Université de Montréal

Impact du choix de la fonction de perte en segmentation d'images et application à un modèle de couleurs

par

Louis-François Poirier

Département de mathématiques et de statistique
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 Statistique

janvier 2006

Université de Montréal

Faculté des études supérieures

Ce mémoire intitulé

Impact du choix de la fonction de perte en segmentation d'images et application à un modèle de couleurs

présenté par

Louis-François Poirier

a été évalué par un jury composé des personnes suivantes :

Christian Léger
(président-rapporteur)

Jean-François Angers
(directeur de recherche)

Max Mignotte
(co-directeur)

Charles Dugas
(membre du jury)

Mémoire accepté le:
23/12/05

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

SOMMAIRE

Les avancements des dernières décennies dans le domaine de l'informatique permettent l'utilisation de méthodes statistiques de plus en plus complexes dans plusieurs domaines d'application. Le domaine de l'imagerie numérique en fait partie sans contredit. En effet, le traitement d'images numériques est devenu essentiel dans plusieurs domaines, mentionnons seulement l'imagerie médicale, l'imagerie satellite ou la robotique. Dans chacun de ces domaines, des décisions doivent être prises basées sur des images, et le traitement d'images vise à améliorer la prise de ces décisions. Les champs de recherche en imagerie sont très vastes et variés, pour notre part nous nous concentrons sur l'aspect de la segmentation d'une image naturelle.

La segmentation d'images est une étape préliminaire pour beaucoup d'applications en vision informatique comme la reconnaissance d'objets ou bien la reconstruction d'images en 3D. Elle consiste à partitionner l'image en régions possédant une ou des caractéristiques communes.

Dans ce mémoire, nous présentons un modèle statistique bayésien basé sur l'article de Destrempes, Mignotte et Angers (2005) pour segmenter une image selon les couleurs. Ce travail vise à évaluer l'impact du choix de la fonction de perte dans la segmentation résultante. Le modèle proposé utilise les champs de Markov cachés conjointement avec un modèle *a priori* de Potts pour modéliser statistiquement l'image. L'estimation du modèle est conduite par l'algorithme EM et les méthodes numériques de Monte Carlo avec chaînes de Markov et de Monte Carlo avec fonction d'importance.

Mots clés : Estimation bayésienne, champs aléatoires de Markov cachés, MCMC, simulations.

SUMMARY

The recent advances in computer technology allow the use of more complex statistical procedures in many fields of application such as digital imagery. Image processing has become essential in many applications like medical imagery, satellite imagery or robot vision. In each of these fields, decisions are made based on images and the role of image processing is to improve the quality of those decisions. Image processing has many ramifications, we will concentrate only on image segmentation.

Image segmentation is frequently seen as a pre-processing step for many higher level tasks in computer vision such as object recognition and 3D image reconstruction. The segmentation of an image consists in partitioning the latter in regions sharing the same properties.

In this thesis we will present a color-based Bayesian statistical segmentation model that takes its roots in the paper of Destrempes, Mignotte & Angers (2005). We will be trying to evaluate the impact of the loss function on the resulting segmentation. The statistical model consists of an hidden Markov random field model and a Potts prior on the underlying segmentation. Estimating the model is done through the EM algorithm and the Monte Carlo Markov Chain and Monte Carlo with importance sampling numerical methods.

Keywords : Bayesian estimation, hidden Markov random fields, MCMC, simulations.

REMERCIEMENTS

Je tiens tout d'abord à remercier Max Mignotte et François Destrempes pour m'avoir donné la chance de travailler sur ce projet qui m'a fortement intéressé. Merci particulièrement à François pour les nombreuses heures que nous avons passées ensemble à discuter du projet. Tu m'as permis d'approfondir mes connaissances sur le sujet et ton aide continuelle fut très appréciée.

Mes deux ans de maîtrise n'auraient certainement pas été aussi agréables si ce n'avait été de mon directeur. Jean-François, ce fut un plaisir de travailler avec toi, ton dévouement pour tes étudiants et la rigueur dans ton travail devraient servir de modèle à tous. Jamais je n'aurais cru qu'un simple appel pour participer à un projet au baccalauréat aurait fini en une si belle relation. Je ne garderai que d'excellents souvenirs de ces années passées à travailler avec toi.

Finalement, je veux remercier ma famille et mes amis qui m'ont toujours supporté tout au long de mes démarches personnelles. Tout particulièrement je veux remercier mes parents qui m'ont donné la chance d'être là où je suis aujourd'hui. Sans vous, rien de tout ça n'aurait été possible, merci mille fois.

INTRODUCTION

Les avancements des dernières décennies dans le domaine de l'informatique permettent l'utilisation de méthodes statistiques de plus en plus complexes dans plusieurs domaines d'application. Le domaine de l'imagerie numérique en fait partie sans contredit. En effet, le traitement d'images numériques est devenu essentiel dans plusieurs domaines, mentionnons seulement l'imagerie médicale, l'imagerie satellite ou la robotique. Dans chacun de ces domaines, des décisions doivent être prises basées sur des images, et le traitement d'images vise à améliorer la prise de ces décisions. Les champs de recherche en imagerie sont très vastes et variés, pour notre part nous nous concentrons sur l'aspect de la segmentation d'une image naturelle.

Le travail présenté dans ce mémoire est fortement basé sur le modèle statistique présenté dans l'article de Destrempes, Mignotte et Angers (2005). Dans cet article, un modèle statistique bayésien est présenté pour répondre au problème de segmentation d'une image selon les couleurs observées. Les nouveautés présentes dans ce modèle sont, entre autres, l'utilisation d'un modèle de vraisemblance bêta au lieu d'un modèle gaussien et l'introduction d'une contrainte globale sur le nombre de régions dans l'image, basée sur la loi cubique pour la taille des régions. Comme dans plusieurs articles, un modèle de Potts est utilisé comme modèle *a priori* sur la segmentation (voir Wu, 1982). L'estimation des paramètres du modèle de segmentation est faite selon une nouvelle approche appelée Exploration/Sélection/Estimation, ou ESE, et consiste en un algorithme de maximisation permettant de trouver le maximum *a posteriori* (MAP). Lors du processus de publication, un des arbitres a questionné l'effet de la fonction de perte dans le modèle bayésien sur la segmentation résultante. Nous utilisons ainsi

le modèle statistique développé par Destrempes *et al.* (2005) pour caractériser une image à partir duquel nous tentons de répondre à la question suivante : quel est l'impact du choix de la fonction de perte en segmentation d'images en utilisant un modèle de couleurs ?

La segmentation d'images est une étape préliminaire pour beaucoup d'applications en vision informatique comme la reconnaissance d'objets ou bien la reconstruction d'un objet en trois dimensions à partir d'images en 2D. Prenons par exemple la reconnaissance d'objets, cette tâche est très utile en robotique industrielle comme par exemple lorsqu'un robot analyse certaines composantes pour identifier des anomalies. La reconnaissance d'objets est également utilisée pour chercher des images à l'intérieur d'une large banque d'images. Pour toutes ces tâches d'analyses supérieures, la segmentation est une étape préalable très importante, premièrement puisqu'elle simplifie le niveau d'information contenu dans l'image et deuxièmement puisqu'elle cible dans l'image des zones partageant une même propriété.

Ainsi, nous pouvons voir la segmentation d'une image comme la partition de celle-ci en un ensemble de régions possédant une ou des caractérisques communes. De manière générale, la segmentation peut être faite selon trois caractérisques de l'image ou bien une combinaison de celles-ci. Ces dernières sont les couleurs, les textures et les formes. Dans le travail présenté dans ce mémoire, nous nous limitons à la segmentation selon les couleurs.

Bien évidemment, il existe plusieurs approches pour segmenter une image. Nous adoptons une approche bayésienne basée sur les champs aléatoires de Markov cachés (Hidden Markov Random Fields, HMRF). La combinaison des champs de Markov cachés avec la théorie de la décision bayésienne nous fournit un cadre de segmentation formel qui incorpore l'information sur les régions sous-jacentes et la couleur observée à chaque pixel. Les champs de Markov cachés sont des outils puissants de plus en plus utilisés en vision informatique, en reconstruction d'images et en reconnaissance vocale. Les premiers auteurs à utiliser les HMRF dans un cadre de segmentation d'images sont Geman et Geman (1984) et Besag (1986). Ils ont pu démontrer l'utilité et la puissance de ceux-ci pour résoudre

le problème de segmentation d'une image. L'utilité de la théorie de la décision bayésienne vient du fait que nous traitons le problème de segmentation comme un problème statistique inverse. En effet, nous supposons un champ aléatoire caché modélisant la segmentation des régions à partir duquel l'image observée aurait été engendrée. Le problème inverse vient du fait que cette segmentation n'est pas connue et nous tentons de l'estimer à partir de l'image. Ainsi, le paradigme de Bayes nous donne un cadre statistique pour inclure des contraintes *a priori* sur la forme et la taille des régions dans l'image.

Plusieurs méthodes furent proposées pour estimer un modèle basé sur les champs de Markov cachés. La majorité des méthodes proposées consistent en des algorithmes itératifs où les paramètres sont mis à jour en fonction de l'information à chaque étape de la procédure. Dans les premiers travaux de Besag (1986) un tel algorithme itératif appelé ICM est proposé. Dans Destrempes *et al.* (2005) une approche bayésienne non décisionnelle basée sur le maximum *a posteriori* (MAP) est utilisée et un algorithme de maximisation appelé ESE est développé. D'autres approches comme le recuit simulé (voir Hajek, 1988) ou le recruit simulé adaptatif (voir Laksshmanan et Derin, 1989) furent proposées pour estimer la segmentation au sens du MAP. Un autre algorithme itératif fréquemment utilisé en segmentation d'images est l'algorithme EM. Ce dernier est utilisé dans Belongie *et al.* (1998) et des variantes de ce dernier sont utilisées entre autre dans Nasios et Bors (2004) où une approche variationnelle du EM est proposée et dans Wu, Yang et Chan (2003). L'algorithme EM est plus intense au sens calculatoire que le ESE par exemple, mais il nous fournit un cadre bien connu dans le domaine statistique. Ainsi, nous utilisons une approche par l'algorithme EM dans ce travail.

La structure du mémoire va comme suit : au premier chapitre nous décrivons les concepts statistiques et mathématiques nécessaires à la compréhension de la modélisation d'une image. Nous caractérisons l'image au sens mathématique en introduisant la notion de champ aléatoire et décrivons brièvement le domaine de représentation des couleurs. Nous introduisons ensuite les champs aléatoires de Markov cachés dans le contexte du traitement d'images et décrivons l'approche

proposée dans Destrempes *et al.* (2005). Finalement nous expliquons les méthodes numériques qui seront utilisées pour estimer le modèle proposé.

Au chapitre 2 nous expliquons brièvement la théorie de la décision bayésienne et décrivons les fonctions de perte que nous proposons. Ensuite, nous proposons une approche décisionnelle bayésienne basée sur le modèle de Destrempes *et al.* (2005). Dans ce chapitre, nous verrons également l'approche sélectionnée pour évaluer la convergence des méthodes MCMC.

Finalement, au troisième chapitre nous expliquons la méthode d'analyse en grappe qui est utilisée pour fusionner les régions et discutons des aspects pratiques des critères d'arrêt pour les méthodes MCMC et l'algorithme EM. Pour terminer, nous présentons des résultats de simulations sur des images synthétiques pour tester l'approche proposée au chapitre 2 et illustrons également la méthode sur différentes images naturelles.

Chapitre 1

CADRE STATISTIQUE

Dans ce chapitre, nous définissons les concepts de base en traitement d'images qui vont nous permettre de modéliser statistiquement une image. Dans un premier temps, nous décrivons le cadre général mathématique dans lequel nous définissons une image. Les notions de pixel, voisinage et clique seront brièvement expliquées de manière à comprendre comment nous pouvons modéliser ces données. Nous décrivons également certains systèmes de référence pour représenter une couleur et plus particulièrement le système Lab qui sera utilisé dans ce mémoire.

Dans un deuxième temps, nous définissons le modèle de segmentation d'une image selon un modèle de couleurs développé dans Destrempes, Mignotte et Angers (2005). Nous expliquons les grandes lignes du cadre statistique utilisé dans cet article et nous expliquons brièvement la méthode ESE qui y fut développée pour maximiser une fonction.

Finalement, nous passons en revue les méthodes d'estimation numérique que nous utilisons pour estimer les différents paramètres du modèle. Nous utilisons une approche basée sur l'algorithme EM pour estimer la segmentation de l'image. Nous définissons alors le problème de segmentation d'une image comme un problème statistique inverse et nous motivons l'utilisation de l'algorithme EM dans un tel contexte. Nous décrivons également les méthodes d'estimation numérique de Monte Carlo avec fonction d'importance et l'algorithme de Metropolis-Hasting que nous utiliserons pour estimer les paramètres du modèle.

1.1. Éléments en traitement d'images

Dans cette section, nous définissons les concepts de base en imagerie et en statistique qui nous permettent de modéliser mathématiquement une image. Nous voulons tout d'abord décrire en langage mathématique ce qu'est une image et ensuite définir certaines propriétés statistiques et mathématiques de celle-ci. Tout au long de cette section, nous utilisons un exemple bien simple pour illustrer chacun de ces concepts.

1.1.1. Définition mathématique d'une image

Dans ce mémoire, nous définissons mathématiquement une image comme un champ aléatoire stationnaire.

Définition 1.1.1. *Un champ aléatoire $Y = \{Y_s : s \in \Omega\}$ est un objet aléatoire sur un ensemble d'indices à deux dimensions, tel que*

- $\Omega = \{(i,j)|1 \leq i \leq n, 1 \leq j \leq m\}$ *est un ensemble d'indices de points sur une grille rectangulaire ;*
- *pour chaque point $s = (i,j) \in \Omega$, appelé pixel ou site, Y_s est une variable aléatoire à valeur réelle ;*
- *et le champ aléatoire Y est caractérisé par une fonction de densité conjointe $f(Y)$ pour le cas d'un champ aléatoire continu ou d'une fonction de masse conjointe $p(Y)$ dans le cas d'un champ aléatoire discret.*

Définition 1.1.2. *Un champ aléatoire est dit stationnaire s'il est invariant par rapport aux translations, c'est-à-dire que sa fonction de densité conjointe dépend uniquement de la distance entre les sites et non pas de leur position absolue.*

Dans le cas d'une image représentée par un champ aléatoire stationnaire noté Y, à chaque site s la variable aléatoire Y_s prend une valeur sur un domaine $\mathcal{D}^p$, où $\mathcal{D}$ est un compact. La dimension p dépend du fait que nous observons une image en noir et blanc ($p = 1$) ou bien une image en couleur ($p = 3$). Pour une image en noir et blanc, Y_s est un singleton dans $\mathcal{D}$, c'est-à-dire que le niveau de gris est représenté par une variable aléatoire en une dimension. Dans le cas d'une image en couleur, Y_s est un triplet dans $\mathcal{D}^3$.

Pour une image en couleur, le domaine $\mathcal{D}^p$ dépend de la manière dont nous représentons la couleur. Il existe plusieurs systèmes de référence pour représenter les couleurs, le plus connu étant le système RGB. Nous expliquons plus en détail à la section 1.1.4 ces systèmes de référence. Pour une image en noir et blanc, le domaine $\mathcal{D}$ se résume à l'intervalle $[0, 255] \bigcap \mathbb{N}$. De manière générale, nous normalisons toujours l'espace de représentation sur le domaine $[0, 1]^3$ pour les images en couleurs et sur $[0, 1]$ pour les images en noir et blanc.

Sur le plan mathématique, nous pouvons voir le support de l'image Y comme un graphe Ω où les noeuds s correspondent aux pixels et les arrêtes sont représentées par le système de voisinage. Nous verrons à la section 1.1.2 le détail des systèmes de voisinage.

1.1.2. Voisinage

Nous introduisons dans cette section la notion de voisinage en imagerie. Nous utilisons la notation $N(s)$ pour définir le voisinage du pixel s. Comme il a été introduit à la section précédente, le système de voisinage représente les arrêtes qui relient les noeuds du graphe. Nous définissons un voisinage en termes mathématiques de la manière suivante.

Définition 1.1.3. *Soit s et t deux noeuds du graphe Ω et les sous-ensembles $N(\cdot)$ les voisinages associés, alors $N(s)$ est un voisinage de s si et seulement si :*

(1) $s \notin N(s)$,

(2) $t \in N(s) \Rightarrow s \in N(t)$.

Les systèmes de voisinage les plus usuels sont les systèmes hiérarchiques. Ces systèmes représentent toujours une région symétrique circulaire autour du pixel s et sont inclusifs, c'est-à-dire que le voisinage d'ordre h du pixel s est toujours inclus dans son voisinage d'ordre $h + 1$. Voir la figure 1.1.1 pour un exemple de voisinage hiérarchique. De plus, un système de voisinage hiérarchique doit avoir la même structure pour tous les pixels de l'image sauf possiblement aux frontières. Il existe trois manières de traiter les points sur les frontières.

(1) Frontière libre : Nous considérons les pixels à l'extérieur du domaine de l'image comme étant manquants. Les points sur les frontières contiennent donc moins de voisins que ceux à l'intérieur de l'image.

(2) Frontière périodique : L'image est répétée dans toutes les directions selon un modèle toroïdal. De cette manière, les pixels de la frontière de droite deviennent adjacents à ceux de la frontière de gauche par exemple et de même pour les frontières inférieure et supérieure.

(3) Frontière étendue : Les points sur les frontières sont dupliqués sur les colonnes et les rangés extérieures. En d'autres mots, l'image est agrandie avec les mêmes valeurs que celles qui sont observées aux frontières.

Dans ce mémoire, nous considérons que l'approche par frontières libres. Ainsi, les ensembles de voisins des pixels aux frontières auront une cardinalité inférieure à celle des pixels intérieurs.

Définition 1.1.4. *Le voisinage d'ordre h du pixel s est défini comme tous les pixels compris dans un cercle de rayon $2^{\frac{h-1}{2}}$ centré au pixel s.*

Les systèmes de voisinage hiérarchiques les plus utilisés sont les systèmes d'ordre 1 et 2. Le voisinage d'ordre 1 d'un pixel s est défini comme l'ensemble des pixels à l'intérieur d'un cercle de rayon 1 autour de s. Le système d'ordre 1 se résume aux pixels adjacents verticalement ou horizontalement à s.

Considérons l'exemple de la figure 1.1.1 qui représente une image de 5×5 pixels, le voisinage de premier ordre de $s_{3,3}$ est donné par

$$N_1(s_{3,3}) = \{s_{2,3}, s_{3,2}, s_{3,4}, s_{4,3}\}.$$

De même, pour un pixel aux frontières, disons $s_{1,1}$, son voisinage de premier ordre est donné par $N_1(s_{1,1}) = \{s_{1,2}, s_{2,1}\}$.

Nous définissons le voisinage d'ordre 2 d'un pixel s comme l'ensemble des pixels compris à l'intérieur d'un cercle de rayon $\sqrt{2}$ centré en s. Ce système de voisinage se résume aux pixels adjacents verticalement, horizontalement ou diagonalement à s.

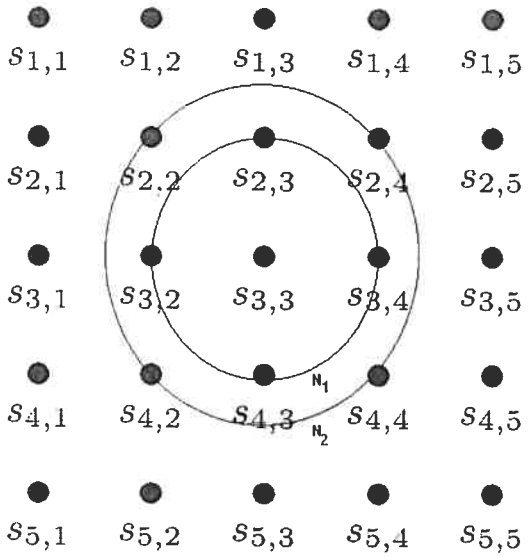

FIGURE 1.1.1. Image de 5×5 pixels avec un exemple d'un voisinage hiérarchique d'ordre 1 et d'ordre 2

Si nous reprenons l'exemple précédant, le voisinage de deuxième ordre de $s_{3,3}$ est donné par

$$N_2(s_{3,3}) = \{s_{2,2}, s_{2,3}, s_{2,4}, s_{3,2}, s_{3,4}, s_{4,2}, s_{4,3}, s_{4,4}\}.$$

Comme nous avons mentionné précédemment, nous pouvons remarquer que $N_1(s_{3,3}) \subset N_2(s_{3,3})$. Tout au long du travail présenté dans ce mémoire, nous utilisons le voisinage d'ordre 2 pour nos modèles.

1.1.3. Cliques et fonction d'énergie

Après avoir défini un système de voisinage, nous pouvons construire un système de cliques $\mathcal{C}$ par rapport au système de voisinage $N(\cdot)$.

Définition 1.1.5. *Une clique $c \in \mathcal{C}$ est un sous-graphe du support de l'image Y où tous ses éléments sont voisins les uns des autres. Par convention, un singleton $\{s\}$ de Ω est également une clique.*

Notons $\mathcal{C}_k$ l'ensemble des cliques de cardinalité k, c'est-à-dire les cliques contenant k pixels. La figure 1.1.2 montre les cliques associées aux voisinages d'ordre 1 et 2. Si nous retournons à l'exemple de la figure 1.1.1, pour un voisinage d'ordre

deux, les cliques de cardinalité 2 contenant le pixel $s_{1,1}$ sont données par

$$c_2(s_{1,1}) = \{(s_{1,1}, s_{1,2}), (s_{1,1}, s_{2,1}), (s_{1,1}, s_{2,2})\}.$$

Les cliques de cardinalité 3 et 4 contenant le pixel s_1 sont données respectivement par

$$c_3(s_{1,1}) = \{(s_{1,1}, s_{1,2}, s_{2,1}), (s_{1,1}, s_{1,2}, s_{2,2}), (s_{1,1}, s_{2,1}, s_{2,2})\},$$

$$c_4(s_{1,1}) = \{(s_{1,1}, s_{1,2}, s_{2,1}, s_{2,2})\}.$$

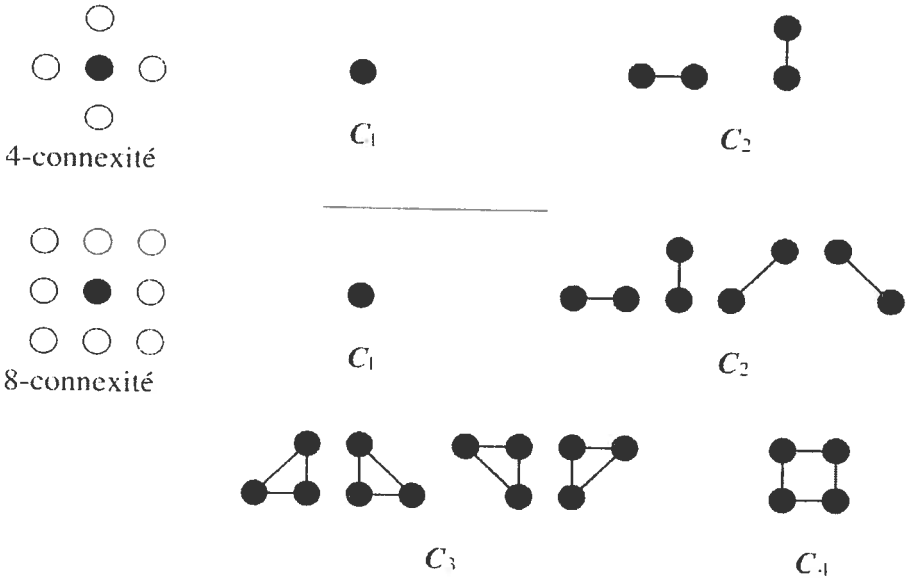

FIGURE 1.1.2. Cliques associées aux voisinages d'ordre 1 et 2 (voir Maître, 2003)

Pour chaque clique c, nous définissons une fonction de potentiel $\mathcal{U}_c$ qui dépend uniquement de la valeur des pixels qui constituent la clique c.

Définition 1.1.6. *Soit Λ l'ensemble des valeurs que peut prendre le champ aléatoire X, Ω le support du champ et $\#(\Omega)$ la cardinalité de Ω, alors $\mathcal{U}_c$ est définie comme $\mathcal{U}_c : \Lambda^{\#(\Omega)} \to \mathbb{R}_+$ tel que*

(1) $\mathcal{U}_c \equiv 0$ si c n'est pas une clique

(2) pour deux configurations x et x' alors,

$$x_s = x'_s \, \forall s \in c \;\Rightarrow\; \mathcal{U}_c(x) = \mathcal{U}_c(x').$$

Nous définissons l'énergie globale de l'image comme la somme des potentiels de toutes les cliques composant l'image, c'est-à-dire,

$$U = \sum_{c \in C} U_c.$$

L'énergie locale pour un pixel s est définie par la somme des potentiels de toutes les cliques qui contiennent le pixel s :

$$U_s = \sum_{c \in C | s \in c} U_c.$$

Nous référons à l'exemple 1.1.1 pour illustrer l'énergie globale dans un cadre concret.

1.1.4. Systèmes de représentation

La colorimétrie est la science qui étudie la vision des couleurs. Un des grands pas historiques en compréhension de la perception des couleurs vient de Maxwell (1831-1879) qui reprit l'étude célèbre de Newton de la décomposition de la lumière par un prisme. Il montra qu'il est possible de recréer la couleur blanche qu'avec trois couleurs, soient le rouge, le vert et le bleu. Ainsi, le principe de *trichromie* est né. Par la suite nous démontrons que toute couleur peut être recréée à partir d'un mélange de trois couleurs primaires correctement choisies. Par contre, ce dernier postulat ne garantit pas que pour trois primaires fixés, on pourra recréer toutes les couleurs. Certaines couleurs vont nécessiter un changement des trois primaires. Pour plus de détails du le sujet, voir Marion (1997).

Comme nous avons mentionné à la section 1.1.1, le système de représentation le plus habituel pour une couleur est le système RGB. Ce système est celui qui est utilisé par la plupart des écrans de télévision réguliers. De manière générale, l'espace RGB est la représentation sur trois plans de la composition de l'image en rouge (Red), vert (Green) et bleu (Blue). Nous pouvons nous demander pourquoi ces trois couleurs en particulier. Des études ont montré que pour obtenir une palette de couleur la plus vaste possible, il faut premièrement des couleurs primaires indépendantes, c'est-à-dire qu'aucune d'elles peut être recréée à partir des deux autres et elles doivent être le plus éloignées les unes des autres dans le spectre.

Il fut ainsi convenu universellement que le rouge, le vert et le bleu sont de bons candidats.

Le domaine par défaut du système RGB est $[0, 255] \bigcap \mathbb{N}$ pour chaque composante. De manière générale, nous normalisons l'image sur le domaine $[0, 1]^3$ pour faciliter le traitement numérique. Nous pouvons visualiser le système RGB comme un cube en trois dimensions (voir la figure 1.1.3).

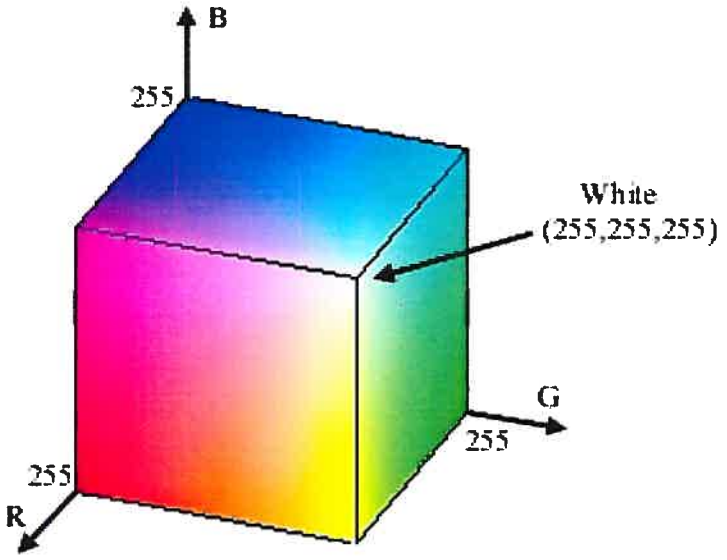

FIGURE 1.1.3. Cube RGB

Un des problèmes reliés à l'utilisation du système RGB en traitement d'images est la notion de distance entre deux couleurs. En effet, prenons deux couleurs qui sont très près en terme de distance euclidienne dans le cube RGB ; la différence perçue par l'oeil entre les deux couleurs peut être très grande et vice versa pour deux couleurs très semblables visuellement, ces dernières peuvent être très distantes dans le cube. Nous disons alors que le système RGB n'est pas *perceptuellement uniforme*. Nous allons voir plus loin que certains systèmes de représentation tentent de corriger cette non-uniformité perceptuelle.

Comme nous avons déjà mentionné, il existe plusieurs systèmes de représentation des couleurs, par exemple les espaces RGB, XYZ, HSV, YIQ, Luv, Lab pour n'en nommer que quelques uns. En fait, toute transformation bijective d'un

système de représentation nous donnera un nouveau système valide. De cette manière, plusieurs dérivés du système RGB on été conçus pour répondre à certains besoins technologiques dans différents domaines. Nous ne présentons ici que les espaces qui nous seront utiles, soient les systèmes XYZ, YIQ et Lab.

Le système XYZ fut développé par la CIE (Commission Internationale de l'Éclairage) en 1931 pour remplacer les coordonnées RGB car il est impossible de représenter toutes les couleurs visibles par des valeurs positives des composantes R, G et B. La composante Y du système XYZ a été intentionnellement conçue pour représenter la luminescence, ou la luminosité de la couleur. La luminescence est définie en physique comme l'intensité d'une source quelconque. Prenons par exemple une image en niveau de gris, puisqu'il n'y a aucun apport en terme de couleur, tout ce que nous observons est la luminescence.

Le système YIQ est une transformation de l'espace RGB développé pour la transmission de signaux télévision. La composante Y représente la luminescence exactement comme pour le système XYZ. Cette dernière composante se voit attribuer plus d'espace dans le signal transmis car l'oeil humain est plus sensible à une variation dans la luminescence que dans la couleur. Les images télévisées ainsi obtenues sont plus nettes à cause de l'emphase mise sur cette caractéristique.

Tous les systèmes mentionnés précédemment ne sont pas perceptuellement uniformes. Plusieurs systèmes ont été proposés par la CIE pour tenter de résoudre ce problème. Un de ces systèmes est la représentation Lab qui fut développée en 1976 par la CIE. Ce dernier est une transformation non linéaire du système XYZ. La représentation visuelle du système Lab est cependant plus difficile que pour le système RGB. La composante L correspond à la luminescence ou bien la luminosité de la couleur. Si nous fixons a et b, alors nous obtenons la variation de L sur les niveaux de gris. La composante a représente la variation vert-rouge et la composante b la variation jaune-bleu. Le domaine de chaque composante n'est plus le même, L varie sur $[0, 100]$ et a et b varient sur $[-60, 60]$. Encore une fois, chacune des composantes sera normalisée sur le domaine $[0, 1]$.

Pour faire le passage du système RGB au système Lab, il faut premièrement passer par le système XYZ. Pour ce faire, nous avons besoin des deux premières

coordonnées projectives des trois composantes R, G et B de référence notée respectivement (x_r, y_r), (x_g, y_g) et (x_b, y_b) ainsi que des composantes du blanc de référence X_W, Y_W et Z_W (nous pouvons également donner ses coordonnées projectives). Pour les trois couleurs primaires de référence, les coordonnées projectives sont obtenues par

$$x_l = \frac{R}{R+G+B},$$
$$y_l = \frac{G}{R+G+B},$$
$$z_l = \frac{B}{R+G+B},$$

où $l = r, g,$ et b. Notons que la somme des coordonnées projectives donne toujours 1, voilà pourquoi nous donnons uniquement que les deux premières coordonnées. Le passage entre les deux systèmes se fait par le produit matriciel suivant

$$\begin{pmatrix} X & Y & Z \end{pmatrix} = \begin{pmatrix} R & G & B \end{pmatrix} M,$$

où

$$M = \begin{pmatrix} S_r X_r & S_r Y_r & S_r Z_r \\ S_g X_g & S_g Y_g & S_g Z_g \\ S_b X_b & S_b Y_b & S_b Z_b \end{pmatrix}.$$

Les composantes X_r, Y_r, Z_r sont données par

$$X_r = x_r / y_r$$
$$Y_r = 1$$
$$Z_r = (1 - x_r - y_r)/y_r$$

et la forme est la même pour le vert (X_g, Y_g, Z_g) et le bleu (X_b, Y_b, Z_b). Les composantes S_r, S_g et S_b sont obtenues par le produit matriciel suivant

$$\begin{pmatrix} S_r & S_g & S_b \end{pmatrix} = \begin{pmatrix} X_W & Y_W & Z_W \end{pmatrix} \begin{pmatrix} X_r & Y_r & Z_r \\ X_g & Y_g & Z_g \\ X_b & Y_b & Z_b \end{pmatrix}^{-1}.$$

TABLEAU 1.1.1. Coordonnées projectives des trois couleurs primaires et du blanc de référence utilisées par Matlab

	x	y
R	0,64	0,33
G	0,30	0,60
B	0,15	0,06
D65	0,3127	0,3290

Il n'y a pas de définition physique ou perceptuelle unique de la couleur blanche. Le blanc est défini comme ce que nous considérons être la lumière blanche. Souvent, le blanc de référence représente la lumière la plus blanche que peut produire un appareil (voir http ://developer.apple.com/documentation/QuickTime/-REF/refVectors.21.htm). Ainsi, nous voyons que le blanc de référence peut dépendre de l'appareil, de l'écran ou du moniteur qui est utilisé, et par conséquent, il n'est pas unique. Le transfert entre le système RGB et XYZ dans Matlab est effectué par rapport au blanc de référence D65. Ce dernier fut développé par la CIE de manière a approximer la lumière naturelle du jour. Les coordonnées projectives pour les trois couleurs primaires utilisées par Matlab ainsi que pour le blanc de référence D65 sont données dans le tableau 1.1.1 (voir www.brucelindbloom.com).

Maintenant voici comment passer du système XYZ au système Lab :

$$L = \begin{cases} 116(Y/Y_W)^{1/3} - 16, & \text{si } Y/Y_W > 0{,}008856\,; \\ 903{,}3(Y/Y_W), & \text{sinon,} \end{cases}$$

$$a = 500(g(X/X_W) - g(Y/Y_W)),$$

$$b = 200(g(Y/Y_W) - g(Z/Z_W)),$$

où

$$g(t) = \begin{cases} t^{1/3}, & \text{si } Y/Y_W > 0{,}008856\,; \\ 7{,}787t + 16/116, & \text{sinon,} \end{cases}$$

et X_W, Y_W et Z_W sont les composantes du blanc de référence.

1.1.5. Champs aléatoires de Markov et champs de Gibbs

Nous présentons dans cette section les bases de la théorie des champs de Markov. Ces derniers sont des outils importants de notre modèle, ainsi il est important de bien comprendre leur fonctionnement. Toutes les définitions et théorèmes de cette section proviennent de Maître (2003) et de Won et Gray (2004).

1.1.5.1. *Définition d'un champ de Markov*

Les premiers travaux sur les champs de Markov ont commencé dans les années 50. Depuis les travaux de Geman et Geman (1984) et Besag (1986), l'utilisation de ces derniers a beaucoup évoluée en imagerie pour les problèmes de segmentation et de restoration d'images.

Considérons x_s la valeur de la variable aléatoire X au pixel s et $x^s = (x_t)_{s \neq t}$ l'ensemble des valeurs pour tous les pixels excepté le site s. Un champ de Markov est alors défini comme suit.

Définition 1.1.7. *Un champ aléatoire X est dit un champ de Markov si et seulement si la probabilité conditionnelle locale en un site n'est fonction que de la configuration du voisinage du site considéré, c'est-à-dire*

$$\Pr\Big(X_s = x_s | x^s\Big) = \Pr\Big(X_s = x_s | x_t, t \in N(s)\Big).$$

Une manière simple d'interpréter la propriété markovienne est que la valeur que prend un pixel ne dépend que des valeurs de ses voisins. Il est intéressant de remarquer que pour une définition de voisinage qui s'étend sur l'image au complet, tout champ aléatoire a la propriété d'être markovien.

Comme il est mentionné dans Maître (2003), l'hypothèse markovienne s'applique bien au cas des images naturelles vu leur composition par des zones relativement homogènes. Ce modèle peut cependant être moins applicable pour certaines images synthétiques.

1.1.5.2. *Équivalence avec les champs de Gibbs*

Commençons tout d'abord par définir ce qu'est un champ de Gibbs.

Définition 1.1.8. *Un champ aléatoire X est un champ aléatoire de Gibbs (ou un champ aléatoire avec une mesure de Gibbs) si sa distribution conjointe est de la forme*

$$P(X = x) = \frac{1}{Z} \exp(-\mathcal{U}(x)), \tag{1.1.1}$$

où $\mathcal{U}(x) = \sum_{c \in C} \mathcal{U}_c(x)$ avec C le système de cliques associé au voisinage $N(\cdot)$ et Z une constante de normalisation appelée fonction de partition.

Notez que l'équation (1.1.1) est dite mesure ou distribution de Gibbs. Il est intéressant de noter ici que la définition d'un champ aléatoire de Gibbs ne dépend pas du choix de la fonction de potentiel choisie. Ainsi, l'utilisateur peut construire sa propre définition de potentiel de manière à mettre l'emphase sur une caractéristique particulière de l'image, en autant que la distribution résultante reste une probabilité. La constante de normalisation Z est donnée par

$$Z = \sum_{x} \exp(-\mathcal{U}(x)),$$

où la sommation est prise sur l'ensemble des configurations possibles. Il est facile de voir que cette constante n'est pas calculable en pratique pour un champ de dimension deux de taille raisonnable. Prenons tout simplement une image de taille $512 \times 512 = 262144$ qui ne prend que des valeurs binaires, c'est-à-dire $\Lambda = \{1, 2\}$. Dans ce cas simple, nous avons que $\text{Card}(\Lambda^{\#(\Omega)}) = 2^{262144} = 1{,}6 \times 10^{78913}$.

Exemple 1.1.1. *L'exemple suivant provient de Won et Gray (2004) et illustre l'impact du choix de la fonction de potentiel. Nous voulons montrer ici comment nous pouvons construire une fonction de potentiel selon le but du problème. Prenons les deux configurations d'images binaires de la figure 1.1.4. Supposons que nous voulons trouver le ratio des probabilités des configurations* $\Pr(X = x^{(2)}) / \Pr(X = x^{(1)})$ *et que nous ne sommes intéressés qu'aux cliques horizontales de cardinalité deux pour un système de voisinage de premier ordre. Il est facile de voir qu'il y a seulement 12 cliques horizontales de cardinalité deux pour ces images de* 4×4 *pixels. Définissons la fonction de potentiel comme suit :*

$$\mathcal{U}_c(x) = \begin{cases} \tau, & \textit{si les deux éléments de la clique horizontale sont égaux;} \\ -\tau, & \textit{sinon.} \end{cases}$$

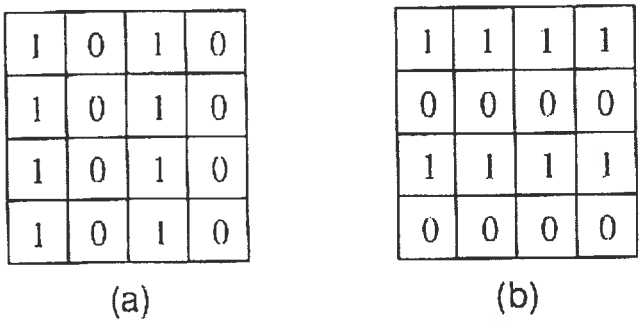

FIGURE 1.1.4. (a) Configuration $x^{(1)}$, (b) Configuration $x^{(2)}$ (Won et Gray, 2004)

Les probabilités de chacunes des deux configurations sont données par

$$P(X = x^{(1)}) = \frac{1}{Z} \exp\left\{\sum_{c \in \mathcal{C}} \mathcal{U}_c(x^{(1)})\right\} = \frac{1}{Z} \exp(-12\tau),$$

$$P(X = x^{(2)}) = \frac{1}{Z} \exp\left\{\sum_{c \in \mathcal{C}} \mathcal{U}_c(x^{(2)})\right\} = \frac{1}{Z} \exp(12\tau).$$

Le ratio des deux probabilités est ainsi donné par

$$\frac{P(X = x^{(2)})}{P(X = x^{(1)})} = \frac{\exp(12\tau)}{\exp(-12\tau)} = \exp(24\tau).$$

Donc, si nous posons $\tau = 1$, nous pouvons conclure que la configuration $x^{(2)}$ est environ $2{,}6 \times 10^{10}$ fois plus probable que la configuration $x^{(1)}$. Le résultat n'est pas surprenant compte tenu de la configuration horizontale de l'image 2. Si nous nous étions intéressés aux cliques verticales au lieu des cliques horizontales, nous aurions les mêmes résultats mais avec les signes inverses pour chaque configuration.

Nous pourrions définir de la même manière plusieurs fonctions de potentiel pour faire ressortir différentes caractéristiques de l'image.

Avec ces définitions en main, il est maintenant possible d'établir le lien qui existe entre les champs de Markov et les champs de Gibbs. Le théorème de Hammersley-Clifford établit cette relation dans le cas discret.

Théorème 1.1.1. *Soit Ω un graphe fini ou dénombrable, $N(\cdot)$ un système de voisinage borné et Λ un espace d'états discrets, alors X est un champ de Markov*

relativement à $N(\cdot)$ *et* $\Pr(X = x) > 0 \quad \forall x \in \Lambda^{\#(\Omega)}$ *si et seulement si* X *est un champ de Gibbs de potentiel associé à* $N(\cdot)$.

Le théorème précédent est très important pour mettre en relation les probabilités conditionnelles locales et globales. En effet, si nous désirons calculer $\Pr(X_s = x_s | X^s = x^s)$, nous avons que

$$\begin{aligned}\Pr(X_s = x_s | X^s = x^s) &= \frac{\Pr(X = x)}{\Pr(X^s = x^s)} \\ &= \frac{\exp(-\mathcal{U}(x_s, x^s))/Z}{\sum_{e\in\Lambda} \exp(-\mathcal{U}(e, x^s))/Z} \\ &= \frac{\exp(-\mathcal{U}(x_s, x^s))}{\sum_{e\in\Lambda} \exp(-\mathcal{U}(e, x^s))},\end{aligned}$$

avec Λ l'ensemble des états que peut prendre X. Si nous définissons l'énergie locale comme

$$\mathcal{U}_s(x_s | x_t, t \in N(s)) = \sum_{c\in\mathcal{C}|s\in c} \mathcal{U}_c(x_s, x_t, t \in N(s)) = \sum_{c\in\mathcal{C}|s\in c} \mathcal{U}_c(x_s, N(s)),$$

alors nous pouvons voir que seulement les voisins interviennent. L'énergie globale de l'image peut maintenant s'écrire comme

$$\mathcal{U}(x) = \sum_{c\in\mathcal{C}|s\notin c} \mathcal{U}_c(x) + \sum_{c\in\mathcal{C}|s\in c} \mathcal{U}_c(x) = \left(\sum_{c\in\mathcal{C}|s\notin c} \mathcal{U}_c(x) \right) + \mathcal{U}_s(x_s | N(s)).$$

Réécrivons maintenant la probabilité conditionnelle du site s par rapport au reste de l'image comme

$$\begin{aligned}\Pr(X_s = x_s | X^s = x^s) &= \frac{\exp\left(- \left[\sum_{x\in\mathcal{C}|s\notin c} \mathcal{U}_c(x) \right] - \mathcal{U}_s(x_s | N(s)) \right)}{\sum_{e\in\Lambda} \exp\left(- \left[\sum_{x\in\mathcal{C}|s\notin c} \mathcal{U}_c(x) \right] - \mathcal{U}_s(e | N(s)) \right)} \\ &= \frac{\exp\left(-\mathcal{U}_s(x_s | N(s)) \right)}{\sum_{e\in\Lambda} \exp\left(-\mathcal{U}_s(e | N(s)) \right)}. \qquad (1.1.2)\end{aligned}$$

L'expression précédente est très utile en ce sens qu'elle nous permet d'éviter le calcul de la fonction de partition en passant par la probabilité locale en chaque site. Cette probabilité locale ne faisant intervenir que les sites voisins au pixel s, nous retrouvons ainsi la propriété markovienne. Tous les algorithmes de simulation d'un champ markovien sont basés sur cette relation. L'équation (1.1.2) est souvent appelée pseudo-vraisemblance.

1.1.5.3. *Application au cas de la segmentation*

Attardons-nous maintenant à l'application des champs de Markov au problème de segmentation d'une image dans un cadre bayésien. Posons $Y = \{Y_s\}$ un champ aléatoire continu de couleurs dont nous observons une réalisation y et $X = \{X_s\}$ un champ aléatoire de Markov discret d'étiquettes non observé qui prend ses valeurs dans l'ensemble fini $\Lambda = \{e_1, e_2, ..., e_k\}$. Puisque les observations y ne sont pas une réalisation du champ X, nous parlons ainsi de champ aléatoire de Markov caché pour X. Le but de la segmentation dans le contexte bayésien est de retrouver une réalisation x de X basée sur les observations y. Nous nous intéressons donc à la probabilité *a posteriori* $\Pr(X = x|Y = y)$ qui à l'aide de la règle de Bayes est donnée par

$$\Pr(X = x|Y = y) = \frac{\Pr(Y = y|X = x)\Pr(X = x)}{\Pr(Y = y)}.$$

L'hypothèse markovienne sur X suppose que nous avons un modèle *a priori* de la forme

$$\Pr(X = x) = \frac{\exp(-\mathcal{U}(x))}{Z},$$

avec une certaine fonction de potentiel $\mathcal{U}$. Nous pouvons exprimer la fonction *a posteriori* sous la forme suivante

$$\begin{aligned} P(X = x|Y = y) &\propto \Pr(Y = y|X = x)\Pr(X = x) \\ &\propto e^{\log \Pr(Y=y|X=x) - \mathcal{U}(x)} \\ &\propto e^{-\mathcal{U}(x|y)}, \end{aligned}$$

avec $\mathcal{U}(x|y) = -\sum_{s\in\Omega} \log p(y_s|x_s) + \sum_{c\in C} \mathcal{U}_c(x)$. Nous voyons que la distribution *a posteriori* reste une distribution de Gibbs. Ainsi, par le théorème de Hammersley-Clifford nous avons que le champ X conditionnellement à y est un champ de Markov.

1.1.6. Modèle isotropique de Potts

Nous avons défini les champs de Gibbs par rapport à une fonction de potentiel. Plusieurs modèles peuvent être utilisés pour modéliser *a priori* le champ de

Markov caché X. Dans notre cas, nous utilisons un modèle isotropique de Potts à deux dimensions donné par

$$f(x) = \frac{1}{Z(\tau)} \exp\left\{-\tau \sum_{<s,t>} (1 - \delta(x_s, x_t))\right\}, \tag{1.1.3}$$

où la sommation est prise sur toutes les paires de voisins, $\tau > 0$ un paramètre et δ la fonction delta de Kronecker. Le dénominateur $Z(\tau)$ représente la fonction de partition qui est donnée par

$$Z(\tau) = \sum_{x} \exp\left\{-\tau \sum_{<s,t>} (1 - \delta(x_s, x_t))\right\}.$$

Il est à rappeler ici que la sommation est prise sur toutes les configurations possibles.

Comme $\tau > 0$, nous remarquons que le modèle favorise les régions plus homogènes. Le modèle de Potts est une généralisation du modèle d'Ising qui a été développé à l'origine pour modéliser l'attraction de particules en ferro-magnétisme en physique statistique. Ce dernier permet seulement deux classes d'étiquettes pour la variable aléatoire X.

Il est également possible de considérer différentes pondérations pour le paramètre τ selon les cliques considérées. En effet, certains auteurs proposent entre autre une pondération de l'ordre $1/\sqrt{2}$ pour les cliques diagonales pour un voisinage d'ordre 2 pour tenir compte de la distance euclidienne dans le plan. Pour notre part, nous n'effectuons pas de distinction entre les directions dans nos modèles.

Nous référons le lecteur à la section 6.1.6 de l'ouvrage de Maître (2003) pour des exemples de simulations de champs aléatoires de Markov pour différentes fonctions de potentiel.

1.2. Approche développée par Destrempes *et al.* (2005)

Dans cette section nous décrivons brièvement la méthode de segmentation d'une image introduite dans Destrempes *et al.* (2005) et les applications à un modèle de couleur.

1.2.1. Système de représentation des couleurs

Comme nous l'avons vu précédemment, l'espace RGB est rarement utilisé en traitement d'images. Dans Destrempes *et al.* (2005), le système de représentation des couleurs qui est utilisé est le système YIQ. Un modèle de vraisemblance sera appliqué à chacune des composantes de l'image et la segmentation sera effectuée conjointement sur le triplet de couleurs. Ainsi, une décorrélation de l'image par rapport à ses trois composantes est nécessaire. La décorrélation du cube est effectuée en appliquant tout d'abord le difféomorphisme $\xi : (0,1)^p \to \mathbb{R}^p$ où $p = 3$, tel que $\xi(x) = \tanh^{-1}(2x-1)$, sur chacune des composantes y_s normalisées sur $[0,1]$. La modélisation est ensuite effectuée sur les composantes décorrélées u_s données par

$$u_s = \xi^{-1}\Big(W_{(k)}^t\big(\xi(y_s) - \nu_{(k)}\big) + \nu_{(k)}\Big),$$

où $\nu_{(k)}$ est le vecteur moyenne des composantes transformées $\xi(y_s)$ du segment k et $W_{(k)}$ une matrice orthogonale de décorrélation. L'indice k fait référence au fait que la décorrélation est effectuée indépendemment pour chaque classe de l'image. Le difféomorphisme ξ^{-1} est utilisé pour s'assurer qu'après la rotation du cube nous soyons toujours dans l'espace $(0,1)^p$.

1.2.2. Modèle statistique bayésien

Le contexte est toujours celui des champs de Markov cachés, c'est-à-dire que nous observons un champ aléatoire Y et nous posons un champ de Markov caché X sous-jacent à l'image. La vraisemblance $\Pr(y|x)$ est définie par le produit sur tous les noeuds du graphe $\prod_{s\in\Omega} P(y_s|x_s)$, c'est-à-dire que sachant x, les observations y sont indépendantes. Dans Destrempes *et al.* (2005), les composantes transformées u_s sont modélisées par des densités bêta,

$$u_{s,r} \sim B(u_{s,r}|\alpha_{k,r}, \beta_{k,r}, x_s = k),$$

avec $r \in \{1,2,3\}$ les trois composantes du système YIQ et $B(x,|\alpha,\beta)$ donné par

$$B(x|\alpha,\beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1}, \tag{1.2.1}$$

où $x \in (0,1)$, $\alpha, \beta > 0$ et Γ est la fonction gamma définie par $\Gamma(z) = \int_0^\infty e^{-v} v^{z-1} dv$. Nous utilisons la notation $\alpha_{k,r}$ et $\beta_{k,r}$ pour bien mettre en évidence la dépendance de la fonction de vraisemblance sur la classe k auquel appartient le pixel s. Par hypothèse d'indépendance suite à la décorrélation, la vraisemblance pour le pixel s est donnée par

$$B(u_s|x_s = k) = \prod_r B(u_{s,r}|\alpha_{k,r}, \beta_{k,r}, x_s = k).$$

La valeur des étiquettes x est modélisée *a priori* par un modèle isotropique de Potts tel que décrit à la section 1.1.6 avec un système de voisinage d'ordre 2 et une contrainte pour identifier les classes allouées. Rappelons que le modèle est défini à l'équation (1.1.3). La contrainte sur le nombre de classes passe par un vecteur ν de taille K qui prend la valeur 1 si la classe est allouée et 0 sinon. Le vecteur ν est évidemment contraint à allouer au moins une classe, c'est-à-dire $1 \leq \sum_{k=1}^K \nu_{(k)} \leq K$. Une segmentation x est dite admise par ν si toutes les classes e_k de x satisfont $\nu_{(k)} = 1$. Soit $\chi(e_k, \nu) = \nu_{(k)}$ nous définissons alors la contrainte par $\prod_s \chi(x_s, \nu)$. La segmentation x est donc modélisée *a priori* par

$$\pi(x|\tau, \nu) = \frac{1}{Z(\tau, \nu)} \exp\left\{-\tau \sum_{<s,t>} (1 - \delta(x_s, x_t))\right\} \prod_s \chi(x_s, \nu).$$

Dans Destrempes *et al.* (2005), le paramètre τ est fixé à 1, la densité *a priori* ne dépend donc que du paramètre ν.

Un autre élément est ajouté au modèle. Nous ajoutons une contrainte globale sur la segmentation pour restreindre le nombre de régions dans l'image. Cette contrainte est basée sur la fonction d'énergie ρ donnée par

$$\rho(x) = \omega n |G|^{1/2} + \sum_{i=1}^n \Big[2\log(|R_i(x)|) + \log(1 - 1/|G|)\Big], \tag{1.2.2}$$

où $R_i(x), i = 1 \ldots n$ sont les n régions connectées dans l'image associées à x. Une région $R_i(x)$ est dite connectée si chaque pixel $s \in R_i(x)$ admet au moins un voisin de même étiquette de région. De plus, $|R_i(x)|, i = 1 \ldots n$ est le nombre de pixels de chacune de ces n régions connectées, ω un paramètre fixé à 1 ou 0 dans Destrempes *et al.*, (2005) et $|G|$ la taille de l'image. Lorsque le paramètre ω est

fixé à 1, la contrainte globale favorise une segmentation avec moins de régions connectées et l'inverse lorsqu'il est fixé à 0.

Comme il est mentionné dans Destrempes *et al.* (2005), il est à noter que la fonction de partition $Z(\tau, \nu)$ augmente à un rythme exponentiel lorsque nous augmentons le nombre de classes allouées. Ceci est tel qu'il devient impossible de comparer des vraisemblances lorsque nous avons un nombre différent de régions. Pour tenir compte de ce fait, nous insérons un terme dans la fonction d'énergie qui vient annuler la fonction de partition. Nous définissons ainsi $\rho(x, \nu) = \rho(x) - \log(Z(\tau, \nu))$. La contrainte globale sur la segmentation est donc donnée par

$$e^{-\rho(x)+\log(Z(\tau,\nu))}.$$

La contrainte globale peut être incluse soit dans le modèle *a priori* ou dans la fonction de perte, les résultats seront les mêmes. Dans l'approche de Destrempes *et al.* (2005), elle est incluse dans la fonction de perte.

1.2.3. Contexte décisionnel

La fonction de perte considérée dans Destrempes *et al.* (2005) est la fonction de perte dichotomique 0-1 pondérée par la contrainte globale. Pour alléger la notation, définissons les paramètres du modèle de vraisemblance par $\Phi = (\underset{\sim}{\alpha}, \underset{\sim}{\beta})$ et ceux du modèle *a priori* par $\Psi = (\tau, \nu)$. La fonction de perte est donc donnée par

$$\mathbb{L}(\Theta, \theta) = 1 - e^{-\rho(X,\Psi)}\delta(\Theta, \theta),$$

où $\Theta = (X, \Phi, \Psi)$, $\theta = (x, \phi, \psi)$ et δ la fonction de Kronecker pour des variables discrètes ou la fonction delta pour des variables continues.

Nous avons maintenant un modèle bayésien complètement spécifié. Nous cherchons maintenant à estimer les paramètres du modèle. Pour résumer avec la notation simplifiée, nous avons la vraisemblance du modèle de champs aléatoires de Markov cachés donnée par $\Pr(y|\Theta) = \Pr(y|X, \Phi)$. L'estimateur bayésien est alors défini comme

$$\hat{\theta}(y) = \arg\min_{\theta} \int_{\Theta} \mathbb{L}(\Theta, \theta) \Pr(\Theta|y) d\Theta$$

$$= \arg\min_{\theta} \int_{\Theta} \mathbb{L}(\Theta, \theta) \frac{\Pr(y|\Theta) \Pr(\Theta)}{\Pr(y)} d\Theta. \tag{1.2.3}$$

Puisque $\Pr(y) = \int_{\Theta} \Pr(y|\Theta) \Pr(\Theta) d\Theta$ ne dépend pas de θ, on a que l'équation 1.2.3 se résume à

$$\hat{\theta}(y) = (x_*, \Phi_*, \Psi_*) = \arg\max_{x,\phi,\psi} e^{-\rho(x,\psi)} \Pr(y|x, \phi) \Pr(x|\psi).$$

L'estimateur obtenu est ainsi un maximum *a posteriori* pondéré par la contrainte globale. Comme il est mentionné dans Destrempes *et al.* (2005), les méthodes existantes pour maximiser cette fonction dans un tel contexte comme le recuit simulé ou le recuit simulé adaptatif ont ou bien trop de paramètres pour être suivies ou bien convergent vers une solution sous optimale.

1.2.4. Estimation du modèle

Une nouvelle méthodologie appelée Exploration/Sélection/Estimation (ESE) est proposée dans l'article pour optimiser les paramètres du modèle. Cette dernière est une généralisation de l'algorithme Exploration/Sélection (E/S) de François (2002). Nous n'effectuons ici qu'un bref survol de la méthodologie ; nous référons à Destrempes *et al.* (2005) pour de plus amples détails sur la procédure.

Un peu comme l'algorithme EM, le ESE est une procédure itérative qui utilise le vecteur de données complètes (x, y), c'est-à-dire que nous supposons le champ des étiquettes x connu et estimons Φ, les paramètres du modèle de vraisemblance, par maximum de vraisemblance. Une fois les estimateurs du maximum de vraisemblance (EMV) calculés, nous cherchons

$$(x^*, \Phi^*) = \arg\max_{x,\phi} e^{-\rho(x,\psi)} P(y|x, \phi) P(x|\psi).$$

Comme il a déjà été mentionné à la section 1.2.2, le paramètre τ dans le modèle de Potts étant fixé à 1, le paramètre Φ se résume au vecteur des étiquettes allouées ν. Le problème se réduit donc à minimiser la fonction

$$f(x, \nu) = \rho(x) - \log(P(y|x, \hat{\Phi})) + \tau \sum_{<s,t>} (1 - \delta(x_s, x_t))$$

sur l'espace A de toute les réalisations (x, ν) telles que x est alloué par ν.

Dans Destrempes *et al.* (2005), la segmentation initiale utilisée pour x est calculée à l'aide de l'algorithme des K-moyennes. Cet algorithme est fréquemment utilisé en traitement d'images pour sa simplicité. Il consiste essentiellement en un modèle de classification basé sur des hypothèses gaussiennes avec des matrices de variance-covariance diagonales ayant la même variance pour chacune des classes.

(1) Premièrement nous supposons un nombre fixe de classes K et nous assignons aléatoirement une classe à chaque pixel ;

(2) nous calculons le vecteur moyen des composantes transformées u_s pour chaque classe ;

(3) nous réassignons chaque pixel à la classe qui a la plus proche moyenne ;

(4) nous répétons les étapes 2 et 3 jusqu'à convergence.

1.3. Éléments de statistique

Dans cette section, nous présentons les algorithmes et les méthodes d'estimation numériques que nous implantons pour mettre en oeuvre le modèle proposé.

1.3.1. Algorithme EM

Même si des versions semblables à l'algorithme EM ont été exprimées par d'autres auteurs auparavant, la référence de base est Dempster, Laird et Rubin (1977). L'algorithme EM est un outil itératif simple et très puissant pour calculer le maximum de vraisemblance lorsque nous considérons notre vecteur d'observations comme des données incomplètes.

1.3.1.1. *Segmentation en tant que problème inverse*

Nous avons mentionné ci-dessus que l'algorithme EM est un outil puissant pour traiter des données incomplètes ou manquantes. Cependant, quelle relation pouvons-nous faire entre le problème de segmentation d'une image et celui des données incomplètes ? Il est possible de traiter la segmentation comme un problème statistique inverse. Considérons tout d'abord le problème direct. En considérant la notation proposée à la sous-section 1.1.5.3, si nous connaissons l'étiquette e_k du pixel s, alors la couleur observée dans l'image à ce pixel peut

être simulée à partir de la densité associée au segment e_k. Le problème vient du fait que le champ des étiquettes n'est pas connu et ainsi nous faisons face à un problème inverse.

Nous voulons donc à partir d'un vecteur d'observations $\underset{\sim}{y} \sim f(\underset{\sim}{y}|\Theta)$ et d'une densité *a priori* $\pi(x|\Phi)$ sur X, estimer les paramètres Θ du modèle de vraisemblance et les étiquettes pour chaque pixel. L'estimation des étiquettes est effectuée en maximisant la densité *a posteriori*

$$x_s^* = \arg\max_k \pi(x_s = e_k|\underset{\sim}{y}, \Theta, \Phi), \quad \forall s.$$

Cependant, cette approche suppose que nous connaissons les paramètres Θ, ce qui n'est pas le cas, et vice versa pour l'estimation des paramètres sachant le champ des étiquettes. Pour ainsi éviter de maximiser tous les paramètres en même temps, nous utilisons l'algorithme EM. L'idée sous-jacente à l'algorithme dans ce contexte vient du fait que si nous connaissons le champ des étiquettes $x_s, \forall s$, alors l'estimation des paramètres peut être faite directement à partir de la densité *a posteriori*. Nous introduisons donc un vecteur de variables latentes $z_s = (z_{s,1}, \ldots, z_{s,k})'$ qui ne sont en fait que des variables indicatrices telles que $z_{s,j} = 1 \Leftrightarrow x_s = e_j$ et 0 sinon. Nous travaillons ensuite avec le vecteur de données complètes $\underset{\sim}{\xi} = (\underset{\sim}{y}, \underset{\sim}{z})$, où $\underset{\sim}{z} = \{z_s\}_{s\in\Omega}$.

La difficulté associée à cette approche est que le vecteur latent $\underset{\sim}{z}$ est inconnu. Cependant, une fois les paramètres du modèle estimés, nous pouvons simuler une réalisation du champ X. Nous pouvons ainsi voir se dessiner une relation conditionnelle itérative entre les deux estimations. Cette relation sera l'essence de l'algorithme EM.

1.3.1.2. *L'algorithme EM*

Sous sa forme générale la plus simple, l'algorithme EM est appliqué comme suit. Soit une fonction $Q(\phi^*|\phi) = \mathbb{E}\Big[\log f(X|\phi^*)|\underset{\sim}{y}, \phi\Big]$, avec l'hypothèse que $Q(\phi^*|\phi)$ existe pour toute paire de paramètres (ϕ^*, ϕ). L'algorithme à l'itération $p+1$ est défini par deux étapes simples :

- *Étape-E :* Calculer $Q(\phi|\phi^{(p)})$.
- *Étape-M :* Estimer $\phi^{(p+1)}$ qui maximise $Q(\phi|\phi^{(p)})$.

Comme il est mentionné dans Dempster, Laird et Rubin (1977), l'idée derrière l'algorithme est que nous cherchons ϕ^* qui maximise $\log f(x|\phi)$. Cependant, puisque $\log f(x|\phi)$ est inconnu, nous maximisons son espérance à l'étape p sachant les données $\underset{\sim}{y}$ et l'estimé actuel $\phi^{(p)}$. Dans le cas de la segmentation d'images, $f(x|\theta)$ est la fonction de vraisemblance modélisant la couleur observée à chacun des pixels.

1.3.2. Algorithme Monte Carlo avec fonction d'importance

L'intégration de Monte Carlo est une méthode numérique très simple pour estimer l'espérance d'une fonction d'un paramètre θ quelconque. Cependant, pour ce faire il faut être capable de simuler à partir de la densité *a posteriori* $\pi(\theta|\underset{\sim}{x})$ et cette tâche n'est pas toujours facile dans un contexte réaliste. Dans cette situation, il est intéressant de se pencher sur l'algorithme de Monte Carlo avec fonction d'importance.

Le principe derrière l'algorithme est bien simple, nous voulons une densité assez près de $\pi(\theta|\underset{\sim}{x})$ pour laquelle il est facile de simuler un échantillon et ensuite pondérer notre estimé pour tenir compte du changement de densité.

Proposition 1.3.1. *Supposons une fonction $h(\theta)$ appelée fonction d'importance, telle que :*

(1) $\{\theta|\pi(\theta|\underset{\sim}{x}) > 0\} \subset \{\theta|h(\theta) > 0\}$,

(2) $\lim_{\theta\to\pm\infty} \frac{\pi(\theta|\underset{\sim}{x})}{h(\theta)} < \infty$.

Alors, nous obtenons que

$$\mathbb{E}[g(\theta)|\underset{\sim}{x}] = \frac{\mathbb{E}^{h(\theta)}[g(\theta)\omega(\theta)]}{\mathbb{E}^{h(\theta)}[\omega(\theta)]},$$

où $\omega(\theta) = \frac{\pi(\theta|\underset{\sim}{x})}{h(\theta)}$ *et* $g(\cdot)$ *une certaine fonction du paramètre d'intérêt.*

DÉMONSTRATION. L'espérance *a posteriori* d'une fonction du paramètre θ est donnée par

$$\begin{aligned}\mathbb{E}[g(\theta)|\underset{\sim}{x}] &= \int_\theta g(\theta)\pi(\theta|\underset{\sim}{x})d\theta \\ &= \frac{\int_\theta g(\theta)\pi(\theta)f(\underset{\sim}{x}|\theta)d\theta}{\int_\theta \pi(\theta)f(\underset{\sim}{x}|\theta)d\theta}\end{aligned}$$

$$= \frac{\int_\theta g(\theta)\frac{\pi(\theta)f(\underline{x}|\theta)}{h(\theta)}h(\theta)d\theta}{\int_\theta \frac{\pi(\theta)f(\underline{x}|\theta)}{h(\theta)}h(\theta)d\theta}$$
$$= \frac{\int_\theta g(\theta)\omega(\theta)h(\theta)d\theta}{\int_\theta \omega(\theta)h(\theta)d\theta}$$
$$= \frac{\mathbb{E}^{h(\theta)}[g(\theta)\omega(\theta)]}{\mathbb{E}^{h(\theta)}[\omega(\theta)]}.$$

□

Nous avons maintenant une fonction $h(\theta)$ pour laquelle il est simple de générer un échantillon et nous pouvons pondérer notre résultat pour tenir compte du changement de densité.

1.3.3. Méthodes MCMC et algorithme de Metropolis-Hasting

Lorsqu'il est impossible de calculer analytiquement la densité *a posteriori* pour un vecteur de paramètres $\underline{\theta}$, nous devons utiliser des méthodes numériques soit pour intégrer la densité ou bien pour en simuler un échantillon. Les méthodes MCMC (Monte Carlo par chaînes de Markov) sont utilisées lorsqu'il est impossible de simuler directement à partir d'une densité $\pi(\underline{\theta}|\underline{x})$. L'idée derrière les simulations MCMC est de simuler un processus de Markov dont sa distribution stationnaire à la convergence est $\pi(\underline{\theta}|\underline{x})$.

Considérons la situation suivante :

- $\underline{x}|\theta_1 \sim f(\underline{x}|\theta_1)$,
- $\theta_1|\theta_2 \sim \pi_1(\theta_1|\theta_2)$,
- $\theta_2 \sim \pi_2(\theta_2)$.

Nous avons alors le théorème suivant pour les densités *a posteriori* marginales.

Théorème 1.3.1. *Les densités a posteriori marginales $\pi_1(\theta_1|\underline{x})$ et $\pi_2(\theta_2|\underline{x})$ sont uniquement déterminées par les lois a posteriori conditionnelles $\pi_1(\theta_1|\theta_2, \underline{x})$ et $\pi_2(\theta_2|\theta_1, \underline{x})$*

DÉMONSTRATION. Prenons la densité *a posteriori* marginale π_1 et montrons qu'elle dépend uniquement des densités *a posteriori* conditionnelles.

$$\pi_1(\theta_1|x) = \int_{\Theta_2} \pi(\theta_1, \theta_2|x)d\theta_2$$

$$
\begin{aligned}
&= \int_{\Theta_2} \pi_1(\theta_1|\theta_2, x)\pi_2(\theta_2|x)d\theta_2 \\
&= \int_{\Theta_2} \pi_1(\theta_1|\theta_2, x) \left[\int_{\Theta_1} \pi_2(\eta, \theta_2|x)d\eta\right] d\theta_2 \\
&= \int_{\Theta_2} \pi_1(\theta_1|\theta_2, x) \left[\int_{\Theta_1} \pi_2(\theta_2|\eta, x)\pi_1(\eta|x)d\eta\right] d\theta_2 \\
&= \int_{\Theta_2} \left[\int_{\Theta_1} \pi_1(\theta_1|\theta_2, x)\pi_2(\theta_2|\eta, x)\pi_1(\eta|x)d\eta\right] d\theta_2 \\
&= \int_{\Theta_1} \left[\int_{\Theta_2} \pi_1(\theta_1|\theta_2, x)\pi_2(\theta_2|\eta, x)d\theta_2\right] \pi_1(\eta|x)d\eta \\
&= \int_{\Theta_1} K(\theta_1, \eta)\pi_1(\eta|x)d\eta. \qquad (1.3.1)
\end{aligned}
$$

Ainsi $\pi_1(\theta_1|x)$ est solution de l'équation intégrale (1.3.1). □

Un algorithme très populaire nommé échantillonneur de Gibbs est complètement basé sur le théorème précédent. Ce dernier consiste tout simplement à générer en alternance une observation à partir des densités conditionnelles *a posteriori*, c'est-à-dire qu'à l'itération i, nous simulons une observation à partir de

$$\pi(\theta_j|\underset{\sim}{\theta}^{(i)}_{(-j)}, x),$$

où

$$\underset{\sim}{\theta}^{(i)}_{(-j)} = \left(\theta_1^{(i)}, \ldots, \theta_{j-1}^{(i)}, \theta_{j+1}^{(i-1)}, \ldots, \theta_J^{(i-1)}\right)$$

est le vecteur de paramètres $\underset{\sim}{\theta}$ sans le paramètre θ_j mis à jour pour les paramètres déjà estimés. Nous continuons à simuler ainsi jusqu'à convergence du processus de Markov vers sa distribution stationnaire.

Cependant, il n'est pas toujours facile, ou voire même possible de simuler à partir des densités conditionnelles *a posteriori*. Dans cette situation, nous utilisons l'algorithme de Metropolis-Hasting. Ce dernier est défini comme suit. Soit une fonction $q_j(\theta_j|\theta^*)$ telle que supp $\pi_j(\theta_j|\theta_{(-j)}, \underset{\sim}{x}) \subset$ supp $q_j(\theta_j|\theta^*), \forall\theta^*$, où supp f est le support de la fonction f. Avec une telle fonction q_i, l'algorithme s'énonce :

- générer $z \sim q_j(\theta_j|\theta^*)$
- calculer $\psi = \min\left(1, \left[\frac{\pi_j(z|\theta^{(i)}_{(-j)}, \underset{\sim}{x})}{\pi_j(\theta_j^{(i-1)}|\theta^{(i)}_{(-j)}, \underset{\sim}{x})}\right] / \left[\frac{q_j(z|\theta^{(i)}_{(-j)}, \underset{\sim}{x})}{q_j(\theta_j^{(i-1)}|\theta^{(i)}_{(-j)}, \underset{\sim}{x})}\right]\right)$

- générer $U \sim \mathrm{U}(0,1)$

$$\theta_j^{(i)} = \begin{cases} z & \text{si } U \leq \psi, \\ \theta_j^{(i-1)} & \text{si } U > \psi. \end{cases}$$

Les fonctions $\pi_i(\cdot)$ sont souvent appelées fonctions cibles et $q_i(\cdot)$ fonctions de proposition.

1.4. Conclusion du chapitre

Nous avons vu dans ce chapitre les notions de base en traitement d'images qui nous servent à caractériser mathématiquement une image. Nous avons également présenté sommairement l'approche de segmentation de Destrempes *et al.* (2005) qui est basée sur un modèle bayésien non décisionnel. Cette approche consiste essentiellement à calculer un maximum *a posteriori* (MAP) pour la segmentation. Nous présentons au chapitre 2 une approche bayésienne décisionnelle avec laquelle nous évaluons l'impact du choix de la fonction de perte pour l'estimation des paramètres du modèle de vraisemblance. L'estimation de la segmentation n'est plus basée sur le calcul du MAP mais plutôt simulée à l'aide de l'algorithme EM et l'estimation des paramètres du modèle de vraisemblance est conduite à l'aide de méthodes MCMC.

Chapitre 2

THÉORIE DE LA DÉCISION ET SEGMENTATION D'UNE IMAGE PAR SIMULATIONS

Dans ce chapitre, nous présentons la méthodologie proposée pour segmenter une image selon un modèle bayésien avec champ de Markov caché. L'approche est fortement basée sur celle de Destrempes, Mignotte et Angers (2005) présentée au chapitre 1, le modèle statistique étant sensiblement le même. Les nouveautés sont principalement au niveau de la représentation de l'image, du cadre décisionnel bayésien et de l'estimation du modèle.

Dans un premier temps, nous expliquons les éléments de base de la théorie de la décision bayésienne. Entre autres les concepts de fonction de perte et d'estimateur bayésien sont présentés. Ensuite nous présentons les fonctions de perte que nous utilisons dans le modèle proposé et les estimateurs bayésiens associés.

Deuxièmement, nous expliquons comment nous représentons l'image dans notre modèle, c'est-à-dire l'espace de couleurs utilisé et comment nous traitons les composantes de cet espace. Nous verrons dans cette section des différences majeures avec l'approche de Destrempes *et al.* (2005).

Ensuite, nous caractérisons l'approche que nous proposons pour estimer le modèle statistique proposé. Contrairement à Destrempes *et al.* (2005), nous utilisons l'algorithme EM pour simuler itérativement la segmentation de l'image basée sur le champ de couleurs. Nous montrons comment nous appliquons cet algorithme et

présentons certains problèmes qui y sont reliés. Nous présentons également l'estimation bayésienne des paramètres du modèle de vraisemblance par des méthodes numériques, soient les méthodes de Monte Carlo avec fonction d'importance et le MCMC.

Nous discutons ensuite de l'approche utilisée pour simuler une image segmentée à partir du modèle estimé. En dernier lieu, nous discutons du choix d'un critère d'arrêt pour les simulations MCMC.

2.1. Cadre décisionnel bayésien

Dans cette section, nous présentons les bases de la théorie de la décision bayésienne. Nous présentons également les fonctions de perte usuelles en statistique bayésienne et nous décrivons les fonctions de perte que nous proposons pour notre modèle.

2.1.1. Concepts et définitions de bases

Nous tenons à mentionner que cette section est fortement inspirée de Robert (2001). Un des buts premier de l'inférence statistique est de fournir des outils de décision. Ces outils sont souvent basés sur l'estimation des paramètres d'un certain modèle statistique. Cependant, il est intéressant de pouvoir analyser l'impact d'une décision sur un paramètre par rapport à sa vraie valeur qui elle est inconnue. La théorie de la décision nous donne exactement un cadre pour effectuer ce type d'analyse de validation d'un estimateur.

Soit Θ l'espace des paramètres, $\mathcal{D}$ l'espace de toutes les décisions possibles et $\mathcal{S}$ l'espace échantillonnal.

Définition 2.1.1. *Une règle de décision $\delta(x)$ est une fonction de $\mathcal{S} \mapsto \mathcal{D}$ telle que*

$$\delta : x \mapsto d = \delta(x).$$

L'exemple suivant présente quelques cas d'espaces de décisions.

Exemple 2.1.1. *Voici des exemples d'espaces de décisions dans des situations habituelles en statistiques.*

- *estimation ponctuelle $\mathcal{D} = \Theta$,*

- *intervalle de confiance* $\mathcal{D} = \{$*tous les sous-ensembles de* $\Theta\}$,
- *test d'hypothèse* $\mathcal{D} = \{0, 1\}$ *où 0 :=acceptation de* H_0 *et 1 :=rejet de* H_0.

Nous cherchons maintenant à quantifier l'erreur que nous commettons en prenant une décision sur un paramètre. Cette quantification est faite à partir d'une fonction de perte.

Définition 2.1.2. *Une fonction de perte est une fonction* $\mathbb{L} : \Theta \times \mathcal{D} \mapsto \mathbb{R}^+$, *c'est-à-dire,*

$$(\theta, a) \mapsto \mathbb{L}(\theta, a).$$

Même si le concept de la théorie de la décision est très souhaitable, il n'en reste pas moins que c'est un sujet controversé. En effet, la quantification de la perte par rapport à une décision quelconque est très reliée au choix de la fonction de perte et ce choix n'est pas unique ni universel.

Nous définissons maintenant certaines mesures de risque reliées à une décision δ sur θ par rapport à une fonction de perte $\mathbb{L}$.

Définition 2.1.3. *Le risque fréquentiste d'une règle de décision* δ *est donné par*

$$\begin{aligned} R(\theta, \delta) &= \mathbb{E}\big[\mathbb{L}\big(\theta, \delta(x)\big)|\theta\big], \\ &= \int_S \mathbb{L}(\theta, \delta(x)) f(x|\theta) dx. \end{aligned}$$

Le risque fréquentiste est en fait un moyennage de la fonction de perte sur toutes les observations possibles. Le choix d'une règle de décision est fait afin de minimiser le risque fréquentiste, de sorte que nous préférons δ_1 à δ_2 si $R(\theta, \delta_1) < R(\theta, \delta_2)\, \forall \theta$. Cependant, ceci est rarement faisable en pratique ; ainsi nous moyennons sur θ pour obtenir le risque intégré.

Définition 2.1.4. *Le risque intégré est défini par*

$$\begin{aligned} r(\pi, \delta) &= \mathbb{E}^{\pi(\theta)}\big[R(\theta, \delta)\big] \\ &= \int_\Theta R(\theta, \delta)\pi(\theta) d\theta. \end{aligned}$$

De même que pour le risque fréquentiste, nous choisissons la décision δ qui donne le plus petit risque intégré. Cette décision qui minimise le risque intégré est dite règle de Bayes.

Définition 2.1.5. *Une règle de Bayes est une règle de décision qui minimise le risque intégré. Ainsi, δ^π est une règle de Bayes si*

$$r(\pi, \delta^\pi) = \inf_\delta r(\pi, \delta)$$

$$\delta^\pi = \arg\min_\delta r(\pi, \delta).$$

Nous appellons $r(\pi) = r(\pi, \delta^\pi)$ le risque de Bayes. Dans le cadre d'estimation ponctuelle, la règle de Bayes est également appelée estimateur bayésien.

Nous montrons maintenant comment il est possible de calculer la règle de Bayes sans passer par le risque fréquentiste.

Définition 2.1.6. *La perte espérée* a posteriori *de l'action $\delta(x) = a$ avec $X = x$ est donnée par*

$$\begin{aligned}\rho(\pi, a) &= \mathbb{E}_\theta\big[\mathbb{L}(\theta, a)|x\big] \\ &= \int_\Theta \mathbb{L}(\theta, a)\pi(\theta|x)d\theta.\end{aligned}$$

La perte espérée *a posteriori* est un moyennage de l'erreur, ou la perte, par rapport à la distribution *a posteriori* du paramètre θ. Ainsi, contrairement à l'approche fréquentiste, nous intégrons sur l'espace des paramètres Θ parce que θ est inconnu et x est observé. Le théorème suivant montre la relation qui existe entre le risque intégré et la perte espérée *a posteriori.*

Théorème 2.1.1. *Le risque intégré est donné par l'espérance de la perte espérée* a posteriori *par rapport à la marginale de X.*

$$r(\pi, \delta) = \mathbb{E}^{m(x)}\big[\rho\big(\pi, \delta(X)\big)\big].$$

DÉMONSTRATION. En utilisant le théorème de Fubini nous permettant d'interchanger l'ordre des intégrales, nous avons

$$\begin{aligned}r(\pi, \delta) &= \mathbb{E}^{\pi(\theta)}\big[R\big(\theta, \delta(x)\big)\big] \\ &= \int_\Theta R\big(\theta, \delta(x)\big)\pi(\theta)d\theta \\ &= \int_\Theta \left[\int_\mathcal{S} \mathbb{L}\big(\theta, \delta(x)\big) f(x|\theta)dx\right] \pi(\theta)d\theta \\ &= \int_\mathcal{S}\int_\Theta \mathbb{L}\big(\theta, \delta(x)\big) f(x|\theta)\pi(\theta)d\theta dx\end{aligned}$$

$$
\begin{aligned}
&= \int_{\mathcal{S}} \left[\int_{\Theta} \mathbb{L}\big(\theta, \delta(x)\big) \pi(\theta|x) d\theta \right] m(x) dx \\
&= \int_{\mathcal{S}} \rho\big(\pi, \delta(x)\big) m(x) dx \\
&= \mathbb{E}^{m(x)} \big[\rho\big(\pi, \delta(X)\big) \big].
\end{aligned}
$$

□

Le théorème précédent nous permet de calculer la règle de Bayes en minimisant directement la perte espérée *a posteriori*, c'est-à-dire

$$\delta^{\pi} = \arg\min_{\delta} \rho(\pi, \delta).$$

2.1.2. Fonctions de perte utilisées dans ce mémoire

Nous présentons dans cette section certaines fonctions de perte fréquemment utilisées en inférence bayésienne et les règles de Bayes qui leur sont associées, ainsi que des nouvelles fonctions de perte que nous proposons dans ce mémoire.

2.1.2.1. *La fonction de perte quadratique*

Cette fonction de perte est sans équivoque la plus usuelle en statistique bayésienne dû à la simplicité de sa forme et de l'estimateur bayésien résultant. Dans le cas unidimensionnel, la fonction de perte quadratique est donnée par

$$\mathbb{L}(\theta, a) = (\theta - a)^2. \tag{2.1.1}$$

La fonction de perte quadratique est souvent critiquée pour la forte pénalité qu'elle applique aux grandes déviations par rapport à la vraie valeur du paramètre. Néanmoins, comme nous l'avons déjà mentionné, cette fonction de perte donne un estimateur bayésien très simple et intuitif.

Proposition 2.1.1. *La règle de Bayes δ^{π} associée à la densité a priori $\pi(\theta)$ et la fonction de perte quadratique est donnée par l'espérance a posteriori*

$$\delta^{\pi}(x) = \mathbb{E}[\theta|x] = \frac{\int_{\Theta} \theta f(x|\theta)\pi(x) d\theta}{\int_{\Theta} f(x|\theta)\pi(x) d\theta} = \int_{\Theta} \theta \pi(\theta|x) d\theta.$$

La preuve de cette proposition est très simple.

DÉMONSTRATION. Nous cherchons à minimiser la perte espérée *a posteriori*

$$\rho(\theta, a) = \mathbb{E}^{\pi}\Big[(\theta - a)^2 | x\Big] = \mathbb{E}^{\pi}[\theta^2|x] - 2a\mathbb{E}^{\pi}[\theta|x] + a^2.$$

Il est facile de voir que le minimum est atteint pour $a = \delta^{\pi}(x) = \mathbb{E}^{\pi}[\theta|x]$. □

Remarque 2.1.1. *En traitement d'images, l'estimateur associé à la fonction de perte quadratique est souvent appelé l'estimateur du champ moyen, ou Mean Field (MF) en anglais.*

2.1.2.2. *La fonction de perte dichotomique 0-1*

Nous présentons la fonction de perte dichotomique 0-1 parmi les autres fonctions de perte décisionnelles, cependant puisqu'elle consiste en un cas dégénéré en un point, elle est souvent associée au cas non décisionnel. La fonction de perte dichotomique 0-1 est définie par l'équation suivante

$$\mathbb{L}(\theta, a) = \begin{cases} 0, & \text{si } \theta = a, \\ 1, & \text{sinon.} \end{cases} \tag{2.1.2}$$

Proposition 2.1.2. *La règle de Bayes associée à la fonction de perte (2.1.2) est donnée par le maximum a posteriori (MAP)*

$$\delta^{\pi}(x) = \arg\max_{\theta} \pi(\theta|x).$$

DÉMONSTRATION. Pour démontrer la proposition précédente, nous utilisons une version légèrement différente de la fonction de perte donnée par l'équation (2.1.2). Soit la fonction de perte dichotomique sur un intervalle ϵ autour de θ donnée par

$$\mathbb{L}(\theta, a) = 1 - \mathbb{I}_{(|\theta - a| < \epsilon)}(\theta), \tag{2.1.3}$$

et nous faisons tendre ϵ vers 0. Calculons maintenant la perte espérée *a posteriori*

$$\begin{aligned} \rho_{\epsilon}(\pi, a|x) &= \mathbb{E}^{\pi}\Big[1 - \mathbb{I}_{(|\theta - a| < \epsilon)}(\theta)|x\Big], \\ &= 1 - \Pr(|\theta - a| < \epsilon|x). \end{aligned}$$

Il est facile de voir que la perte espérée *a posteriori* est minimale lorsque $\Pr(|\theta - a| < \epsilon|x)$ est maximale. Par conséquent, lorsque $\epsilon \longrightarrow 0$ nous voyons que δ^{π} est donnée par $\arg\max_{\theta} \pi(\theta|x)$. □

Comme nous l'avons déjà mentionné, l'estimateur défini par le MAP est souvent associé à de l'inférence dite non décisionnelle puisque la fonction de perte dichotomique est un cas limite d'une fonction de perte. Cet estimateur est également fréquemment utilisé parce qu'il s'apparente à l'estimateur du maximum de vraisemblance.

Les deux fonctions de perte que nous venons de voir résultent en des estimateurs bayésiens bien connus et faciles à implanter. Nous allons maintenant voir deux autres fonctions de perte qui nous permettront de quantifier différement le risque. En effet, les deux prochaines fonctions de perte vont nous permettre d'inclure une tolérance sur l'erreur commise en estimant θ par a. La première fonction est la généralisation de la fonction de perte 0-1 sur un intervalle ϵ autour de θ que nous avons vu à l'équation (2.1.3). La deuxième fonction de perte proposée est une modification de la fonction de perte de Huber.

2.1.2.3. *Fonction de perte dichotomique sur un intervalle*

Lorsque nous regardons une image, nous ne sommes pas en mesure de distinguer chaque petite variation de couleur. Nous sommes donc intéressés à prendre une décision sur un paramètre de sorte que nous jugeons faire une erreur seulement lorsque cette erreur est perceptible. La fonction de perte dichotomique sur un intervalle nous donne un cadre pour effectuer ce type de validation. Cette dernière est donnée par

$$\mathbb{L}(\theta, a) = \begin{cases} 0, & \text{si } |\theta - a| < \epsilon, \\ 1, & \text{sinon.} \end{cases} \tag{2.1.4}$$

Nous pouvons interpréter ϵ comme une tolérance sur l'erreur faite sur θ. Nous définissons maintenant l'estimateur bayésien (ou la règle de Bayes) associé à la fonction de perte dichotomique sur un intervalle.

Proposition 2.1.3. *La règle de Bayes associée à la fonction de perte donnée par l'équation (2.1.4) est donnée par*

$$\delta^{\pi}(x) = \arg\max_{a} \int_{a-\epsilon}^{a+\epsilon} \pi(\theta|x) d\theta. \tag{2.1.5}$$

Nous montrons que la règle de Bayes donnée par l'équation (2.1.5) se résume à une valeur de a telle que

$$\pi(a+\epsilon|x) = \pi(a-\epsilon|x). \tag{2.1.6}$$

DÉMONSTRATION. Commençons tout d'abord par évaluer la perte espérée *a posteriori* d'une décision a sur le paramètre θ, c'est-à-dire

$$\begin{aligned}
\rho(\theta, a) &= \mathbb{E}^{\pi(\theta|x)}\Big[\mathbb{L}(\theta, a)\Big], \\
&= \int_{-\infty}^{a+\epsilon} \pi(\theta|x)d\theta + \int_{a+\epsilon}^{\infty} \pi(\theta|x)d\theta, \\
&= \int_{-\infty}^{\infty} \pi(\theta|x)d\theta - \int_{a-\epsilon}^{a+\epsilon} \pi(\theta|x)d\theta, \\
&= 1 - \int_{a-\epsilon}^{a+\epsilon} \pi(\theta|x)d\theta.
\end{aligned}$$

Nous voulons maintenant minimiser la perte espérée *a posteriori*, prenons la dérivée par rapport à a.

$$\begin{aligned}
\frac{d}{da}\rho(\theta, a) &= -\frac{d}{da}\int_{a-\epsilon}^{a+\epsilon} \pi(\theta|x)d\theta, \\
&= \pi(a-\epsilon|x) - \pi(a+\epsilon|x) = 0, \\
&\Leftrightarrow \pi(a+\epsilon|x) = \pi(a-\epsilon|x).
\end{aligned}$$

□

Remarque 2.1.2. *Dans le cas d'une distribution unimodale comme la densité bêta dans notre situation, la valeur a satisfaisant l'équation (2.1.6) est unique.*

2.1.2.4. *Fonction de perte de Huber*

La fonction de perte de Huber est un mélange de la fonction de perte quadratique et de la fonction de perte de Laplace. La fonction de perte de Laplace est tout simplement une fonction linéaire par morceaux en a, à savoir $\mathbb{L}(\theta, a) = |\theta - a|$. La fonction de perte de Huber est définie par l'équation suivante :

$$\mathbb{L}(\theta, a) = \begin{cases} \frac{1}{4\epsilon}(\theta - a)^2, & \text{si } |\theta - a| \leq 2\epsilon\,; \\ |\theta - a| - \epsilon, & \text{sinon.} \end{cases} \tag{2.1.7}$$

La figure 2.1.1 compare la fonction de perte de Huber à la portion quadratique de l'équation (2.1.7). Nous remarquons que nous pondérons plus faiblement le risque pour une grande erreur que pour la fonction de perte quadratique.

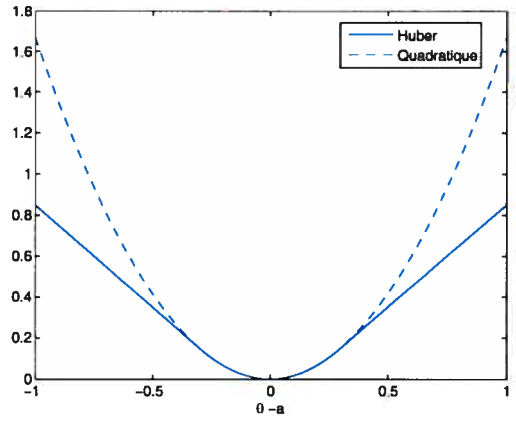

FIGURE 2.1.1. Représentation de la fonction de perte de Huber pour $\epsilon = 0{,}15$ par rapport à la partie quadratique de la fonction.

Cependant, il est potentiellement d'intérêt de ne pas vouloir pénaliser outre mesure lorsque nous commettons une grande erreur. Par exemple dans le contexte de l'imagerie, si nous cherchons à estimer la couleur d'un pixel et disons que sa vraie couleur est bleu, il serait alors intéressant de quantifier l'erreur lorsque nous estimons une couleur près du vrai bleu et considérer égale la perte d'estimer le bleu par du jaune, du violet ou bien du vert par exemple. Nous présentons maintenant une modification de la fonction de perte de Huber qui prend en compte cet aspect.

Soit la fonction de perte donnée par l'équation (2.1.8). Nous remarquons encore la présence de la composante quadratique dans un voisinage ϵ de la vraie valeur du paramètre ; mais maintenant tous ce qui sort de ce voisinage se voit attribuer une perte égale ϵ^2. Ainsi, nous avons

$$\mathbb{L}(\theta, a) = \min(\epsilon^2, (\theta - a)^2), \tag{2.1.8}$$

où ϵ est considéré comme une tolérance sur le paramètre θ. La figure 2.1.2 montre le graphique de la fonction de perte de Huber modifiée.

Nous cherchons maintenant la règle de Bayes associée à la fonction de perte de Huber modifiée.

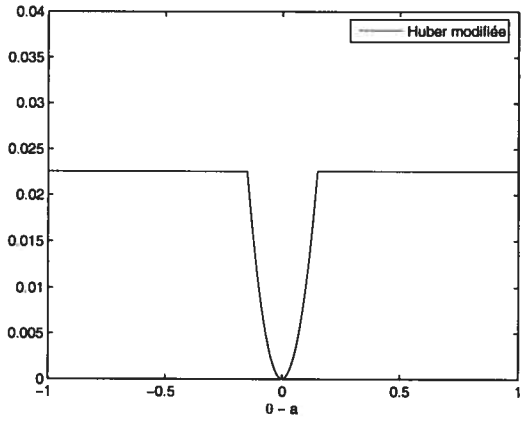

FIGURE 2.1.2. Représentation graphique de la fonction de perte de Huber modifiée pour $\epsilon = 0{,}15$.

Proposition 2.1.4. *La règle de Bayes associée à la fonction de perte de Huber modifiée est donnée par une valeur de a telle que*

$$a = \frac{\mathbb{E}\Big[\theta \mathbb{I}_{(a-\epsilon,a+\epsilon)}(\theta)|x\Big]}{\Pi(a+\epsilon|x) - \Pi(a-\epsilon|x)},$$

où $\Pi(\cdot)$ *est la fonction de répartition.*

Remarque 2.1.3. *La règle de Bayes présentée à la proposition précédente revient à calculer l'espérance de la distribution* a posteriori *tronquée sur* $(a-\epsilon; a+\epsilon)$.

DÉMONSTRATION. Nous commençons toujours par évaluer la perte espérée *a posteriori.*

$$\begin{aligned}
\rho(\theta, a) &= \mathbb{E}^{\pi(\theta|x)}\Big[\mathbb{L}(\theta, a)\Big], \\
&= \int_{-\infty}^{\infty} \min\big(\epsilon^2, (\theta-a)^2\big)\pi(\theta|x)d\theta, \\
&= \epsilon^2\Big[\Pi(a-\epsilon|x) + 1 - \Pi(a+\epsilon|x)\Big] + \int_{a-\epsilon}^{a+\epsilon} (\theta-a)^2\pi(\theta|x)d\theta.
\end{aligned}$$

Nous cherchons à minimiser le risque intégré *a posteriori* ; prenons la dérivée par rapport à a, c'est-à-dire

$$\begin{aligned}
\frac{d}{da}\rho(\theta, a) = \epsilon^2\Big[\pi(a-\epsilon|x) - \pi(a+\epsilon|x)\Big] - 2\int_{a-\epsilon}^{a+\epsilon} (\theta-a)\pi(\theta|x)d\theta \\
+ \epsilon^2\pi(a+\epsilon|x) - \epsilon^2\pi(a-\epsilon|x)
\end{aligned}$$

$$
\begin{aligned}
&= -2\Big[\int_{a-\epsilon}^{a+\epsilon} \theta\pi(\theta|x) - a\int_{a-\epsilon}^{a+\epsilon} \pi(\theta|x)\Big] \\
&= -2\Big[\mathbb{E}[\theta\mathbb{I}_{(a-\epsilon,a+\epsilon)}(\theta)|x] - a\Big(\Pi(a+\epsilon|x) - \Pi(a-\epsilon|x)\Big)\Big] = 0 \\
&\Leftrightarrow a\Big(\Pi(a+\epsilon|x) - \Pi(a-\epsilon|x)\Big) = \mathbb{E}[\theta\mathbb{I}_{(a-\epsilon,a+\epsilon)}(\theta)|x].
\end{aligned}
$$

□

Nous avons mentionné que la fonction de perte de Huber est une généralisation des fonctions de perte quadratique et de Laplace. À son tour, la fonction de perte de Huber modifiée est une généralisation des fonctions de perte quadratique et dichotomique 0-1. En effet, il est facile de voir que lorsque $\epsilon \to \infty$, l'équation (2.1.8) tend vers la fonction de perte quadratique donnée par (2.1.1). De même lorsque $\epsilon \to 0$, la composante quadratique perd de l'importance et la fonction de perte de Huber modifiée tend vers une fonction de Dirac centrée en a.

Il faut toujours garder en tête que malgré qu'il soit intéressant d'avoir un cadre décisionnel qui nous permet d'inclure une tolérance sur les paramètres estimés, cette tolérance n'est pas un élément facile à déterminer. En effet, le choix du ϵ en question n'est pas unique et peut dépendre du système de référence (voir chapitre 3).

2.2. Cadre de représentation de l'image dans notre modèle décisionnel bayésien

Dans cette section nous expliquons comment nous représentons et traitons l'image dans le modèle de segmentation proposé. Nous mettons en lumière les différences par rapport à l'approche de Destrempes *et al.* (2005).

Commençons tout d'abord par le système de représentation des couleurs. Comme nous l'avons vu à la section précédente, nous voulons tester l'influence des fonctions de perte qui incluent une notion de tolérance sur le paramètre estimé. Dans notre contexte d'imagerie, une tolérance sur une couleur se résume à une distance acceptable dans l'espace de couleur utilisé. Ainsi, nous voulons choisir un système de représentation qui est perceptuellement uniforme. Comme nous

l'avons vu à la section 1.1.4, le système de représentation Lab possède cette propriété. De plus, nous choisissons ce système de représentation pour des raisons de simplicité de saisie avec le logiciel Matlab. Il est important de noter que d'autres systèmes de représentation perceptuellement uniforme existent et peuvent également constituer de bons candidats (voir Marion, 1997).

Une fois le système de représentation choisi, nous décidons comment nous traitons les trois composantes de ce système, en l'occurence les composantes L, a et b. Comme nous avons vu à la section 1.2.1, dans Destrempes *et al.* (2005) une décorrélation des trois composantes de l'espace est effectuée segment par segment et ce sont les triplets décorrélés qui sont ensuite modélisés. Pour notre part, nous utilisons une autre approche qui consiste à modéliser et segmenter indépendamment les trois composantes. Un des avantages de procéder ainsi est que nous évitons de décorréler les données. Par contre, nous nous retrouvons avec trois segmentations différentes, soit une pour chacune des composantes. Cette dernière constatation n'est pas un problème en soit, c'est plutôt le nombre de segments résultants qui nous cause un problème. En effet, supposons que nous souhaitons avoir K segments dans l'image finale, si nous segmentons chaque composante avec K classes chacune, lorsque nous recréons l'image en superposant les trois composantes segmentées, nous obtenons un nombre de segments beaucoup plus grand (une possibilité de K^3 segments finaux). Pour résoudre ce problème, il faudra utiliser une procédure pour regrouper ces segments provenant du produit cartésien des trois composantes. Nous verrons au chapitre 3 la méthode que nous utilisons pour regrouper les régions.

Nous avons mentionné plus haut qu'il existe d'autres systèmes de représentation que le système Lab qui pourraient être utilisés. Il faut cependant remarquer que les résultats de la segmentation seraient différents avec un autre système car nous segmentons l'image composante par composante. Ainsi, la segmentation du plan a ne serait pas la même que celle du plan u dans le système Luv par exemple.

Pour conclure sur le traitement des données, nous appliquons une transformation linéaire sur les données de chacune des composantes de l'image pour contrôler la variance. Cette transformation n'est en fait qu'une normalisation de l'étendue

des données sur l'intervalle $[0, 1]$. Pour ce faire, nous calculons l'étendue des valeurs des pixels pour chacune des composantes. Cependant, au lieu de normaliser ces valeurs observées, nous augmentons par un facteur de 5% chacune des bornes de ces étendues et décrétons que ces nouvelles bornes augmentées sont en fait celles que nous observons, c'est-à-dire

$$y'_s = \frac{y_s - (\min(y) - 0{,}05)}{\max(y) - \min(y) + 0{,}10}.$$

De cette manière, nous évitons d'obtenir des valeurs exactement aux extrémités de l'intervalle de normalisation qui pourraient causer certaines instabilités numériques. Pour simplifier la notation dans le reste du texte, nous utiliserons quand même la notation y_s pour y'_s.

2.3. Modèle décisionnel bayésien

Dans cette section, nous décrivons l'approche que nous utilisons pour segmenter une image selon un modèle de couleurs. Le modèle statistique est celui développé dans Destrempes *et al.* (2005). Nous présentons les changements que nous apportons au modèle dans un contexte décisionnel bayésien hiérarchique et surtout la méthode d'estimation qui est utilisée. Contrairement à Destrempes *et al.* (2005) qui utilisent le ESE pour maximiser la segmentation de l'image, nous utilisons l'algorithme EM.

2.3.1. Modèle de vraisemblance

Le modèle de vraisemblance utilisé sur le champ de couleurs observé Y est le même que celui défini à la section 1.2.2, soit un modèle bêta. Cependant, puisque nous modélisons indépendamment chaque composante du système Lab, le modèle de vraisemblance se résume à

$$B(y|\underset{\sim}{\alpha}, \underset{\sim}{\beta}, x) = \prod_s B(y_s|\alpha_k, \beta_k, x_s = k),$$

où la fonction de densité d'une loi bêta est donnée par l'équation (1.2.1). Remarquons que la dépendance de la distribution du pixel s par rapport à la classe à laquelle il appartient implique que les pixels ne sont pas identiquement distribués.

Nous désirons maintenant estimer les paramètres (α_k, β_k), $1 \leq k \leq K$. Nous souhaitons utiliser une densité *a priori* non informative. Nous avons tout d'abord essayé une densité *a priori* uniforme sur le plan (α_k, β_k). Les résultats étaient concluants mais lorsque les données étaient très concentrées, les paramètres estimés étaient très grands et nous obtenions ainsi une variance extrêmement petite. Nous proposons donc d'utiliser la densité *a priori* de Jeffreys. Dans le cas d'un modèle bêta, cette dernière n'est cependant pas facile à calculer ; nous devons donc l'approximer.

Définition 2.3.1. *La densité* a priori *de Jeffreys notée* $\pi_J(\cdot)$ *est définie comme*

$$\pi_J(\underline{\theta}) \propto \sqrt{|I(\underline{\theta})|},$$

où $|I(\underline{\theta})|$ *est le déterminant de la matrice d'information de Fisher de* $\underline{\theta}$.

Dans Sun et Berger (1998), nous trouvons que la matrice d'information de Fisher pour le vecteur de paramètres (α, β) est donnée par

$$I(\alpha, \beta) = \begin{pmatrix} \psi'(\alpha) - \psi'(\alpha+\beta) & -\psi'(\alpha+\beta) \\ -\psi'(\alpha+\beta) & \psi'(\beta) - \psi'(\alpha+\beta) \end{pmatrix},$$

où ψ' est la fonction polygamma d'ordre 1 donnée par

$$\begin{aligned} \psi'(x) &= \frac{\partial^2}{\partial x^2} \log(\Gamma(x)) \\ &= \frac{\Gamma''(x)\Gamma(x) - \Gamma'(x)^2}{\Gamma(x)^2}. \end{aligned}$$

Nous savons alors que le déterminant de la matrice d'information de Fisher est donné par

$$|I(\alpha, \beta)| = \big(\psi'(\alpha) - \psi'(\alpha+\beta)\big)\big(\psi'(\beta) - \psi'(\alpha+\beta)\big) - \big(\psi'(\alpha+\beta)\big)^2. \quad (2.3.1)$$

Nous cherchons une approximation de l'équation (2.3.1) afin d'étudier son comportement aux bornes. Nous travaillons avec l'identité suivante

$$\psi'(x) = \sum_{l=0}^{\infty} \frac{1}{(x+l)^2}.$$

À l'aide de Mathematica, nous pouvons vérifier les deux limites suivantes

$$\lim_{x \to 0} x^2 \psi'(x) = 1 \quad \Rightarrow \lim_{x \to 0} \psi'(x) = \mathrm{O}(x^{-2}),$$

$$\lim_{x\to\infty} x\psi'(x) = 1 \quad \Rightarrow \lim_{x\to\infty} \psi'(x) = \mathrm{O}(x^{-1}).$$

Ce que nous pouvons conclure sur le comportement de la fonction polygamma à partir des deux limites suivantes est qu'elle se comporte comme la fonction $\frac{1}{x^2}$ lorsque nous approchons 0 et $\frac{1}{x}$ lorsque nous allons vers l'infini. Ainsi, nous proposons l'approximation suivante pour la fonction polygamma d'ordre 1

$$\psi'(x) \propto \frac{1}{x^2} + \frac{1}{x}. \tag{2.3.2}$$

Comme nous pouvons le voir dans le graphique de la figure 2.3.1, l'approximation proposée à l'équation (2.3.2) est plus que satisfaisante. En effet, les deux courbes sont complètement superposées par rapport à l'échelle affichée.

À l'aide de cette approximation, nous calculons maintenant la fonction *a priori* de Jeffreys. Nous rappelons que pour ce faire nous devons calculer le déterminant de la matrice d'information de Fisher.

Proposition 2.3.1. *Le déterminant de la matrice d'information de Fisher est donné approximativement par*

$$\begin{aligned}
|I(\alpha,\beta)| &= \big(\psi'(\alpha) - \psi'(\alpha+\beta)\big)\big(\psi'(\beta) - \psi'(\alpha+\beta)\big) - \big(\psi'(\alpha+\beta)\big)^2 \\
&\simeq \left(\frac{1}{\alpha} + \frac{1}{\alpha^2} - \frac{1}{\alpha+\beta} - \frac{1}{(\alpha+\beta)^2}\right)\left(\frac{1}{\beta} + \frac{1}{\beta^2} - \frac{1}{\alpha+\beta} - \frac{1}{(\alpha+\beta)^2}\right) \\
&\quad - \left(\frac{1}{\alpha+\beta} + \frac{1}{(\alpha+\beta)^2}\right)^2 \\
&= \left(\frac{\beta}{\alpha(\alpha+\beta)} + \frac{\beta(\beta+2\alpha)}{\alpha^2(\alpha+\beta)^2}\right)\left(\frac{\alpha}{\beta(\alpha+\beta)} + \frac{\alpha(\alpha+2\beta)}{\beta^2(\alpha+\beta)^2}\right) \\
&\quad - \left(\frac{1}{(\alpha+\beta)} + \frac{1}{(\alpha+\beta)^2}\right)^2 \\
&= \frac{\alpha+2\beta}{\beta(\alpha+\beta)^3} + \frac{\beta+2\alpha}{\alpha(\alpha+\beta)^3} - \frac{2}{(\alpha+\beta)^3} + \frac{(\alpha+2\beta)(\beta+2\alpha)}{\alpha\beta(\alpha+\beta)^4} - \frac{1}{(\alpha+\beta)^4} \\
&= \frac{1}{\alpha\beta(\alpha+\beta)^3}\Big(\alpha^2+\beta^2+2\alpha\beta\Big) + \frac{2}{\alpha\beta(\alpha+\beta)^4}\Big(\alpha^2+\beta^2+2\alpha\beta\Big) \\
&= \frac{1}{\alpha\beta(\alpha+\beta)}\Big(1 + \frac{2}{(\alpha+\beta)}\Big) \\
&\simeq \frac{1}{\alpha\beta(\alpha+\beta)}.
\end{aligned}$$

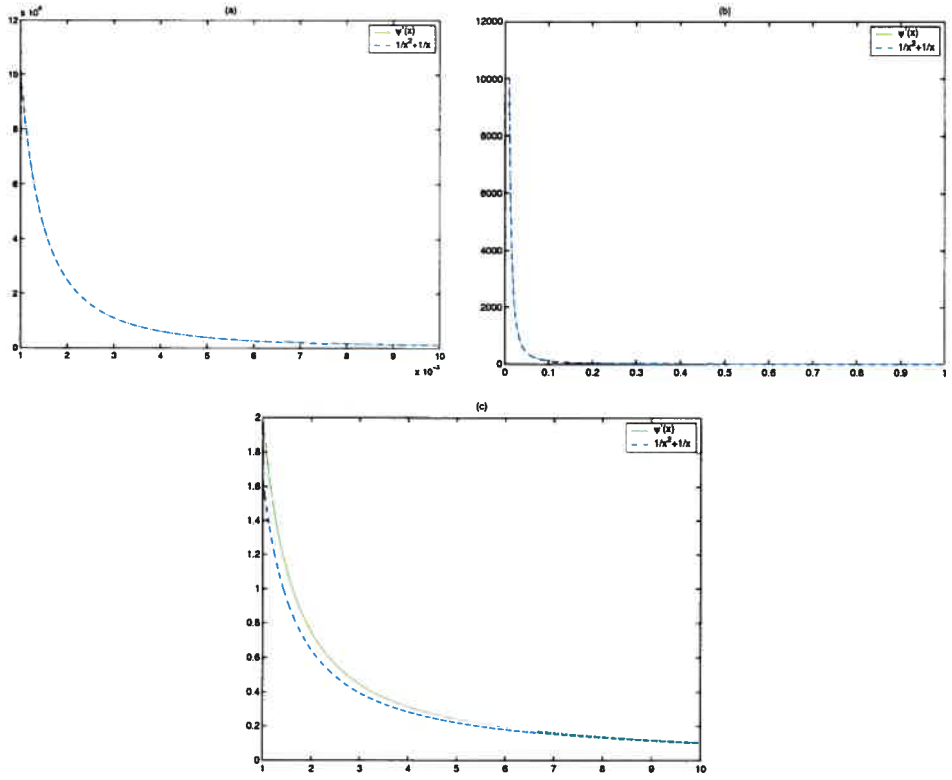

FIGURE 2.3.1. Comparaison de la fonction polygamma d'ordre 1 et de l'approximation proposée pour différents intervalles de valeurs de x

À l'aide de l'approximation précédente, nous proposons comme approximation de la densité *a priori* de Jeffreys la densité suivante

$$\pi_J^*(\alpha, \beta) \propto \frac{1}{\sqrt{\alpha\beta(\alpha+\beta)}}. \tag{2.3.3}$$

La densité définie par l'équation (2.3.3) est pratique car elle nous permet de forcer *a priori* les paramètres à ne pas être trop grand, nous permettant ainsi de conserver une variance légèrement plus grande lorsque nous avons une région où la couleur des pixels est relativement uniforme. La densité *a posteriori* conjointe pour α_k et β_k sachant la segmentation x pour une classe k est donnée par

$$\pi(\alpha_k, \beta_k | y, x) \propto \frac{\Gamma(\alpha_k+\beta_k)^n}{\Gamma(\alpha_k)^n\Gamma(\beta_k)^n\sqrt{\alpha_k\beta_k(\alpha_k+\beta_k)}} \prod_{s|x_s=k} y_s^{\alpha_k-1}(1-y_s)^{\beta_k-1}. \tag{2.3.4}$$

Nous remarquons que nous ne pouvons pas intégrer analytiquement cette densité. Nous utilisons alors l'algorithme de Metropolis-Hasting pour simuler un échantillon des densités marginales.

Premièrement, nous approximons les fonctions gamma par la formule de Stirling. L'équation (2.3.4) devient donc

$$\pi(\alpha_k, \beta_k | y, x) \propto \frac{(\alpha_k + \beta_k)^{\alpha_k + \beta_k - 1/2}}{\alpha_k^{\alpha_k - 1/2} \beta_k^{\beta_k - 1/2}} \frac{1}{\sqrt{\alpha_k \beta_k (\alpha_k + \beta_k)}} \times e^{-(\alpha_k - 1) \sum_{s|x_s = k} \log(y_s)^{-1}} e^{-(\beta_k - 1) \sum_{s|x_s = k} \log(1 - y_s)^{-1}} \tag{2.3.5}$$

$$= \frac{(\alpha_k + \beta_k)^{\alpha_k + \beta_k - 1}}{\alpha_k^{\alpha_k} \beta_k^{\beta_k}} e^{-(\alpha_k - 1) \sum_{s|x_s = k} \log(y_s)^{-1}} \times e^{-(\beta_k - 1) \sum_{s|x_s = k} \log(1 - y_s)^{-1}}. \tag{2.3.6}$$

Pour être en mesure d'utiliser l'algorithme de Metropolis-Hasting pour estimer la densité *a posteriori*, il faut tout d'abord choisir une fonction de proposition. Pour des raisons de simplicité et compte tenu de bons résultats préalables, nous utilisons une densité binormale tronquée sur $\mathbb{R}_+^2$ comme fonction de proposition pour (α_k, β_k), qui est donnée par

$$\begin{pmatrix} \alpha_k \\ \beta_k \end{pmatrix} \sim N_2 \left(\begin{pmatrix} \widehat{\alpha}_k \\ \widehat{\beta}_k \end{pmatrix}, \widehat{\Sigma} \right) \cdot \mathbb{I}_{(0,\infty)^2},$$

où $\widehat{\alpha}_k$ et $\widehat{\beta}_k$ sont les estimateurs du maximum de vraisemblance de α_k et β_k respectivement et $\widehat{\Sigma}$ la matrice de variance-covariance asymptotique de $\widehat{\alpha}_k$ et $\widehat{\beta}_k$ définie par

$$\widehat{\Sigma} = \mathbb{E}^{f(y|\alpha_x, \beta_x, x)} \left[\left(-\frac{\partial^2}{\partial \theta_i \partial \theta_j} \log f(y|\theta, x) \right)_{i,j}^{-1} \right],$$

où $\underset{\sim}{\theta} = (\alpha_k, \beta_k)'$. Finalement, $\widehat{\Sigma}$ est défini comme

$$\widehat{\Sigma} = \begin{pmatrix} \sigma_{\alpha_k}^2 & r\sigma_{\alpha_k}\sigma_{\beta_k} \\ r\sigma_{\alpha_k}\sigma_{\beta_k} & \sigma_{\beta_k}^2 \end{pmatrix},$$

où r est la corrélation entre α et β. Nous cherchons maintenant les densités de proposition marginales.

Proposition 2.3.2. *Pour une densité binormale donnée par*

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix} \sim N_2 \left(\begin{pmatrix} \mu_\alpha \\ \mu_\beta \end{pmatrix}, \begin{pmatrix} \sigma_\alpha^2 & r\sigma_\alpha\sigma_\beta \\ r\sigma_\alpha\sigma_\beta & \sigma_\beta^2 \end{pmatrix} \right),$$

les densités marginales sont données par

$$\alpha|\beta \sim N\left(\mu_\alpha + r\frac{\sigma_\alpha}{\sigma_\beta}(\beta - \mu_\beta), \sigma_\alpha^2(1-r^2)\right), \tag{2.3.7}$$

$$\beta|\alpha \sim N\left(\mu_\beta + r\frac{\sigma_\beta}{\sigma_\alpha}(\alpha - \mu_\alpha), \sigma_\beta^2(1-r^2)\right). \tag{2.3.8}$$

Avec le résultat de la proposition précédente, nous sommes maintenant en mesure de simuler un échantillon des densités marginales *a posteriori* selon l'algorithme défini à la section 1.3.3.

2.3.2. Modèle *a priori* sur la segmentation

Comme nous l'avons vu à la sous-section 1.1.5.3, nous voulons appliquer un modèle statistique au champ de Markov caché discret X. Comme dans Destrempes *et al.* (2005), nous utilisons un modèle de Potts défini par l'équation (1.1.3). Cependant, contrairement au modèle de Destrempes *et al.* (2005) le nombre de classes K est fixe dans notre modèle. Par conséquent, le modèle *a priori* se résume à l'équation suivante

$$\pi(x|\tau) = \frac{1}{Z(\tau)} e^{-\tau \sum_{<s,t>} \left(1-\delta(x_s,x_t)\right)}, \tag{2.3.9}$$

et nous rappelons que la sommation est prise sur toutes les paires de voisins. Il est possible de réécrire l'équation (2.3.9) en terme des pixels et de leur voisinage

$$\pi(x|\tau) = \prod_s \frac{1}{Z(\tau)} e^{-\tau \sum_{t\in N_2(s)} \left(1-\delta(x_s,x_t)\right)}. \tag{2.3.10}$$

Nous voulons estimer le paramètre τ qui représente en quelque sorte le poids du modèle markovien *a priori*. Comme nous avons vu à la sous-section 1.1.5.2, il est impossible de travailler avec la fonction de partition $Z(\tau)$ vu le très grand nombre de configurations de segmentations possibles. Nous allons donc utiliser le résultat que nous avons présenté à la même sous-section qui met en relation les probabilités locales et globales. En utilisant l'équation (1.1.2), nous trouvons la pseudo-vraisemblance de notre modèle *a priori*

$$\pi(x|\tau) = \prod_s \frac{\exp\left\{-\tau \sum_{t\in N_2(s)}(1-\delta(x_s,x_t))\right\}}{\sum_{e\in\Lambda} \exp\left\{-\tau \sum_{t\in N_2(s)}(1-\delta(e_s,x_t))\right\}}, \tag{2.3.11}$$

où $\Lambda = \{1, \ldots, K\}$ est l'ensemble des classes que peut prendre le champ de Markov caché discret X. Avec la représentation donnée à l'équation (2.3.11), nous estimons le paramètre τ à l'aide de l'algorithme de Monte Carlo avec fonction d'importance.

Nous modélisons τ *a priori* par une densité uniforme $\pi(\tau) \propto \mathbb{I}_{(0,\infty)}$. Comme nous avons vu à la sous-section 1.3.2, nous devons proposer une fonction d'importance $h(\tau)$ à partir de laquelle nous allons simuler les valeurs de τ. Comme pour les paramètres de vraisemblance α_k et β_k, nous utilisons la densité asymptotique de τ donnée par

$$h(\tau) \sim N(\hat{\tau}, \sigma_\tau^2),$$

où $\hat{\tau}$ est l'estimateur du maximum de vraisemblance de τ et σ_τ^2 sa variance asymptotique.

Une autre différence de notre approche par rapport à Destrempes *et al.* (2005) est le traitement de la contrainte globale sur la segmentation. Dans leur approche, cette dernière est intégrée à la fonction de perte dichotomique 0-1 et vient ainsi pondérer cette dernière. Pour notre part, nous l'intégrons au modèle *a priori*. Le modèle devient donc

$$\begin{aligned} \pi(x|\tau) &= \frac{1}{Z(\tau)} e^{-\tau \sum_{<s,t>} \left(1-\delta(x_s,x_t)\right)} e^{-\rho(x)+\log(Z(\tau))}, \\ &\propto e^{-\tau \sum_{<s,t>} \left(1-\delta(x_s,x_t)\right)} e^{-\rho(x)} \end{aligned}$$

où $\rho(x)$ est donné par l'équation (1.2.2) dans laquelle nous fixons le paramètre ω à 0.

2.3.3. Estimation de la segmentation par l'algorithme EM

Nous avons maintenant un modèle statistique complètement spécifié, nous sommes donc en mesure d'estimer la segmentation x à partir de l'algorithme EM tel que défini à la section 1.3.1. Pour accélérer la simulation, nous utilisons une segmentation initiale non aléatoire. Nous utilisons une segmentation par seuil très simple qui consiste à diviser l'étendue de couleur observée en K classes, K étant fixé préalablement, et assigner chaque pixel à sa classe respective. Basé sur des résultats empiriques préliminaires, nous remarquons que l'algorithme EM

converge plus rapidement lorsque nous ajoutons une légère composante aléatoire à cette segmentation initiale par seuillage. Ainsi, nous assignons aléatoirement une région initiale à 25% des pixels. Nous notons la segmentation initiale par $x^{(0)}$. L'implantation de l'algorithme EM est en soit très simple, nous résumons ici en quelques étapes la marche à suivre.

(1) Simuler les paramètres $\alpha_k^{(i)}$ et $\beta_k^{(i)}$ sachant la segmentation $x^{(i)}$ par l'algorithme de Metropolis-Hasting tel que vu à la section 2.3.1 pour chaque classe $k \in (1, \ldots, K)$;

(2) Estimer α et β selon les différentes fonctions de perte ;

(3) pour chaque pixel s, générer une nouvelle classe $x_s^{(i+1)}$ selon une loi multinomiale de paramètres K et

$$w_e \propto f(y_s|x_s = e, \alpha_e^{(i)}, \beta_e^{(i)}) \exp\left\{-\tau \sum_{t \in N(s)} (1 - \delta(e, x_t))\right\} \times \exp\{-\rho(x|x_s = e)\}, \quad (2.3.12)$$

avec $e = \{1, \ldots, K\}$. De cette manière nous obtenons la segmentation $x^{(i+1)}$;

(4) recommencer les étapes 1 et 2 jusqu'à convergence de la segmentation.

Nous avons cependant un problème d'ordre computationnel lorque nous utilisons l'algorithme EM tel qu'il est décrit ci-dessus. En effet, le calcul de la contrainte globale $\rho(x)$ dans l'équation (2.3.12) est trop long pour être utilisé dans les simulations. Cela s'explique par le fait qu'il faut calculer la taille des n régions connectées à chaque pixel et pour chacune des K classes proposées, et ce, à chaque itération.

Premièrement nous avons tenté d'optimiser le calcul de la contrainte globale, mais en vain, les améliorations que nous voyions ne changeaient pas l'ordre de grandeur du temps de calcul de manière significative. Prenons par exemple une image de taille 200×200 avec 10 classes à estimer. Il faudra calculer à chaque itération la contrainte globale 400 000 fois. Disons que le calcul prend dix secondes, le temps de calcul total sera d'environ 46 jours. Si nous optimisons le

temps de calcul à une seconde, le temps total sera de 4,6 jours. L'amélioration est significative mais le temps de calcul est toujours inacceptable.

Une autre possibilité nous permettant de contourner le problème serait de développer une sorte de pseudo-contrainte globale basée uniquement sur les voisins des pixels. Mais cette proposition se ramène en partie au modèle markovien qui prend déjà en compte la valeur des pixels dans un voisinage local.

Nous proposons donc une variante à l'algorithme qui peut s'apparenter à l'algorithme de Metropolis-Hasting. Nous allons tout simplement générer les nouvelles classes selon le modèle sans la contrainte globale, c'est-à-dire

$$q(x|y,\underset{\sim}{\alpha},\underset{\sim}{\beta}) = \prod_s f(y_s|x,\underset{\sim}{\alpha},\underset{\sim}{\beta}) \exp\left\{-\tau \sum_{<s,t>} (1-\delta(x_s,x_t))\right\}.$$

La fonction q sera notre fonction de proposition pour l'algorithme de Metropolis-Hasting. Maintenant, basé sur le même algorithme que nous avons décrit précédemment, nous allons générer une nouvelle classe selon cette fonction simplifiée. Ainsi, à l'étape (2) de l'algorithme, pour chaque pixel s, nous générons une nouvelle classe $x_s = e$ selon les nouvelles probabilités

$$w_e \propto f(y_s|x_s = e, \alpha_e^{(i)}, \beta_e^{(i)}) \exp\left\{-\tau \sum_{t\in N(s)} (1-\delta(e,x_t))\right\},$$

avec $e = \{1,\ldots,K\}$. Ensuite nous devons décider si nous acceptons ou rejetons la segmentation proposée. L'acceptation-rejet se fait ensuite selon le rapport que nous avons vu à la section 1.3.3, c'est-à-dire

$$\psi = \min\left(1, \left[\frac{\pi(x^{(i+1)}|y,\underset{\sim}{\alpha}^{(i)},\underset{\sim}{\beta}^{(i)})}{\pi(x^{(i)}|y,\underset{\sim}{\alpha}^{(i)},\underset{\sim}{\beta}^{(i)})}\right] \Big/ \left[\frac{q(x^{(i+1)}|y,\underset{\sim}{\alpha}^{(i)},\underset{\sim}{\beta}^{(i)})}{q(x^{(i)}|y,\underset{\sim}{\alpha}^{(i)},\underset{\sim}{\beta}^{(i)})}\right]\right).$$

Dans notre cas, le rapport ψ se simplifie à

$$\psi = \min\left(1, \frac{\exp(-\rho(x^{(i+1)}))}{\exp(-\rho(x^{(i)}))}\right).$$

Cette approche est très intéressante car elle implique que nous calculons seulement une fois la contrainte globale à chaque étape et nous conservons la vocation de "globalité" de cette contrainte sur la segmentation.

Avec cette dernière modification à l'algorithme EM, nous sommes maintenant en mesure d'estimer la segmentation de l'image. Nous voulons faire remarquer que

la segmentation de l'image par l'algorithme EM implique que cette dernière sera simulée à partir de l'information disponible à chaque itération (voir Dempster *et al.* (1977)). Ainsi, les fonctions de perte ne sont pas appliquées à la segmentation, mais plutôt à l'estimation des paramètres de vraisemblance du modèle. Dans Destrempes *et al.* (2005), la segmentation est estimée au sens du MAP.

Nous allons maintenant voir à la section suivante comment simuler une image à partir de cette segmentation.

2.4. Simulation de l'image segmentée

Une fois l'image segmentée et les paramètres du modèle estimés, nous désirons simuler une image selon cette segmentation, c'est-à-dire que pour chaque classe de pixels nous désirons simuler la couleur estimée *a posteriori*. Cette tâche n'est pas très compliquée car la couleur des pixels est modélisée par une densité bêta et nous avons déjà estimé les paramètres du modèle. Ainsi, nous simulons une nouvelle couleur pour chaque classe à partir de la densité prédictive.

Définition 2.4.1. *La densité prédictive pour une nouvelle observation $\tilde{y}$ est donnée par*

$$m(\tilde{y}|y) = \int_{\Theta} f(\tilde{y}|\theta)\pi(\theta|y)d\theta. \tag{2.4.1}$$

Dans notre cas, l'équation (2.4.1) pour un segment k devient

$$m(\tilde{y}|y) = \int_{\alpha_k}\int_{\beta_k} B(\tilde{y}|\alpha_k,\beta_k)\pi(\alpha_k,\beta_k|y)d\alpha_k d\beta_k. \tag{2.4.2}$$

Il est évident que nous ne pouvons pas résoudre analytiquement l'équation (2.4.2). Ainsi nous utilisons une approche numérique pour estimer la couleur. Un moyen simple d'obtenir un échantillon de la densité prédictive est de simuler une valeur de α_k^l et de β_k^l à partir de la densité *a posteriori* $\pi(\alpha_k,\beta_k|y)$ et ensuite de simuler une nouvelle valeur $\tilde{y}$ à partir de la vraisemblance selon ces paramètres, c'est-à-dire $B(\tilde{y}|\alpha_k^l,\beta_k^l)$. Cependant, nous avons déjà obtenu un échantillon de la densité *a posteriori* pour les paramètres α_k et β_k lorsque nous avons estimé les paramètres du modèle. Nous allons donc utiliser ces vecteurs pour générer l'échantillon de la densité prédictive. Il est à remarquer que l'estimation de la densité prédictive se fait indépendamment segment par segment.

Comme pour l'estimation des paramètres, nous voulons également comparer l'impact des quatre fonctions de perte sur la couleur estimée dans les classes. Dans le cas de la fonction de perte quadratique et la fonction de perte dichotomique 0-1, les estimateurs bayésiens résultants sont simples et il est facile de les calculer à partir de l'échantillon de la densité prédictive. Dans le cas des deux autres fonctions de perte, il est un peu plus difficile de calculer l'estimateur sans passer par une forme analytique pour la densité prédictive. Dans ce cas, nous utilisons la forme analytique suivante qui est une estimation par Metropolis-Hasting

$$m(\tilde{y}|y) = \frac{1}{T}\sum_{t=1}^{T} f(\tilde{y}|\alpha_t, \beta_t).$$

La forme analytique précédente nous permet d'obtenir un estimé de la densité prédictive pour toute valeur de $\tilde{y}$. Ainsi, à l'aide de méthodes numériques comme la méthode de bisection, nous pouvons maintenant calculer les estimateurs bayésiens résultants de la fonction de perte dichotomique 0-1 sur un intervalle et de la fonction de perte de Huber modifiée.

2.5. Évaluation de la convergence du MCMC

Un des grands défis des simulations de Monte Carlo par chaînes de Markov est l'évaluation de la convergence. En effet, comme nous l'avons mentionné à la section 1.3.3, il faut exécuter les simulations jusqu'à convergence vers la distribution stationnaire. Cependant, l'évaluation de cette convergence n'est pas une tâche facile et plusieurs auteurs ont tenté de trouver des méthodes pour le faire. Les méthodes les plus utilisées restent encore un mélange de plusieurs approches dont des techniques graphiques d'évaluation de la convergence. Cependant, dans notre contexte de segmentation d'images, nous cherchons à mettre sur pied une méthodologie de segmentation non dirigée par l'utilisateur qui fonctionne pour toutes les images. Ainsi, il n'est pas concevable de penser utiliser des méthodes graphiques pour évaluer la convergence. De plus, nous avons un très grand nombre de paramètres à évaluer, vu la nature itérative de l'algorithme EM. Nous rappelons que nous devons estimer deux paramètres pour chacune des K classes à chaque itération, et ce seulement pour le modèle de vraisemblance. Ainsi, nous

devons utiliser une méthode indépendante de l'utilisateur, qui donne des résultats précis et relativement rapides.

Plusieurs approches ont été proposées dans la litérature pour développer une règle d'évaluation automatique de la convergence pour le MCMC. Certains auteurs (voir Brooks, 1998) utilisent des statistiques basées sur les pentes des droites qui relient chacun des points de la chaîne de Markov, d'autres (voir Chauveau et Diebolt, 1998) utilisent des statistiques basées sur le théorème limite centrale ou bien des méthodes non paramétriques (voir Brooks, Giudici et Philippe, 2003). Cependant, ces méthodes ne donnent pas toujours de bons résultats dans notre situation. Nous rappelons que nous cherchons une méthode qui fonctionne dans la majorité des cas et surtout qui ne force pas les chaînes à rouler trop longtemps. Le facteur temps est très important, considérant encore une fois le nombre de paramètres que nous devons estimer. D'autres techniques d'évaluation de la convergence proposées par certains auteurs (voir Gelman et Rubin, 1992, Brooks et Gelman, 1998) sont plutôt basées sur des chaînes multiples ainsi que la variance inter et intra-chaînes. Il s'avère que dans notre situation, cette dernière approche donne les meilleurs résultats.

Nous décrivons ici brièvement la méthode de Gelman et Rubin (1992) pour le cas univarié. Soit $\xi = (\xi_1, \ldots, \xi_M)$, M chaînes quelconques de longueur T. Nous calculons tout d'abord

$$W = \sum_{m=1}^{M} s_m^2/M = \sum_{m=1}^{M} \frac{1}{M} \sum_{t=1}^{T} \frac{(\xi_{mt} - \bar{\xi}_{m\cdot})^2}{T-1},$$

et

$$B = \frac{T}{M-1} \sum_{m=1}^{M} (\bar{\xi}_{m\cdot} - \bar{\xi}_{\cdot\cdot}),$$

où $\bar{\xi}_{m\cdot} = \frac{1}{T}\sum_{t=1}^{T} \xi_{m,t}$ et $\bar{\xi}_{\cdot\cdot} = \frac{1}{M}\sum_{m=1}^{M} \bar{\xi}_{m\cdot}$. Les composantes W et B sont respectivement les estimés de la moyenne des variances intra-chaînes et de la variance inter-chaînes. Le critère d'arrêt est ensuite basé sur la statistique suivante

$$\widehat{R} = \frac{\widehat{V}}{W} \frac{df}{df-2}, \tag{2.5.1}$$

où

$$\widehat{V} = \frac{(T-1)}{T}W + \frac{M+1}{MT}\frac{B}{W},$$
$$df = \frac{2\widehat{V}^2}{\widehat{\mathrm{Var}}(\widehat{V})},$$

avec

$$\widehat{\mathrm{Var}}(\widehat{V}) = \left(\frac{T-1}{T}\right)^2 \frac{1}{M}\widehat{\mathrm{Var}}(s_m^2) + \left(\frac{M+1}{MT}\right)^2 \frac{2}{M-1}B^2 + 2\frac{(M+1)(T-1)}{MT^2}\frac{T}{M}\left(\widehat{\mathrm{Cov}}(s_m^2, \bar{\xi}_{m\cdot}^2) - 2\bar{\xi}_{\cdot\cdot}\widehat{\mathrm{Cov}}(s_m^2, \bar{\xi}_{m\cdot})\right),$$

où $\widehat{\mathrm{Cov}}(s_m^2, \bar{\xi}_{m\cdot}^2)$ est la covariance échantillonnale entre les variances et le carré des moyennes échantillonnales des m chaînes et l'équivalent pour $\widehat{\mathrm{Cov}}(s_m^2, \bar{\xi}_{m\cdot})$.

La composante df vient du fait que les auteurs utilisent une approximation de la densité de ξ par une loi de Student de degrés de liberté df. Le facteur $\frac{df}{df-2}$ dans l'équation (2.5.1) est proposé dans Brooks et Gelman (1998). En effet, nous pouvons remarquer que pour des petits degrés de liberté, le facteur peut être négatif. Le critère corrigé est donné par l'équation suivante

$$\widehat{R} = \frac{\widehat{V}}{W}\frac{df+3}{df+1}. \tag{2.5.2}$$

La statistique $\widehat{R}$ estime le facteur de réduction de l'étendue de la distribution de ξ que nous pouvons espérer si nous continuons les simulations jusqu'à $T \to \infty$. En d'autres mots, c'est le rapport de la variance de l'estimé de ξ sur la variance intra-chaînes avec un facteur d'ajustement pour la variance de la distribution de Student. Ainsi, lorsque $\widehat{R} \to 1$, nous pouvons conclure que de rouler les chaînes plus longtemps ne changerait pas les résultats. Nous trouvons dans Robert (1998) que la distribution de $\widehat{R}$ peut être déduite de l'approximation $B/W \sim F_{M-1;\psi}$, où F est la distribution de Fisher, $\psi = 2W^2/\omega$ et

$$\omega = \frac{1}{M^2}\left(\sum_{m=1}^{M} s_m^4 - \frac{1}{M}\left(\sum_{m=1}^{M} s_m^2\right)^2\right).$$

Ainsi, nous cherchons une valeur ε telle que

$$\Pr\left[\widehat{R} < 1+\varepsilon\right] = 0{,}95$$

$$\Leftrightarrow \Pr\left[\frac{T-1}{T} + \frac{M+1}{MT}\frac{B}{W} < 1+\varepsilon\right] = 0{,}95$$
$$\Leftrightarrow \Pr\left[\frac{M+1}{MT}\frac{B}{W} < 1+\varepsilon-\frac{T-1}{T}\right] = 0{,}95$$
$$\Leftrightarrow \Pr\left[\frac{B}{W} < \frac{MT}{M+1}\left(1+\varepsilon-\frac{T-1}{T}\right)\right] = 0{,}95$$
$$\Leftrightarrow \frac{MT}{M+1}\left(1+\varepsilon-\frac{T-1}{T}\right) = F_{M-1;\psi;0{,}95}.$$

Avec cette approximation de la distribution de la statistique $\widehat{R}$, nous sommes en mesure de tester si la statistique est suffisament près de 1.

2.6. Conclusion de chapitre

Au chapitre 2 nous avons présenté les notions de base en théorie de la décision bayésienne. Celles-ci nous ont permis de proposer quatre fonctions de perte et les règles bayésiennes qui leur sont associées. Ensuite, nous avons décrit le cadre que nous considérons pour représenter une image dans le modèle proposé. Nous avons mis en évidence certaines différences majeures par rapport à l'approche de Destrempes *et al.* (2005). Finalement, nous avons proposé une nouvelle approche décisionnelle basée sur le modèle de Destrempes *et al.* (2005) et nous avons décrit la méthode d'estimation de ce modèle à l'aide de l'algorithme EM. Nous avons terminé en décrivant l'approche utilisée pour simuler une image à partir de la segmentation estimée.

Au prochain chapitre, nous allons présenter des résultats de l'application de la méthode proposée sur des images réelles et synthétiques.

Chapitre 3

ANALYSE EN GRAPPE, SIMULATIONS ET RÉSULTATS

Le but principal de ce dernier chapitre est de présenter des résultats de simulation en mettant en oeuvre la méthode proposée au chapitre 2. En premier lieu, nous décrivons la technique d'analyse en grappe que nous utilisons pour grouper les régions formées par la superposition des trois composantes segmentées indépendemment.

Deuxièmement, nous présentons des résultats de simulations sur des images synthétiques pour tester la performance de notre méthodologie par rapport aux différentes fonctions de perte proposées. Nous allons comparer l'effet du choix de la fonction de perte en fonction du bruit dans l'image et en fonction du nombre de segments que nous utilisons.

Finalement, nous illustrons la méthode proposée à l'aide de quelques images naturelles. Nous verrons également des représentations d'une image segmentée à plusieurs étapes de l'algorithme de l'analyse en grappe pour observer la dégradation de l'image estimée à mesure que le nombre de segments diminue.

3.1. ANALYSE EN GRAPPE

Comme nous avons vu à la section 2.2, la méthode de segmentation que nous proposons consiste à traiter séparément chaque composante de l'image, c'est-à-dire les composantes L, a et b. Cependant, lorsque nous reconstruisons l'image finale en superposant les trois composantes segmentées, nous nous retrouvons

avec un nombre de segments beaucoup plus grand que celui désiré. Ainsi, nous proposons d'utiliser une méthode d'analyse en grappe pour fusionner certaines régions et ainsi obtenir le nombre de segments voulu.

Le champ de recherche qui se concentre sur la classification d'objets ou l'identification de groupes à l'intérieur d'un jeu de données est très vaste et pourrait à lui seul faire l'objet d'un mémoire complet. Puisque nous ne voulons pas nous étendre trop sur le sujet nous présentons seulement l'approche que nous utilisons. La technique d'analyse en grappe que nous avons sélectionnée est une méthode bien connue en statistique et fait partie de la famille de techniques d'analyse en grappe dites hiérarchiques d'agglomération. Plus précisément, nous utilisons la technique d'analyse en grappe par la méthode du centroïde.

Nous référons à l'annexe A pour plus de détails sur cet algorithme d'analyse en grappe et la manière dont nous l'implantons.

3.2. Discussion sur les critères d'arrêt

Dans cette section nous voulons faire un retour sur les notions de critères d'arrêt d'un point de vue plus pratique. Nous discutons du critère d'arrêt pour les chaînes de Markov et du critère d'arrêt pour l'algorithme EM.

Nous avons présenté à la section 2.5 le critère d'arrêt que nous utilisons pour établir la convergence du MCMC. Comme nous l'avons déjà mentionné, un des défis auquel nous faisons face est de proposer une méthode qui est complètement indépendante de l'utilisateur et qui n'est pas trop lourde à évaluer. Par conséquent, même la méthode de Gelman et Rubin (1992) présente des faiblesses à cet égard. Il arrive parfois que les paramètres $(\underset{\sim}{\alpha}, \underset{\sim}{\beta})$ du modèle de vraisemblance ne convergent pas au sens du critère de Gelman et Rubin (1992) mais que la couleur simulée à partir de la densité prédictive ait convergé. Ceci est surtout dû à la nature du modèle de vraisemblance que nous utilisons. En effet, dans un modèle bêta il n'y a pas de paramètres de position et d'échelle comme dans un modèle gaussien par exemple. Ainsi, une combinaison des deux paramètres α et β est utilisée pour déterminer la position et le facteur d'échelle. Par conséquent, il est possible de faire varier les paramètres de sorte que la position reste la même et

que seulement la variance change. Rappelons que l'espérance et la variance d'une variable aléatoire $\theta \sim B(\alpha, \beta)$ sont données respectivement par

$$\mathbb{E}(\theta) = \frac{\alpha}{\alpha + \beta},$$
$$V(\theta) = \frac{\alpha\beta}{(\alpha + \beta)^2(\alpha + \beta + 1)},$$

de sorte que si les deux paramètres croissent ou décroissent de manière proportionnelle, l'espérance restera sensiblement la même et uniquement la variance sera affectée. Il s'avère que cette situation se présente souvent dans nos simulations et puisque notre intérêt final est la couleur estimée dans chacune des régions, nous proposons alors de suivre la convergence de la couleur estimée selon la densité prédictive. Nous sommes conscient que cette approche est un raccourci et que théoriquement nous devrions tester la convergence des deux paramètres surtout que ces derniers sont utilisés pour les probabilités de sélection dans l'algorithme EM. Cependant, à la lumière de résultats empiriques qui montrent que les estimés obtenus à l'aide de cette approche sont stables et acceptables, nous jugeons que les avantages de traiter la convergence de la couleur estimée par la densité prédictive surpassent les incohérences théoriques.

En somme, voici comment nous mettons en oeuvre le critère d'arrêt de Gelman et Rubin (1992) pour évaluer la convergence des chaînes de Markov.

- Nous effectuons au moins 500 itérations avant d'évaluer la convergence des chaînes ;
- l'évaluation de la convergence est effectuée sur la couleur estimée à partir de la densité prédictive ;
- nous utilisons une période de chauffe de $n/2$ itérations ;
- après avoir atteint la convergence, nous simulons un échantillon de $n = 1000$ observations à partir de la distribution stationnaire.

Discutons maintenant du critère d'arrêt pour l'algorithme EM. Cette étape n'est pas simple puisqu'il existe très peu de références dans lesquelles le critère d'arrêt utilisé est détaillé. En effet, la majorité des auteurs qui utilisent l'algorithme EM ou bien un dérivé de ce dernier pour segmenter une image ne mentionnent pas l'approche utilisée pour en évaluer la convergence. Dans un contexte

d'imagerie, l'évaluation de la convergence de l'algorithme EM n'est pas une tâche facile car elle dépend de plusieurs facteurs dont la complexité de l'image et le nombre de segments à estimer. En effet, plus il y a de segments dans l'image, plus il est probable que des pixels soient assignés de manière équiprobable à plus d'une région. Ainsi, dans une telle situation, nous n'obtiendrons jamais une convergence absolue pour un nombre fini d'itérations.

De manière générale, nous pouvons classer la majorité des approches proposées dans la littérature en deux groupes. Le premier consiste à suivre l'assignation des régions aux pixels d'une itération à l'autre et de construire une statistique basée sur ce changement d'assignation. Une telle approche est utilisée dans Gu et Sun (2005). Le deuxième groupe englobe les mesures de convergence qui traitent l'évolution de la vraisemblance de l'image estimée à chaque itération. Le papier de Nasios et Bors (2004) présente un critère d'arrêt basé sur cette approche.

En ce qui concerne le premier groupe de mesures basées sur l'assignation des classes, il est évident que pour une étape intermédiaire dans le processus de convergence, le nombre de pixels qui sont assignés à une classe différente d'une itération à l'autre est beaucoup plus grand qu'à une étape près du point de convergence. Dans ce sens, Gu et Sun (2005) proposent une statistique bien simple qui correspond au nombre de pixels qui se sont vu assigner une classe différente entre l'itération i et l'itération $i+1$. Une telle statistique devrait approcher 0 lorsque nous atteignons la convergence ; Gu et Sun (2005) proposent d'arrêter l'algorithme lorsque cette statistique devient inférieure à 1%. Cependant, en pratique nous observons que cette statistique dépend de l'image. En effet, si nous initialisons l'algorithme avec une segmentation complètement aléatoire, dans les premières itérations pratiquement aucun pixel se voit réassigner la même étiquette de région et plus l'algorithme avance, plus la statistique diminue pour finalement atteindre un plateau à partir d'un certain moment. Le problème vient du fait que ce plateau ne se situe jamais au même niveau d'une image à l'autre. Pour cette raison, nous n'utilisons pas ce critère d'arrêt.

Dans le cas du deuxième type d'approches, nous trouvons que les mesures basées sur la vraisemblance de l'image estimée sont plus stables. L'approche que

nous proposons au chapitre 2 utilise un modèle bayésien ; ainsi, nous proposons de suivre l'évolution de la densité *a posteriori* au lieu de la vraisemblance uniquement. En effet, puisque l'algorithme EM est utilisé pour estimer la densité *a posteriori*, il est plus logique de proposer une mesure basée sur cette même densité. La mesure proposée par Nasios et Bors (2004) est tout simplement l'accroissement relatif de la log-vraisemblance d'une itération à l'autre et ils suggèrent d'arrêter les simulations lorsque cette mesure est inférieure à 1%. Un des problèmes que nous observons dans nos simulations est que la densité *a posteriori* n'est pas strictement croissante d'une itération à l'autre (ce qui est également vrai pour la vraisemblance), mais suggère une allure généralement croissante qui semble converger à partir d'un certain nombre d'itérations. Ainsi, nous proposons de suivre la statistique correspondant à la pente de la droite de régression des dix derniers points de la trace du log de la densité *a posteriori*. De la même manière, cette statistique s'approche de 0 lorsque nous atteignons la convergence, mais elle permet de lisser la trace pour minimiser l'effet d'une baisse de la densité *a posteriori* à une itération quelconque. Il est important de noter ici qu'il y a un facteur d'échelle à considérer. En effet, le log de la densité *a posteriori* peut être d'un ordre relativement élevé pour une image naturelle de taille raisonnable. Ainsi, il devient absurde de considérer une borne fixe indépendante de l'image. Nous proposons alors d'arrêter les simulations lorsque la pente de la droite de régression est inférieure à 1% du maximum du log de la densité *a posteriori*. Par exemple, pour une image où le log de la densité *a posteriori* varie dans un ordre de 10^4, nous arrêtons les simulations lorsque la statistique est inférieure à 100. De plus, par mesure de précaution nous arrêtons l'algorithme à la deuxième occurence où la statistique est inférieure à la borne ; de cette manière nous sommes beaucoup plus conservateurs.

La figure 3.2.1 montre la trace du log de la densité *a posteriori* et la mesure proposée pour le critère d'arrêt pour une image synthétique de 160×160 pixels. Comme nous l'avons déjà mentionné, la densité *a posteriori* n'est pas strictement croissante : à partir du moment où la convergence est atteinte elle peut osciller légèrement. Le critère proposé au paragraphe précédent aurait stoppé les

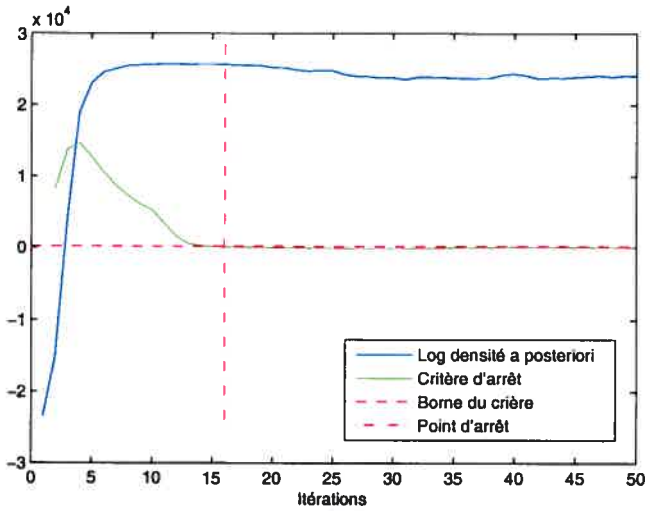

FIGURE 3.2.1. Trace du log de la densité *a posteriori*

simulations à l'itération 16 et nous pouvons voir dans le graphique 3.2.1 que la convergence semble atteinte à cette étape de l'algorithme.

3.3. CHOIX DE LA TOLÉRANCE POUR LES FONCTIONS DE PERTE

Comme nous avons vu au chapitre précédent, nous incluons un facteur de tolérance dans la fonction de perte 0-1 sur un intervalle et dans la fonction de perte de Huber modifiée. Jusqu'à présent, nous n'avons pas discuté du choix de cette tolérance. Nous rappelons que le facteur de tolérance consiste en une erreur que nous jugeons acceptable sur l'estimation du paramètre. Nous discutons ici brièvement du choix de cette valeur.

Nous basons notre choix sur la perception d'une différence entre deux niveaux de gris. En effet, puisque nous traitons chacune des composantes indépendemment dans notre modèle, chacune d'elle devient comme une observation en noir et blanc. Puisque nous ramenons toujours les étendues de chaque composante sur l'intervalle $[0, 1]$ en considérant le facteur de majoration de 5% comme nous l'avons expliqué à la section 2.2, nous utilisons toujours la même valeur de tolérance relative sur l'intervalle $[0, 1]$ pour chacune des composantes.

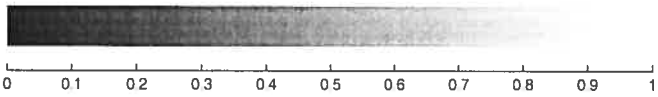

FIGURE 3.3.1. Étendue des niveaux de gris

Comme nous pouvons observer dans la figure 3.3.1, sur chaque intervalle d'une longueur 0,10 sur la bande des niveaux de gris il est difficile d'identifier une différence à l'oeil nu. Ainsi, nous utilisons une tolérance bilatérale de 5% sur la couleur estimée et jugeons qu'après cette distance l'erreur commise est trop grande.

Dans le cas de l'estimation des paramètres de vraisemblance $(\underset{\sim}{\alpha}, \underset{\sim}{\beta})$, nous utilisons la même tolérance relative, mais cette fois nous l'appliquons à la valeur moyenne de la chaîne. Par exemple, pour une chaîne qui oscille en moyenne autour de 125, la tolérance appliquée serait de $0{,}05 \cdot 125 = 6{,}25$.

Nous allons voir à la section suivante des résultats de simulations sur des images synthétiques. Ces dernières sont des images artificielles où nous connaissons la valeur des paramètres initiaux. Une des simulations qui sera présentée compare différents niveaux de tolérance.

3.4. RÉSULTATS DE SIMULATIONS SUR DES IMAGES SYNTHÉTIQUES

Même si la méthode proposée est construite pour segmenter des images naturelles, nous utilisons tout de même cette dernière sur des images synthétiques pour pouvoir valider les résultats obtenus. En effet, c'est seulement sur une image synthétique qu'il est possible de connaître les vraies classes sous-jacentes à une image. Ainsi, à partir d'une image dont nous connaissons les régions, nous simulons une nouvelle image en y ajoutant un bruit gaussien centré en 0 avec une certaine variance, et ce pixel par pixel. Nous présentons dans cette section les résultats obtenus avec la méthode de segmentation proposée selon les quatre fonctions de perte pour trois images synthétiques en noir et blanc.

Dans un premier temps nous présentons des résultats en fonction du niveau de bruit qui est appliqué aux trois images synthétiques. Par niveau de bruit nous faisons référence à la variance utilisée pour le bruit gaussien. Nous appliquons trois niveaux de bruit, soient des variables aléatoires gaussiennes centrées en 0 avec une variance de 5%,10% et 20% de l'étendue des données. Dans un deuxième temps, nous présentons des résultats de simulations en fonction du nombre de classes demandé. Pour une seule image et un niveau de bruit choisi, nous faisons varier le nombre de classes demandé de manière à avoir soit moins ou plus de classes que le nombre réel pour observer le comportement de l'algorithme.

L'utilisation d'images en noir et blanc pour effectuer ces simulations n'a pas réellement d'impact car la méthode que nous proposons se fait indépendemment pour chaque composante du système Lab, et puisque le but premier de cette simulation n'est pas de tester l'algorithme d'analyse en grappe alors nous réduisons le temps de calcul en utilisant des images en niveaux de gris.

Nous rappelons que la segmentation est effectuée au sens de l'algorithme EM ; les étiquettes de régions sont donc simulées à partir de l'information disponible à chaque itération. Ainsi, les fonctions de perte sont appliquées seulement sur l'estimation des paramètres de vraisemblance du modèle est sur l'estimation de la couleur résultante. Dans l'article de Destrempes *et al.* (2005) la segmentation est estimée au sens du MAP. Il existe également une autre fonction de perte qui est utilisée en segmentation d'images dont l'estimateur résultant est appelé modes des marginales *a posteriori* (MPM). Cette fonction de perte est moins restrictive que celle associée au MAP car elle pénalise la configuration au prorata du nombres de sites mal étiquetés (voir Marroquin, Mitter et Poggio (1987)). Par conséquent, les résultats concernant le MAP présentés dans cette section font référence à l'utilisation du MAP pour l'estimation des paramètres de vraisemblance et non pas de la segmentation.

3.4.1. Impact du choix de la fonction de perte en fonction du bruit

Dans cette section nous voulons observer le comportement de la méthode proposée avec les quatre fonctions de perte selon le niveau de bruit qui est appliqué

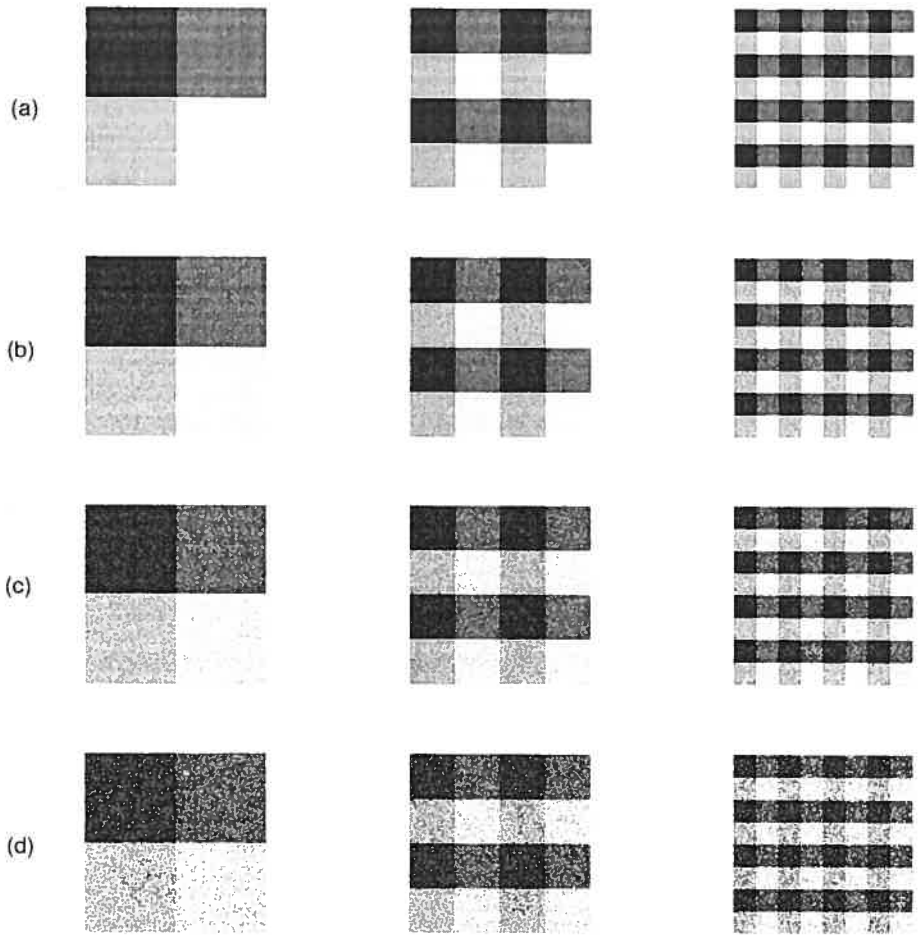

FIGURE 3.4.1. (a) Segments réels. Images synthétiques bruitées (b) à 5% (c) à 10% (d) à 20%.

aux images synthétiques. La figure 3.4.1 montre les neuf images de départ que nous utiliserons pour les simulations. Chacune de ces images est de taille 160×160 pixels. Comme nous pouvons le voir avec les images de la ligne (a) dans la figure 3.4.1, chaque image est engendrée à partir de quatre segments réels. Avec ces trois images, nous voulons évaluer l'impact de la fonction de perte sur des images où les régions deviennent de plus en plus petites en fonction du bruit. Suite aux résultats ainsi obtenus, nous tenterons de tirer des conclusions sur l'impact du choix de la fonction de perte sur la segmentation de ces images.

TABLEAU 3.4.1. Identification des régions

Région 1	Noir
Région 2	Gris foncé
Région 3	Gris pâle
Région 4	Blanc

Le protocole que nous suivons pour mettre en place la simulation est bien simple. Pour chacune des trois images synthétiques et chacun des trois niveaux de bruit, nous générons une image avec laquelle nous appliquons la méthode proposée selon les quatre fonctions de perte. De cette manière, nous utilisons toujours la même image pour chaque fonction de perte. Nous répétons ensuite cet exercice 100 fois de manière à obtenir un échantillon de 100 segmentations pour les quatre fonctions de perte proposées pour chacune des neuf images synthétiques bruitées.

Dans un premier temps nous effectuons les simulations avec une tolérance de 5% pour les fonctions de perte de Huber modifiée et 0-1 sur un intervalle. Nous présentons également des résultats pour une tolérance de 1% et 10% sur une des images, soit l'image 2×2 avec un niveau de bruit de 20%.

Nous présentons les résultats des simulations de deux manières. Premièrement, nous calculons pour chaque pixel de l'image l'erreur absolue et l'écart type par rapport aux 100 simulations. Ces résultats sont ensuite présentés de manière graphique comme dans la figure 3.4.2 qui montre les résultats obtenus avec la fonction de perte de Huber modifiée et un bruit gaussien de 20%. Les figures représentant les résultats graphiques pour chaque fonction de perte et chaque niveau de bruit sont présentées en annexe B. Dans un deuxième temps nous calculons la moyenne de ces deux statistiques sur chacune des vraies régions. Les résultats pour la simulation avec un bruit de 20% sont présentés dans les tableaux 3.4.2 et 3.4.3. Nous identifions toujours les régions dans le même ordre, c'est-à-dire de gauche à droite et de haut en bas comme le montre le tableau 3.4.1.

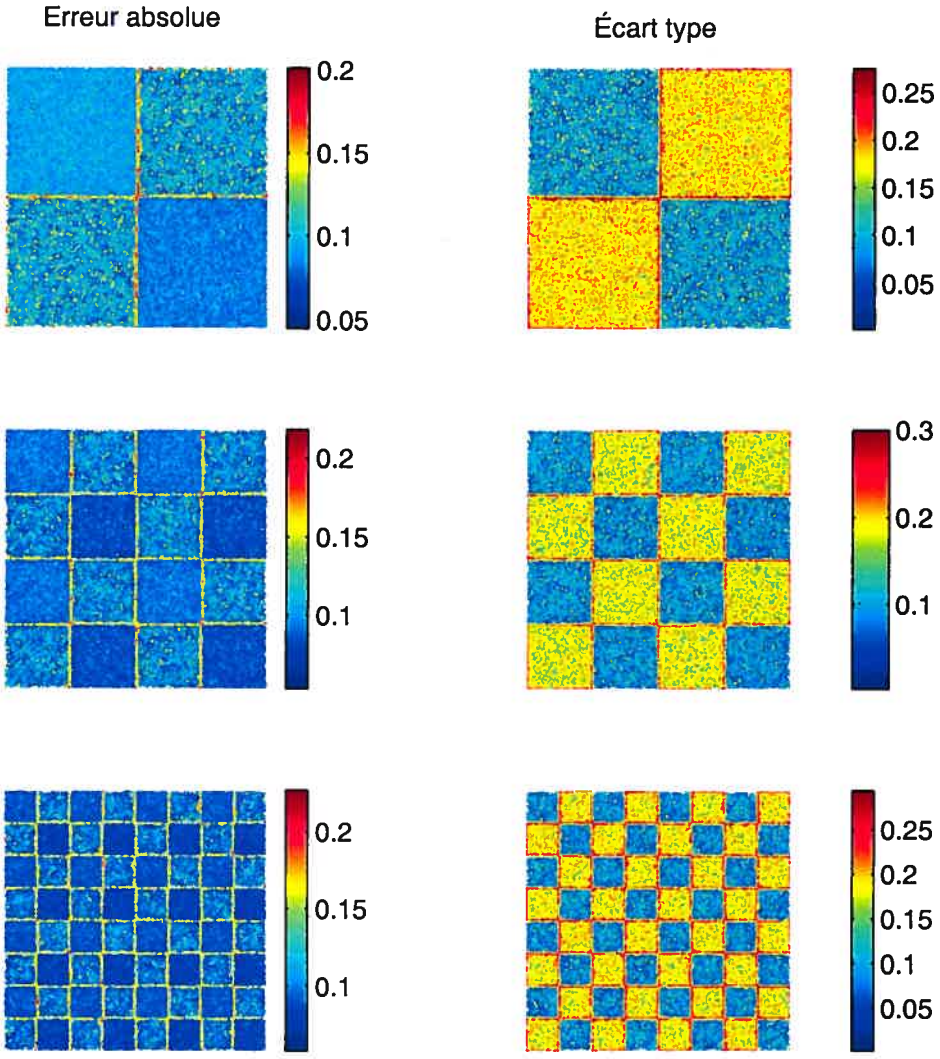

FIGURE 3.4.2. Résultats des simulations avec un bruit gaussien de 20% pour la fonction de perte de Huber modifiée

Basé sur les résultats graphiques, nous pouvons constater que pour la fonction de perte de Huber modifiée avec une tolérance de 5% et un bruit gaussien de 20%, nous commettons une plus grande erreur absolue pour les régions 2 et 3, c'est-à-dire les régions grises. Nous remarquons également que cette erreur croît légèrement lorsque les régions deviennent de plus en plus petites. Ceci s'observe également dans le tableau 3.4.2 pour les valeurs moyennes sur les régions. En effet nous voyons que l'erreur absolue moyenne pour la région 4 par exemple passe de

TABLEAU 3.4.2. Erreur absolue moyenne par région pour les simulations avec un bruit de 20%

Image	Fonction de perte	Région 1	Région 2	Région 3	Région 4
Image 1	Huber modifiée	0,0958	0,1041	0,1052	0,0870
	0-1 sur un intervalle	0,0683	0,0920	0,0932	0,0592
	Quadratique	0,0316	0,0745	0,0766	0,0358
	MAP	0,0928	0,1113	0,1137	0,0843
Image 2	Huber modifiée	0,0969	0,1096	0,1092	0,0879
	0-1 sur un intervalle	0,0695	0,0976	0,0967	0,0601
	Quadratique	0,0336	0,0795	0,0807	0,0370
	MAP	0,0939	0,1135	0,1143	0,0855
Image 3	Huber modifiée	0,0986	0,1175	0,1162	0,0900
	0-1 sur un intervalle	0,0720	0,1054	0,1045	0,0627
	Quadratique	0,0374	0,0868	0,0877	0,0413
	MAP	0,0955	0,1238	0,1245	0,0870

0,0870 à 0,0879 à 0,0900 pour les trois images respectivement. Cependant nous pouvons observer que l'erreur est toujours plus grande aux frontières et puisque le nombre de frontières augmente lorsque les régions deviennent de plus en plus petites alors il est normale que l'erreur absolue moyenne croisse.

Comme nous pouvons le constater dans le tableau 3.4.2, c'est la fonction de perte quadratique qui donne les meilleurs résultats en terme d'erreur absolue moyenne pour chaque région et pour chaque image également. En deuxième et troisième place viennent la fonction de perte 0-1 sur un intervalle et la fonction de perte de Huber modifiée et finalement le MAP. Il est important de se rappeler que ces résultats sont obtenus avec une tolérance de 5% pour les fonctions de perte 0-1 sur un intervalle et de Huber modifiée.

En ce qui concerne la moyenne des écarts types par régions, les résultats sont sensiblement les mêmes sauf qu'il arrive pour certaines régions que la fonction de perte de Huber modifiée donne des résultats plus dispersés que pour le MAP (voir tableau 3.4.3).

TABLEAU 3.4.3. Écart type moyen par région pour les simulations avec un bruit de 20%

Image	Fonction de perte	Région 1	Région 2	Région 3	Région 4
Image 1	Huber modifiée	0,1053	0,1792	0,1807	0,1034
	0-1 sur un intervalle	0,0959	0,1617	0,1630	0,0950
	Quadratique	0,0851	0,1479	0,1495	0,0851
	MAP	0,1050	0,1823	0,1826	0,1029
Image 2	Huber modifiée	0,1074	0,1846	0,1847	0,1042
	0-1 sur un intervalle	0,0978	0,1675	0,1669	0,0956
	Quadratique	0,0866	0,1522	0,1536	0,0857
	MAP	0,1068	0,1855	0,1859	0,1051
Image 3	Huber modifiée	0,1097	0,1926	0,1923	0,1086
	0-1 sur un intervalle	0,1011	0,1756	0,1756	0,0992
	Quadratique	0,0889	0,1590	0,1598	0,0876
	MAP	0,1095	0,1942	0,1948	0,1077

Si nous observons les résultats montrés en annexe B pour les autres niveaux de bruit, nous trouvons des résultats sensiblement identiques. En effet, nous pouvons voir dans les tableaux B.1 et B.3 que pour un niveau de bruit de 5% et 10% l'ordonnancement des fonctions de perte est le même que pour un bruit de 20%. Comme pour le niveau de bruit de 20%, nous obtenons toujours une plus grande erreur absolue moyenne pour les régions grises, soit les régions 2 et 3.

En ce qui concerne la dispersion au travers des régions pour les niveaux de bruit plus faible, nous pouvons observer certaines différences dans l'ordonnancement des fonctions de perte par rapport à celui obtenu pour l'erreur absolue. En effet l'ordre de préférence des fonctions de perte dépend de la couleur estimée et de l'image. Prenons le tableau B.4 par exemple, nous remarquons que pour un bruit de 10% la fonction de perte privilégiée pour l'image 1 et la région 1 est la fonction de perte 0-1 sur un intervalle avec un écart type de 0,0034 et vient ensuite la fonction de perte de Huber modifiée avec un écart type de 0,0039. Pour la région 2, c'est la fonction de perte quadratique avec un écart type de 0,0162

TABLEAU 3.4.4. Erreur absolue moyenne et écart type moyen pour l'image 2×2 avec un bruit gaussien de 20% pour différent niveaux de tolérance

Statistique	Fonction de perte	Région 1	Région 2	Région 3	Région 4
Erreur absolue moyenne	Huber avec 1% tol.	0,1110	0,1045	0,1051	0,1020
	0-1 avec 1% tol.	0,1060	0,1030	0,1035	0,0969
	Huber avec 10% tol.	0,0788	0,0963	0,0968	0,0708
	0-1 avec 10% tol.	0,0209	0,0774	0,0776	0,0304
Écart type moyen	Huber avec 1% tol.	0,1080	0,1807	0,1820	0,1070
	0-1 avec 1% tol.	0,1070	0,1784	0,1794	0,1059
	Huber avec 10% tol.	0,0996	0,1689	0,1699	0,0985
	0-1 avec 10% tol.	0,0831	0,1400	0,1409	0,0818

qui donne la plus petite dispersion. Dans le cas de l'image 3, nous observons que la fonction de perte 0-1 sur un intervalle montre toujours la plus petite dispersion mais c'est maintenant la fonction quadratique qui tient la deuxième position. En somme, en ce qui a trait à l'écart type, il n'y a pas de fonction de perte qui donne constamment de meilleurs résultats.

Suite aux résultats présentés jusqu'à présent, nous pouvons affirmer que pour l'erreur absolue, la fonction de perte quadratique est celle qui donne les meilleurs résultats pour les trois images synthétiques utilisées et les trois niveaux de bruit lorsqu'une tolérance de 5% est utilisée pour les fonction de perte de Huber modifiée et 0-1 sur un intervalle. Nous allons maintenant regarder les résultats obtenus lorsque nous utilisons d'autres niveaux de tolérance pour ces deux fonctions de perte, soit 1% et 10%. Pour cette simulation nous nous limitons à l'image 2×2 avec un bruit gaussien de 20%. Les résultats graphiques sont présentés à la figure 3.4.3 et les moyennes par régions dans le tableau 3.4.4.

Analysons tout d'abord les résultats obtenus pour la fonction de perte de Huber modifiée. Rappelons que cette fonction de perte est une généralisation

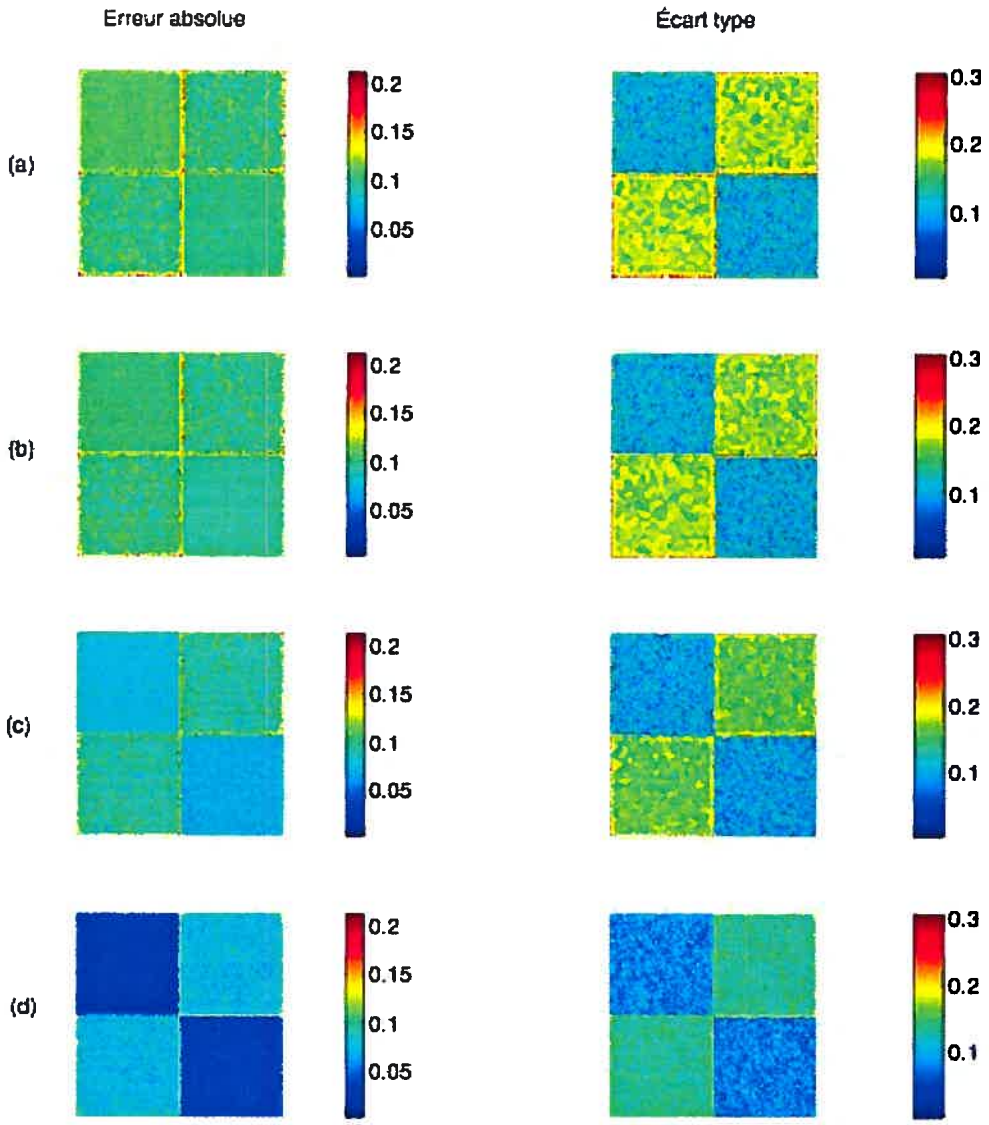

FIGURE 3.4.3. Résulats avec un bruit gaussien de 20% pour la fonction de perte de Huber modifiée avec une tolérance de (a) 1% (c) 10% et la fonction de perte 0-1 sur un intervalle avec tolérance de (b) 1% (d) 10%.

de la fonction de perte quadratique et du MAP. Dans le cas de l'erreur absolue moyenne, nous comparons avec les résultats présentés dans le tableau 3.4.2. Nous voyons que l'erreur absolue moyenne est toujours plus faible lorsque nous utilisons

une tolérance de plus en plus grande et ce pour chaque région. En effet, prenons par exemple les résultats de la région 1, pour une tolérance de 1% nous obtenons une erreur absolue moyenne de 0,1110 contre 0,0958 et 0,0788 pour une tolérance de 5% et 10% respectivement. Ces résultats sont en ligne avec la théorie puisque nous avons remarqué que l'erreur absolue moyenne obtenue avec la fonction de perte quadratique est la plus faible et la fonction de perte de Huber modifiée généralise la fonction de perte quadratique lorsque la tolérance est grande.

Dans le cas de la fonction de perte 0-1 sur un intervalle, lorsque la tolérance est faible cette dernière généralise le MAP. Au niveau des résultats de simulations, nous remarquons que les résultats sont toujours moins bons lorsque la tolérance est plus faible. De plus, nous voyons que pour une tolérance de 10%, les résultats obtenus avec la fonction de perte 0-1 sur un intervalle sont meilleurs en ce qui a trait à l'erreur absolue moyenne que la fonction de perte quadratique pour les régions 1 et 4 et très semblables pour les deux autres régions. Ce résultat est très intéressant puisqu'il fait ressortir l'intérêt de considérer une tolérance sur la couleur estimée.

Dans le cas de la dispersion moyenne à l'intérieur des régions, nous pouvons faire sensiblement le même constat, c'est-à-dire que les résultats obtenus sont toujours moins dispersés pour une plus grande tolérance autant pour la fonction de perte de Huber modifiée que pour la fonction de perte 0-1 sur un intervalle. De plus, dans le cas de l'écart type, nous remarquons que les résultats obtenus avec la fonction de perte 0-1 sur un intervalle avec une tolérance de 10% sont uniformément les meilleurs en terme de dispersion moyenne à travers les régions.

À la lumière des résultats présentés dans cette section, nous pouvons affirmer qu'il y a un intérêt d'inclure un facteur de tolérance dans les fonctions de perte et que selon la valeur de ce facteur, les résultats peuvent varier.

3.4.2. Impact du choix de la fonction de perte en fonction du nombre de régions

Dans cette section nous voulons évaluer l'impact du nombre de régions à estimer dans le modèle selon la fonction de perte qui est utilisée. Les simulations

TABLEAU 3.4.5. Erreur absolue moyenne et écart type moyen pour l'image 2×2 avec un bruit gaussien de 20% lorsque l'algorithme est lancé avec 3 régions

Statistique	Fonction de perte	Région 1	Région 2	Région 3	Région 4
Erreur absolue moyenne	Huber modifiée	0,0893	0,1671	0,1676	0,0802
	0-1 sur un intervalle	0,0613	0,1621	0,1625	0,0518
	Quadratique	0,0276	0,1530	0,1556	0,0318
	MAP	0,0860	0,1721	0,1616	0,0767
Écart type moyen	Huber modifiée	0,0814	0,1687	0,1698	0,0803
	0-1 sur un intervalle	0,0750	0,1560	0,1571	0,0746
	Quadratique	0,0648	0,1357	0,1384	0,0658
	MAP	0,0827	0,1747	0,1724	0,0793

sont encore basées sur des images synthétiques et puisque nous avons remarqué à la section précédente que la composition des images n'a pas un grand effet, nous utilisons seulement l'image synthétique 2×2 avec un bruit gaussien de 20%.

Nous allons lancer l'algorithme 100 fois comme à la simulation précédente avec 3 et 5 régions à estimer. Nous pouvons voir dans les tableaux 3.4.5 et 3.4.6 les résultats lorsque l'algorithme est lancé avec 3 ou 5 régions. Nous pouvons voir en comparant les résultats du tableau 3.4.2 lorsque l'algorithme fut lancé avec 4 régions avec ceux du tableau 3.4.5 pour trois régions que l'erreur absolue moyenne pour les régions 1 et 4, c'est-à-dire le noir et le blanc, est du même ordre pour toutes les fonctions de perte. C'est pour les régions 2 et 3 que l'erreur moyenne est beaucoup plus grande, nous voyons ainsi qu'en moyenne l'algorithme a estimé ces deux régions à partir d'une même valeur centrale. Nous pouvons observer cette même tendance à la figure 3.4.4.

De plus, il est intéressant de remarquer que la fonction de perte ne semble pas avoir d'impact lorsque moins de régions sont utilisées dans l'algorithme. En effet, nous pouvons voir que le même ordonnancement des régions est obtenu que lorsque 4 régions furent utilisées.

TABLEAU 3.4.6. Erreur absolue moyenne et écart type moyen pour l'image 2 × 2 avec un bruit gaussien de 20% lorsque l'algorithme est lancé avec 5 régions

Statistique	Fonction de perte	Région 1	Région 2	Région 3	Région 4
Erreur absolue moyenne	Huber modifiée	0,0913	0,1045	0,1053	0,0824
	0-1 sur un intervalle	0,0623	0,0711	0,0730	0,0527
	Quadratique	0,0282	0,0726	0,0729	0,0324
	MAP	0,0876	0,1037	0,1077	0,0790
Écart type moyen	Huber modifiée	0,0853	0,1740	0,1752	0,0836
	0-1 sur un intervalle	0,0729	0,1272	0,1282	0,0712
	Quadratique	0,0679	0,1410	0,1406	0,0673
	MAP	0,0834	0,1676	0,1731	0,0814

Par contre, lorsque la simulation est lancée avec 5 régions, nous observons que les résultats sont légèrement supérieurs à ceux obtenus avec 4 régions. Les résultats graphiques sont présentés à la figure 3.4.5. En effet, prenons par exemple la fonction de perte quadratique, dans le cas où nous avons utilisé 4 régions, l'erreur moyenne pour les quatre régions est respectivement 0,0316, 0,0745, 0,0766 et 0,0358 et les résultats avec 5 régions pour la même fonction de perte sont respectivement 0,0282, 0,0726, 0,0729 et 0,0324 comme nous pouvons le voir au tableau 3.4.6.

Ces résultats ne sont cependant pas surprenants, en effet lorsque nous utilisons moins de régions que le nombre réel connu, il est bien évident qu'au moins une région aura un taux d'erreur beaucoup plus élevé. À l'inverse, lorsqu'un plus grand nombre de régions est utilisé, au pire deux régions estimerons la même couleur. Il est cependant intéressant de remarquer que l'algorithme estime toujours très bien les frontières lorsque moins de régions sont utilisées. En effet, nous pouvons

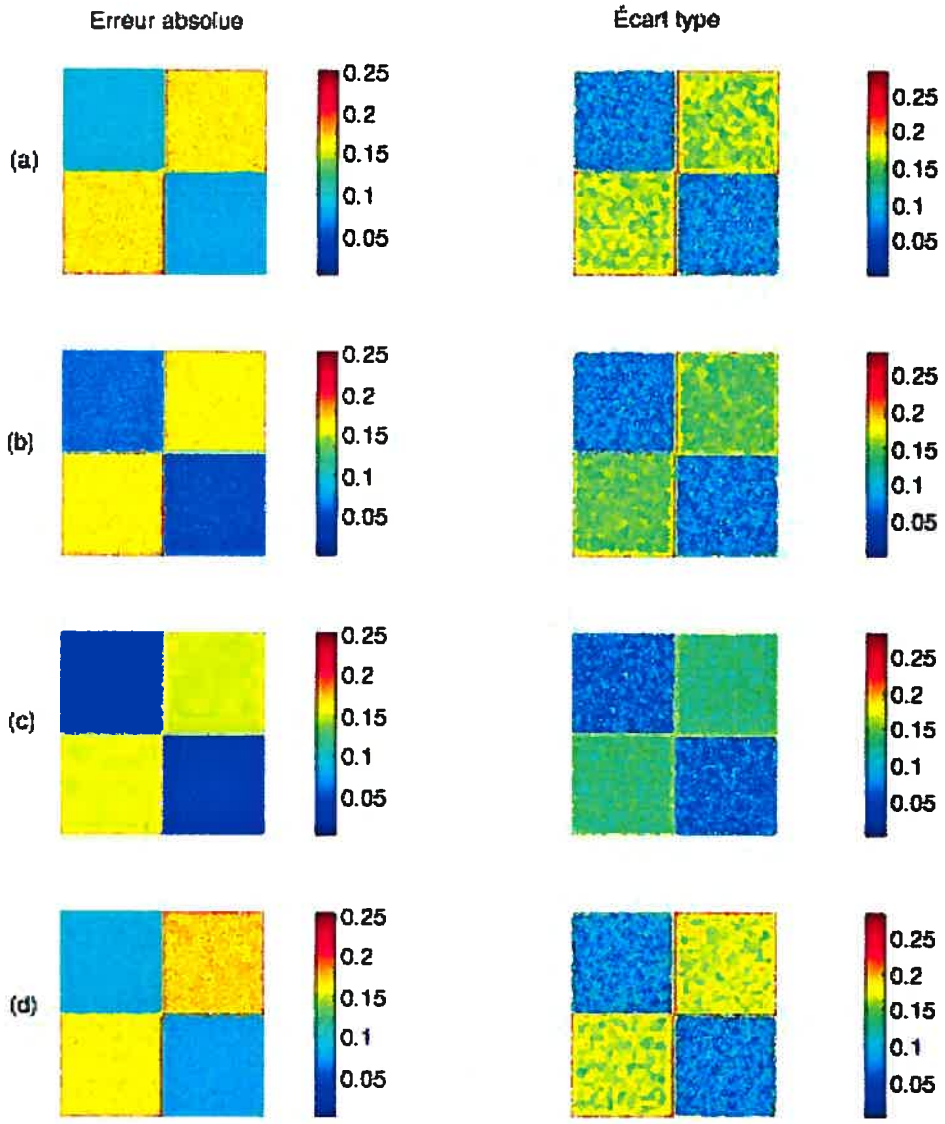

FIGURE 3.4.4. Résultats des simulations avec un bruit gaussien de 20% avec 3 régions pour la fonction de perte (a) de Huber modifiée (b) 0-1 sur un intervalle (c) quadratique (d) MAP.

remarquer à la figure 3.4.4 que l'erreur absolue et l'écart type sont très bien définis autour des régions réelles.

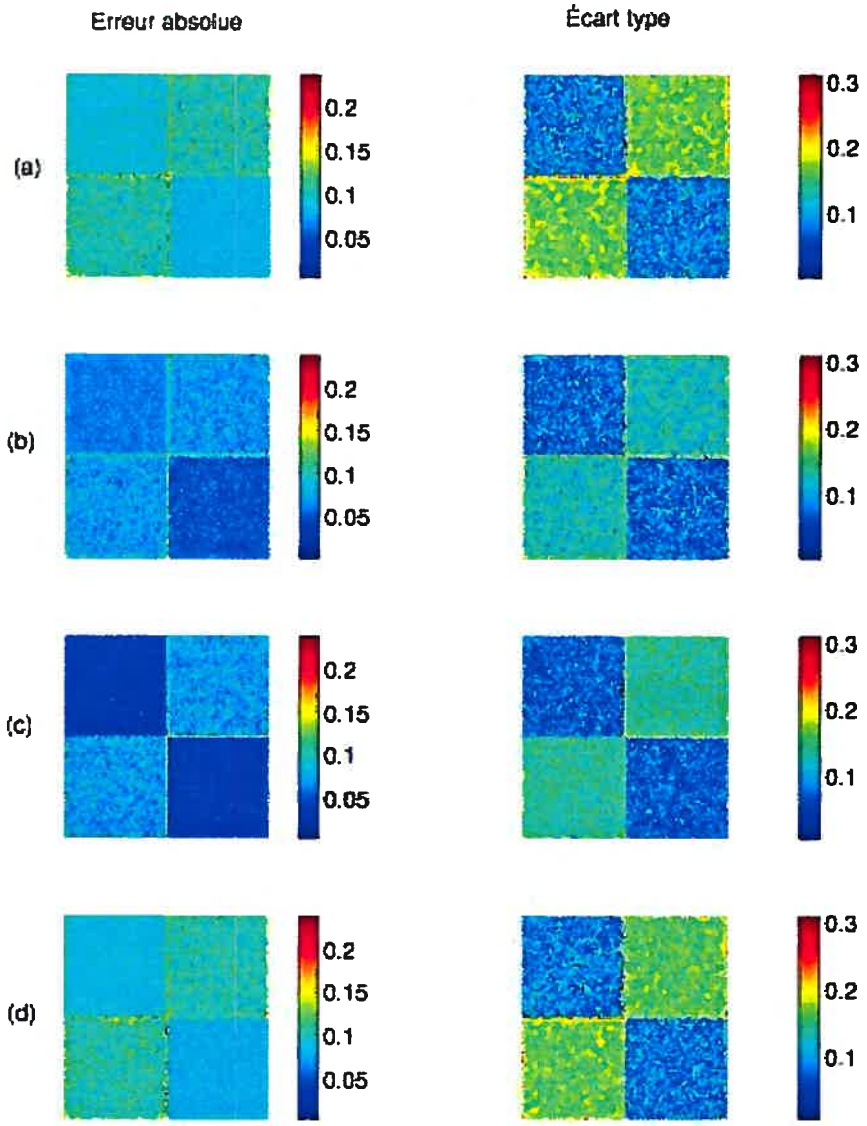

FIGURE 3.4.5. Résultats des simulations avec un bruit gaussien de 20% avec 5 régions pour la fonction de perte (a) de Huber modifiée (b) 0-1 sur un intervalle (c) quadratique (d) MAP.

FIGURE 3.5.1. Image naturelle originale utilisée pour comparer les fonctions de perte

3.5. ILLUSTRATION DE LA MÉTHODE PROPOSÉE SUR DES IMAGES NATURELLES

Dans cette section nous présentons des résultats de segmentation d'images naturelles pour illustrer la méthode proposée au chapitre 2. Dans un premier temps nous montrons les résultats de la segmentation d'une image à l'aide des quatre fonctions de perte proposées. Ensuite, pour une fonction de perte choisie nous montrons l'image estimée à plusieurs étapes de l'algorithme d'analyse en grappe pour illustrer la dégradation de celle-ci. Finalement nous concluons le chapitre en montrant d'autres exemples d'images naturelles segmentées à l'aide de notre approche.

La figure 3.5.1 montre l'image qui est utilisée pour comparer les fonctions de perte. Cette dernière est de taille 200×300 pixels. Pour chacune des quatre fonctions de perte, nous mettons en oeuvre la méthode avec 10 segments initiaux par composantes. Nous présentons ensuite une série de quatre figures, chacune d'elles montre les trois composantes L, a et b originales et la segmentation de ces dernières selon les quatre fonctions de perte. De plus, nous affichons également la représentation de la superposition des trois composantes segmentées avant la fusion de ces dernières par l'algorithme d'analyse en grappe. Les résultats pour les quatre fonctions de perte sont présentés aux figures 3.5.2, 3.5.3, 3.5.4 et 3.5.5. En

regardant minutieusement ces quatre figures, nous remarquons qu'il y a certaines différences au niveau de la segmentation et des couleurs estimées mais elles sont quasi imperceptibles à l'oeil nu. De plus, il nous est impossible de déterminer quelle fonction de perte donne les meilleurs résultats car la segmentation réelle est inconnue. Ainsi, il semble que pour une image naturelle l'impact du choix de la fonction de perte est minime.

Nous voulons également illustrer la dégradation de l'image lorsque que nous fusionnons de plus en plus de régions par l'analyse en grappe. Nous pouvons voir à la figure 3.5.6 que le fait de réduire le nombre de régions n'a pas beaucoup d'effet jusqu'à un certains nombre et ensuite la dégradation est très rapide. En effet les images estimées avec 20, 15, 10 et même 8 régions sont vraiment très semblables et ensuite la dégradation est marquante lorsque nous baissons à 6 classes.

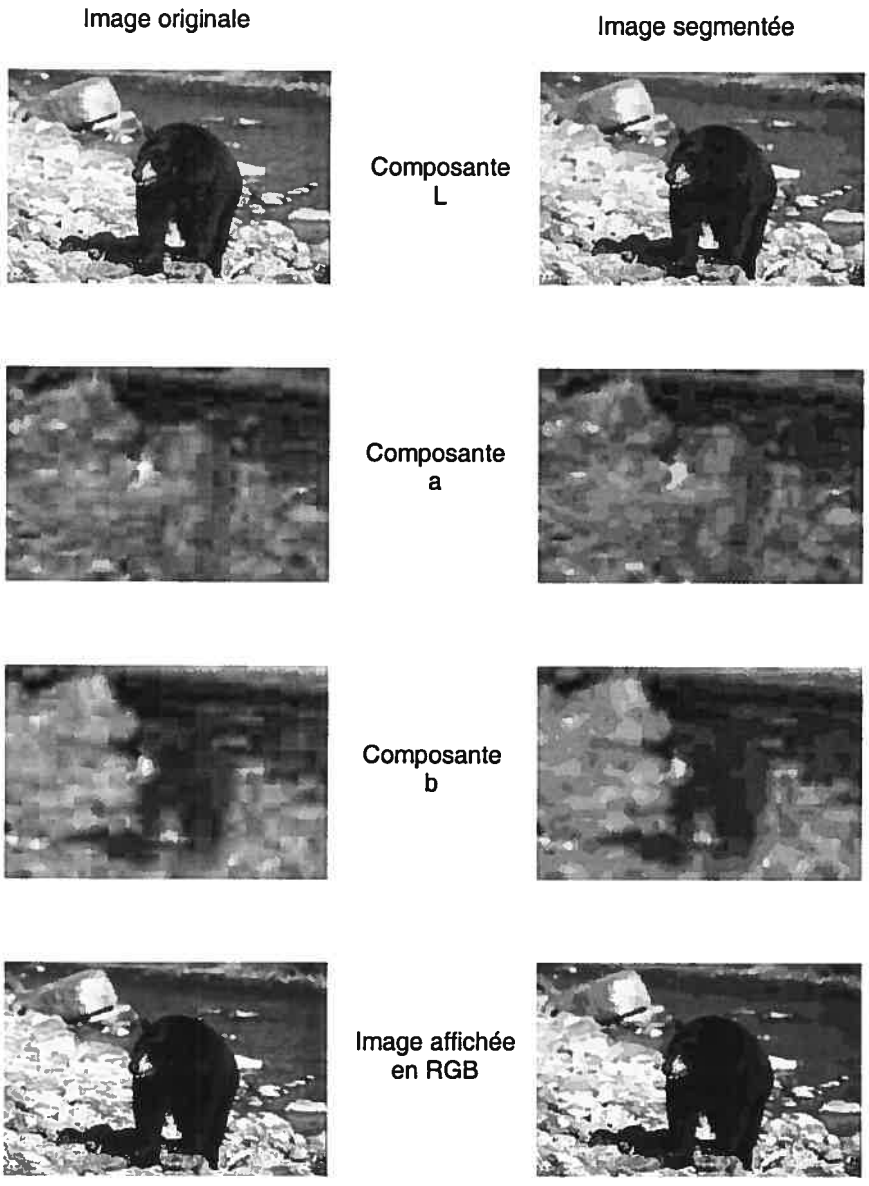

FIGURE 3.5.2. Représentation des trois composantes segmentées à partir de la fonction de perte de Huber avec 10 régions pour chaque composante

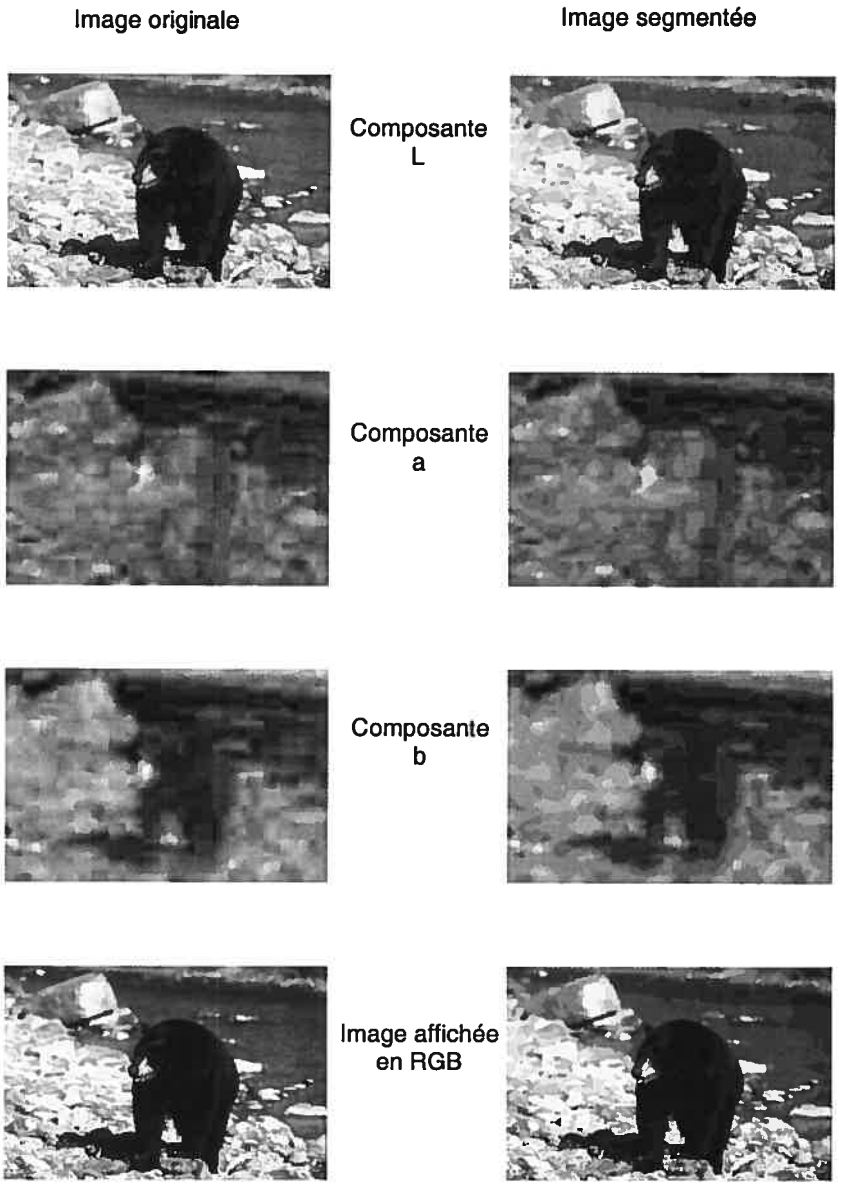

FIGURE 3.5.3. Représentation des trois composantes segmentées à partir de la fonction de perte 0-1 sur un intervalle avec 10 régions pour chaque composante

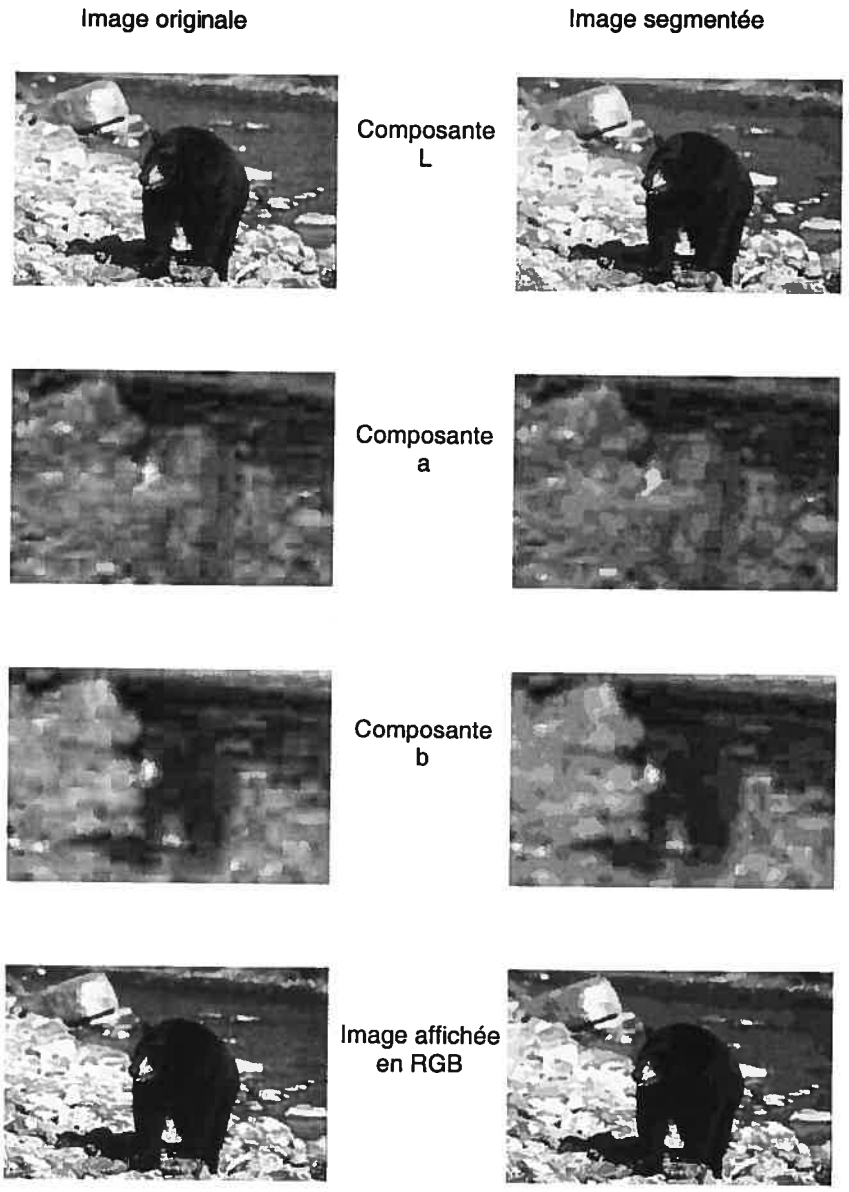

FIGURE 3.5.4. Représentation des trois composantes segmentées à partir de la fonction de perte quadratique avec 10 régions pour chaque composante

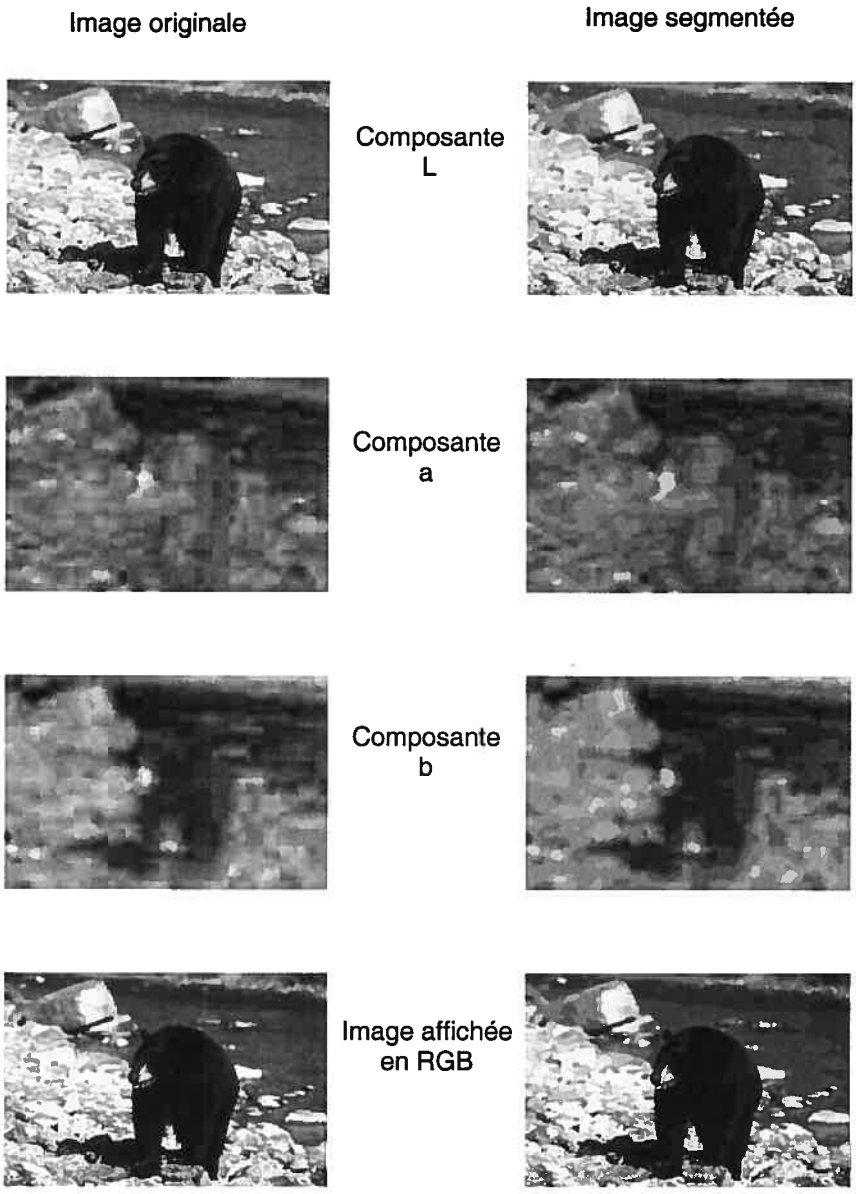

FIGURE 3.5.5. Représentation des trois composantes segmentées à partir du MAP avec 10 régions pour chaque composante

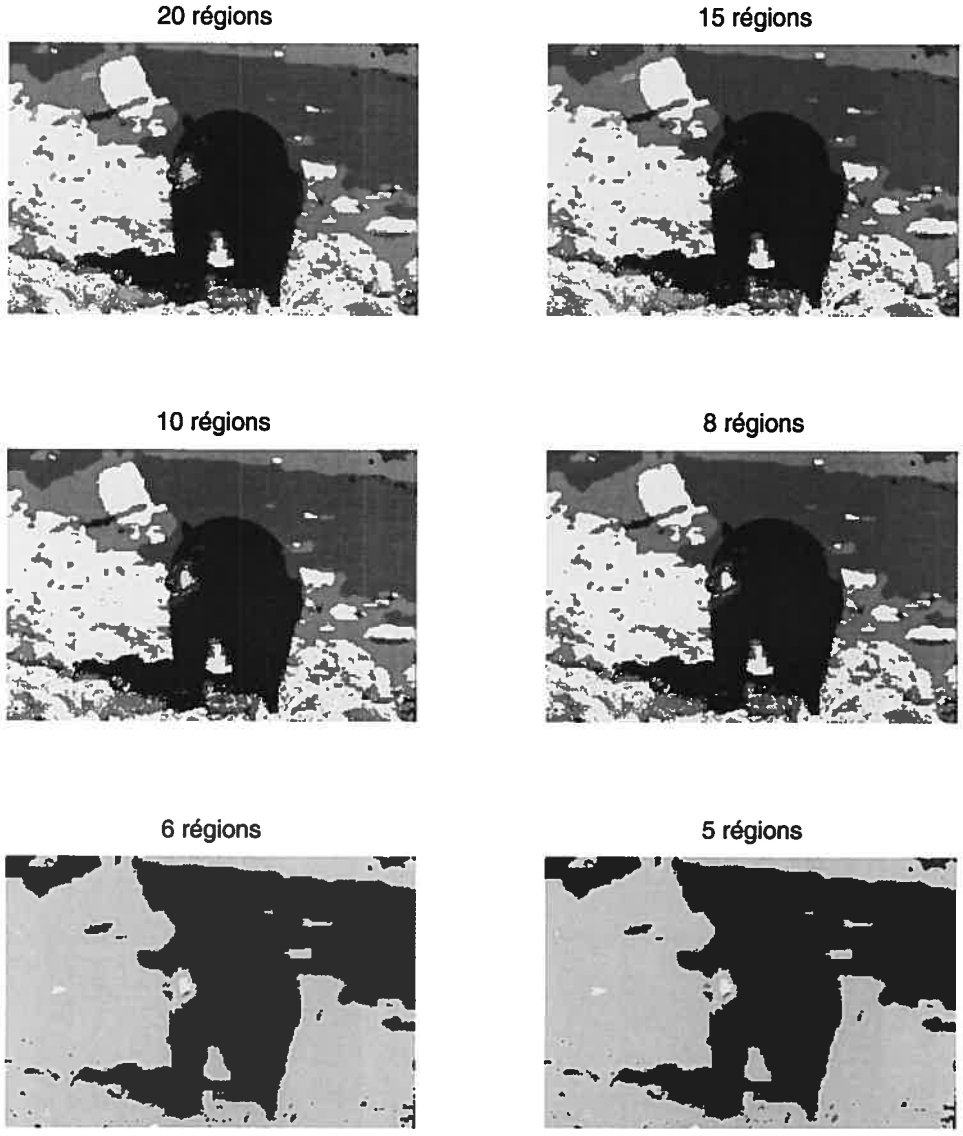

FIGURE 3.5.6. Segmentation de l'image avec différents nombres de régions pour l'analyse en grappe à l'aide de la fonction de perte de Huber modifiée

3.5.1. Autres exemples de segmentation d'images naturelles

Dans cette section nous présentons quelques exemples de segmentation d'images naturelles à partir de la méthode proposée au chapitre 2. Dans chacun de ces exemples, la segmentation est effectuée à partir de la fonction de perte quadratique. Le choix de cette fonction est basé principalement sur le fait que pour les images synthétiques les résultats obtenus à partir de la fonction de perte quadratique étaient supérieurs en terme d'erreur absolue ou de dispersion de la couleur estimée et également parce que le temps de calcul est moindre avec cette dernière qu'avec les autres fonctions de perte. Les images sont présentées à la figure 3.5.7. Dans le cas de la première image avec le poisson, cette dernière est estimée avec 10 segments par composantes et nous appliquons l'algorithme d'analyse en grappe pour obtenir 10 segments finaux également. Pour l'image avec la femme, 6 segments furent utilisés par composantes et nous fusionnons les régions résultant de la superposition des composantes pour obtenir 7 régions finales, pour l'image avec la maison, 6 régions sont utilisés dans les deux cas et finalement, pour la dernière image avec les zèbres 8 régions ont été utilisées dans les deux cas.

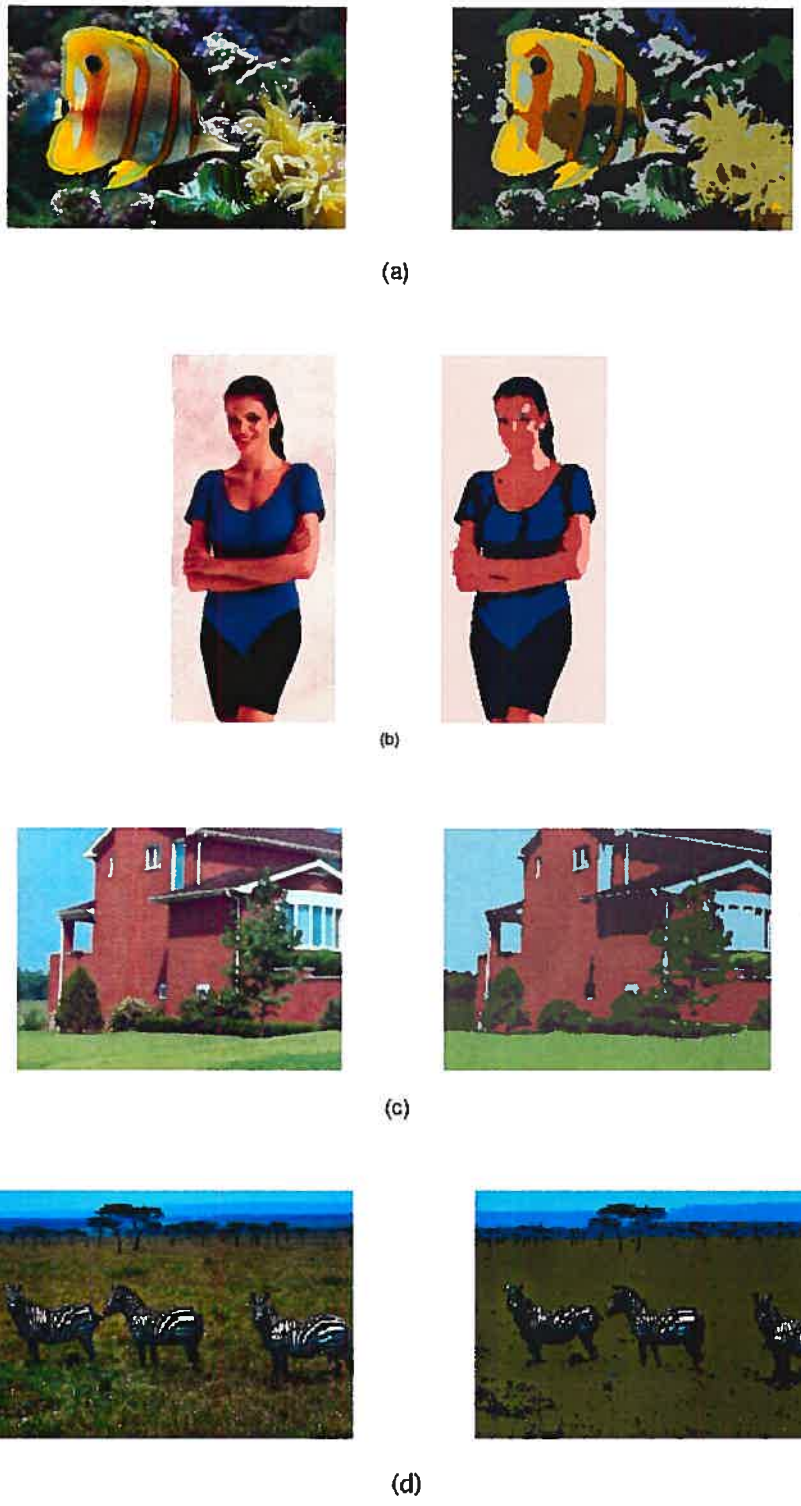

FIGURE 3.5.7. Comparaison de l'image originale avec l'image segmentée avec (a) 10 régions (b) 7 régions (c) 6 régions (d) 8 régions.

3.6. Conclusion de chapitre

Dans ce dernier chapitre nous avons présenté des résultats pratiques de la mise en oeuvre de la méthode de segmentation proposée au chapitre 2 pour évaluer l'impact de la fonction de perte sur la segmentation d'une image selon les couleurs. Nous avons rapporté des résultats de simulations basés sur des images synthétiques et nous avons montré des résultats de segmentations d'images naturelles. Dans le cas des simulations sur des images synthétiques, nous avons trouvé que la fonction de perte quadratique est celle qui donne les meilleurs résultats au sens des deux critères observés, c'est-à-dire l'erreur absolue et l'écart type par rapport à la vraie couleur.

Cependant, dans le cas des images naturelles, nous avons remarqué que le choix de la fonction de perte n'a pas réellement d'impact perceptible à l'oeil nu. Ainsi, nous avons présenté des exemples de segmentation d'images naturelles seulement pour la fonction de perte quadratique car cette dernière est celle qui est la plus rapide à calculer et elle s'est montrée la plus efficace lors des simulations sur images synthétiques.

CONCLUSION

Dans ce mémoire, nous avons répondu à la question d'un arbitre concernant l'impact du choix de la fonction de perte pour la segmentation d'images dans un modèle de couleurs. Au premier chapitre, nous avons mis en contexte l'aspect statistique de la modélisation d'une image en décrivant brièvement la théorie permettant de traiter mathématiquement une image comme un graphe. Nous avons vu les notions de champ aléatoire de Markov caché, de clique et de voisinage. Ensuite nous avons présenté l'approche proposée dans Destrempes, Mignotte et Angers (2005) sur laquelle notre modèle est grandement basé. Nous avons terminé ce chapitre en présentant un survol des méthodes numériques utilisées tout au long de ce mémoire pour estimer le modèle proposé.

Au deuxième chapitre, nous avons présenté la théorie de la décision bayésienne en décrivant sommairement les concepts de fonction de perte et d'estimateur bayésien. Nous avons proposé quatre fonctions de perte que nous avons utilisées pour estimer le modèle. Ensuite, nous avons décrit le modèle statistique que nous proposons pour segmenter une image, qui est basé sur celui défini dans Destrempes *et al.* (2005). Nous avons également suggéré le critère d'arrêt de Gelman et Rubin (1992) pour évaluer la convergence des méthodes MCMC.

Finalement, nous entamons le dernier chapitre en discutant de certains aspects pratiques de notre modèle et de l'estimation de ce dernier. Nous avons discuté de la mise en application du critère d'arrêt pour les méthodes MCMC et avons proposé un nouveau critère pour l'évaluation de la convergence de l'algorithme EM basé sur la densité *a posteriori*. En deuxième partie, nous avons présenté des résultats de simulations basés sur des images synthétiques, qui nous permettent d'évaluer l'impact du choix de la fonction de perte sur la segmentation. Ensuite,

nous avons montré des résultats de segmentation d'une image naturelle selon les quatre fonctions de perte proposées. Nous avons terminé le chapitre en montrant quelques exemples de segmentation d'images naturelles.

En somme, les simulations sur des images synthétiques nous ont permis de remarquer que tout en étant minime, le choix de la fonction de perte a un impact sur la segmentation. Basé sur les résultats obtenus, nous avons vu que la fonction de perte quadratique est celle qui semble donner les meilleurs résultats dans presque toutes les situations. En effet, seulement la fonction de perte 0-1 sur un intervalle avec une tolérance de 10% a donné de meilleurs résultats au sens de l'erreur absolue et de l'écart type par rapport à la vraie couleur de chaque pixel. Ceci nous pousse à se questionner sur le choix de ce facteur de tolérance et nous jugeons qu'il serait intéressant dans le futur de pousser un peu plus loin les recherches dans ce sens.

Dans le cas des images naturelles, l'impact du choix de la fonction de perte est beaucoup plus difficile a évaluer considérant le fait que la segmentation réelle sous-jacente à l'image est inconnue. Cependant, basé sur les résultats obtenus, nous devons admettre que les images segmentées à partir des différentes fonctions de perte n'affichent pas de différence perceptible à l'oeil nu assez substantielle pour poser un jugement sur le choix de la meilleure fonction de perte. Ainsi, pour des raisons de simplicité de l'estimateur résultant, de popularité de ce dernier et de la qualité des résultats basés sur des images synthétiques, nous avons proposé d'utiliser la fonction de perte quadratique pour les autres exemples de segmentation d'images naturelles.

Pour terminer, une des grandes différences dans le modèle que nous proposons par rapport à celui de Destrempes *et al.* (2005) est qu'il n'est pas totalement indépendant de l'utilisateur, ou non supervisé pour utiliser la terminologie du domaine de traitement d'images. En effet, il y a un seul paramètre que nous n'estimons pas et ce dernier est le nombre de classes. Nous suggérons donc de poursuivre les recherches pour inclure ce paramètre dans l'estimation de notre modèle et ainsi proposer une méthode non supervisée. Des pistes de recherches

possibles pour estimer le nombre de classes sont l'intégration de critères de sélection de modèle comme le critère AIC ou BIC dans l'algorithme d'analyse en grappe ou bien l'intégration du paramètre directement dans l'algorithme EM.

Annexe A

TECHNIQUE D'ANALYSE EN GRAPPE PAR LE CENTROÏDE

Comme nous l'avons mentionné, l'analyse en grappe également appelée classification ou *cluster analysis* en anglais est un champ de recherche très large et nous présentons ici qu'un bref survol. Les méthodes les plus utilisées et souvent les plus simples à mettre en oeuvre sont celles appartenant à la famille des techniques hiérarchiques. La technique par arborescence, la technique du voisin le plus près ou du voisin le plus éloigné font partie de cette famille de méthodes hiérarchiques, et elles sont dites d'agglomération. La terminologie technique d'agglomération fait référence au fait qu'au départ toutes les observations sont considérées comme étant un groupe distinct et peu à peu ces dernières sont regroupées selon un certain algorithme. Nous présentons ici un algorithme basé sur une notion de distance, plus précisément la distance euclidienne. Pour plus de détails sur les méthodes d'analyse en grappe, nous référons à Everitt (1974).

Commençons tout d'abord par définir la distance euclidienne.

Définition A.0.1. *La distance euclidienne entre deux éléments* $\mathbf{x}$ *et* $\mathbf{y}$ *tous deux dans* $\mathbb{R}^p$ *est définie comme*

$$d(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{p} (x_i - y_i)^2 \right)^{1/2}.$$

Il existe plusieurs autres mesures de distance qui pourraient être utilisées, mais la distance euclidienne est la plus usuelle d'entre elles.

La technique d'analyse en grappe par la méthode du centroïde consiste à construire une matrice qui représente toutes les distances deux-à-deux. Dans notre cas, les objets de notre analyse en grappe sont les triplets de couleurs estimées des régions résultantes du produit cartésien des trois segmentations par composantes, notés

$$(\tilde{\mathbf{y}}_1, \ldots, \tilde{\mathbf{y}}_n) = \left\{ \begin{pmatrix} \tilde{y}_{1,1} \\ \tilde{y}_{2,1} \\ \tilde{y}_{3,1} \end{pmatrix}, \ldots, \begin{pmatrix} \tilde{y}_{1,n} \\ \tilde{y}_{2,n} \\ \tilde{y}_{3,n} \end{pmatrix} \right\},$$

en supposant que nous obtenons n régions suite au produit cartésien. Ainsi, nous construisons une matrice des distances deux-à-deux des triplets $\tilde{\mathbf{y}}_i$

$$D = \begin{pmatrix} d(\tilde{\mathbf{y}}_1, \tilde{\mathbf{y}}_1) & \ldots & d(\tilde{\mathbf{y}}_1, \tilde{\mathbf{y}}_n) \\ & \ddots & \vdots \\ 0 & & d(\tilde{\mathbf{y}}_n, \tilde{\mathbf{y}}_n) \end{pmatrix}.$$

La technique d'analyse en grappe par le centroïde se résume en trois étapes.

(1) Regrouper les deux régions dont la distance euclidienne est la plus petite,

(2) Calculer la couleur résultante de la fusion des deux régions en calculant la moyenne pondérée des valeurs composante par composante,

(3) Recalculer la matrice des distances D avec les objets restants.

Nous continuons ainsi jusqu'à ce que nous ayons atteint le nombre de régions voulu.

Nous ajoutons une dernière étape à l'algorithme qui ne concerne pas du tout l'analyse en grappe par la méthode du centroïde. Nous sommes quelque peu inconfortable avec le fait de conserver des paramètres issus d'une simple moyenne entre les valeurs des régions regroupées. Ainsi, avec la segmentation finale, qui en passant est maintenant la même sur chacune des composantes, nous allons réestimer les paramètres de notre modèle composante par composante comme nous l'avons vu au chapitre 2. Puisque nous gardons la segmentation finale fixe, nous n'avons pas à estimer le modèle *a priori* sur la segmentation et avons seulement qu'à estimer par la densité prédictive la nouvelle couleur pour chacune des classes. De cette manière, nous obtenons une couleur estimée qui est réellement basée sur

notre modèle de vraisemblance et les données observées, et non pas seulement une moyenne des couleurs estimées avant regroupement des régions.

Annexe B

RÉSULTATS GRAPHIQUES DE L'ERREUR ABSOLUE ET DE L'ÉCART TYPE

TABLEAU B.1. Erreur absolue moyenne par région pour les simulations avec un bruit de 5%

Image	Fonction de perte	Région 1	Région 2	Région 3	Région 4
Image 1	Huber modifiée	0,0333	0,0026	0,0028	0,0357
	0-1 sur un intervalle	0,0282	0,0022	0,0023	0,0294
	Quadratique	0,0061	0,0010	0,0011	0,0063
	MAP	0,0389	0,0080	0,0085	0,0324
Image 2	Huber modifiée	0,0332	0,0025	0,0028	0,0358
	0-1 sur un intervalle	0,0281	0,0021	0,0024	0,0294
	Quadratique	0,0061	0,0010	0,0011	0,0063
	MAP	0,0393	0,0080	0,0086	0,0318
Image 3	Huber modifiée	0,0335	0,0026	0,0028	0,0357
	0-1 sur un intervalle	0,0284	0,0022	0,0024	0,0294
	Quadratique	0,0061	0,0011	0,0011	0,0063
	MAP	0,0400	0,0077	0,0083	0,0318

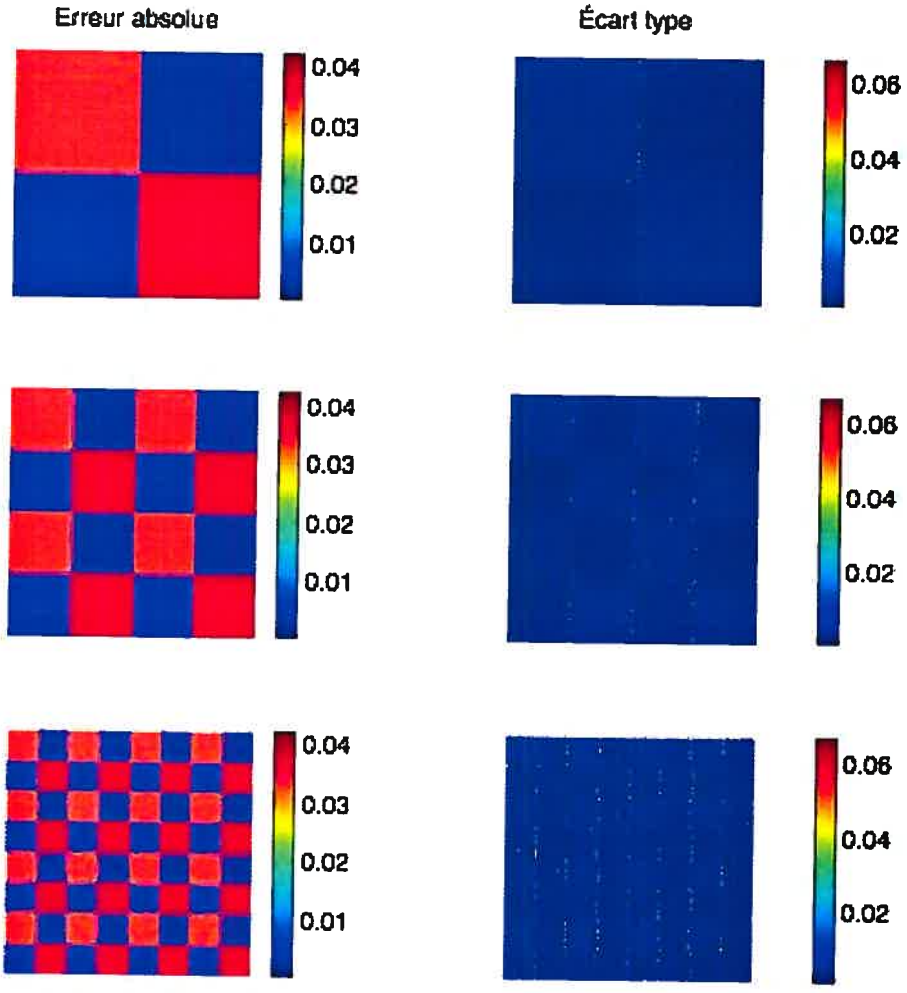

FIGURE B.1. Résultats des simulations avec un bruit gaussien de 5% pour la fonction de perte de Huber modifiée

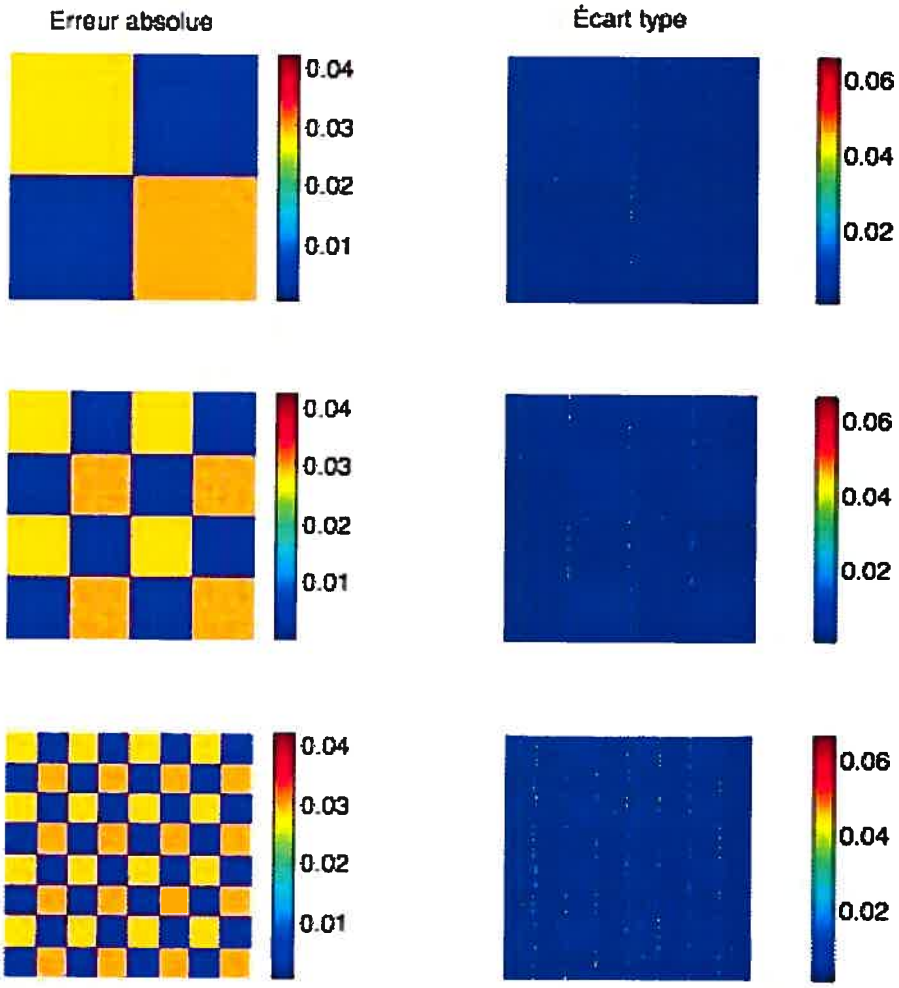

FIGURE B.2. Résultats des simulations avec un bruit gaussien de 5% pour la fonction de perte 0-1 sur un intervalle

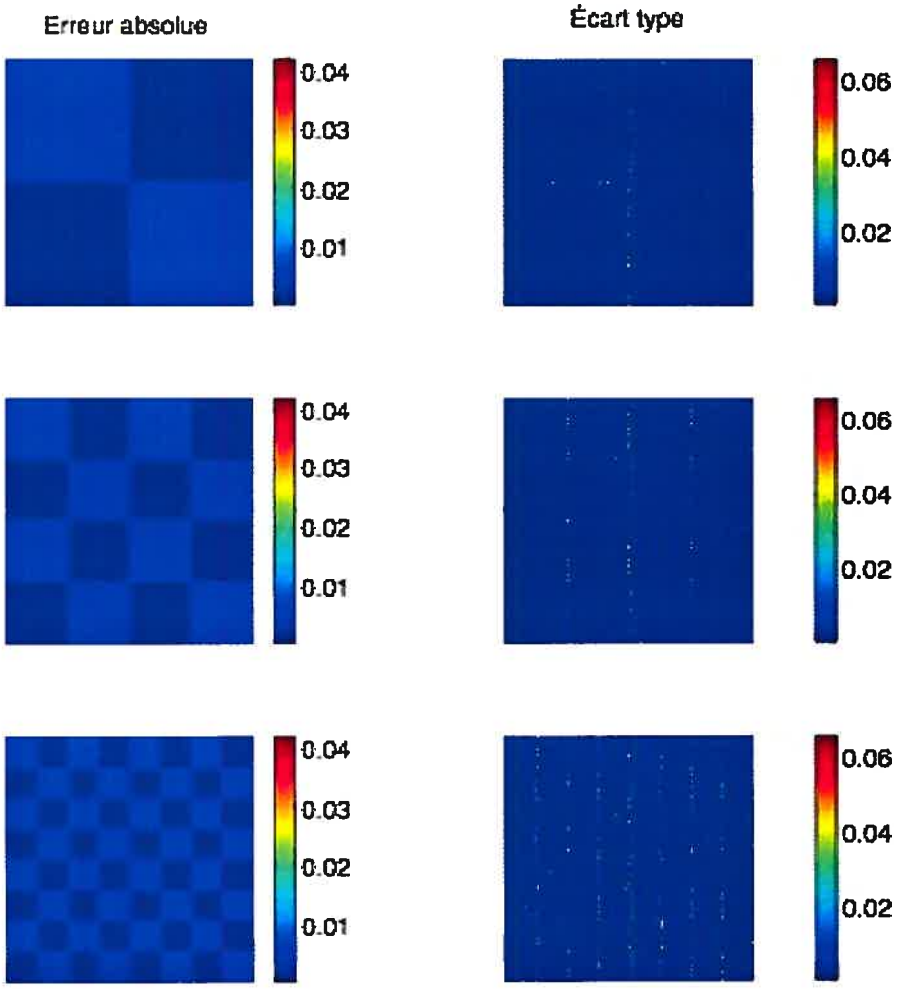

FIGURE B.3. Résultats des simulations avec un bruit gaussien de 5% pour la fonction de perte quadratique

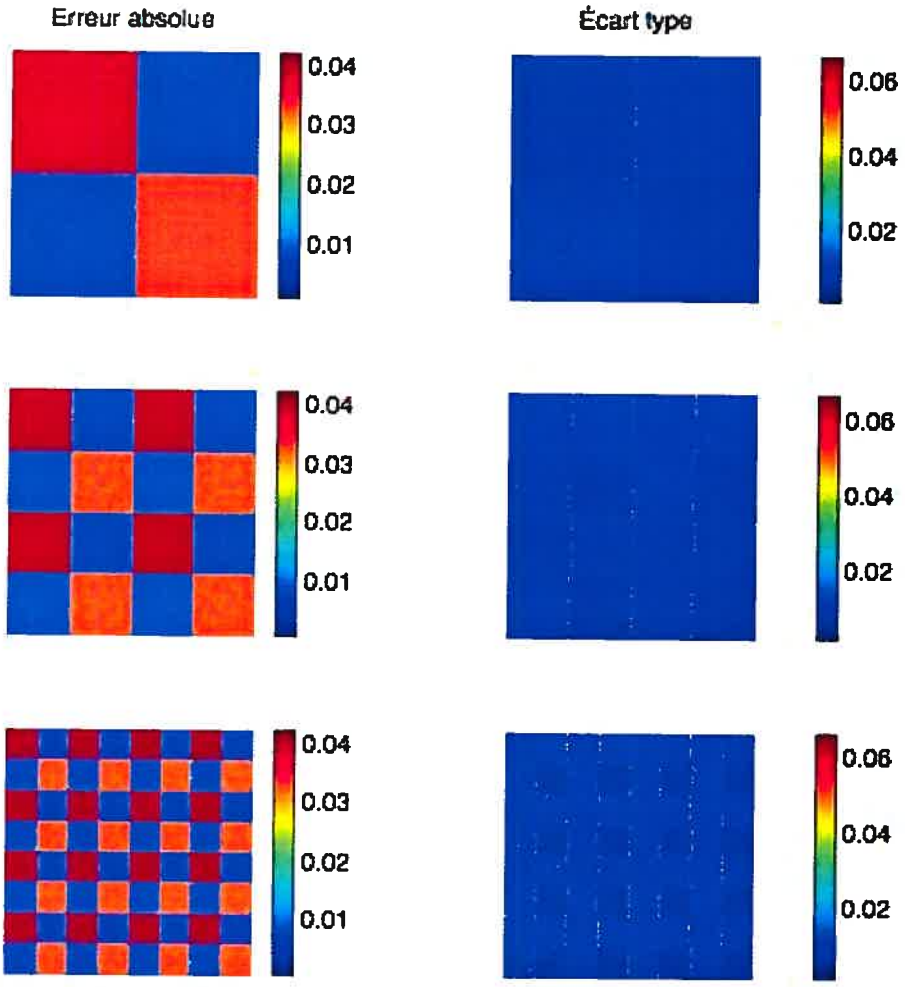

FIGURE B.4. Résultats des simulations avec un bruit gaussien de 5% pour le MAP

TABLEAU B.2. Écart type moyen par région pour les simulations avec un bruit de 5%

Image	Fonction de perte	Région 1	Région 2	Région 3	Région 4
Image 1	Huber modifiée	0,0013	0,0007	0,0008	0,0011
	0-1 sur un intervalle	0,0012	0,0007	0,0008	0,0010
	Quadratique	0,0007	0,0007	0,0008	0,0007
	MAP	0,0059	0,0090	0,0093	0,0061
Image 2	Huber modifiée	0,0011	0,0009	0,0008	0,0012
	0-1 sur un intervalle	0,0010	0,0010	0,0008	0,0011
	Quadratique	0,0006	0,0009	0,0008	0,0007
	MAP	0,0065	0,0092	0,0096	0,0061
Image 3	Huber modifiée	0,0012	0,0015	0,0016	0,0010
	0-1 sur un intervalle	0,0011	0,0014	0,0014	0,0009
	Quadratique	0,0007	0,0014	0,0014	0,0007
	MAP	0,0065	0,0088	0,0095	0,0051

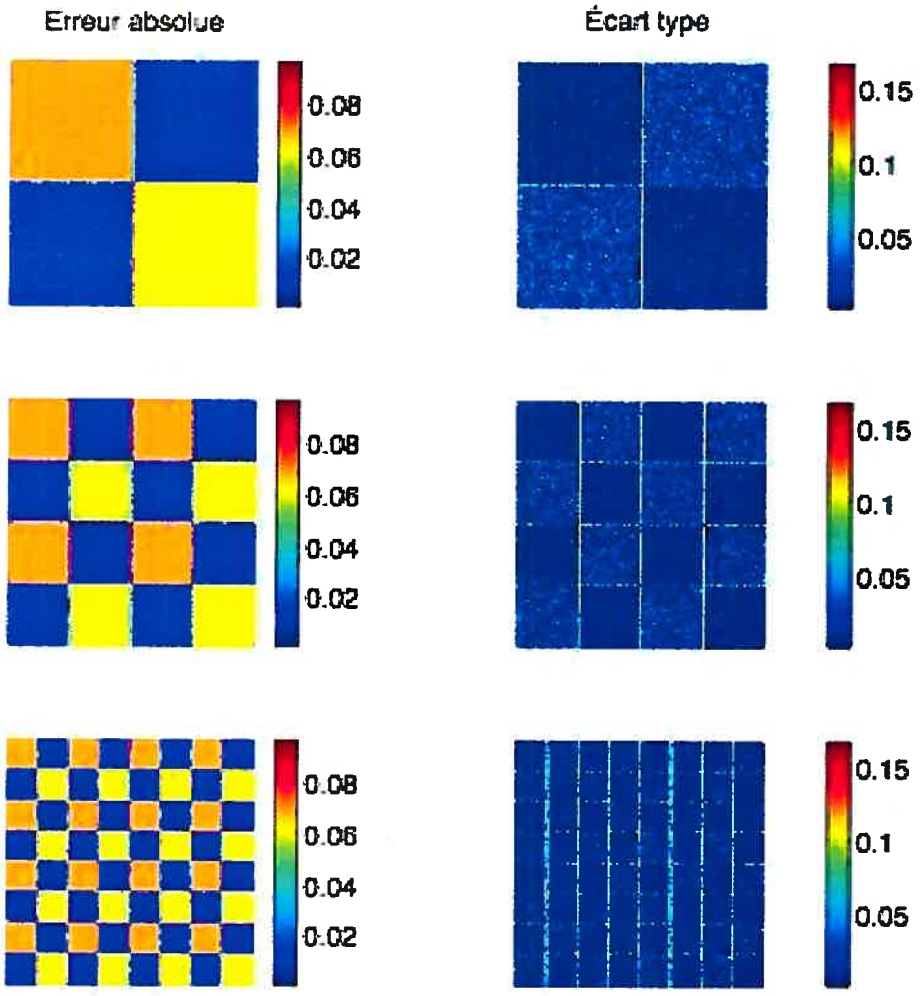

FIGURE B.5. Résultats des simulations avec un bruit gaussien de 10% pour la fonction de perte de Huber modifiée

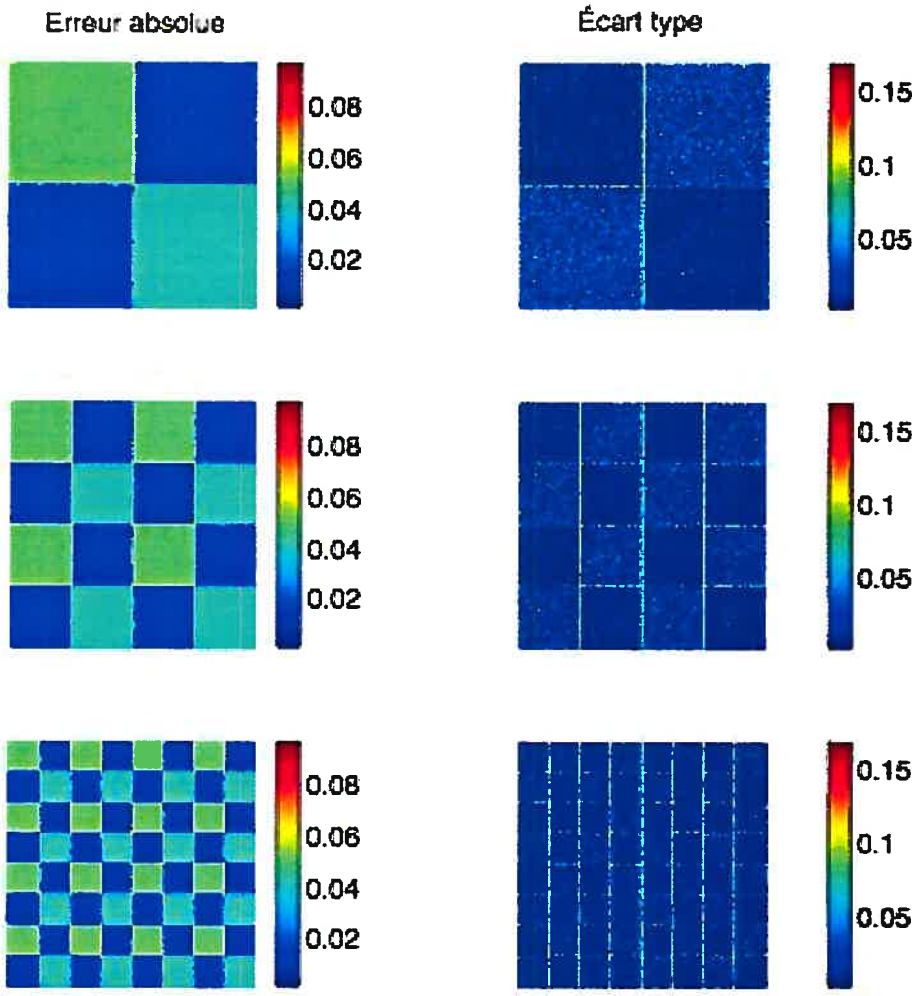

FIGURE B.6. Résultats des simulations avec un bruit gaussien de 10% pour la fonction de perte 0-1 sur un intervalle

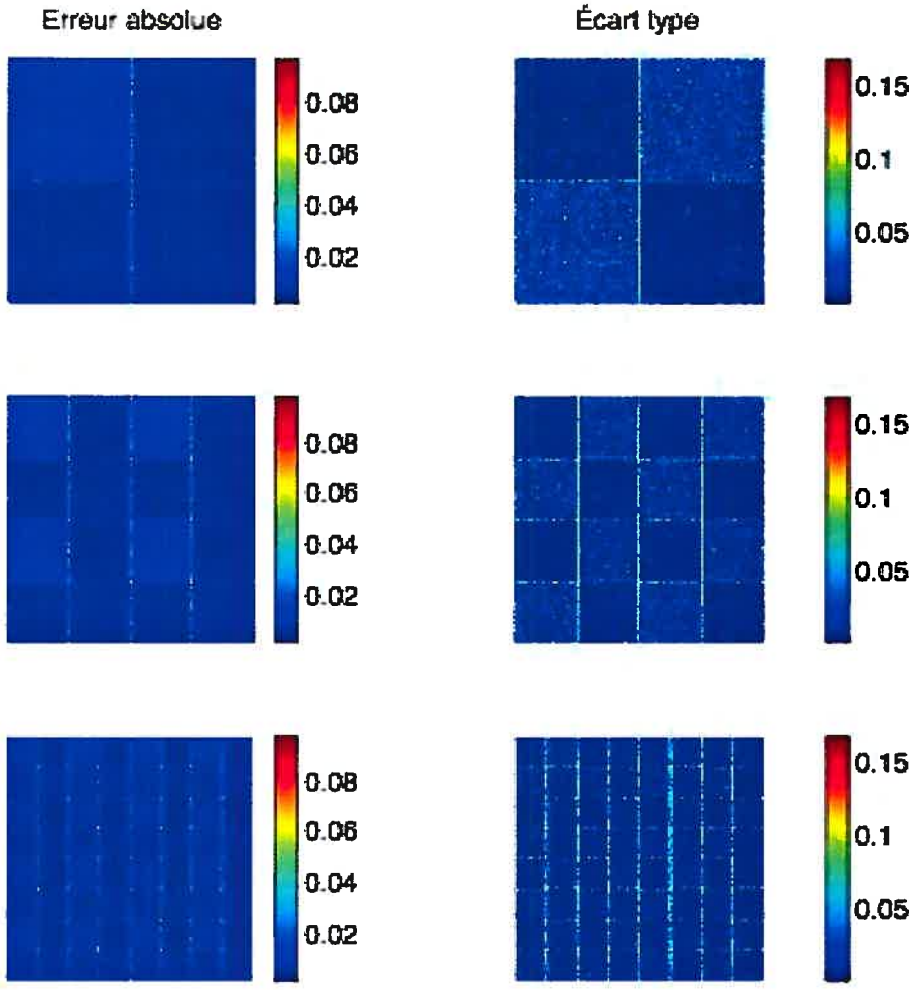

FIGURE B.7. Résultats des simulations avec un bruit gaussien de 10% pour la fonction de perte quadratique

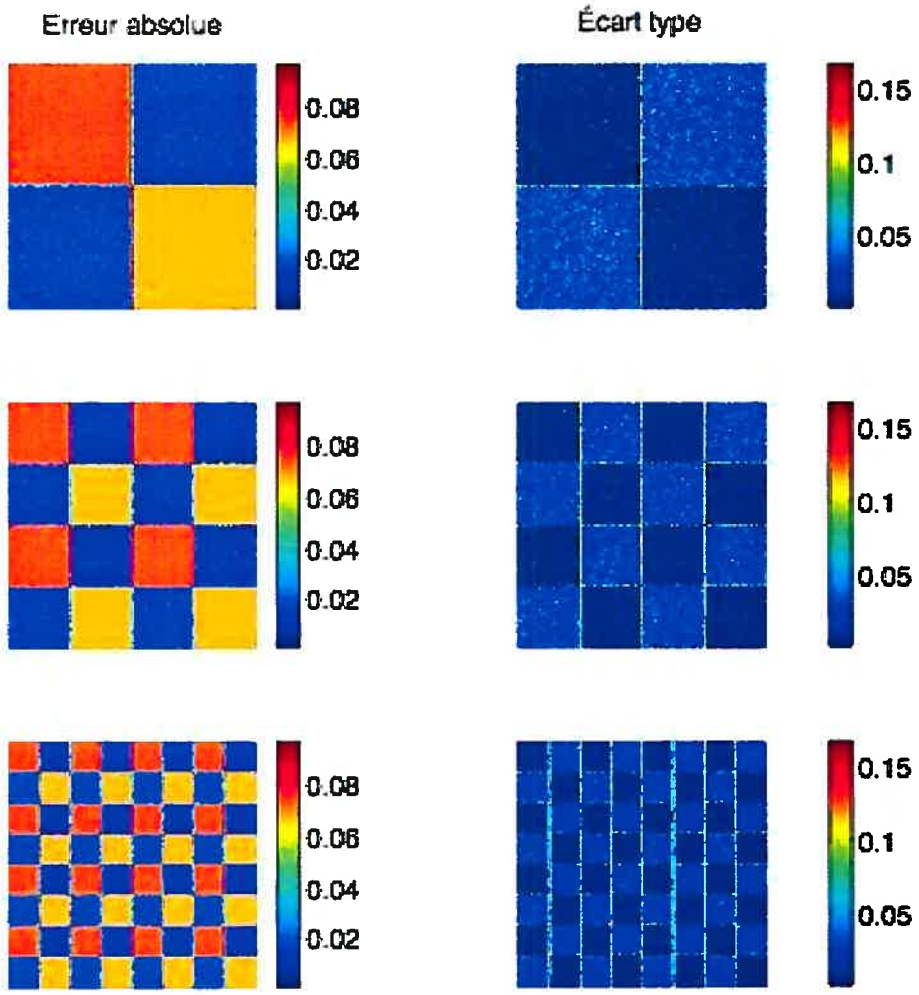

FIGURE B.8. Résultats des simulations avec un bruit gaussien de 10% pour le maximum *a posteriori*

TABLEAU B.3. Erreur absolue moyenne par région pour les simulations avec un bruit de 10%

Image	Fonction de perte	Région 1	Région 2	Région 3	Région 4
Image 1	Huber modifiée	0,0687	0,0140	0,0149	0,0609
	0-1 sur un intervalle	0,0503	0,0135	0,0143	0,0404
	Quadratique	0,0067	0,0044	0,0044	0,0041
	MAP	0,0751	0,0180	0,0187	0,0669
Image 2	Huber modifiée	0,0689	0,0139	0,0151	0,0612
	0-1 sur un intervalle	0,0505	0,0134	0,0145	0,0406
	Quadratique	0,0068	0,0041	0,0044	0,0043
	MAP	0,0752	0,0168	0,0178	0,0667
Image 3	Huber modifiée	0,0690	0,0145	0,0158	0,0614
	0-1 sur un intervalle	0,0507	0,0140	0,0153	0,0408
	Quadratique	0,0067	0,0043	0,0046	0,0045
	MAP	0,0750	0,0178	0,0203	0,0667

TABLEAU B.4. Écart type moyen par région pour les simulations avec un bruit de 10%

Image	Fonction de perte	Région 1	Région 2	Région 3	Région 4
Image 1	Huber modifiée	0,0039	0,0182	0,0181	0,0038
	0-1 sur un intervalle	0,0034	0,0176	0,0171	0,0032
	Quadratique	0,0040	0,0162	0,0162	0,0039
	MAP	0,0055	0,0293	0,0304	0,0057
Image 2	Huber modifiée	0,0045	0,0151	0,0149	0,0045
	0-1 sur un intervalle	0,0039	0,0140	0,0138	0,0040
	Quadratique	0,0046	0,0129	0,0134	0,0044
	MAP	0,0059	0,0261	0,0285	0,0064
Image 3	Huber modifiée	0,0059	0,0115	0,0116	0,0057
	0-1 sur un intervalle	0,0051	0,0104	0,0107	0,0052
	Quadratique	0,0054	0,0095	0,0099	0,0055
	MAP	0,0076	0,0254	0,0293	0,0075

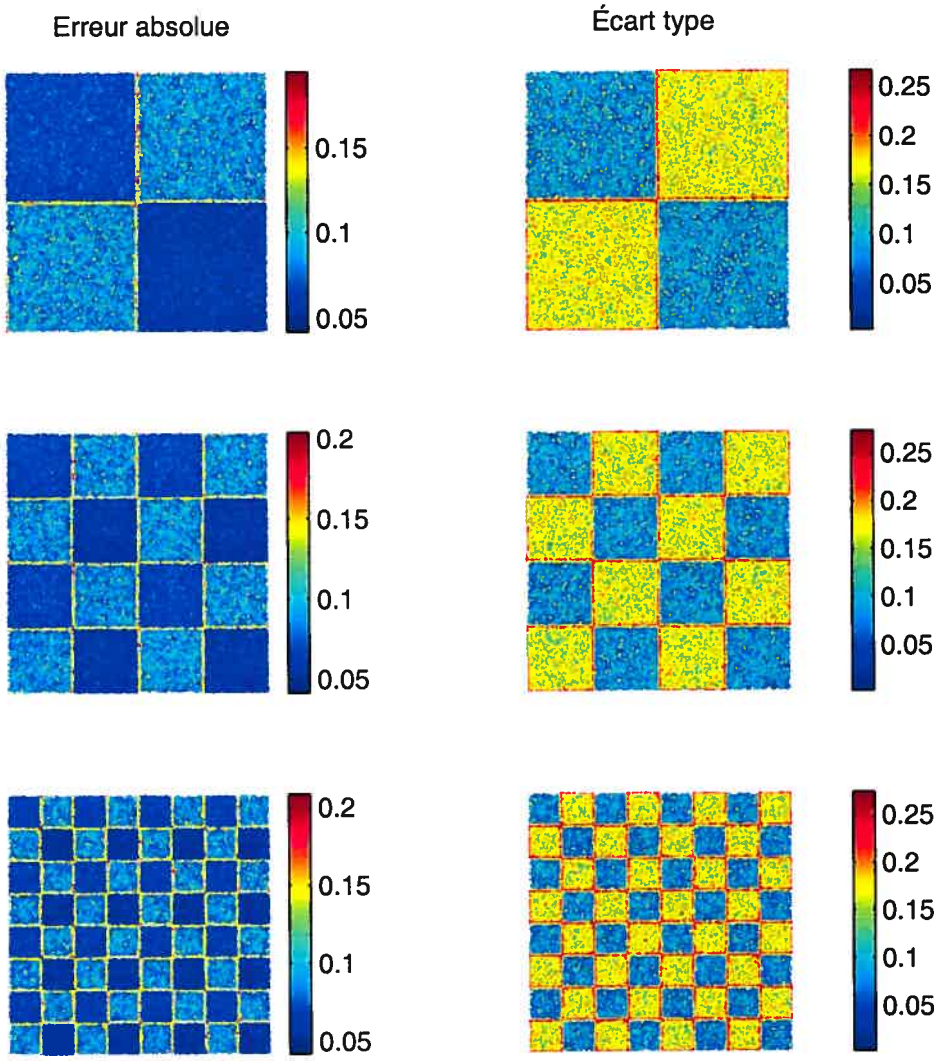

FIGURE B.9. Résultats des simulations avec un bruit gaussien de 20% pour la fonction de perte 0-1 sur un intervalle

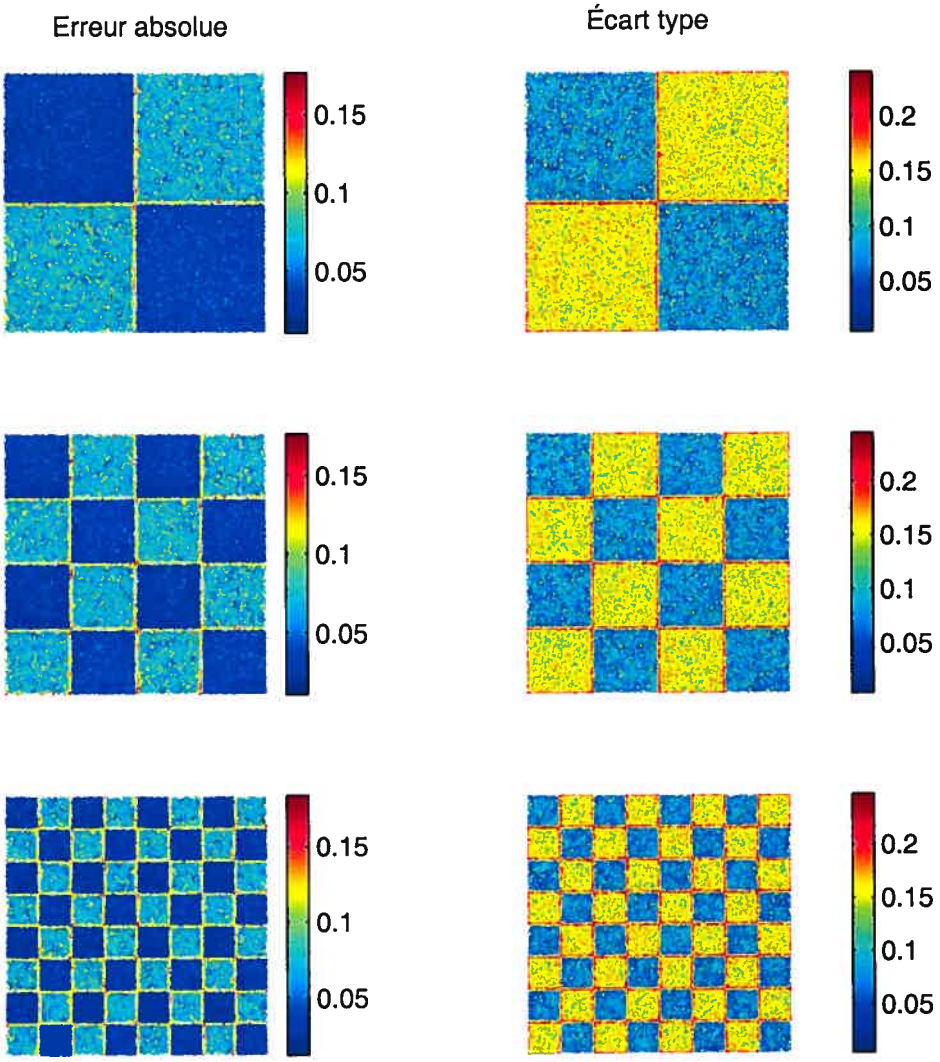

FIGURE B.10. Résultats des simulations avec un bruit gaussien de 20% pour la fonction de perte quadratique

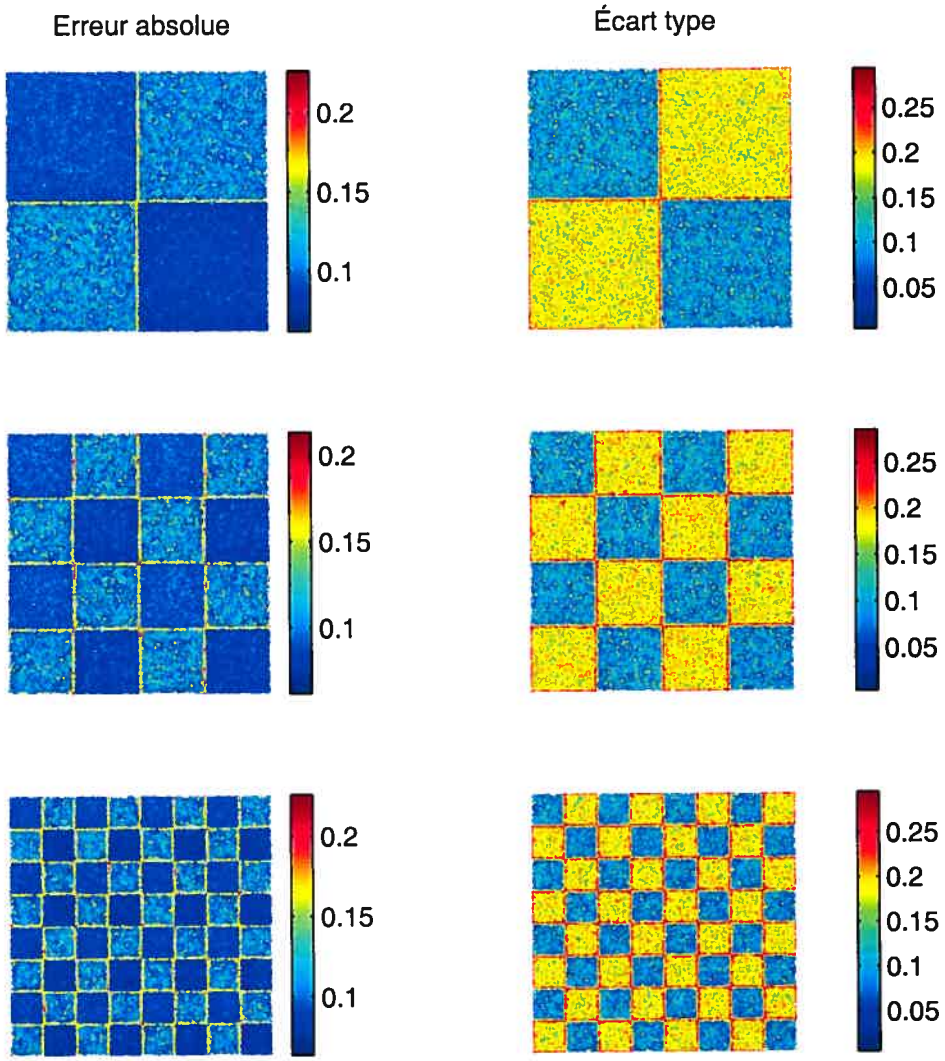

FIGURE B.11. Résultats des simulations avec un bruit gaussien de 20% pour le maximum *a posteriori*

BIBLIOGRAPHIE

BELONGIE, S., CARSON, C., GREENSPAM, H., MALIK, J., (1998), Color- and Texture-Based Image Segmentation Using EM and Its Application to Content-Based Image Retrieval, *ICCV*, pp. 675-682.

BESAG. J., (1986), On the statistical analysis of dirty pictures, *Journ. of the Royal Statistical Society*, **B-48(3)**, pp. 259-302.

BROOKS, S.P., (1998), Quantitative convergence assessment for Markov chain Monte Carlo via cusums, *Statistics and Computing*, **8**, pp. 267-274.

BROOKS, S.P. ET GELMAN, A., (1998) General Methods for Monitoring Convergence of Iterative Simulations, *Journal of Comp. and Graphical Statistics*, **7(4)**, pp.434-455.

BROOKS, S.P., GIUDICI, P., PHILIPPE, A., (2003), Nonparametrix Convergence Assessment for MCMC Model Selection, *Journal of Comp. and Graphical Statistics*,**12(1)**, pp. 1-22.

CHAUVEAU, D. ET DIEBOLT J., An automated stopping rule for MCMC Convergence assessment, (1998), *Computational Statistics*, **14(3)**, pp.419-442.

DESTREMPES, F., MIGNOTTE, M., ANGERS, J.F. (2005), A Stochastic Method for Bayesian Estimation of Hidden Markov Random Field Models with Application to a Color Model, *IEEE Trans. on Image Processing*, **14(8)**, pp. 1096-1124.

DEMPSTER, A.P., LAIRD, N.M., RUBIN, D.B., (1977), Maximum Likelihood from Incomplete Data via the *EM* Algorithm, *Journ. of the Royal Stat. Soc.. Serie B(Methodological)*, **39(1)**, pp. 1-38.

EVERITT, B. (1974), *Cluster Analysis*, Heinemann Educationnal Books Ltd, London.

FRANÇOIS, O., (2002), Global Optimization with Exploration/Selection Algorithm and Simulated Annealing, *Ann. Appl. Probability*, **12(1)**, pp. 248-271.

GELMAN, A. ET RUBIN, D.B., (1992), Inference from Iterative Simulation Using Multiple Sequences,*Statistical Science*, **7(4)**, pp. 457-472.

GEMAN, S. ET GEMAN, D., (1984), Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **6(6)**, pp.721-741.

GU, D.B. ET SUN, J.X. (2005), EM image segmentation algorithm based on an inhomogeneous hidden MRF model, *IEEE Proc. Vis. Image Signal Process*, **152(2)**, pp.184-190.

HAJEK, B., (1988), Cooling schedule for optimal annealing, *Math. Oper. Res.*, **13**, pp. 311-329.

LAKSHMANAN, S. ET DERIN, H., (1989), Simultaneous parameter estimation and segmentation of Gibbs random fields using simulated annealings, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **11(8)**, pp. 799-813.

MAÎTRE, H. (2003), *Le traitement des images*, Lavoisier, Paris.

MARION A. (1997), *Acquisition & visualisation des images*, Eyrolles, Paris.

MARROQUIN, J., MITTER, S. ET POGGIO, T., (1987), Probabilistic solution of ill-posed problems in computational vision, *Journ. of American Stat. Ass.*, **82(397)**, pp.76-89.

NASIOS, N. ET BORS, A.G. (2004), A Variational Approach for Color Image Segmentation, *ICPR (1)*, pp. 680-683.

ROBERT, C.P. (1998), *Discretization and MCMC Convergence Assessment*, Lecture Notes in Statistics, Springer-Verlag, New York.

ROBERT, C.P., (2001), *The Bayesian Choice*, Springer, New York.

SUN, D. ET BERGER, J.O., (1998), Reference Priors with Partial Information, *Biometrika*, **85(1)**, pp-55-71.

WON, C.S, ET GRAY, M. (2004), *Stochastic image processing*,Kluwer Academic/Plenum Publishers, New York.

WU, F.Y., (1982), The Potts Model, *Review of Modern Physics*, **54(1)**, pp.235-268.

Wu, Y., Yang, X. et Chan, K.L., (2003), Unsupervised Color Image Segmentation Based on Gaussian Mixture Model, *Proc. IEEE Pacific-Rim Conference on multimedia (PCM2003).*