optimal au sens du MV, de la forme $T = W\Xi + \nu + \varepsilon$, où $\varepsilon \sim \mathcal{N}(0, \sigma^2 I_d)$, W est une matrice $2d \times q$, et $\Xi \sim \mathcal{N}(0, I_q)$, en posant $W = U_q(\Lambda_q - \sigma^2 I_q)^{1/2}$, $\nu = \bar{t}$, et σ comme au chapitre 6. Nous obtenons ainsi les déformations non-linéaires de la forme moyenne $\gamma_0 = \bar{t}$, terminant ainsi la phase d'entraînement.

Nous considérons également les déformations rigides du plan données par les translations (τ_x, τ_y), les homothéties s, et les rotations ψ, appliquées point par point sur la courbe. On obtient ainsi un vecteur de déformation

$$\theta = (\tau_x, \tau_y, s, \psi, \xi)$$

de dimension $q + 4$. La transformation θ est calculée de la manière suivante : d'abord calculer $\gamma_\xi = U_q(\Lambda_q - \sigma^2 I_d)^{1/2}\xi + \bar{t}$; puis, appliquer s et ψ ; finalement, translater par τ. La courbe qui résulte de la transformation est dénotée γ_θ.

9.1.2 Distribution a priori

Soit Θ le vecteur aléatoire des déformations. Nous modélisons la distribution de Θ par

$$P_\Theta(\theta) = \mathcal{U}(\tau_x, \tau_y, s, \psi) P_T(U_q(\Lambda_1 - \sigma^2 I_q)^{1/2}\xi + \bar{t})$$

où $\mathcal{U}$ dénote la distribution uniforme, et $T = U_q(\Lambda_1 - \sigma^2 I_q)^{1/2}\Xi + \bar{t} + \varepsilon$ représente les données complètes. On obtient donc

$$P_\Theta(\theta) \propto \exp\Big(-\frac{1}{2}\xi^t(I_q - \sigma^2\Lambda_q^{-1})\xi\Big)$$

par le lemme du chapitre 6.

9.1.3 Vraisemblance et distribution a posteriori

Soit un chemin $c = (s_0, ..., s_m)$ dans le graphe G de la section 7.1. On considère comme à la section 8.1, et semblablement à [47] et [32], la fonction de vraisemblance $P_{Y \mid C}(y \mid c)$ donnée par

$$k(y) \prod_{i=0}^{m} \frac{\mathcal{M}(y_{s_i}; w_j, \mu_j, \sigma_j)}{\mathcal{W}(y_{s_i}; \min, C, \alpha)} \prod_{i=1}^{m} \frac{k_0\, \mathcal{E}(y_{s_{i-1},s_i}; \alpha_0)}{\mathcal{U}(y_{s_{i-1},s_i}; 0, \frac{\pi}{2})}$$

où

$$k(y) = \prod_{s \in S} P_{\text{off}}(y_s) \prod_{(s,t) \in E} P_{\text{off}}(y_{st}).$$

On définit alors la vraisemblance d'une déformation θ en posant

$$P_{Y \mid \Theta}(y \mid \theta) = P_{Y \mid c_\theta}(y \mid c_\theta)$$

où c_θ est obtenue de γ_θ par interpolation de la courbe polygonale correspondante.

Finalement, nous obtenons la distribution *a posteriori*

$$P_{\Theta \mid Y}(\theta \mid y) \propto P_{Y \mid \Theta}(y \mid \theta) P_\Theta(\theta).$$

9.2 Méthode stochastique de localisation

Tout comme dans [38], la localisation non-supervisée d'un patron de forme γ_0 dans une image est formulée comme la recherche d'une déformation θ qui maximise la distribution *a posteriori* $P_{\Theta \mid Y}(\theta \mid y)$.

Écrivant $c_\theta = (s_0, s_1, ..., s_m)$, il s'agit donc de minimiser le champ de Gibbs

$$\begin{aligned} U(\theta, y) &= \sum_{i=0}^{m} \big(\log(\mathcal{W}(y_{s_i}; \min, C, \alpha) - \log(\mathcal{M}(y_{s_i}; w_j, \mu_j, \sigma_j))\big) \\ &+ \sum_{i=1}^{m} \big(\log(\mathcal{U}(y_{s_{i-1},s_i}; 0, \frac{\pi}{2}) - \log(k_0\, \mathcal{E}(y_{s_{i-1},s_i}; \alpha_0))\big) \\ &+ \frac{1}{2}\xi^t(I_q - \sigma^2\Lambda_q^{-1})\xi \end{aligned}$$

comme fonction de θ.

Pour résoudre ce problème, nous procédons en plusieurs étapes. Soit une image de dimension $M \times N$.

1. **Phase d'initialisation**

   (a) Dans un premier temps, on considère 10 valeurs initiales distinctes pour chaque paramètre affine τ_x, τ_y, s, et 20 valeurs initiales distinctes pour le paramètre affine ψ, ces valeurs étant échantillonnées uniformément sur les intervalles respectifs $[0, M]$, $[0, N]$, $[0.2\sqrt{M^2+N^2}, 0.6\sqrt{M^2+N^2}]$, et $[0, 2\pi]$.

   (b) On calcule la valeur du champs de Gibbs U pour chacune des positions initiales, en prenant $\xi = (0, 0, ...0)$. On effectue ensuite une classification des $m\%$ meilleures positions en K clusters à l'aide de l'algorithme des K-moyennes, avec pour attributs τ_x, τ_y, s, et ψ. Dans nos tests, nous avons choisis $m = 10$ et $K = 10$.

2. **Optimisation locale**

   Pour chaque cluster obtenu à l'étape précédente, on a recourt à l'algorithme d'optimisation présenté au chapitre 5, afin de trouver une solution optimale au problème de minimisation de $U(\tau_x, \tau_y, s, \psi, \xi, y)$ sur un domaine restreint défini par le centre géométrique du cluster. Si $\tau_x(i), \tau_y(i), s(i), \psi(i)$ est le centre du

i-ème cluster, on optimise U sur le domaine

$$\begin{aligned}
\tau_x(i) - \frac{M}{6} \leq \quad \tau_x \quad &\leq \tau_x(i) + \frac{M}{6}, \\
\tau_y(i) - \frac{N}{6} \leq \quad \tau_y \quad &\leq \tau_y(i) + \frac{N}{6}, \\
0.9\, s(i) \leq \quad s \quad &\leq 1.1\, s(i), \\
\psi(i) - \frac{\pi}{3} \leq \quad \psi \quad &\leq \psi(i) + \frac{\pi}{3}, \\
-3 \leq \quad \xi_i \quad &\leq 3.
\end{aligned}$$

Pour chaque cluster, on fait plusieurs essais avec des germes différents pour l'algorithme d'optimisation stochastique. À la fin des essais, on conserve la meilleure solution pour le cluster.

3. **Optimisation globale**

   On optimise la fonction U sur le domaine complet de l'image

$$\begin{aligned}
0 \leq \quad \tau_x \quad &\leq M, \\
0 \leq \quad \tau_y \quad &\leq N, \\
0.2\sqrt{M^2+N^2} \leq \quad s \quad &\leq 0.6\sqrt{M^2+N^2}, \\
0 \leq \quad \psi \quad &\leq 2\pi \\
-3 \leq \quad \xi_i \quad &\leq 3.
\end{aligned}$$

   en prenant pour vecteur de solutions initiales L répliques de chacune des K solutions obtenues à l'étape précédente. Dans nos tests, on prend $L = 10$.

Remarque 1. Afin d'éviter les formes qui se retrouvent partiellement à l'extérieur de l'image, nous ajoutons à U un terme égal à 100 fois le nombre de points situés en dehors de l'image. Il s'agit là d'un terme régulateur qui force l'algorithme à ne considérer que les formes situées à l'intérieur de l'image. Le choix du lagrangien (ici 100) est inoffensif ; il suffit de prendre une valeur assez grande.

Remarque 2. Le choix pour l'intervalle associé au paramètre de changement d'échelle s suppose implicitement un *a priori* sur la dimension de la forme cherchée. Certes, on peut considérer un domaine plus grand, mais alors le problème d'optimisation qui en résulte est plus complexe. Dans ce cas, l'algorithme mettra plus de temps à converger vers une solution acceptable. Par ailleurs, dans les applications de la localisation de formes, on connaît souvent la taille relative de l'objet cherché (par exemple, un objet anatomique en imagerie médicale).

Remarque 3. Pour l'algorithme d'optimisation E/S, il semble préférable de faire plusieurs tentatives avec des germes différents, comme nous le proposons, plutôt que d'augmenter le nombre d'itérations avec un même germe.

Remarque 4. La phase d'initialisation que nous proposons est quelque peu arbitraire : il faut spécifier la valeur des constantes m et K. Dans des travaux à venir, nous proposerons une toute autre méthode pour la phase d'initialisation, basée sur un modèle statistique des régions d'une image. Pour l'instant, la phase d'initialisation permet de trouver quelques déformations parmi lesquelles s'en trouvent une suffisamment près de la forme recherchée. Voir figure 9.1.

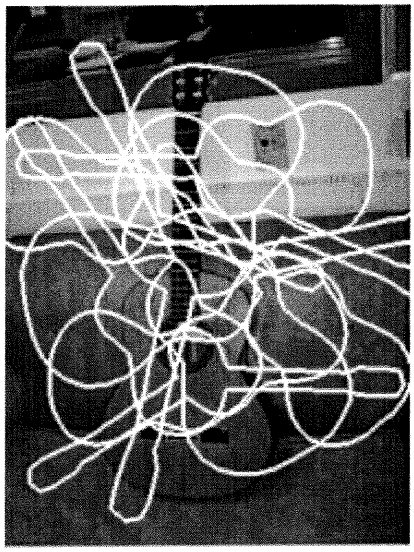


FIG. 9.1 – *Exemple de déformations obtenues à l'étape d'initialisation.*

9.3 Résultats empiriques

Notre modèle est suffisant pour les déformations d'un objet manufacturé. Nous avons testé ce modèle avec la forme d'une guitare classique du vingtième siècle.

La base d'apprentissage comporte 28 images. Nous tenons à remercier les boutiques d'instruments de musique Archambault (500 Ste-Catherine E., Montréal) et Steve's Music (51 St-Antoine O., Montréal) pour nous avoir gracieusement permis de prendre des photos de guitares. Les prix et les marques de guitares n'apparaissent sur aucune de nos images. Chaque courbe est représentée par 70 points obtenus de la façon suivante. On choisit 7 points clés sur la courbe (voir figure 9.2) ; puis, on prend 10 points équidistants sur chaque portion de la courbe délimitée par deux points clés adjacents.

FIG. 9.2 – *Points clés sur la courbe.*

La phase d'entraînement réduit la dimension à 1, avec erreur de reconstruction moyenne inférieure à 0.01. La figure 9.3 présente trois déformations non-linéaires du patron de forme et la valeur du paramètre non-linéaire correspondante. La figure 9.4 présente trois déformations non-linéaires du patron de forme selon le modèle de

Jain et al. On constate que dans le cas du modèle de Jain, les déformations obtenues sont farfelues et improbables. Par contre, dans le cas du modèle statistique, les déformations obtenues sont tout à fait plausibles. Notez que dans notre exemple, les formes obtenues varient très peu de la forme moyenne. Ce dernier point s'explique par le fait que les guitares classiques du vingtième siècle présentent des proportions presque identiques. Si on avait choisit un objet manufacturé de forme plus variable, le modèle statistique en aurait tenu compte.

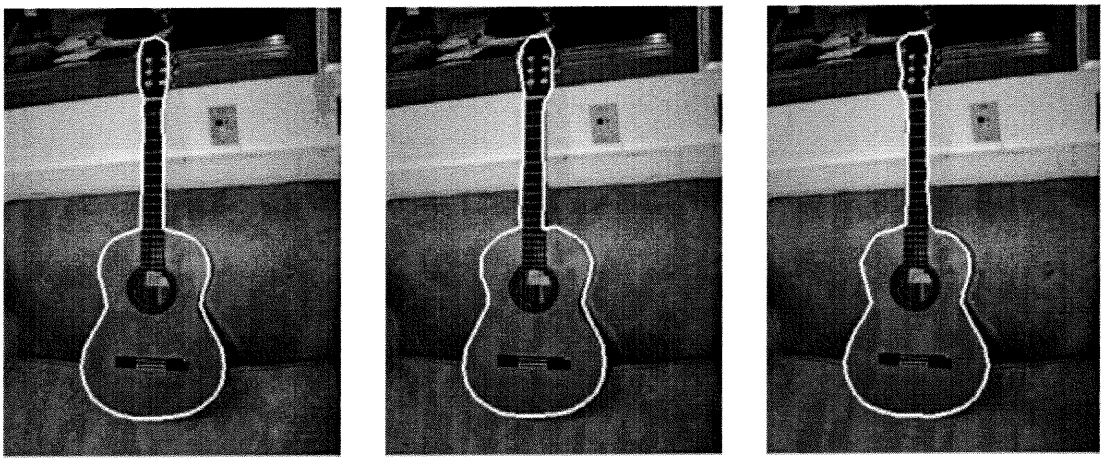


FIG. 9.3 – *Exemple de déformations non-linéaires d'une forme selon le modèle de Tipping et Bishop. Valeurs des paramètres non-linéaires (de gauche à droite) :* $\xi_1 = 0$ *;* $\xi_1 = 3$ *;* $\xi_1 = -3$.

La valeur des paramètres estimés pour la première image de la figure 9.18 par la procédure ECI de la section 7.3, sont présentés au tableau 9.1. On peut voir à la figure 9.5 une représentation du champ de Gibbs heuristique de Jain et al, ainsi que de notre champs de Gibbs statistique. Notez que dans le cas du modèle de Jain, il faut spécifier deux constantes : la constante de lissage ρ, et la variance σ (voir section 2.3.1). Par contre, dans le cas de notre modèle statistique, les paramètres sont estimés automatiquement par la procédure de la section 7.2.

On présente aux figures 9.6 à 9.13, les valeurs du champs de Gibbs statistique

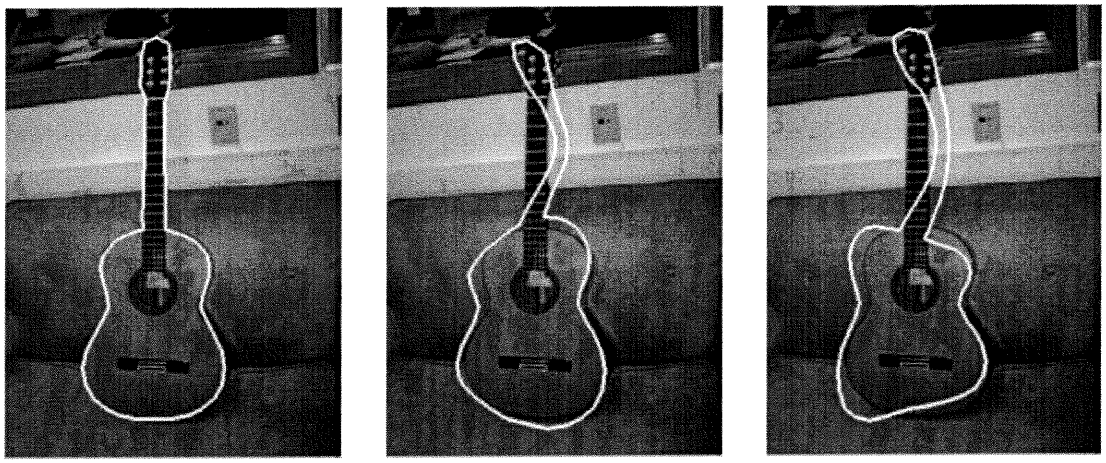


FIG. 9.4 – *Exemple de déformations non-linéaires d'une forme selon le modèle de Jain. Valeurs des paramètres non-linéaires (de gauche à droite) :* $\xi^x_{mn} = \xi^y_{mn} = 0$, *pour* $m, n = 1, 2$ *;* $\xi^x_{mn} = \xi^y_{mn} = 1$, *pour* $m, n = 1, 2$ *;* $\xi^x_{mn} = -1$ *et* $\xi^y_{mn} = 1$, *pour* $m, n = 1, 2$.

min	C	α
-0.001	0.715127	3.80892
w_1	μ_1	σ_1^2
0.331191	50.5294	268.267
w_2	μ_2	σ_2^2
0.668809	28.7663	93.2973
α_0	k_0	ν
0.422855	1.02497	96.6096

TAB. 9.1 – *Valeurs des paramètres estimés par la procédure ECI pour la première image de la figure 9.18.*

pour diverses déformations de la forme moyenne. Dans le cas des figures 9.10 à 9.12, on retrouve une déformation erronée pour laquelle la valeur du champs de Gibbs correspondante est située entre les valeurs de deux déformations acceptables de la forme. Toutefois, la déformation minimale est bien une déformation désirée. Par contre, dans

le cas de la figure 9.13, la valeur du champs de Gibbs de la déformation erronée est inférieure à celle de la déformation désirée. Nous pensons que ce phénomène est du à l'occlusion. Nous n'avons trouvé aucun autre exemple de ce type. On espère donc que le minimum global du champs de Gibbs corresponde bien à une déformation acceptable de la forme, lorsque la forme n'est pas trop occultée.

Nous présentons 15 exemples de localisation de formes aux figures 9.14 à 9.18. Ces image ne font pas partie de l'ensemble de 28 images qui a servi pour la phase d'entraînement. Pour l'image 1, la procédure d'estimation ECI prend 1min ; la phase d'initialisation 7sec ; les 50 (5 germes $\times 10$ clusters) optimisations locales prennent 5min pour un total de 10000 itérations ; l'optimisation globale prend 32sec pour 1000 itérations sur une station 1,2GHz. Au total, notre procédure comporte 11000 itérations et prend 6min et 40sec, ou jusqu'à deux fois plus de temps pour les images testées plus grandes. Chaque image a été testée en prenant 40 germes différents pour l'algorithme stochastique. Le pourcentage des germes menant à une solution erronée, acceptable, ou excellente est présenté au tableau 9.2. Notez qu'il y a très peu de différence entre une solution acceptable et une solution excellente. On y présente également les résultats lorsque 10 germes différents sont utilisés pour chacun des 10 clusters (21000 itérations en 11min 30sec pour l'image 1), ainsi que pour 15 germes (31000 itérations en 16min 10sec pour l'image 1). Les images 12 à 15 présentent des difficultés importantes. Dans le cas de l'image 13, la difficulté s'explique par l'effet de perspective alors que nous n'avons pas modéliser les transformations projectives. Pour l'image 15, la forme est trop occultée ; or, notre modèle ne tient compte que des contours. Pour les images 12 et 14, nous croyons que les résultats seraient nettement améliorés si on tenait compte des régions.

Finalement, on présente aux figures 9.19 à 9.23 les résultats d'une descente de gradient à direction alternée et à pas fixe dans chaque direction effectuée sur 20000 initialisations distinctes sur le champs de Gibbs heuristique de Jain et al (voir section 2.3.1). On fixe le nombre d'itérations maximal à 1000 pour chacune des 20000

No	5 germes			10 germes			15 germes		
	err.	acc.	exc.	err.	acc.	exc.	err.	acc.	exc.
1	0%	0%	100%	0%	0%	100%	0%	0%	100%
2	12.5%	5%	82.5%	0%	2.5%	97.5%	0%	0%	100%
3	15%	0%	85%	2.5%	0%	97.5%	2.5%	0%	97.5%
4	2.5%	2.5%	95%	0%	0%	100%	0%	0%	100%
5	27.5%	30%	42.5%	20%	50%	30%	0%	62.5%	37.5%
6	12.5%	20%	67.5%	5%	15%	80%	0%	2.5%	97.5%
7	0%	12.5%	87.5%	0%	0%	100%	0%	2.5%	97.5%
8	0%	0%	100%	0%	0%	100%	0%	0%	100%
9	5%	10%	85%	0%	0%	100%	0%	0%	100%
10	2.5%	5%	92.5%	0%	0%	100%	0%	0%	100%
11	0%	5%	95%	0%	0%	100%	0%	0%	100%
12	65%	0%	35%	10%	0%	90%	20%	0%	80%
13	55%	0%	45%	62.5%	0%	37.5%	70%	0%	30%
14	100%	0%	0%	100%	0%	0%	100%	0%	0%
15	95%	0%	5%	97.5%	0%	2.5%	97.5%	0%	2.5%

TAB. 9.2 – *Proportion des germes menant à une solution erronée, acceptable, ou excellente pour les images présentées aux figures 9.14 à 9.18.*

descentes de gradient. De plus, le domaine de recherche est restreint à

$$\begin{aligned} 0 \leq \quad \tau_x \quad &\leq M, \\ 0 \leq \quad \tau_y \quad &\leq N, \\ 0.2\sqrt{M^2+N^2} \leq \quad s \quad &\leq 0.6\sqrt{M^2+N^2}, \\ 0 \leq \quad \psi \quad &\leq 2\pi \\ -3 \leq \quad \xi_i \quad &\leq 3. \end{aligned}$$

Nous avons considérer deux ensembles de paramètres : $\rho = 50/\delta$, où δ est le diamètre

No	15 germes			Jain			Jain		
	err.	acc.	exc.	err.	acc.	exc.	err.	acc.	exc.
1	0%	0%	100%			x	x		
2	0%	0%	100%			x			x
3	2.5%	0%	97.5%			x			x
4	0%	0%	100%			x			x
5	0%	62.5%	37.5%			x		x	
6	0%	2.5%	97.5%		x			x	
7	0%	2.5%	97.5%		x			x	
8	0%	0%	100%			x			x
9	0%	0%	100%	x				x	
10	0%	0%	100%		x		x		
11	0%	0%	100%	x			x		
12	20%	0%	80%	x			x		
13	70%	0%	30%	x					x
14	100%	0%	0%	x			x		
15	97.5%	0%	2.5%		x		x		

TAB. 9.3 – *Comparaison de notre méthode avec celle de Jain et al. À gauche : proportion des germes menant à une solution erronée, acceptable, ou excellente pour les images présentées aux figures 9.14 à 9.18. Au centre : résultat d'une descente de gradient effectuée sur le champs de Gibbs de Jain à partir de 20000 initialisations avec* $\rho = 50/\delta$ *et* $\sigma = \sqrt{10}$*, où* δ *est le diamètre de l'image. À droite : résultat d'une descente de gradient effectuée sur le champs de Gibbs de Jain à partir de 20000 initialisations avec* $\rho = 100/\delta$ *et* $\sigma = 1$.

de l'image, et $\sigma = \sqrt{10}$; $\rho = 100/\delta$ et $\sigma = 1$. Le temps de calcul est de 2 à 3 heures par image. On observe que la localisation n'a pleinement réussie que pour 6 des 15 images. Bien sûr, on pourrait ajuster le pas de la descente de gradient ainsi que la valeur des paramètres ρ et σ afin d'obtenir une meilleure localisation, mais il s'agit

là d'une supervision et il n'y a aucune garantie de convergence.

9.4 Conclusion

Notre méthode de localisation de formes présente les avantages fondamentaux suivants :

- La fonction de vraisemblance comporte un véritable sens statistique. Dès lors, les paramètres statistiques de la vraisemblance peuvent être estimés, tant et si bien que le modèle de la vraisemblance est non-supervisé.
- La distribution *a priori* comporte un véritable sens statistique, et peut donc être apprise (par l'ACPP). En particulier, le modèle de l'*a priori* est non-supervisé.
- L'algorithme d'optimisation que nous avons adopté converge asymptotiquement vers une solution optimale au sens du MAP. Donc, en principe, notre méthode ne risque pas de s'enliser dans une solution sous-optimale.

Nous identifions les points suivants à améliorer dans notre méthode :

- Notre méthode permet de localiser une forme si elle se trouve dans l'image, mais non de décider si la forme y est présente. Dès lors, notre méthode ne peut être utilisée directement pour la détection de formes.
- La phase d'initialisation doit tenir compte des régions afin de mieux localiser les positions potentielles de l'objet recherché.
- Le modèle de déformations doit tenir compte des régions afin de faciliter la recherche de la déformation désirée.
- Le modèle de déformation doit tenir compte des transformations projectives.

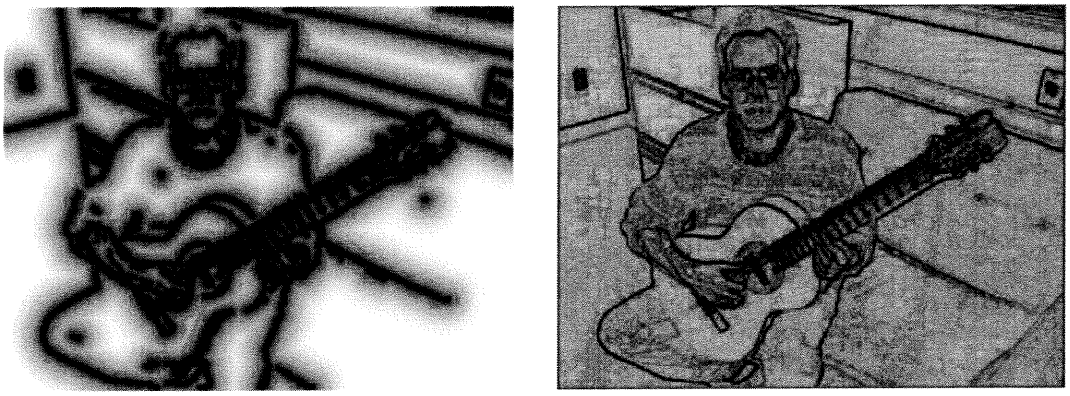


FIG. 9.5 – *À gauche : Champ de Gibbs heuristique de Jain et al (sans tenir compte des angles), avec constante de lissage* $\rho = 50/\delta$, *où* δ *est le diamètre de l'image. À droite : Représentation de la fonction* $\log(p_{off}(y_s)/p_{on}(y_s))$.

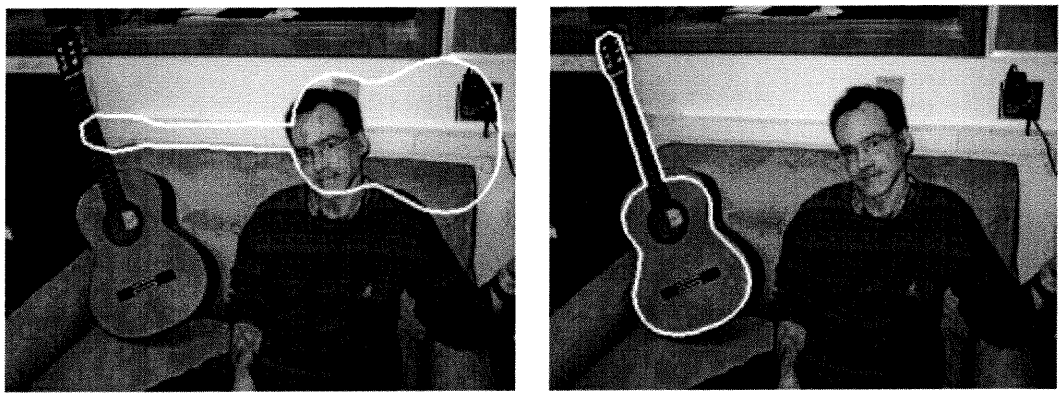


FIG. 9.6 – *Valeur du champs de Gibbs statistique pour deux positions (image 3). À gauche :* -28.9902 *(position erronée). À droite :* -528.508 *(excellente position).*

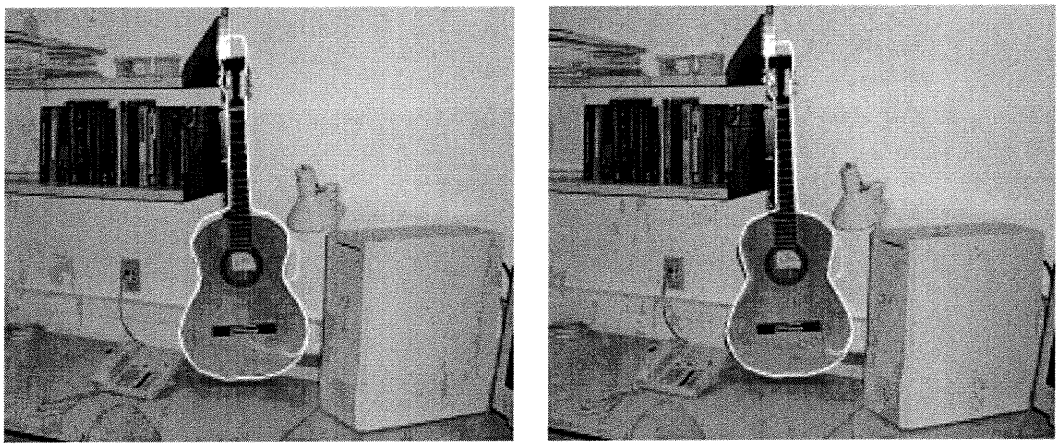


FIG. 9.7 – *Valeur du champs de Gibbs statistique pour deux positions (image 5). À gauche :* -1423.32 *(position acceptable). À droite :* -780.725 *(excellente position).*

FIG. 9.8 – *Valeur du champs de Gibbs statistique pour deux positions (image 6). À gauche :* -872.3 *(position acceptable). À droite :* -1090.39 *(excellente position).*

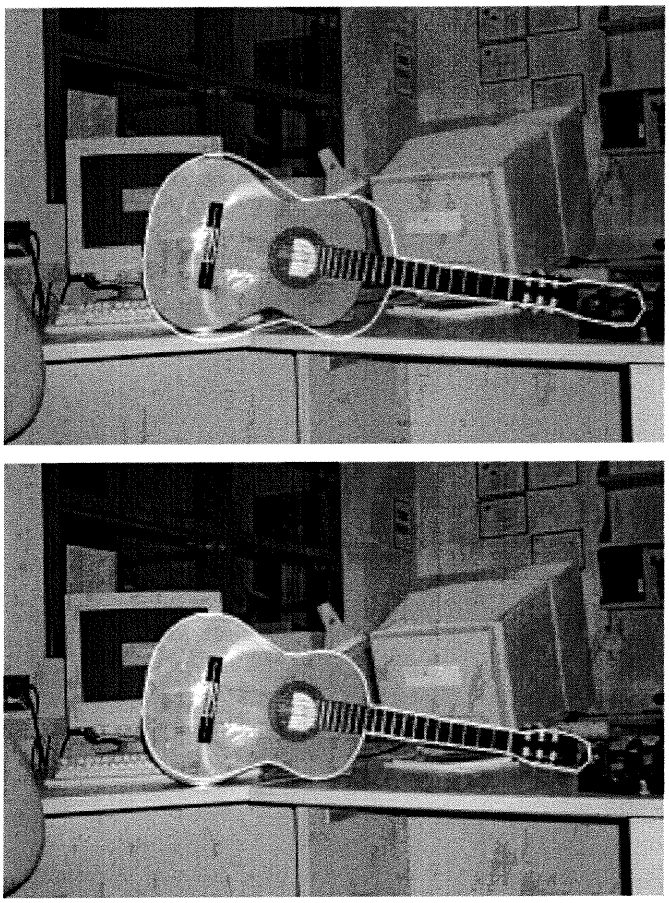


FIG. 9.9 – *Valeur du champs de Gibbs statistique pour deux positions (image 7). En haut :* 186.982 *(position acceptable). En haut :* −1459.74 *(excellente position).*

FIG. 9.10 – *Valeur du champs de Gibbs statistique pour trois positions (image 12). En haut :* 683.276 *(position erronée). En bas à gauche :* 715.945 *(excellente position). En bas à droite :* -231.786 *(excellente position).*

FIG. 9.11 – *Valeur du champs de Gibbs statistique pour trois positions (image 13). En haut :* -384.512 *(position erronée). Au centre :* -286.108 *(position acceptable). En bas :* -450.445 *(position acceptable).*

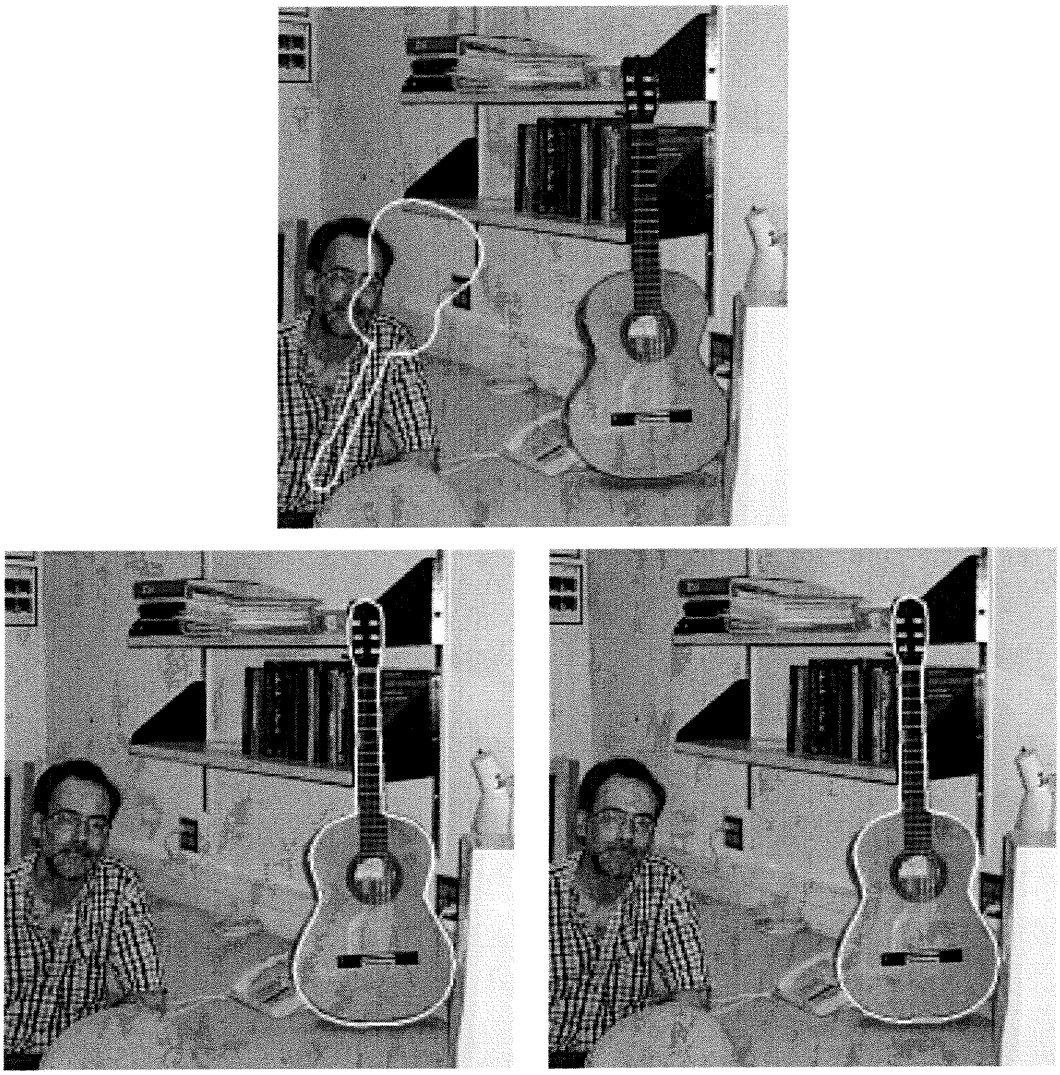


FIG. 9.12 – *Valeur du champs de Gibbs statistique pour trois positions (image 14). En haut :* 531.614 *(position erronée). En bas à gauche :* 964.5 *(excellente position). En bas à droite :* 384.411 *(excellente position).*

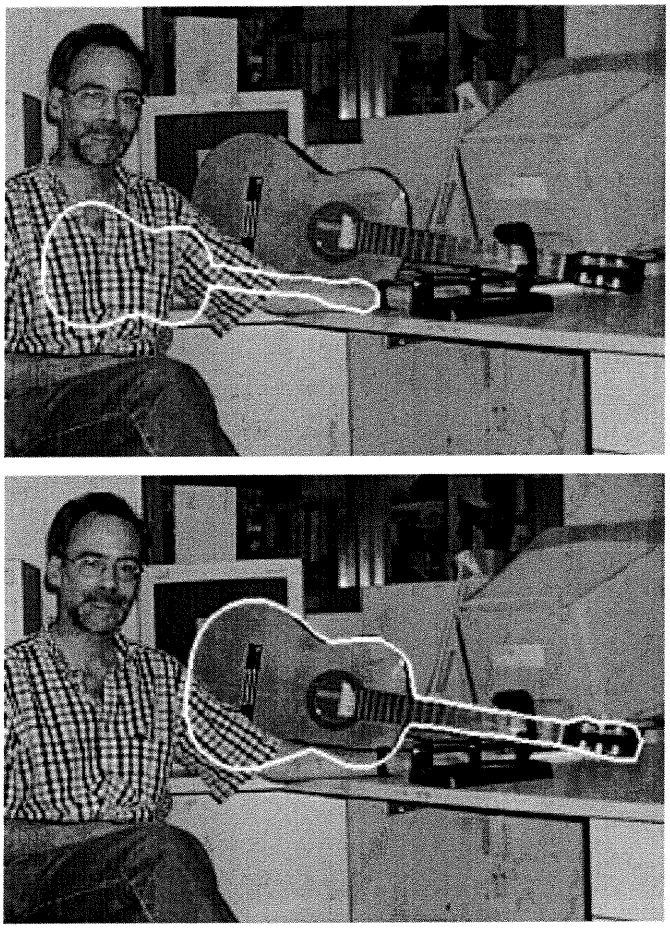


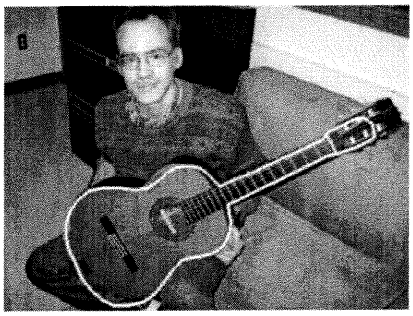
FIG. 9.13 – *Valeur du champs de Gibbs statistique pour deux positions (image 15). À gauche :* -46.0724 *(position erronée). À droite :* 111.581 *(excellente position).*

(1)

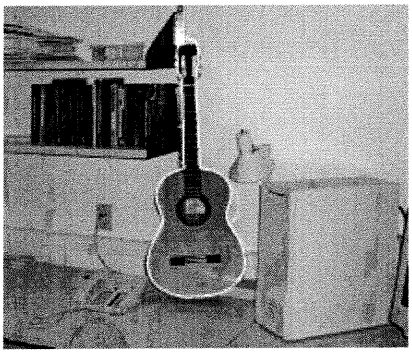
(2)

(3)

FIG. 9.14 – *Exemples de localisations de formes obtenues par optimisation stochastique du champs de Gibbs statistique basé sur les paramètres estimés par la procédure ECI.*



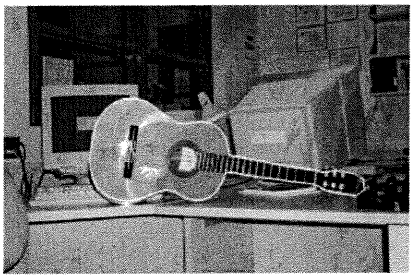
(4)



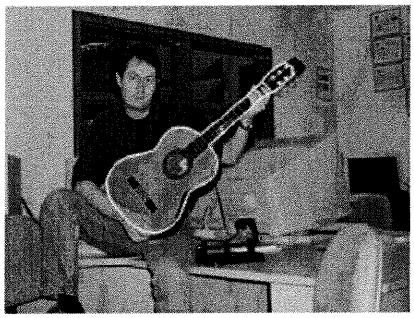
(5)

(6)

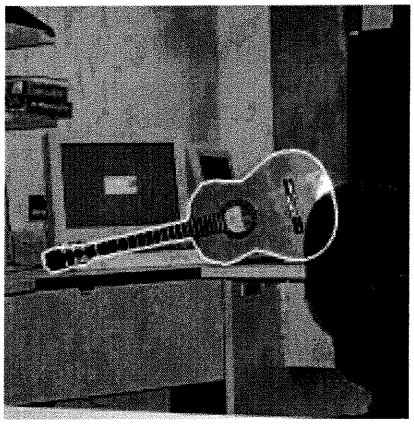
FIG. 9.15 – *Exemples de localisations de formes obtenues par optimisation stochastique du champs de Gibbs statistique basé sur les paramètres estimés par la procédure ECI.*



(7)

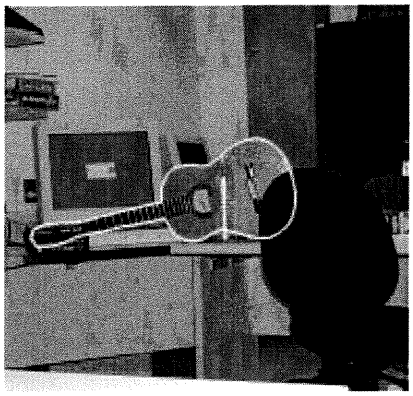


(8)

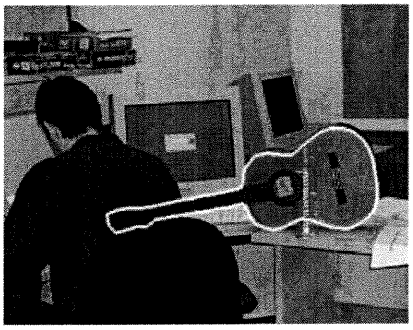


(9)

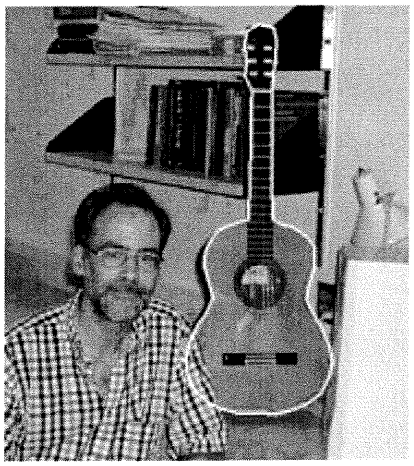
FIG. 9.16 – *Exemples de localisations de formes obtenues par optimisation stochastique du champs de Gibbs statistique basé sur les paramètres estimés par la procédure ECI.*



(10)



(11)

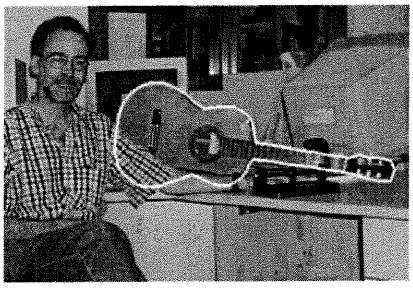


(12)

FIG. 9.17 – *Exemples de localisations de formes obtenues par optimisation stochastique du champs de Gibbs statistique basé sur les paramètres estimés par la procédure ECI.*

(13)

(14)



(15)

FIG. 9.18 – *Exemples de localisations de formes obtenues par optimisation stochastique du champs de Gibbs statistique basé sur les paramètres estimés par la procédure ECI.*

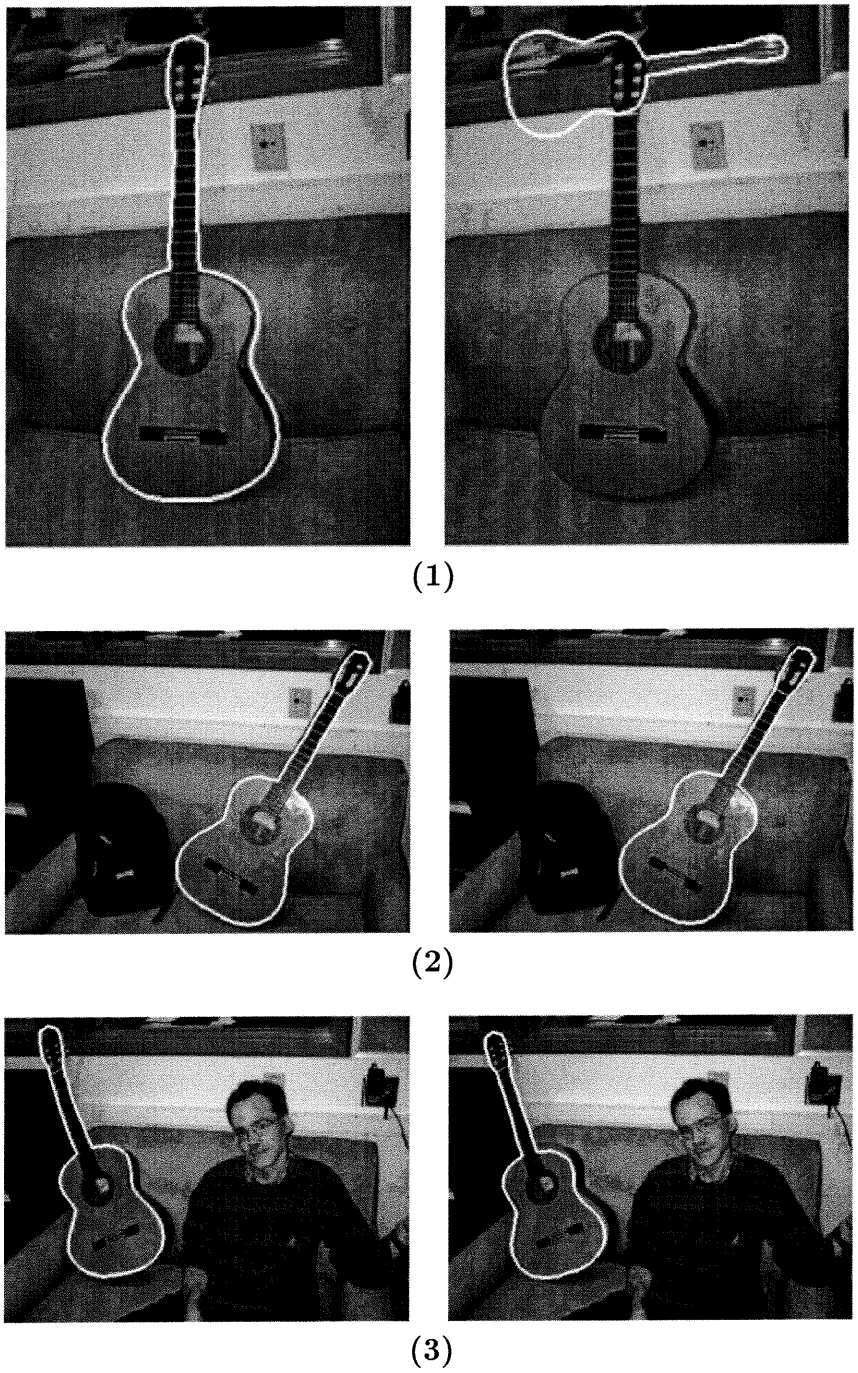
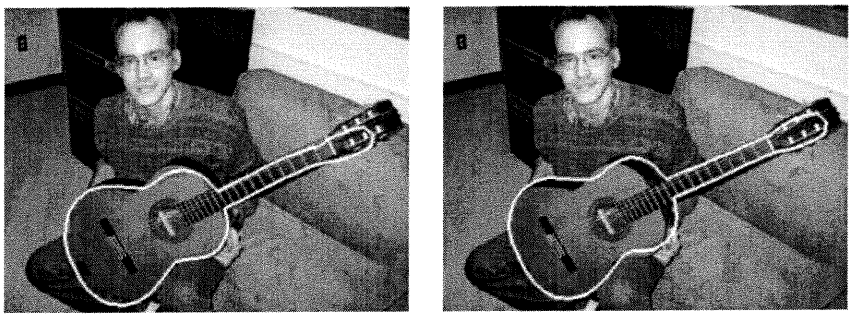
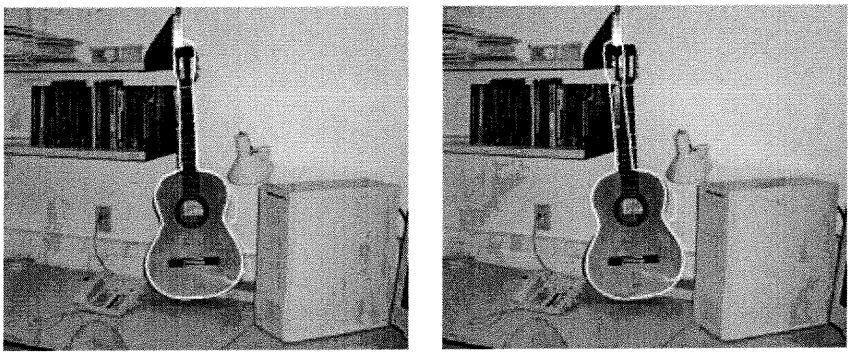


FIG. 9.19 – *Exemples de localisations de formes obtenues par descente de gradient sur le champs de Gibbs heuristique de Jain et al. À gauche :* $\rho = 50/\delta$*, où* δ *est le diamètre de l'image, et* $\sigma = \sqrt{10}$*. À droite :* $\rho = 100/\delta$ *et* $\sigma = 1$*.*



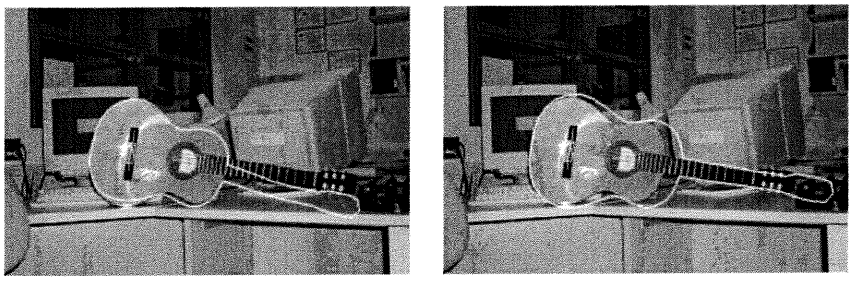
(4)



(5)

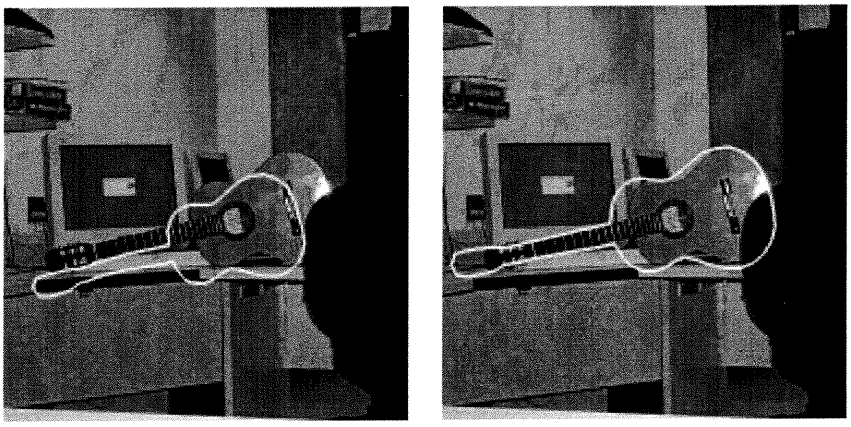
(6)

FIG. 9.20 – *Exemples de localisations de formes obtenues par descente de gradient sur le champs de Gibbs heuristique de Jain et al. À gauche :* $\rho = 50/\delta$*, où* δ *est le diamètre de l'image, et* $\sigma = \sqrt{10}$*. À droite :* $\rho = 100/\delta$ *et* $\sigma = 1$*.*



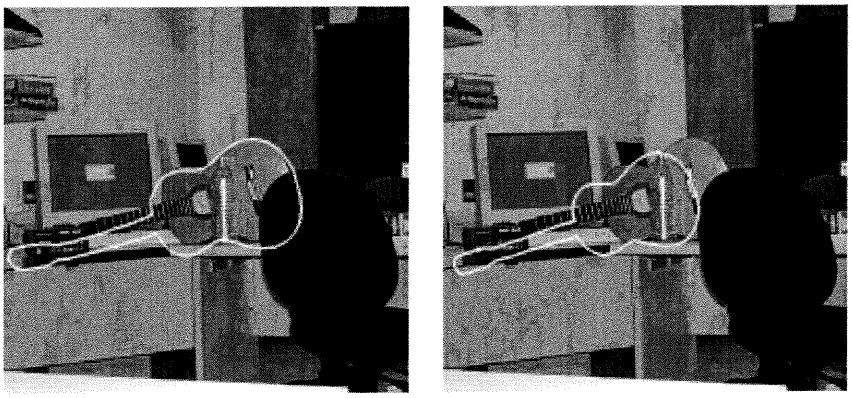
(7)

(8)

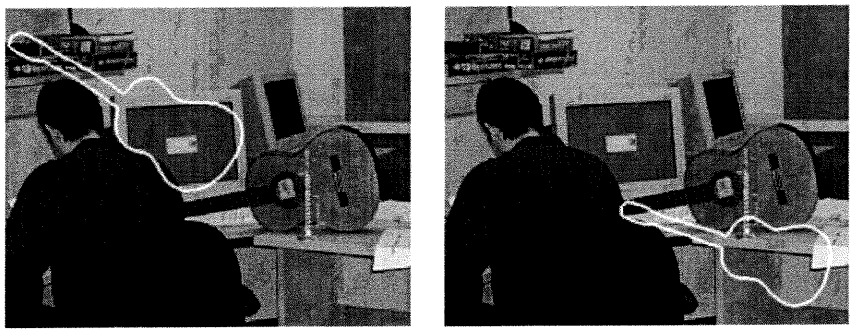


(9)

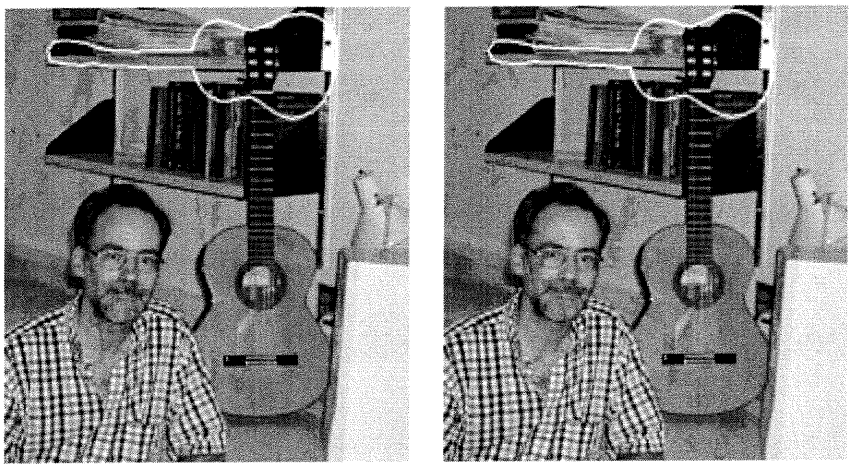
FIG. 9.21 – *Exemples de localisations de formes obtenues par descente de gradient sur le champs de Gibbs heuristique de Jain et al. À gauche :* $\rho = 50/\delta$*, où* δ *est le diamètre de l'image, et* $\sigma = \sqrt{10}$*. À droite :* $\rho = 100/\delta$ *et* $\sigma = 1$*.*



(10)



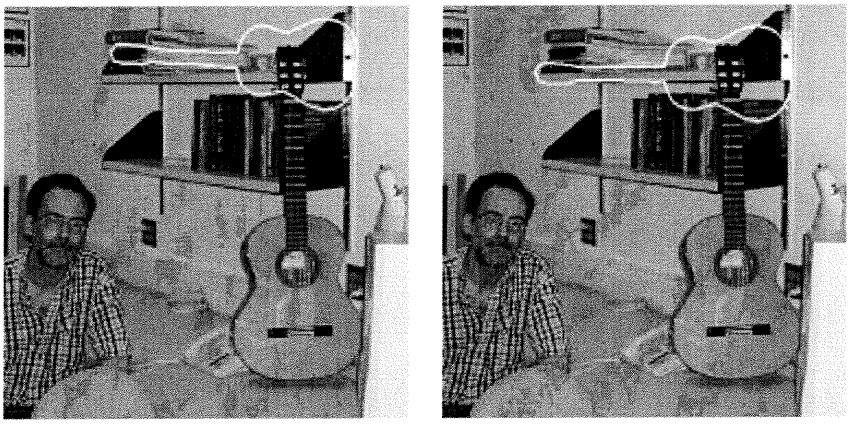
(11)



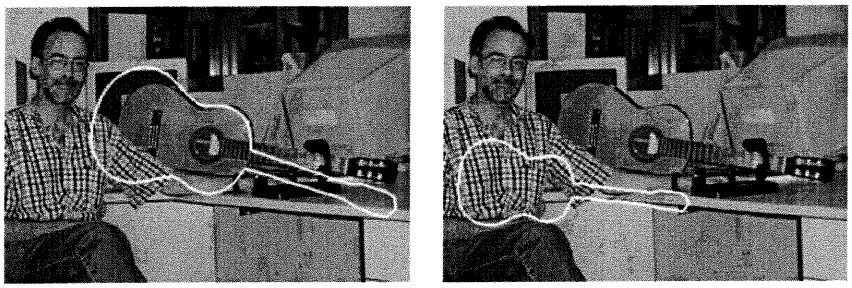
(12)

FIG. 9.22 – *Exemples de localisations de formes obtenues par descente de gradient sur le champs de Gibbs heuristique de Jain et al. À gauche :* $\rho = 50/\delta$*, où* δ *est le diamètre de l'image, et* $\sigma = \sqrt{10}$*. À droite :* $\rho = 100/\delta$ *et* $\sigma = 1$*.*

(13)



(14)



(15)

FIG. 9.23 – *Exemples de localisations de formes obtenues par descente de gradient sur le champs de Gibbs heuristique de Jain et al. À gauche :* $\rho = 50/\delta$*, où* δ *est le diamètre de l'image, et* $\sigma = \sqrt{10}$*. À droite :* $\rho = 100/\delta$ *et* $\sigma = 1$*.*

Chapitre 10

Conclusion

Poser les fondements théoriques en vue de la segmentation d'une image au sens des contours n'est certes pas une tâche aisée. En effet, on serait bien embêté de fournir une définition formelle de l'ensemble des points de contours d'une image. Il n'est pourtant pas impossible de s'aventurer dans un tel projet.

Dans ce mémoire, la solution proposée consiste à formuler les problèmes à résoudre comme minimisation de fonctions, lesquelles sont définies à partir d'un modèle cohérent des contours qui repose sur la distribution empirique d'une variable aléatoire toute simple, le champs vectoriel du gradient des niveaux de gris. L'application de la théorie à la résolution de trois problèmes en traitement d'images suggère la validité, ne serait-ce que d'un point de vue fonctionnel, du modèle proposé.

Ceci dit, dans le cas de la localisation de formes, la fonction à minimiser que l'on obtient est d'une telle complexité dès que l'image est le moindrement encombrée, que le recours à l'optimisation stochastique demeure une solution provisoire. Dans des travaux à venir, il nous faudra donc développer une toute autre méthode pour trouver une solution initiale suffisamment près de la forme cherchée. Ce but une fois atteint, il sera d'autant plus facile de résoudre le problème de la localisation de formes.

Nous avons observé que notre méthode de détection de contours donne parfois lieu

à des résultats difficilement exploitables dû à la présence d'une trop grande quantité de détails inutiles. C'est là qu'une bonne segmentation de l'image au sens des régions devient importante. On peut alors détecter les contours les plus significatifs, c'est-à-dire ceux qui sont frontières entre régions. De plus, certaines images ne permettent une bonne segmentation qu'à partir des textures, faute de contours précis, et dans ce cas notre méthode doit être ajustée sensiblement.

Ces modifications apportées, nous espérons que ce mémoire trouvera des applications utiles, en particulier dans le domaine de l'imagerie médicale.

Bibliographie

[1] E. H. L. Aarts and J. H. M. Korst. *Simulated Annealing and Boltzmann Machines.* New York : Wiley, 1988.

[2] R. Azencott. *Simulated Annealing : Parallelization Techniques.* John Wiley and Sons, New-York, 1992.

[3] T. Bäck. *Evolutionary Algorithms in Theory and Practice.* New York : Oxford Univ. Press, 1996.

[4] S. Banks. *Signal processing, image processing and pattern recognition.* Prentice Hall, 1990.

[5] D. J. Bartholomew. *Latent Variable Models and Factor Analysis.* Charles Griffin & Co. Ltd., London, 1987.

[6] B. Braathen, P. Masson, and W. Pieczynski. Global and local methods of unsupervised Bayesian segmentation of images. *GRAPHICS and VISION*, 2(1) :39–52, 1993.

[7] J. F. Canny. A computational approach to edge detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 8(6) :679–698, 1986.

[8] S. Castan, J. Zhao, and J. Shen. Une famille de détecteurs de contours basée sur le filtre exponentiel optimal. In $7^{ème}$ *Congrès AFCET-RFIA, Paris*, 1989.

[9] O. Catoni. Rough large deviation estimates for simulated annealing : Application to exponential schedules. *Ann. Prob.*, 20 :1109–1146, 1992.

[10] G. Celeux and J. Diebolt. L'algorithme SEM : un algorithme d'apprentissage probabiliste pour la reconnaissance de mélange de densités. *Revue de statistiques appliquées*, 34(2) :35–52, 1986.

[11] R. Cerf. The dynamics of mutation-selection algorithms with large population sizes. *Ann. Inst. Henri Poincaré Probab. Statist.*, 32(4) :455–508, 1996.

[12] R. Cerf. A new genetic algorithm. *Ann. Appl. Probab.*, 6(3) :778–817, 1996.

[13] T.-S. Chang and Y. Chow. A limit theorem for a class of inhomogeneous markov processes. *Ann. Probab.*, 17 :1483–1502, 1989.

[14] T.-S. Chang and Y. Chow. Asympotic behavior of eigenvalues and random updating schemes. *Appl. Math. Optim.*, 28 :259–175, 1993.

[15] T. F. Cootes and C. J. Taylor. A mixture model for representing shape variation. *Image and Vision Computing*, 17 :567–573, 1999.

[16] T. F. Cootes and C. J. Taylor. *Statistical Models of Appearance for Computer Vision.* Wolfson Image Analysis Unit, Imaging Science and Biomedical Engineering, University of Manchester - http ://www.wiau.man.ac.uk, 2000.

[17] T. E. Davies and J. C. Principe. A markov chain framework for the simple genetic algorithm. *Evolutionary Computation*, 1(3) :269–288, 1993.

[18] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Royal Statistical Society*, pages 1–38, 1976.

[19] R. Deriche. Using canny's criteria to derive a recursively implemented optimal edge detector. *International Journal of Computer Vision*, 1(2) :167–187, 1987.

[20] F. Destrempes and M. Mignotte. Unsupervised detection and semi-automatic extraction of contours using a statistical model and dynamic programming. In *ICSIP' 2002*, accepté.

[21] F. Destrempes and M. Mignotte. Unsupervised detection of contours using a statistical model. In *ICIP' 2002*, accepté.

[22] F. Destrempes and M. Mignotte. Unsupervised localization of shapes using statistical models. In *ICSIP' 2002*, accepté.

[23] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1 :269–270, 1959.

[24] M.-P. Dubuisson Jolly, S. Lakshmanan, and A. K. Jain. Vehicle segmentation and classification using deformable templates. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, pages 293–308, Mars 1996.

[25] G. D. Forney Jr. The Viterbi algorithm. *Proc. IEEE*, 61(3) :263–278, Mars 1973.

[26] O. François. Convergence in simulated evolution algorithms. *Complex Syst.*, 10 :311–319, 1996.

[27] O. François. An evolutionary strategy for global minimization and its markov chain analysis. *IEEE trans. on Evolutionary Computation*, 2(3), 1998.

[28] O. François. Global optimization with exploration/selection algorithms and simulated annealing. *Ann. Appl. Probab.*, 12(1) :248–271, 2002.

[29] M. I. Freidlin and A. D. Wentzell. *Random Perturbations of Dynamical Systems.* Springer-Verlag, New-York, 1984.

[30] D. Geman and S. Geman. Relaxation and annealing with constraints. *Division Appl. Math., Brown Univ., Complex Systems Tech. Rep. 35*, 1987.

[31] D. Geman, S. Geman, C. Graffigne, and P. Dong. Boundary detection by constrained optimixation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 12(7) :609–628, Juillet 1990.

[32] D. Geman and B. Jedynak. An active testing model for tracking roads in satellite images. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 18(1) :1–14, 1996.

[33] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 6(6) :721–741, 1984.

[34] D. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Reading, MA : Addison-Wesley, 1989.

[35] B. Hajek. Cooling schedule for optimal annealing. *Math. Oper. Res.*, 13 :311–329, 1988.

[36] C.-R. Hwang and S.-J. Sheu. Large time behavior of perturbated diffusion markov processes with applications to the second eigenvalue problem for fokker-planck operators and simulated annealing. *Acta Appl. Math*, 19 :253–295, 1990.

[37] C.-R. Hwang and S.-J. Sheu. Singular perturbed markov chains and exact behaviors of simulated annealing process. *J. Theoret. Probab.*, 5 :223–249, 1992.

[38] A. K. Jain, Y. Zhong, and S. Lakshmanan. Object matching using deformable templates. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, pages 267–278, Mars 1996.

[39] A. K. Jain and D. Zongker. Representation and recognition of handwritten digits using deformable templates. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, pages 1386–1391, Décembre 1997.

[40] L. Liu and S. Sclaroff. Deformable shape detection and description via model-based region grouping. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Juin 1999.

[41] N. S. Matloff. *Probability Modeling and Computer Simulation.* PWS-KENT Publishing Company, 1988.

[42] M. Mignotte, C. Collet, P. Pérez, and P. Bouthemy. Three-class markovian segmentation of high resolution sonar images. *Computer Vision and Image Understanding*, 76(3) :191–204, Décembre 1999.

[43] M. Mignotte, C. Collet, P. Pérez, and P. Bouthemy. Hybrid genetic optimization and statistical model-based approach for the classification of shadow shapes in sonar imagery. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(2) :129–141, Février 2000.

[44] M. Mignotte, J. Meunier, and J.-C. Tardif. Endocardial boundary estimation and tracking in echocardiographic images using deformable templates and markov random fields. *Pattern Analysis and Applications*, 4(4) :256–271, Novembre 2001.

[45] E. N. Mortensen and W. A. Barrett. Interactive segmentation with intelligent scissors. *GMIP*, 60(5) :349–384, Septembre 1998.

[46] E. N. Mortensen, B. S. Morse, W. A. Barrett, and J. K. Udupa. Adaptative boundary detection using live-wire two-dimensional dynamic programming. In *IEEE Proceedings of Computers in Cardiology*, pages 635–638, Octobre 1992.

[47] M. Pérez, A. Blake, and M. Gangnet. Jetstream : Probabilistic contour extraction with particles. In *Int. Conf. on Computer Vision, ICCV' 2001*, Vancouver, Canada, Juillet 2001.

[48] W. Pieczynski. Champs de Markov cachés et estimation conditionnelle itérative. *Revue Traitement Du Signal*, 11(2) :141–153, 1994.

[49] F. Salzenstein and W. Pieczynski. Parameter estimation in hidden fuzzy markov random fields and image segmentation. *Graphical Models and Image Processing*, 59(4) :205–220, 1997.

[50] J. Shen and S. Castan. An optimal linear operator for step edge detection. *CVGIP*, 54 :112–133, 1992.

[51] M. E. Tipping and C. M. Bishop. Mixtures of probabilistic principal component analyzers. *Technical report NCRG/97/003*, Juillet 1998.

[52] A. Trouvé. *Parallélisation massive du recuit simulé.* Ph. D. thesis, Université d'Orsay, 1992.

[53] A. Trouvé. *Partially parallel simulated annealing : Low and high temperature approach to the invariant measure.* In Proceedings Volume of the US-French Workshop on Applied Stochastic Analysis (Rutgers University, 29 April-2 May 1991), Lecture Notes in Control and Infor. Sci. 177, Springer-Verlag,New-York, 1992.

[54] A. Trouvé. Cycle decomposition and simulated annealing. *SIAM J. Control Optim.*, 34(3) :966–986, 1996.

[55] A. Trouvé. Rough large deviation estimates for the optimal convergence speed exponent of generalized simulated annealing algorithm. *Ann. Inst. Henri Poincaré Probab. Statist.*, 32, 1996.

[56] Y. Zhong and A. K. Jain. Object localization using color, texture and shape. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 33 :671–684, 2000.