# Hypergraphe de Voisinage Spatiocolorimétrique. Application en traitement d'images : Détection decontours et du bruit.

Soufiane Rital

# Hypergraphe de Voisinage Spatiocolorimétrique. Application en traitement d'images : Détection de contours et du bruit.

## THÈSE

présentée et soutenue publiquement le 05 Juillet 2004

pour l'obtention du

**Doctorat de l'Université de Bourgogne – Dijon**

**(spécialité informatique)**

par

Soufiane Rital

**Composition du jury**

| | | |
|---|---|---|
| *Rapporteurs :* | Pr. Sylvie Philipp-Foliguet | Université de Cergy-Pontoise |
| | Pr. Serge Miguet | Université Lyon II |
| *Examinateurs :* | Pr. Francis Castanie | ENSEEIHT, Toulouse |
| | Pr. Driss Aboutajdine | Université Mohammed V, Rabat Maroc |
| | Pr. Alain Bretto | Université de Caen |
| *Directeur de thèse :* | Pr. Hocine Cherifi | Université de Bourgogne |

**Laboratoire LIRSA. Laboratoire d'Informatique, Réseaux et Systèmes Audiovisuels**

Mis en page avec la classe thloria.

# Avant-propos

*Pendant toute la durée de ma thèse, j'ai eu la chance de côtoyer, et parfois même de rencontrer, des personnalités réellement extraordinaires. Qu'il me soit permis ici de leur rendre humblement hommage et de les remercier pour tout ce qu'elles m'ont apporté : pour l'aide et les conseils qu'elles m'ont prodigués, pour leur passion contagieuse, mais aussi et surtout pour la formidable image de la Recherche et de l'Humanité en général qu'elles incarnent à mes yeux. Un travail de thèse met à contribution beaucoup de personnes. Durant trois années, M. Hocine Cherifi Professeur du laboratoire LIRSA de l'Université de Bourgogne de Dijon a dirigé mes travaux de recherches. Cette thèse doit énormément à sa grande disponibilité, son dynamisme, ses encouragements, son ouverture d'esprit et bien sûr ses qualités scientifiques exceptionnelles. Il m'a permis de me consacrer à cette thèse dans des conditions tout simplement parfaites. Je lui adresse mes plus sincères remerciements. Il s'est profondément investi dans tous les travaux qui sont présentés ici. J'ai eu grand plaisir à travailler avec lui.*

*Que Mme Sylvie Philipp-Foliguet, Professeur à l'Ecole Nationale Supérieure de l'Electronique et de ses Applications (ENSEA) de Cergy Pontoise, trouve ici l'expression de ma gratitude pour le temps qu'elle a bien voulu consacrer à la lecture approfondie de ce manuscrit, en tant que rapporteur. Ses remarques ont fortement contribué à l'amélioration de ce document. Je remercie également très sincèrement M. Serge Miguet, Professeur au Laboratoire d'InfoRmatique en Images et Systèmes d'information (LIRIS) de l'Université de Lumière de Lyon, pour avoir accepté d'examiner ma thèse. Ses remarques ont contribué à améliorer le document et à approfondir certains aspects de mon travail. En tant qu'Attaché Temporaire d'Enseignement et de Recherche (ATER), j'ai le grand plaisir d'enseigner à ses cotés.*

*J'adresse mes sincères remerciements à M. Francis Castanie, Professeur à l'Institut National Polytechnique de Toulouse (I.N.P.T.), pour avoir accepté d'examiner ce travail. C'est un grand honneur que cet éminent spécialiste me fait en participant au jury. Je remercie très vivement M. Alain Bretto, Professeur du Groupe de Recherche en Informatique, Image, Automatique et Instrumentation de l'Université de Caen, pour nos longues discussions sur la théorie des Hypergraphes ainsi que pour l'aide qu'il m'a apportée lors des prémices de ces travaux de recherches.*

*Que M. Driss Aboutajdine, Professeur du Groupe Signaux et Communications de l'Université Mohammed V de Rabat, trouve ici mes plus vifs remerciements pour m'avoir impliqué dans ce domaine de recherche et son profond investissement dans tous les travaux qui sont présentés ici. J'ai eu un réel plaisir à collaborer avec lui.*

*D'autre part ...*

*On ne passe pas trois années dans un établissement sans tisser des liens d'amitié. Aussi dans le désordre qui me caractérise, je vais tâcher de n'oublier personne. Je voudrais tout d'abord remercier les thésards et permanents du laboratoire LIRSA qui ont contribué à rendre ce séjour agréable. Je voudrais en particulier remercier Mohammed Elhassouni, Vincent Bricard-Vieu, Jean Noël Campaner, Samir Salaymeh et Hafsa Benaboud.*

*Je voudrais aussi remercier les thésards et permanents du Laboratoire LIRIS (Université Lumière Lyon 2) avec qui j'ai passé ma deuxième année de poste ATER. Je voudrais en particulier remercier, les membres de FOXSTREAM, Tweed Tiffany, Kinda Hassan et Laure Tougne.*

*Oublier ceux qui m'ont accompagné serait inqualifiable. Je tiens donc à associer à ce travail toutes les personnes qui m'ont aidées, de près ou de loin. Merci à Muriel Moret d'avoir relu ce document.*

*Mes plus sincères remerciements vont aussi à ma famille. Aucun mot ne saurait retranscrire ici sans l'affaiblir le bonheur qu'elle m'a toujours apporté, ni l'ampleur de ce que je lui dois.*

*Merci à tous*

*A mes parents*
*A mes sœurs et A mes frères*

# Résumé

Dans ce document, nous nous intéressons à la modélisation de l'image par le biais de la théorie des hypergraphes. Notre contribution est essentiellement axée sur la détermination des propriétés issues de cette théorie et sur l'analyse de leur adéquation avec des problématiques de l'image et particulièrement la détection de contours et la suppression de bruit.

Dans un premier temps, nous étudions la représentation par hypergraphes de voisinage spatiocolorimétrique de l'image. Trois représentations sont présentées incorporant des propriétés globales, locales, des mesures de similarité et des mesures de dissimilarité.

Ensuite, on utilise les propriétés des hypergraphes engendrées par la représentation afin de définir des modèles structurels de bruit et de contour. Ceci nous permet ainsi de déduire des algorithmes de suppression de bruit et d'extraction de contours sur des images à niveaux de gris et couleur. Les performances des approches proposées sont comparées aux solutions classiquement utilisées. Enfin, la représentation par hypergraphe de voisinage spatiocolorimétrique s'est avérée efficace pour le traitement des images bas niveaux.



**Mots Clés** : graphe, hypergraphe, combinatoire, détection de contours, détection de bruit, espace couleur, système de voisinage, modélisation d'image, mesure de similarité, mesure de dissimilarité.

# Abstract

In this document, we are interested in image modeling by the means of the hypergraph theory. Our contribution is essentially centered on the determination of the properties resulting from this theory and on the analysis from their adequacy with image problems, particularly edge and noise detection.
First, we study the image spatiocolorimetric neighborhood hypergraph representation. Three representations are respectively presented incorporating global properties, local properties and similarity functions.
Then, we use the hypergraph properties generated by the representation in order to define the structural models of noise and edge. This enables us to deduce the algorithms of noise suppression and edge detection on gray scale and color images. The performances of the proposed approaches are compared with the solutions classically used. Finally, the representation by neighborhood hypergraph consistently seems to be efficient in low level image processing.

**KeyWords** : graph, hypergraph, combinatory, edge detection, noise detection color spaces, neighborhood system, image modeling, similarity functions.

# Table des matières

# Table des figures

# Liste des tableaux

# Introduction générale

L'être humain reçoit en permanence des informations très diverses par l'intermédiaire de ces cinq sens. Malgré l'abondance de ces informations, il est capable de construire une représentation cohérente du monde qui l'entoure. Son cerveau réussit à chaque instant à organiser et hiérarchiser les données éparses qu'il reçoit et restitue chaque objet observé comme un 'tout'.

L'analyse d'images cherche à reproduire de façon automatique ce même processus. Elle a pour but la description du contenu d'une image, en vue de son interprétation et d'une prise de décision. Les champs d'application de cette discipline touchent des domaines aussi variés que l'imagerie satellite, médicale ou biologique, la robotique, la géologie...

Les données à traiter proviennent de l'échantillonnage d'une région de l'espace par un système d'acquisition (caméra, microscope confocal, rayons X, ...). Ces données se présentent traditionnellement sous la forme d'images bidimensionnelles et sont le résultat d'une série de mesures effectuées à intervalles réguliers dans un plan. Mais, il peut également s'agir de blocs de données tridimensionnels résultant d'un empilement de coupes planes. Des systèmes plus perfectionnés comme le système à balayage laser permettent quant à eux d'obtenir un nuage de points 3D mesurés directement sur la surface d'objet. Avec l'amélioration constante des systèmes d'acquisition, les mesures sont de plus en plus précises et les données à traiter de plus en plus volumineuses.

L'analyse d'images a pour but de décrire de façon quantitative ou qualitative les objets présents dans les données. Elle transforme pour cela l'information de bas niveau issue de la phase d'acquisition en une information de haut niveau où les formes et les structures sont décrites de façon synthétique. Les grandes étapes de ce processus sont le prétraitement, la segmentation, la description et l'interprétation. Les processus de bas niveau nécessitent très peu d'informations sur le contenu sémantique des images. Il s'agit ici des processus de filtrage, d'amélioration et de restauration des images. Les processus de haut-niveau fonctionnent en aval de ceux de bas-niveau, et peuvent nécessiter des informations sur le contenu des images. Il s'agit de la reconstruction tridimensionnelle, la reconnaissance de formes, les processus cognitifs de façon générale.

La cadre général dans lequel s'inscrit cette thèse est le traitement d'image bas-niveau. Ces traitements peuvent être opérés à partir de la représentation spatiocolorimétrique des images ou en utilisant un système de représentation alternatif (représentation fréquentielle, spatiofréquentielle, ...). Nos travaux reposent sur la représentation spatiocolorimétrique. Néanmoins, le modèle proposé peut être adapté à d'autres types de représentation.

Depuis près de quarante ans, parallèlement et grâce au développement des calculateurs, l'image numérique est intensivement étudiée. Les développements récents des divers champs de ce domaine ont mis l'accent sur la nécessité de formalisation et de modélisation de l'image numérique.

Le cadre théorique général dans lequel s'inscrit ce travail est celui de la détection et des caractérisations des singularités en utilisant une modélisation et les propriétés issues de la théorie des Hypergraphes.

Nous ne nous plaçons pas dans ce document sous l'angle de la sémantique que l'on peut donner à l'image. Notre souci est de présenter une modélisation et de tenter de la comparer avec

quelques outils existants pour l'une des principales problématiques de ce domaine, réduction de bruit et préservation des contours.

Une image numérique peut être considérée comme la réunion de composantes spatiales et radiométriques élémentaires : *les pixels*. Ainsi, travailler sur l'organisation interne de ces éléments entre eux peut être envisagé du point de vue de la pure combinatoire, puisque celle-ci est par définition la science de l'étude de l'organisation des objets mathématiquement formalisés. C'est probablement l'une des raisons pour laquelle la théorie des graphes et les techniques qui en sont issues, ceci en tant que domaine connexe de la combinatoire, ont rapidement émergé en traitement d'image et ont été largement utilisées par de nombreux chercheurs [Mar92].

Une approche attractive consiste en l'application d'une modélisation basée sur une généralisation de la théorie des graphes [Ber87a] : *la théorie des hypergraphes* [Ber87b]. C'est par ce biais que nous abordons la problématique précédemment citée. Les domaines d'applications visés concernent les images à niveaux de gris et les images couleur.

La théorie des hypergraphes est un outil efficace dans de nombreux problèmes discrets d'optimisation [Fag83][GM95][Pet97]. Son introduction en traitement d'images participe au développement de nouvelles modélisations et à de nouveaux algorithmes dans ce domaine. Les premiers résultats obtenus grâce à ce modèle sont très encourageants. Dans ([BACL97]), A. Bretto et al. ont présenté une modélisation d'image par hypergraphe de voisinage. Cette modélisation est utilisée ensuite pour la segmentation des images. L'approche utilisée pour la segmentation est l'approche région. Les régions de l'image sont détectées en s'appuyant sur une notion d'homogénéité globale.

Dans ce document, nous présentons d'autres applications de cette modélisation, telles que la suppression de bruit et la détection des contours. Dans un premier temps, nous avons travaillé sur la modélisation d'image par hypergraphe de voisinage en prenant en considération l'ensemble des paramètres qui permettra de construire des hypergraphes de voisinage tels que la distance sur la grille et la distance colorimétrique. Les deux stratégies d'application de l'hypergraphe de voisinage à l'image sont prises en considération : hypergraphe à information locale et à information globale. L'efficacité de la modélisation est ensuite testée par les deux applications : suppression de bruit et détection de contours, en s'appuyant premièrement sur les notions d'inhomogénéité locale pour détecter le bruit et deuxièmement sur la notion d'homogénéité locale pour détecter les contours.

Notre document s'articule autour de trois chapitres.

- Le premier chapitre est consacré aux présentations des structures discrètes et traitement d'image numérique que nous aurons à manipuler dans les chapitres suivants. Dans les premières sections, nous commençons par les définitions des graphes et hypergraphes. Les notions de base en image numérique : le pavage du plan, la grille, le système de voisinage, etc. seront abordées dans la troisième section. Dans la quatrième section, nous présentons l'hypergraphe de voisinage d'une image à niveaux de gris. Après une brève description du traitement de l'image multicomposante et des espaces couleur, l'extension de la représentation aux images couleurs est le thème de la dernière section.
- Le deuxième chapitre présente une application relative au débruitage en employant la représentation par Hypergraphe de Voisinage HV. Dans la littérature, on distingue deux approches de suppression de bruit. La première est basée sur une estimation tandis que la deuxième repose sur une détection et ensuite une estimation. Dans la dernière approche, l'estimation est conditionnée à la détection de pixels de bruit, tandis que l'approche fondée uniquement sur l'estimation conduit à la modification de tous les pixels de l'image. L'algorithme proposé dans ce chapitre entre dans le cadre des méthodes de détection et estimation de bruit.

  Dans les première et deuxième sections, nous présentons les modèles de bruit. Une description succincte des outils utilisés dans ces approches est présentée dans la troisième section. Les deux

dernières concernent la présentation des algorithmes proposés pour la suppression de bruit dans des images à niveaux de gris et des images couleur. Les extensions du modèle aux images couleurs par le biais des approches marginale et vectorielle sont prises en considération. Enfin, nous comparons à l'aide de mesures objectives, nos résultats aux algorithmes couramment utilisés dans la littérature.

- Le troisième chapitre porte sur l'approche de détection des contours proposée. Nous définissons formellement ce que nous appelons modèle de contour d'objet et nous proposons un algorithme de détermination de ceux-ci pour des images à niveaux de gris et des images couleur. Ensuite, nous regroupons ce modèle de contour et le modèle de bruit défini dans le chapitre 2 et nous donnons un algorithme de détection de contours dans des images bruitées.

  Les résultats de la détection de contours dans des images bruitées ou non bruitées seront comparés avec les algorithmes couramment utilisés.

- En conclusion générale, nous résumons notre contribution et tirons les conclusions sur l'approche développée. Nous proposons, enfin, quelques unes des améliorations qui pourraient être apportées.

# Chapitre 1

# STRUCTURES DISCRETES ET TRAITEMENT D'IMAGE

**Résumé :**

*Dans les systèmes où les relations entre objets manipulés jouent un rôle prépondérant (la chimie, l'électronique et la modélisation de réseaux etc ...), la théorie des graphes trouve une place privilégiée. Une image numérique peut être vue comme un ensemble discret de données. Il est donc naturel de l'envisager du point de vue de la combinatoire ou sous l'angle de la théorie des graphes. Dans une première partie, ce chapitre a pour but de présenter quelques concepts fondamentaux de cette théorie. Nous voyons dans un deuxième temps que la théorie des graphes peut dans un certain sens être généralisée à travers le concept d'hypergraphe et nous donnons quelques définitions essentielles pour notre étude de ce domaine. Nous procédons ensuite à une définition formelle de l'image numérique, pour aboutir finalement à une notion de structure d'hypergraphe de voisinage qu'il est possible de construire sur les composantes élémentaires de l'image numérique. Tout d'abord, nous définissons l'hypergraphe de voisinage de l'image à niveaux de gris HVI. Ensuite, nous présentons, l'hypergraphe de voisinage d'une image couleur. Cette dernière représentation sera nommée : l'Hypergraphe de Voisinage SpatioColorimétrique HVSC.*

## 1.1 Graphes

Il est difficile de dater très précisément le moment où l'idée de graphe a pu germer dans l'esprit des mathématiciens. Si les travaux de Leibniz ou d'Euler sont probablement précurseurs, il est intéressant de noter que la majeure partie des problèmes de combinatoire ou de recherche opérationnelle peut s'exprimer au moyen de la théorie des graphes.

**Définition d'un graphe.** Etant donné un ensemble quelconque $S$, nous notons $S^{(2)}$ l'ensemble des paires d'éléments de $S$.

**Définition 1.1.1** *: Graphe*
*([Ber87a] ; [Die97] ; [Bol98])*
*Etant donné un ensemble $S$, un graphe est défini comme un couple d'ensembles disjoints $(S, E)$ où $E$ est un sous-ensemble de $S^{(2)}$.*

Les éléments de $S$ sont dénommés *sommets* du graphe, ceux de $E$ *arêtes*. La cardinalité de l'ensemble $S$ est appelée la *taille* du graphe. Si $e = \{x, y\}$ est une arête du graphe, les sommets $x$ et $y$ sont dénommés *extrémités* de $e$. Graphiquement, on représente souvent les noeuds par des points, et une arête $e = \{s, t\}$ par un arc de courbe joignant les deux points $s$ et $t$ (Fig. 1.1).

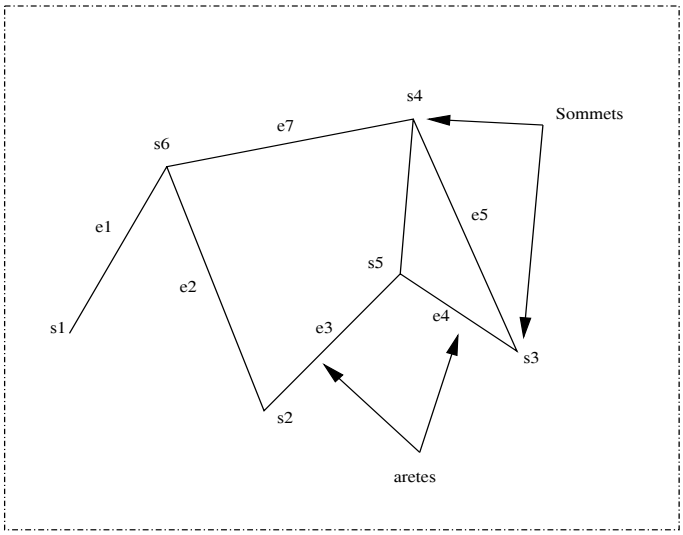



FIG. *1.1: Exemple d'un graphe non orienté.*

Le lecteur aura peut-être remarqué que la définition précédente déroge quelque peu aux habituelles définitions de la théorie des graphes. En effet, bien souvent, l'ensemble des arêtes d'un graphe est défini comme un sous-ensemble du multi-ensemble des couples d'éléments de $S \times S$. De tels graphes sont alors dits *orientés* et un élément de $E$ est appelé *arc*. De plus, rien dans cette définition classique n'interdit l'existence de plusieurs arcs entre deux sommets : ceci définit la notion de multi-arête. De même, rien n'interdit non plus qu'il existe un arc entre un sommet et lui-même (une boucle). Nous nous plaçons donc dans le cadre de graphes *non orientés*, simples et sans boucle.

Une généralisation récente du concept de graphe, introduite par Claude Berge, est celle d'hypergraphe [Ber73], où les arêtes peuvent être de taille arbitraire et non plus seulement de taille deux. Nous rappelons ici la terminologie de la théorie des graphes qui nous sera utile lorsque nous aborderons l'étude des hypergraphes. Pour des notions plus détaillées ou plus générales, nous renvoyons le lecteur à l'ouvrage de Gondran et Minoux [GM95].

**Systèmes de voisinages :** On voit qu'un graphe est parfaitement déterminé par la donnée de l'ensemble $S$ des sommets et de l'application $\mathcal{C}$ qui à tout sommet associe son voisinage, i.e. l'ensemble de ses voisins :

$$\begin{aligned} \mathcal{C} : S &\longmapsto \mathcal{P}(S) \\ s &\longmapsto \mathcal{C}_s = \{t \in S | \{s, t\} \in E\} \end{aligned} \tag{1.1}$$

Ainsi dans l'exemple de la figure 1.1, on a :

$$\begin{aligned} &\mathcal{C}_{s1} = \{s_6\}, \quad \mathcal{C}_{s2} = \{s_5, s_6\}, \quad \mathcal{C}_{s3} = \{s_5, s_4\}, \\ &\mathcal{C}_{s4} = \{s_3, s_5, s_6\}, \quad \mathcal{C}_{s5} = \{s_2, s_3, s_4\}, \quad \mathcal{C}_{s6} = \{s_1, s_2, s_4\} \end{aligned} \tag{1.2}$$

Le graphe $G = (S, E)$ pourra être noté de façon équivalente : $G = (S, \mathcal{C})$. L'ensemble $\mathcal{C}(S) = \{\mathcal{C}_s, s \in S\}$ de tous les voisinages sur $S$ est appelé système de voisinage. Par définition de

l'ensemble $E$ des arêtes du graphe, un système de voisinage possède les propriétés caractéristiques suivantes :

$$\begin{array}{ll} (i) & \forall s \in S, s \notin \mathcal{C}_s \\ (ii) & \forall \{s,r\} \subset S, r \in \mathcal{C}_s \Longleftrightarrow s \in \mathcal{C}_r. \end{array} \tag{1.3}$$

**Chemin ou Chaîne :** C'est une succession des arêtes parcourues dans le même sens. Le nombre des arêtes parcourues s'appelle la *longueur* du chemin. On parle de *chaîne* si l'on ne tient pas compte de la direction des arêtes ; on parle ainsi de chaîne dans les graphes non orientés.

## 1.2 Hypergraphes

Le concept d'hypergraphe a été introduit pour la première fois par Berge ([Ber73] ; [Ber79]) au début des années soixante-dix. En généralisant la notion de Graphe, Berge a pu montrer assez simplement certains théorèmes qu'il était extrêmement délicat et long à démontrer par la théorie des graphes classiques. Comme Berge le dit lui-même "...en généralisant, on peut souvent simplifier ; un seul énoncé, parfois étonnamment simple, peut rassembler plusieurs théorèmes de graphes dans le même moule". Mais, donnons sans plus attendre la définition d'un hypergraphe ([Ber87b]) :

**Définition 1.2.1** *:* **Hypergraphe :**
*Soit $X$ un ensemble. Un Hypergraphe $H$ consiste en la donnée d'une famille $(E_i)_{i\in I}$ de parties non vides de $X$ dont la réunion sur $I$ est $X$. Plus formellement :*

1. $\forall i \in I, \quad E_i \neq \emptyset$
2. $\bigcup_{i=1} E_i = X$

Les éléments de $X$ sont, comme dans la théorie des graphes, dénommés *sommets* de l'hypergraphe, alors que les éléments $(E_i)_{i\in I}$ sont des *hyperarêtes*. Ainsi, la notion d'hypergraphe généralise bien celle de graphe puisque les relations entre les sommets ne sont plus exclusivement binaires du fait qu'aux arêtes sont substitués des ensembles de taille arbitraire.

Notons que, comme pour la notion de graphe, un hypergraphe peut a priori être défini sur un ensemble $X$ éventuellement de taille non finie. Dans la mesure où en traitement d'image, les ensembles de données traitées sont de cardinalité finie, nous nous plaçons dans ce contexte dans toute la suite de ce manuscrit.

Il est enfin possible d'associer à tout graphe $(X, E)$ un hypergraphe de la manière suivante : à chaque sommet $x$ du graphe, on peut associer le voisinage $\Gamma(x)$ de ce sommet défini comme l'ensemble :

$$\Gamma(x) = \{y \in X \quad tel \quad que \quad \{x, y\} \in E\} \tag{1.4}$$

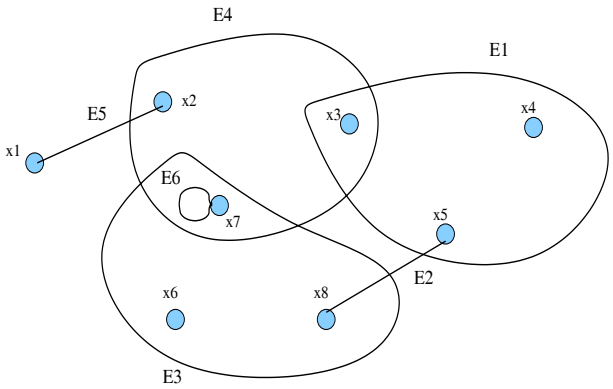
Il est à noter que cette notion de voisinage peut être étendue plus généralement à un sous-ensemble $S$ des sommets en définissant $\Gamma(s)$ comme $\bigcup_{s\in S} \Gamma(s)$.

Un tel voisinage $\Gamma(x)$ est qualifié d'ouvert, par opposition au voisinage fermé du sommet $x$ défini comme $\{x\} \cup \Gamma(x)$. Lorsque le terme non qualifié de voisinage apparaît dans ce document, il s'agit du voisinage ouvert. Il devient alors possible d'associer à un graphe $G = (X, E)$ son hypergraphe de voisinage en le définissant par :

$$H_G = (X, (\{x\} \cup \Gamma(x))_{x\in X}) \tag{1.5}$$

Nous appelons l'ensemble $\{x\} \cup \Gamma(x)$ hyperarête *engendrée* (ou *générée*) par le sommet $x$ du graphe. Ainsi, l'ensemble des sommets de l'hypergraphe de voisinage est identifié à l'ensemble des sommets du graphe. Ses hyperarêtes sont formées par les voisinages fermés de chacun de ses sommets. Remarquons qu'une telle définition est correcte puisque toute hyperarête est non vide (elle contient au moins le sommet qui l'a engendrée) et que la réunion de celles-ci est bien égale à $X$ tout entier (par double inclusion).

Dans la suite, on notera $|A|$ la cardinalité de $A$. Un hypergraphe $H$ sera souvent représenté en dessinant sur le plan des points représentant les sommets. L'arête $E_j$ sera représentée par un trait continu joignant ses deux éléments si $|E_j| = 2$ ; par une boucle si $|E_j| = 1$ ; ou par un trait plein entourant ses éléments si $|E_j| \geq 3$. On peut aussi définir un hypergraphe $H$ par sa matrice d'incidence $A = ((a_i^j))$, les colonnes représentant les arêtes $E_1, E_2, \ldots, E_m$, les rangées représentant les sommets $x_1, x_2, \ldots, x_n$ avec $a_i^j = 0$ si $x_i \notin E_j$, et $a_i^j = 1$ si $x_i \in E_j$ (Fig. 1.2).




| | E1 | E2 | E3 | E4 | E5 | E6 |
|---|---|---|---|---|---|---|
| x1 | 0 | 0 | 0 | 0 | 1 | 0 |
| x2 | 0 | 0 | 0 | 1 | 1 | 0 |
| x3 | 1 | 0 | 0 | 1 | 0 | 0 |
| x4 | 1 | 0 | 0 | 0 | 0 | 0 |
| x5 | 1 | 1 | 0 | 0 | 0 | 0 |
| x6 | 0 | 0 | 1 | 0 | 0 | 0 |
| x7 | 0 | 0 | 1 | 1 | 0 | 1 |
| x8 | 0 | 1 | 1 | 0 | 0 | 0 |

FIG. *1.2: Représentation d'un hypergraphe $H$ et sa matrice d'incidence.*

**Hyperarête isolée :** Une hyperarête $E_i$ est isolée si et seulement si :

$$\forall j \in I, j \neq i \ \ tels \ \ que \ \ E_i \cap E_j \neq \emptyset, \ \ alors \ \ E_j \subseteq E_i \tag{1.6}$$

Autrement dit, toute hyperarête distincte de $E_i$ et intersectant avec celle-ci est nécessairement incluse dans $E_i$. La figure 1.3 permet d'illustrer cette notion.

**Chaîne disjointe :** C'est une succession des hyperarêtes non connectées deux par deux. Elle est disjointe, si la cardinalité des hyperarêtes est égale à 1. Un exemple est présenté dans la figure 1.4.

Pour une représentation complète sur la théorie des hypergraphes, nous invitons le lecteur à consulter les ouvrages de Berge [Ber87b], Gondran et Minoux [GM95].

## 1.3 Image numérique

### 1.3.1 Notion d'image numérique

Il existe plusieurs types d'images, image d'intensité, image couleur, image thermique, .... En général, une image peut être décrite par une fonction à deux dimensions $f(x, y)$, avec $(x, y)$ représentant les coordonnées spatiales et $f(x, y)$ son intensité. Selon l'équation 1.7, une image

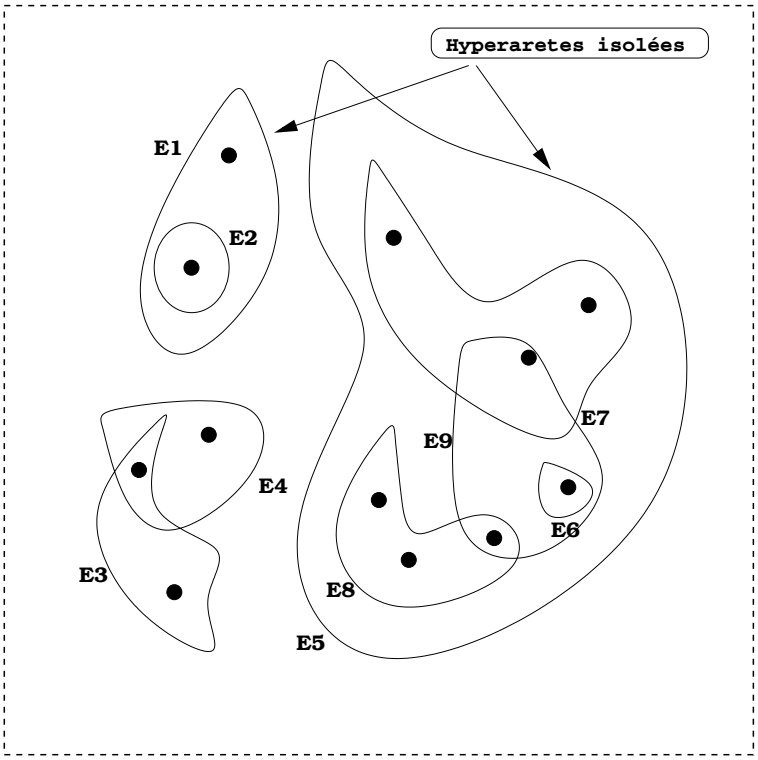



FIG. *1.3: Exemple d'hyperarêtes isolées.*

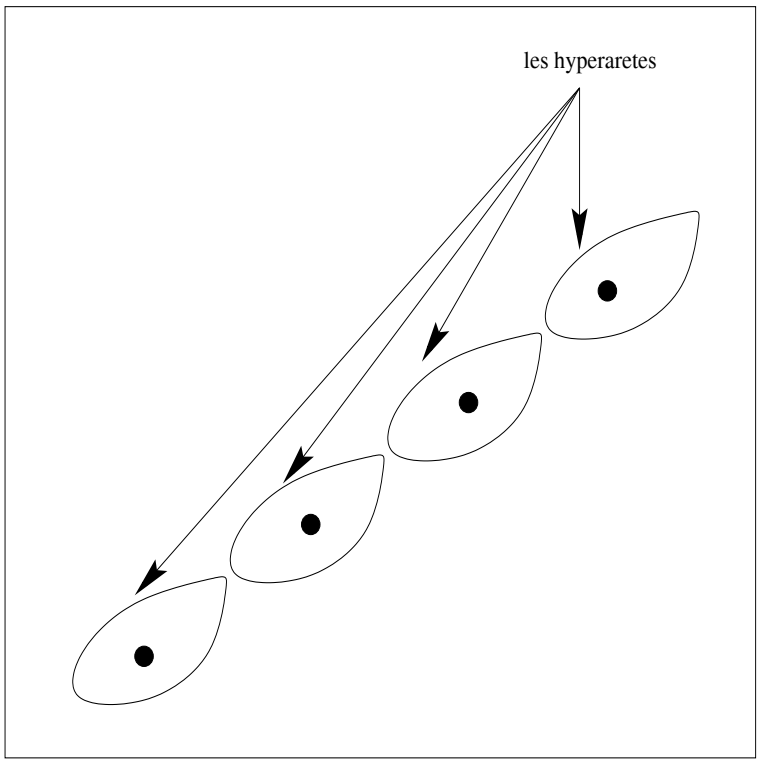



FIG. *1.4: Exemple d'une chaîne disjointe.*

numérique peut être définie comme une application $I$ d'un sous-ensemble $X$ généralement fini de $\mathbb{Z}^2$ à valeurs dans $\Omega$, sous ensemble de $\mathbb{Z}^n$ où $n$ désigne un entier naturel non nul. ([CP95]).

$$I : \quad X \subseteq \mathbb{Z}^2 \Longrightarrow \Omega \subseteq \mathbb{Z}^n \qquad avec \;\; n \geq 1 \tag{1.7}$$

Les éléments de $X$ sont appelés *pixels*, ceux de $\Omega$ caractéristiques radiométriques ou plus simplement couleur.

### 1.3.2 Pavage du plan

Un pavage de $\mathbb{R}^2$ consiste en la donnée d'une partition de $\mathbb{R}^2$. Les pavages généralement étudiés sont restreints à un nombre limité de configurations géométriques dénommées tesselles. Si cette notion est à l'origine de nombreux travaux en mathématiques, elle est aussi source d'inspiration de quelques artistes, citons le plus connu d'entre eux : M.C. Escher (Fig. 1.5). Un pavage est dit régulier lorsqu'une même tesselle est utilisée pour effectuer la partition. L'idée principale en traitement et en analyse d'images est la répétitivité à l'infini et la juxtaposition régulière des tesselles ([CM91], [Bor84], [Bor86]). Les pavages du plan les plus souvent utilisés sont les pavages carré, triangulaire et hexagonal. Le choix d'un point quelconque à l'intérieur d'une tesselle permet d'associer au pavage un maillage : deux points situés dans deux tesselles différentes sont reliés si les deux tesselles partagent un côté. En règle générale, pour les pavages réguliers, ce sont les centres de gravité des tesselles qui sont choisis comme points d'appui du maillage. Ce terme de maillage est parfois remplacé chez certains auteurs par celui de réseau ou de trame ([Jah95]). Ainsi, aux pavages carré, hexagonal et triangulaire sont respectivement associés les maillages carré, triangulaire et hexagonal comme le montre la figure 1.6. En traitement d'images, c'est l'usage des trames carrées qui prévaut à cause des impératifs technologiques imposés par l'industrie et surtout parce qu'elles permettent de structurer les données sous une forme matricielle classique. C'est la raison pour laquelle nous nous limitons dans ce document à une étude sur des structures carrées.

### 1.3.3 Grille et voisinage

Si la donnée d'une distance sur $\mathbb{Z}^2$ permet de définir un graphe non orienté, simple, sans boucle, connexe et régulier sur un maillage, cette distance sera appelée distance de grille. Ce concept permet ainsi de définir la notion de grille. C'est grâce à cette notion de grille qu'il devient dès lors possible de parler de pixel voisin d'un autre et définir le voisinage d'un pixel. Il est à noter que cette notion de voisinage est intimement liée à la définition de la distance $d$ sur $\mathbb{Z}^2$. (Fig. 1.7) :

**Définition 1.3.1** *Voisinage élémentaire d'un pixel : Soient $x$ un pixel de $X$ et $d$ une distance définissant une grille sur $\mathbb{Z}^2$, le voisinage $V(x)$ de taille 1 du pixel $x$ est défini comme :*

$$V(x) = \{y \in X \;\; tel \;\; que \;\; d(x,y) \leq 1\}$$

Il est dès lors possible d'étendre de manière constructive la définition à un voisinage $V_n$ de taille $n$ :

FIG. *1.5: Un pavage du plan : Eight heads. M.C. Escher (1939). All M.C. Escher works (c) 2001 Cordon Art - Baarn - Holland. All rights reserved*

**Définition 1.3.2** *Voisinage de taille n d'un pixel : Soient x un pixel de X, d une distance sur* $\mathbb{Z}^2$ *et n un entier naturel strictement supérieur à* 1.

1. $V_1(x) = V(x)$
2. $V_n(x) = \{y \in V_1(z), \forall z \in V_{n-1}(x)\}$

Notons qu'il est aussi possible de définir $V_n(x)$ comme l'ensemble des points de $X$ dont la distance à x est inférieure ou égale à n. Par convention, le voisinage $V_0(x)$ est réduit au singleton $x$ lui-même.

Deux distances sont principalement utilisées sur $\mathbb{Z}^2$ : la distance dite de Manhattan (city block distance ou square distance) définie comme la somme des valeurs absolues des différences des coordonnées des pixels et la distance de l'échiquier (chessboard distance ou diamond distance) définie comme le maximum des valeurs absolues des différences des coordonnées des pixels. Le voisinage élémentaire d'un pixel associé à la distance de Manhattan est classiquement appelé voisinage à 4-adjacents (dit aussi voisinage de Von Neumann), celui associé à la distance de l'échiquier 8-adjacents (appelé parfois voisinage de Moore).

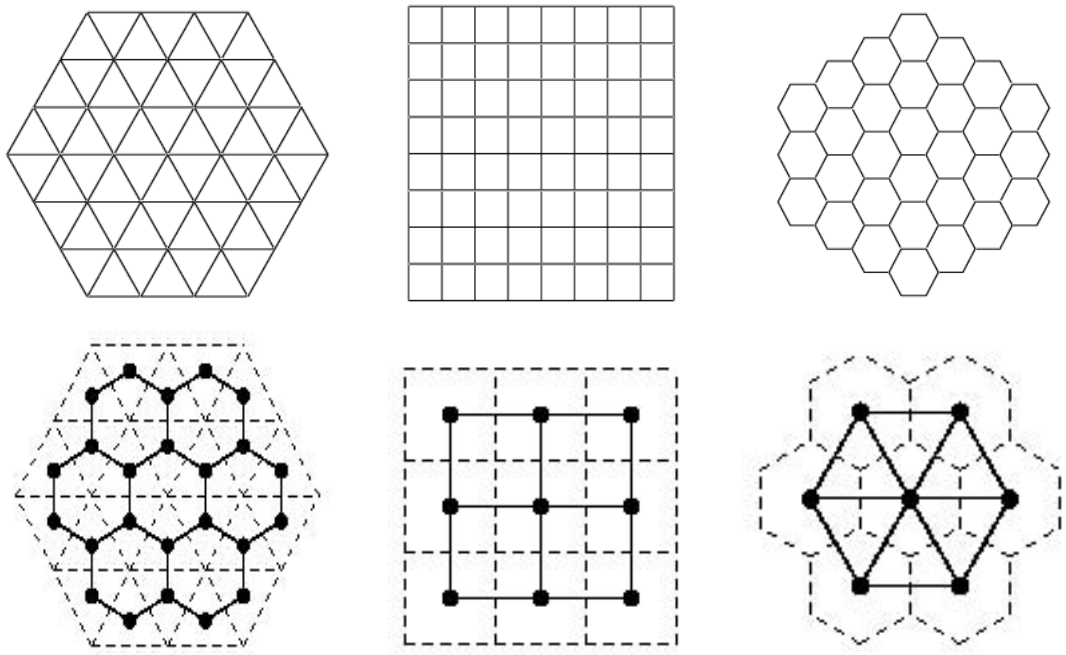


FIG. *1.6: Pavages triangulaire, carré et hexagonal du plan et maillages qui leur sont respectivement associés*

## 1.4 Hypergraphes de voisinage d'une image à niveau de gris

### 1.4.1 Principe

Considérons une distance $d'$ sur $\Omega$ et une distance de grille sur $X$. Soient $\alpha$ et $\beta$ deux réels strictement positifs. A chaque pixel $x$ de $X$, il est possible d'associer un unique voisinage $\Gamma_{\alpha,\beta}(x)$ pour l'image $I$ :

$$\Gamma_{\alpha,\beta}(x) = \left\{ y \in X, y \neq x \mid d'(\mathcal{I}(x), \mathcal{I}(y)) \leq \alpha \text{ et } d(x,y) \leq \beta \right\} \tag{1.8}$$

Nous dénommons du fait de leur domaine de définition $d$ distance de grille et $d'$ distance colorimétrique. Il est à noter que la définition de distances colorimétriques corrélées avec la perception qu'a notre cerveau, est encore aujourd'hui un domaine d'étude extrêmement complexe des neurosciences.

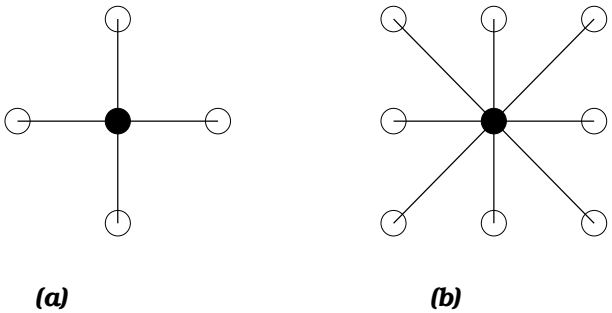



FIG. *1.7: Système de voisinage. (a) 4-adjacents et (b) 8-adjacents.*

Commentons quelque peu la définition précédente. La première partie de celle-ci fait intervenir un écart colorimétrique entre le pixel et ceux de l'image (a priori tous) alors que la seconde partie joue sur une distance inter-pixels sur la grille. Nous appellerons pour cette raison les quantités $\alpha$ et $\beta$ respectivement seuil *colorimétrique* et seuil *spatial*. Nous qualifierons de plus $\Gamma_{\alpha,\beta}$ de *voisinage spatiocolorimétrique*.

Cette notion de voisinage spatiocolorimétrique permet donc de décrire un certain niveau de cohérence (nous utiliserons aussi souvent le terme d'homogénéité) d'un pixel avec son environnement dans l'image, ceci sous réserve que les seuils $\alpha$ et $\beta$ soient suffisamment "bien" choisis et ce, au sens des deux distances considérées. Par "bien" choisis, nous entendons qu'ils soient discriminants par rapport à l'échelle des données. Par exemple, si l'on considère une image carrée à niveaux de gris (c'est-à-dire dont les intensités sont représentées par des entiers naturels de l'intervalle $[0; 255]$) de 100 pixels, un seuil $\alpha$ de 256 et un seuil $\beta$ de 10 permettront d'associer à n'importe quel pixel de l'image... tous les pixels de l'image elle-même ! Probablement peu intéressant donc pour celui qui cherche à étudier et à manipuler le contenu de l'image numérique.

Notons que ce voisinage ainsi défini possède une propriété extrêmement intéressante : sa croissance au sens de l'inclusion en $\alpha$ et en $\beta$. En effet, il est aisé de s'apercevoir que si l'on choisit deux seuils colorimétriques $\alpha$ et $\alpha'$ tels que $\alpha < \alpha'$ par exemple, nécessairement tout pixel $y$ voisin de $x$ à $\alpha$ près le sera aussi à $\alpha'$. Dès lors, si $\beta$ reste fixé, l'inclusion $\Gamma_{\alpha,\beta}(x) \subseteq \Gamma_{\alpha',\beta}(x)$ est vraie. Le raisonnement est tout à fait similaire pour le seuil spatial et il est donc possible de montrer que :

$$\forall x \in X \quad \forall(\alpha, \beta, \beta') \in \mathbb{R}^3 \;\; \beta < \beta' \Longrightarrow \Gamma_{\alpha,\beta}(x) \subseteq \Gamma_{\alpha,\beta'}(x) \tag{1.9}$$

Cette propriété de croissance des voisinages induit donc la croissance de l'hypergraphe de voisinage d'image au sens de l'inclusion des hyperarêtes. Cette propriété est remarquable pour le scientifique qui cherchera à manipuler les images numériques par cet outil : sans parler de "continuité" entre deux hypergraphes puisque l'image numérique est avant toute chose, un objet discret du fait même de sa construction, cette croissance induit au moins une certaine régularité dans le modèle. Qui n'a jamais été confronté au comportement chaotique de certains algorithmes dont les paramètres n'avaient pourtant été que faiblement modifiés ?

A l'aide des définitions précédentes, la définition de l'hypergraphe de voisinage d'une image est dès lors possible ([BACL97]). Si, à l'ensemble des sommets $X$, on associe l'ensemble des hyperarêtes définies par la caractérisation de l'équation 1.8, nous obtenons un hypergraphe de voisinage que nous notons $H_{\alpha,\beta}$ :

$$H_{\alpha,\beta} = (X, (\{x\} \cup \Gamma_{\alpha,\beta}(x))_{x \in X}) \tag{1.10}$$

Il est utile de préciser que cette définition correspond bien à celle d'un hypergraphe puisque chacune des hyperarêtes est non vide : elle contient au moins le pixel $x$. Nous appellerons par ailleurs $E_{\alpha,\beta}(x)$ hyperarête centrée en $x$ (ou engendrée par $x$). Enfin, pour terminer la preuve que cette définition est bien celle d'un hypergraphe, du fait que : $\forall x \in X, \{x\} \subset (\{x\} \cup \Gamma_{\alpha,\beta}(x)) \subset X$, on a bien :

$$\bigcup_{x \in X} \{x\} \subset \bigcup_{x \in X} \{x\} \cup \Gamma_{\alpha,\beta}(x) \subset X \tag{1.11}$$

et $\bigcup_{x \in X} = X$, alors l'équation 1.11 devient :

$$X \subset \bigcup_{x \in X} \{x\} \cup \Gamma_{\alpha,\beta}(x) \subset X \tag{1.12}$$

et donc :

$$\bigcup_{x \in X} \{x\} \cup \Gamma_{\alpha,\beta}(x) = X \quad (par \quad double \quad inclusion) \tag{1.13}$$

### 1.4.2 Choix du paramètre $\alpha$

L'hypergraphe de voisinage associé à une image est fonction de la distance utilisée $d'(I(x), I(y))$ et du seuil de décision $\alpha$. Le seuillage peut s'effectuer de deux manières. Dans le premier cas, le seuil est déterminé globalement pour tous les pixels de l'image, alors que dans le second cas, le seuil est généré localement puis appliqué de manière adaptative à l'ensemble des pixels. Les techniques de seuillage global s'avèrent non satisfaisantes car elles ne prennent pas en compte la non stationnarité des propriétés des images. Pour pallier à cet inconvénient, les méthodes de seuillage locales utilisent des mesures des propriétés locales de l'image.

Le seuillage de l'image est un domaine qui n'a pas cessé de se développer depuis les premiers travaux de Weszka et Rosenfeld [WR78].

Une classification réalisée par Weszka [Wes] divise les méthodes de seuillage en trois classes. Une opération de seuillage peut être représentée par une transformation $T$ dépendant des trois paramètres ci-dessous :

- $(i, j)$ : les coordonnées cartésiennes d'un pixel donné ;
- $I_{i,j}$ : la luminance de ce pixel ;
- $Z(i, j)$ : un ensemble de propriétés dans un voisinage donné de ce pixel.

Weszka donne une classification selon le nombre de paramètres pris en compte. Si seule $I_{i,j}$ est prise en compte, on parle de méthodes globales. Leur principe est de seuiller l'ensemble de l'image avec un seuil identique obtenu à partir d'une mesure globale (tel l'histogramme des niveaux de gris). Si à la fois $I_{i,j}$ et $Z(i, j)$ sont prises en compte, les méthodes correspondantes sont dites locales. Le seuil optimum est fonction de $I_{i,j}$ et des propriétés locales choisies. Enfin, lorsque tous les paramètres sont considérés, les méthodes sont dites dynamiques. Elles seuillent l'image avec un seuil choisi selon une mesure locale telle que le contraste, l'uniformité, le gradient .... Dans ces méthodes, la classification d'un pixel $(i, j)$ dépend de ses voisins.

**Seuil global.** Dans ce cas, le seuil global $\alpha$ est déterminé pour tous les pixels de l'image $I$. Supposons que $\beta = 1$ (Voisinage 8-Adjacents), et que l'ensemble des valeurs $I_i$ avec $i = 1, \ldots, 8$ représente les valeurs susceptibles d'appartenir à l'hyperarête $E_x$ engendrée par $x$. Alors pour qu'un pixel $I_i \in E_x$, il faut qu'il vérifie la condition : $|I_i - I_x| \leq \alpha$.

**Seuil dynamique.** Dans ce cas, le paramètre de seuillage $\alpha$ est déterminé en fonction à la fois de $(i, j)$, $I_{i,j}$ et des propriétés du voisinage $Z_{(i,j)}$. Dans la littérature, il existe plusieurs méthodes de seuillage dynamique [CP95][Ros98][NBBB99][SS03]. Dans ce paragraphe, nous présentons deux de ces méthodes.

Une méthode de seuillage utilisant le gradient de l'image est définie dans [NBBB99]. Soit $W_{i,j}$ la fenêtre d'analyse centrée en $(i, j)$ : pour chaque pixel $(i, j)$, le seuil dynamique est défini par :

$$\hat{\tau}_{i,j} = \sum_{\{m,n\} \in W_{ij}} \phi(\Delta_{m,n}) I_{m,n} / \sum_{\{m,n\} \in W_{ij}} \phi(\Delta_{m,n}) \tag{1.14}$$

$\hat{\tau}_{i,j}$ est le niveau de gris moyen des contours estimé dans la fenêtre d'analyse $W_{ij}$. L'intérêt de la fonction de pondération $\phi(\Delta_{m,n})$ , qui dépend du gradient $\Delta_{m,n}$ est de donner plus de poids aux points du contours. Le choix de cette fonction peut se faire de manière heuristique en

exigeant simplement un poids plus important pour les pixels contours et un poids moindre pour les éléments situés à l'intérieur des zones homogènes. La fonction utilisée dans cette méthode est simplement le module du gradient obtenu par application d'un des opérateurs classiques et simples tels que le Sobel, les opérateurs de Rosenfeld, Prewit, etc. [RZ82] [CP95].

Pour chaque couple $(i, j)$, on calcule $\hat{\tau}_{ij}$ au moyen de la relation 1.14 et on effectue l'opération de seuillage donnée ci-dessous :

$$I_{i,j} \in E_{\alpha,\beta} \quad si \quad d'(I(x), I(y)) \leq \hat{\tau}_{i,j} \quad sinon \quad I_{i,j} \notin E_{\alpha,\beta} \tag{1.15}$$

On peut par ailleurs utiliser un seuillage basé sur la dispersion des données. Pour chaque couple $(i, j)$, on calcule l'écart-type $\sigma$ du voisinage. Ce dernier est défini par :

$$\hat{\sigma} = \sqrt{\frac{1}{N} \times \sum (I_{ij} - moy)^2} \tag{1.16}$$

où, *moy* est la moyenne locale d'intensité des pixels. Le seuillage est présenté par :

$$I_{i,j} \in E_{\alpha,\beta} \quad si \quad d'(I(x), I(y)) \leq \sigma_{i,j} \quad sinon \quad I_{i,j} \notin E_{\alpha,\beta} \tag{1.17}$$

### 1.4.3 Mesure colorimétrique

Pour analyser les proximités entre les caractéristiques des pixels, on utilise des mesures qui peuvent être classées en deux catégories : dissimilarité et similarité. Par définition, une mesure de similarité ou dissimilarité est toute application à valeurs numériques qui permet de mesurer le lien entre les individus d'un même ensemble ou entre les variables. Pour une similarité le lien est d'autant plus fort que sa valeur est grande.

– Une mesure de similarité sur un ensemble $X$ est une application $\mu$ de $X \times X$ dans $\mathbb{R}^+$ qui vérifie les deux conditions suivantes :

$$\mu \quad \text{symétrique} \quad \forall (x, x') \in X \times X; \quad \mu(x, x') = \mu(x', x) \tag{1.18}$$

$$\forall (x, x') \in X \times X \quad \text{avec} \quad x \neq x' \quad \mu(x, x) = \mu(x', x') > \mu(x, x'). \tag{1.19}$$

– Une mesure de dissimilarité est une application $d$ qui satisfait à la condition 1.18 et à 1.20 qui suit :

$$\forall x \in X \quad d(x, x) = 0. \tag{1.20}$$

– Une distance est une mesure de dissimilarité qui vérifie l'inégalité triangulaire :

$$\forall \text{ x,x' et x" } \in X, \quad d(x, x") \leq d(x, x') + d(x', x"). \tag{1.21}$$

Les mesures normalisées produisent des valeurs dans l'intervalle $[0, 1]$. L'interprétation de ces valeurs dépend de la mesure utilisée. Dans le cas d'une fonction de dissimilarité, les pixels qui sont proches, auront une valeur égale à 0 (dissimilarité faible ou similarité élevée). Dans le cas d'une fonction de similarité, les pixels qui sont très éloignés, auront une valeur 0 (dissimilarité élevée ou faible similarité). La figure 1.8 illustre la différence entre les mesures de dissimilarité et similarité.

Les mesures de similarité peuvent être converties en des mesures de dissimilarité, et inversement, en appliquant une transformation appropriée $T$. Si $d_{norm}$ est une mesure de dissimilarité normalisée entre deux vecteurs, alors la mesure de similarité équivalente $\mu$ est donnée par :

$$\mu = T(d_{norm}) \tag{1.22}$$

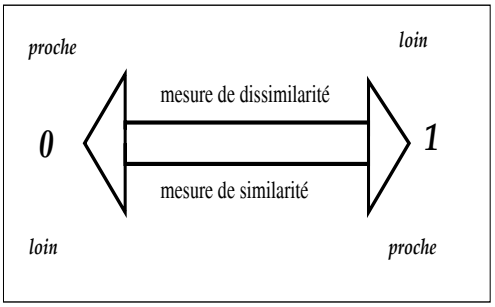



FIG. *1.8: Les mesures de similarité pour des pixels proches retournent une valeur 1. Cependant les mesures de dissimilarité retournent une valeur égale à 0.*

Citons comme exemple, le cas d'une simple transformation définie par :

$$\mu = 1 - d_{norm} \tag{1.23}$$

La détermination d'une mesure appropriée conditionne la qualité de la représentation par hypergraphe. Dans la littérature, il existe plusieurs mesures de dissimilarité entre vecteurs, citons par exemple les distances de Canberra et de Minkowski généralisées (Ces distances seront détaillées dans la section 1.7.2). Il est à noter que ces mesures ne sont pas normalisées.
Dans ce qui suit, nous donnons une description succincte des mesures de similarité [SPC+02]. En effet, nous utilisons ces fonctions de similarité pour générer une représentation par hypergraphe de voisinage d'une image.

Le problème de la définition d'une mesure de similarité entre deux pixels peut se ramener à une utilisation d'une fonction symétrique, où les valeurs de cette dernière sont larges quand deux pixels donnés sont similaires. Une fonction de similarité $\mu$ est définie par :

$$\mu : [0; \infty) \longrightarrow \mathbb{R} \tag{1.24}$$

Les fonctions de similarité doivent vérifier les trois conditions suivantes :

$$\begin{array}{ll} 1. & \mu \text{ est monotone décroissante dans } [0; \infty) \\ 2. & \mu(0) = 1 \\ 3. & \mu(\infty) = 0 \end{array} \tag{1.25}$$

Le fait que $\mu$ doit être non croissant, signifie que la similarité entre deux pixels est petite si la distance entre eux est grande. La deuxième propriété est une normalisation de la fonction de similarité. De cette façon, la similarité entre deux pixels avec le même niveau de gris est égale à 1, et la similarité entre deux pixels avec des intensités différentes est égale à 0. Pour une image à niveaux de gris, la mesure de similarité entre deux pixels est :

$$\begin{array}{l} L_1 : \mu(|I(x) - I(y)|) \\ L_2 : \mu(|I(x) - I(y)|^2) \end{array}$$

Pour que tel pixel $I_i \in E_{\alpha,\beta}(x)$, il faut qu'il vérifie la condition :

$$I_i \in E_{\alpha,\beta}(x) \quad if \quad d'(I_i, I_x) = \mu(I_i, I_x) \geq \alpha. \tag{1.26}$$

Dans cette représentation, la valeur de $\alpha$ doit être comprise entre 0 et 1.

### 1.4.4 Algorithme de construction de l'hypergraphe de voisinage d'une image à niveaux de gris

Ci-après, nous présentons un algorithme[1] de génération de trois représentations par hypergraphe de voisinage de l'image à niveaux de gris. Ces trois représentations diffèrent par les paramètres utilisés. La première représentation (nommée dans la suite par représentation A) utilise un seuillage global, la deuxième un seuillage dynamique (représentation B), et la dernière utilise des fonctions de similarité.

La représentation A utilise un seuil fixe $\alpha$ déterminé pour toute l'image $I$, une distance sur la grille $\beta$ et une distance d'intensité $d'$ (mesure de dissimilarité).

**Algorithme de construction de la représentation A**
(Données : Image $I$, $\alpha$ fixe, $\beta$)

- Construction de l'hypergraphe de voisinage $H_{\alpha,\beta}$ sur $I$
  - a. $X \longleftarrow \emptyset$
  - b. Pour chaque pixel $(x, I(x))$ de $I$
    1. $\Gamma_{\alpha,\beta}(x) \longleftarrow \emptyset$
    2. pour $(y, I(y))$ de $\Gamma_\beta(x)$, si $d(I(x), I(y)) \leq \alpha$ alors $\Gamma_{\alpha,\beta}(x) \longleftarrow \Gamma_{\alpha,\beta}(x) \cup \{y\}$
    3. $X \longleftarrow X \cup \{x\}$
    4. $E_{\alpha,\beta}(x) \longleftarrow \{\Gamma_{\alpha,\beta}(x) \cup \{x\}\}$
  - c. $H_{\alpha,\beta}(x) \longleftarrow (X, (E_{\alpha,\beta}(x))_{x\in X})$
- Codage de l'hypergraphe.

La deuxième représentation emploie un seuil dynamique $\alpha$. Pour chaque couple $(i, j)$, on estime l'écart type $\hat{\sigma}$ de son voisinage 8-Adjacents. Dans ce cas, la distance d'intensité (mesure de dissimilarité) entre pixels $I(x)$ et $I(y)$ est comparée avec $\hat{\sigma}$.

**Algorithme de construction de la représentation B**
(Données : Image $I$, $\beta$)

- Construction de l'hypergraphe de voisinage $H_{\alpha,\beta}$ sur $I$
  - a. $X \longleftarrow \emptyset$
  - b. Pour chaque pixel $(x, I(x))$ de $I$
    1. calcul de l'écart-type $\hat{\sigma}$ ;
    2. $\Gamma_{\alpha,\beta}(x) \longleftarrow \emptyset$
    3. pour $(y, I(y))$ de $\Gamma_\beta(x)$, si $d'(I(x), I(y)) \leq \hat{\sigma}$ alors $\Gamma_{\alpha,\beta}(x) \longleftarrow \Gamma_{\alpha,\beta}(x) \cup \{y\}$
    4. $X \longleftarrow X \cup \{x\}$
    5. $E_{\alpha,\beta}(x) \longleftarrow \{\Gamma_{\alpha,\beta}(x) \cup \{x\}\}$
  - c. $H_{\alpha,\beta}(x) \longleftarrow (X, (E_{\alpha,\beta}(x))_{x\in X})$
- Codage de l'hypergraphe.

La troisième représentation emploie un seuil fixe $\alpha$ mais à la place d'une distance colorimétrique $d'(I(x), I(y))$, on utilise des mesures de similarité. Ces mesures sont générées à l'aide des fonctions de similarité $\mu(x)$.

**Algorithme de construction de la représentation C**
(Données : Image $I$, $\alpha$, $\beta$, fonction de similarité $\mu(x)$)

- Construction de l'hypergraphe de voisinage $H_{\alpha,\beta}$ sur $I$

[1] Remarque : pour ces trois représentations, l'algorithme est le même, ce qui change c'est seulement les paramètres de la génération de l'hypergraphe de voisinage.

- a. $X \longleftarrow \emptyset$
- b. Pour chaque pixel $(x, I(x))$ de $I$
  1. $\Gamma_{\alpha,\beta}(x) \longleftarrow \emptyset$
  2. pour $(y, I(y))$ de $\Gamma_\beta(x)$, si $\mu(I(x), I(y)) \geq \alpha$ alors $\Gamma_{\alpha,\beta}(x) \longleftarrow \Gamma_{\alpha,\beta}(x) \cup \{y\}$
  3. $X \longleftarrow X \cup \{x\}$
  4. $E_{\alpha,\beta}(x) \longleftarrow \{\Gamma_{\alpha,\beta}(x) \cup \{x\}\}$
- c. $H_{\alpha,\beta}(x) \longleftarrow (X, (E_{\alpha,\beta}(x))_{x \in X})$

• Codage de l'hypergraphe.

Dans le cas de l'image à niveaux de gris, $I(x)$ est un scalaire. L'image couleur par contre est une image à trois composantes. L'élaboration de la notion d'homogénéité est dans ce cas plus complexe. Dans les sections suivantes, nous présentons les possibilités de génération de l'hypergraphe de voisinage d'une image couleur. Nous présentons par la suite, les espaces de représentation couleur et les stratégies de traitements associées. Ceci nous conduit à définir les hypergraphes de voisinage qui peuvent être associés aux images couleur.

## 1.5 Représentation de la couleur

Dérivés de domaines techniques (télévision, ...) ou développés spécifiquement pour le traitement des images, un certain nombre d'espaces couleur ont été proposés, chacun correspondant généralement à un objectif particulier. Hormis l'espace RVB, ces espaces peuvent être classés en deux grandes catégories :

• la première catégorie cherche à définir des espaces où la distance euclidienne entre deux couleurs reste proche de la différence perçue par l'être humain. Parmi ces espaces, appelés perceptuellement uniformes, les plus utilisés sont les espaces CIELab et CIELuv [CD98]. La plupart du temps, ces bases demandent la connaissance des conditions d'acquisition des images (illuminant de référence, ...).

• la seconde catégorie vise à construire trois nouvelles composantes exprimant approximativement les notions psychophysiques de Teinte, Luminance et Saturation facilement perçues par l'être humain. Là encore, un certain nombre de formulations ont été proposées [CD98].

Le choix entre les différentes bases existantes va dépendre d'un ensemble de facteurs parmi lesquels on retrouve essentiellement la nature de l'application (contrôle colorimétrique, segmentation, ...), mais aussi la réversibilité de cette transformation, la connaissance des conditions d'acquisition. Un certain nombre de travaux ont analysé la manière de choisir l'espace de représentation en fonction des images analysées [Que97][Cha98]. Nous présentons dans ce qui suit quelques uns des espaces les plus couramment utilisés en traitement d'images.

### 1.5.1 L'espace RVB

Dans le champ d'application de l'imagerie au sens large du terme, on a l'habitude de manipuler les trois composantes $RVB$ d'une couleur en tant que trois entiers codés sur 8 bits. Ceci principalement parce que cela correspond à la manière dont les cartes d'acquisition et les cartes vidéo échangent leurs informations avec les périphériques - écran ou caméra. Cependant, ces composantes ne nous donnent pas d'information sur la couleur elle-même. Le modèle $RVB$ utilise le système de coordonnées cartésiennes. La diagonale du blanc $(1, 1, 1)$ au noir $(0, 0, 0)$ représente les niveaux de gris.

### 1.5.2 Espace couleur uniforme

#### 1.5.2.1 L'espace CIE 1931 XYZ

Les résultats issus lors de l'expérience déterminant le codage photopique, permettent de déterminer les fonctions d'appariement des couleurs. Cependant, pour certaines valeurs du spectre, nous obtenons une valeur négative de la primaire rouge. Même s'il est possible de traiter des valeurs négatives en colorimétrie par une désaturation du stimulus, la CIE a décidé de produire un espace dans lequel les valeurs négatives n'apparaissent plus. De nouvelles primaires ont donc été introduites : les primaires imaginaires de la CIE. La gamme des couleurs produite par le mélange des primaires imaginaires couvre toutes les couleurs réelles. Ainsi, en 1931, la CIE adopte un nouvel espace couleur, pour tenter de caractériser la réponse visuelle, appelé l'espace couleur CIE 1931 XYZ. La transformation colorimétrique entre l'espace CIE $XYZ$ et l'espace CIE $RVB$ est donnée par :

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.78502 & 0.2726 & 0.06527 \\ 0.28303 & 0.7134 & 0.003518 \\ 0 & 0.00896 & 0.3221 \end{bmatrix} \begin{bmatrix} R \\ V \\ B \end{bmatrix} \tag{1.27}$$

Les trois stimulus X, Y et Z représentent les quantités relatives des primaires imaginaires requises pour recréer chaque couleur en utilisant un mélange additif de couleurs. Cet espace présente la propriété qu'à un niveau constant de luminance, une distance égale entre deux paires de points dans des régions différentes du diagramme ne correspond pas nécessairement à une distance égale de perception de la couleur. Cette non uniformité [Mac85] est plus marquée pour une région de vert que pour une région de bleu ou de rouge.

#### 1.5.2.2 L'espace CIE 1960 $UVW$.

Ainsi, en 1960 la CIE recommande la transformation du plan de chromaticité afin de rendre ce dernier plus homogène au sens de la différence perceptuelle de couleur. L'espace CIE 1960 $UVW$ dont la transformation est issue de l'espace $XYZ$ est défini comme suit :

$$\begin{bmatrix} U \\ V \\ W \end{bmatrix} = \begin{bmatrix} 2/3 & 0 & 0 \\ 0 & 1 & 0 \\ -1/2 & 3/2 & 1/2 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{1.28}$$

Dans cet espace, l'uniformisation du plan de chrominance a été rendue possible en transformant les ellipses de MacAdam en cercles de rayon relativement homogène.

Les systèmes colorimétriques non linéaires, introduits par la CIE, tendent tous d'inclure la dimension de luminance dans le calcul de différence couleur[2].

[2]Pendant des années, plusieurs propositions ont été faites pour parfaire ce calcul.

#### 1.5.2.3 L'espace CIE 1976 $L^*a^*b^*$.

En 1976, la CIE propose l'espace couleur $L^*a^*b^*$, appelé aussi le CIE 1976 $L^*a^*b^*$, dont les transformations, issues de l'espace $XYZ$, sont définies par :

$$
\begin{aligned}
L^* = & \begin{cases} 116(\frac{Y}{Y_0})^{1/3} - 16 & \forall \ \frac{Y}{Y_0} \leq 0.008856 \\ 903.3(\frac{Y}{Y_0}) & sinon \end{cases} \\
a^* = & \begin{cases} 500\frac{X}{X_0} - 500\frac{Y}{Y_0} & si \ \frac{X}{X_0} \leq 0.008856 \ et \ \frac{Y}{Y_0} \leq 0.008856, \\ 500\left(\left(\frac{X}{X_0}\right)^{1/3} - \left(\frac{Y}{Y_0}\right)^{1/3}\right) & si \ \frac{X}{X_0} > 0.008856 \ et \ \frac{Y}{Y_0} > 0.008856, \\ 500\left(\frac{X}{X_0}\right)^{1/3} - 500\frac{Y}{Y_0} & si \ \frac{X}{X_0} > 0.008856 \ et \ \frac{Y}{Y_0} \leq 0.008856, \\ 500\frac{X}{X_0} - 500\left(\frac{Y}{Y_0}\right)^{1/3} & si \ \frac{X}{X_0} \leq 0.008856 \ et \ \frac{Y}{Y_0} > 0.008856, \end{cases} \\
b^* = & \begin{cases} 200\frac{Y}{Y_0} - 200\frac{Z}{Z_0} & si \ \frac{Y}{Y_0} \leq 0.008856 \ et \ \frac{Z}{Z_0} \leq 0.008856, \\ 200\left(\left(\frac{Y}{Y_0}\right)^{1/3} - \left(\frac{Z}{Z_0}\right)^{1/3}\right) & si \ \frac{Y}{Y_0} > 0.008856 \ et \ \frac{Z}{Z_0} > 0.008856, \\ 200\left(\frac{Y}{Y_0}\right)^{1/3} - 200\frac{Z}{Z_0} & si \ \frac{Y}{Y_0} > 0.008856 \ et \ \frac{Z}{Z_0} \leq 0.008856, \\ 200\frac{Y}{Y_0} - 200\left(\frac{Z}{Z_0}\right)^{1/3} & si \ \frac{Y}{Y_0} \leq 0.008856 \ et \ \frac{Z}{Z_0} > 0.008856, \end{cases}
\end{aligned}
\tag{1.29}
$$

Dans ces équations, les grandeurs $X_0$, $Y_0$ et $Z_0$ représentent le tristimulus du blanc de référence choisi dans le référentiel XYZ.

Cet espace couleur est une approximation d'un espace couleur uniforme dans lequel l'amplitude perceptuelle de la couleur est définie en terme d'échelles de couleur opposées. Dans un tel espace, la différence de couleur est représentée par une échelle numérique de différences de couleur entre une paire de stimuli tridimensionnels. La différence calculée est associée à l'amplitude visuelle perçue entre les paires d'échantillons. Ainsi, dans ce type d'espace, la mesure de la distance colorimétrique couleur tente de se rapprocher de la différence perceptuelle de couleur. En effet, la fonction racine cubique introduite dans les formules de calcul des axes permet de corriger l'effet de dispersion intra-oculaire.

#### 1.5.2.4 L'espace CIE 1976 $L^*u^*v^*$

La même année, l'espace Luv basé sur le système colorimétrique de MUNSELL est également devenu un standard. Ce modèle de représentation fournit une mesure de couleur simple, calculable en accord avec le système de représentation couleur de MUNSELL [Kun93]. Dans le système colorimétrique de MUNSELL, les couleurs sont identifiées à l'aide de trois composantes qui sont la teinte, la saturation et l'intensité (Hue, Chroma and Value). L'espace $L^*u^*v^*$ est défini à partir

de l'espace XYZ par les équations suivantes :

$$\begin{array}{rcl} L^* & = & \begin{cases} 116(\frac{Y}{Y_0})^{1/3} - 16 & \forall \ \frac{Y}{Y_0} \leq 0.008856 \\ 903.3(\frac{Y}{Y_0}) & sinon \end{cases} \\ u^* & : = & 13L^*(4X/A - 4X_0/A_0), \\ v^* & : = & 13L^*(9X/A - 9X/A_0) \end{array} \tag{1.30}$$

avec : $A = X + 15Y + 3Z$ et $A_0 = X_0 + 15Y_0 + 3Z_0$. Dans l'équation 1.30, les grandeurs $X_0$, $Y_0$ et $Z_0$ représentent le tristimulus du blanc de référence choisi dans le référentiel XYZ.

Les différences de couleur calculées dans ces deux espaces tentent de représenter les différences de couleur perçues par l'être humain. La supériorité, en terme de différence de couleur, d'un espace par rapport à l'autre n'a jamais été démontrée. Ces deux espaces colorimétriques sont utilisés en tant qu'espaces couleur uniformes. Cependant, même si les espaces couleur parfaitement uniformes ne peuvent être atteints dans un modèle euclidien tridimensionnel, les experts de la couleur continuent de travailler au développement d'un espace donnant, de façon substantielle, une meilleure corrélation avec le jugement visuel.

### 1.5.3 Espace couleur perceptuel

L'espace perceptuel $L^*H^*C^*$ [Cel95],[TLL95] tente de rendre plus homogène les régions colorimétriques. Il dérive de l'espace $L^*a^*b^*$ en utilisant les formules suivantes :

$$\begin{array}{l} L = L^*, \\ H^* = \begin{cases} 0 & si \ \ a^* = 0 \\ \arctan\left(\frac{b^*}{a^*}\right) & si \ \ a^* > 0 \ \ et \ \ b^* \geq 0 \\ \arctan\left(\frac{b^*}{a^*}\right) + 2\pi & si \ \ a^* > 0 \ \ et \ \ b^* > 0 \\ \arctan\left(\frac{b^*}{a^*}\right) + \pi & si \ \ a^* < 0 \end{cases} \\ C^* = \sqrt{a^{*2} + b^{*2}} \end{array} \tag{1.31}$$

Cet espace fait partie de la famille des espaces dits de "Luminance, teinte, saturation", où :

- la luminance représente l'intensité de la couleur,
- la teinte représente la couleur (rouge, vert, jaune, turquoise,etc.),
- la saturation représente le degré de pureté de la couleur.

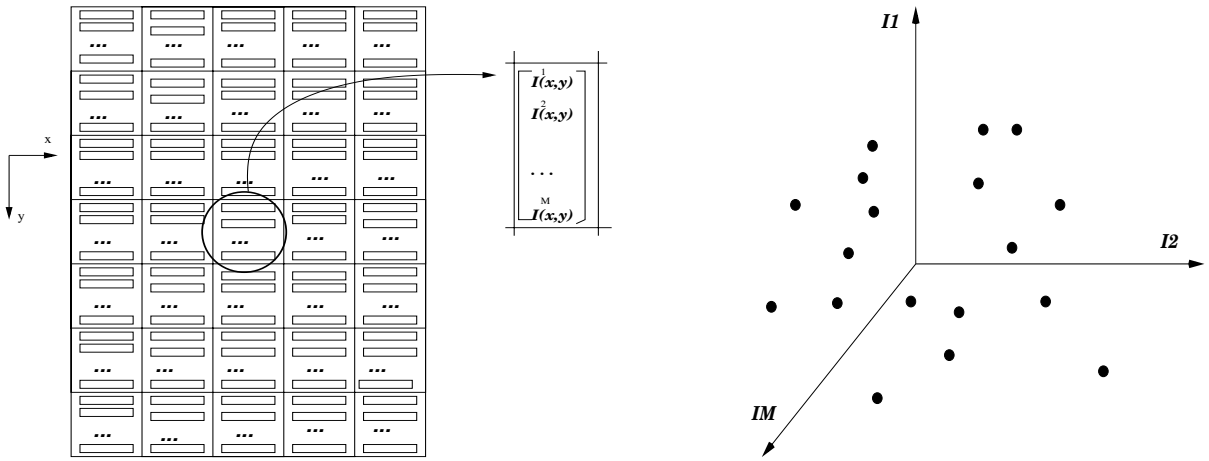
## 1.6 Traitement d'image multicomposante

Dans cette section, nous donnons une brève synthèse des principales caractéristiques rencontrées dans les approches multicomposantes. Cette synthèse a été élaborée en s'appuyant sur l'analyse des travaux consacrés à ce sujet dans [Lam02].

Nous différencions principalement trois types de représentation : la représentation "multiscalaire", la représentation "volumique" et la représentation "vectorielle" [Lam02]. Dans notre cas, nous nous intéressons à la représentation vectorielle de l'image multicomposante, et particulièrement l'image couleur. Le choix entre ces différentes représentations dépend essentiellement de

la nature de l'image. Mais, dans la mesure où une image peut accepter plusieurs représentations, ce choix pourra parfois être lié à la stratégie de traitement envisagée.

L'illustration graphique de la représentation vectorielle peut se faire en considérant une image plane où chaque pixel est associé à un vecteur (Fig. 1.9-a). Ce type de représentation s'accompagne également de représentations dans l'espace des composantes (Fig. 1.9-b), où l'on perd alors l'information sur la disposition spatiale des pixels.




(a) Image $7 \times 5$ (b) Espace des composantes

FIG. *1.9: Représentation vectorielle dans le cas d'une image* $7 \times 5$.

### 1.6.1 Les stratégies de traitement

Selon l'importance accordée à la prise en compte de l'aspect multicomposante, on peut distinguer deux grands types de stratégie dans l'élaboration des traitements :

- **les stratégies marginales** (ou scalaires) où le contexte multicomposante ne sera pas exploité ;
- **les stratégies vectorielles**, élaborées en tenant compte de la nature multicomposante des images.

**Stratégie marginale :** Cette stratégie, qui s'appuie sur la représentation multiscalaire, consiste à traiter chaque composante séparément en utilisant des traitements monocomposantes, empruntés à l'imagerie monochrome (Fig. 1.10). De part sa structure, cette approche ignore totalement la dépendance pouvant exister entre les différentes composantes, délaissant ainsi une information pouvant participer à l'amélioration des performances des traitements. Cette stratégie demande également autant de traitements qu'il y a de composantes, ce qui peut se révéler coûteux en temps de calcul. L'intérêt de cette approche est qu'elle n'utilise que des traitements scalaires. Toutes les méthodes définies en imagerie monochrome sont alors directement exploitables, sans aucune adaptation. On peut, bien évidemment, employer des traitements différents pour chaque composante, en fonction de la nature de ces composantes.

**Stratégie vectorielle :** L'alternative à cette approche est la stratégie vectorielle qui traite de manière globale l'ensemble des composantes. Cette stratégie est illustrée dans la figure 1.11. Cette stratégie est a priori plus satisfaisante du point de vue de la prise en compte du contexte

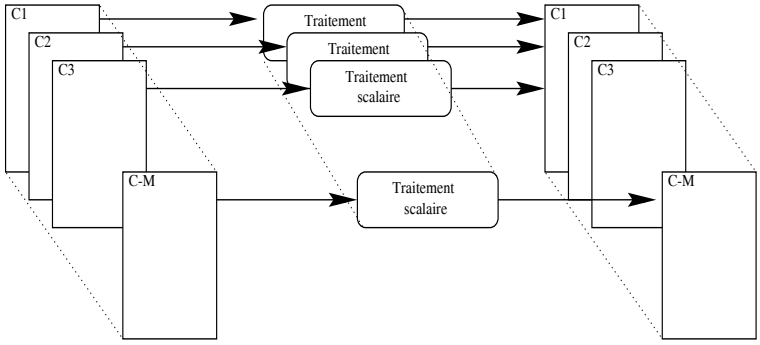



FIG. *1.10: Stratégie marginale.*

multicomposante. Cette approche présente également l'intérêt de ne nécessiter qu'un seul traitement, quel que soit le nombre de composantes. Cet avantage doit être modéré par une complexité accrue de ce seul traitement, complexité directement dépendante de composantes et des interactions entre composantes à l'intérieur des traitements.

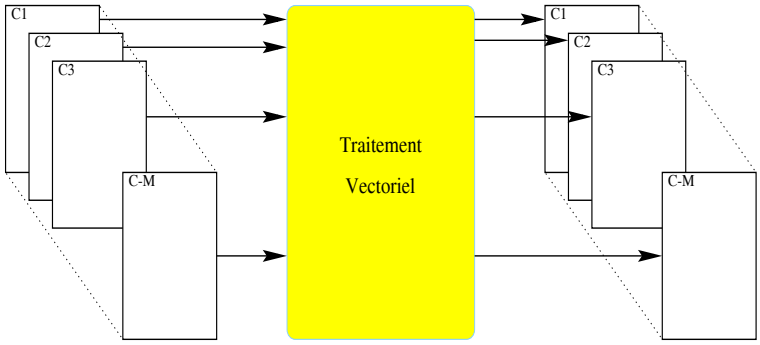



FIG. *1.11: Approche vectorielle.*

### 1.6.2 Les espaces de représentation et de traitement

Il est assez naturel de se poser la question du choix de l'espace dans lequel les traitements vont être effectués. Faut-il travailler dans l'espace initial où sont obtenues les données, ou, au contraire, faut-il chercher un espace mieux adapté aux traitements envisagés ?

La réponse à cette question n'est pas unique et dépend bien sûr de la nature des images multicomposantes et des traitements que l'on désire effectuer. Les deux raisons essentielles suscitant la recherche d'un nouvel espace de travail sont la volonté de réduire la dimensionnalité de l'espace initial des données ou bien la recherche d'un espace mettant en relief les informations nécessaires à l'élaboration des traitements. La réversibilité de la transformation permettant de se projeter dans le nouvel espace peut être un élément guidant le choix de la transformation, car elle permet, après traitement, de retrouver une image de même nature que l'image initiale. La figure 1.12 illustre ce principe. La représentation utilisée est la représentation multi-scalaire et le traitement est vectoriel, mais cette manière de procéder peut naturellement s'adapter à toute représentation, et à toute stratégie de traitement.

Le choix de l'espace de travail fait apparaître trois grandes familles de solutions :

- la solution consistant à rester dans l'espace initial ;

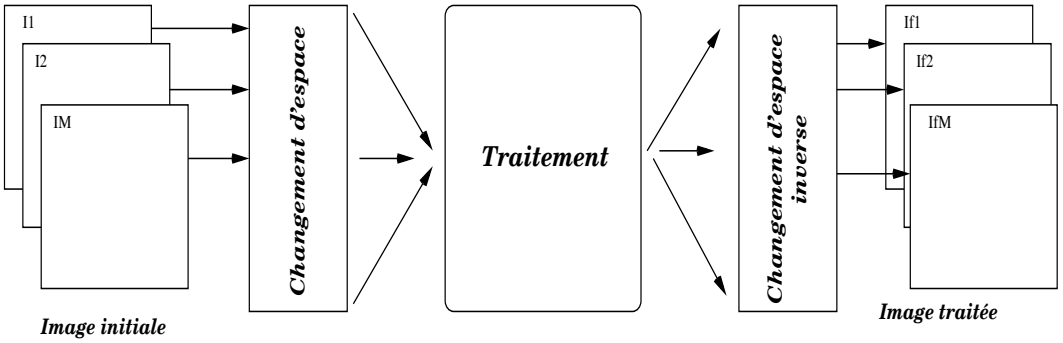



FIG. *1.12: Traitement après projection dans un espace de travail approprié.*

- une famille de solutions visant à réduire la dimensionnalité de l'espace initial et cherchant généralement à décorréler les données ;
- et enfin, la recherche d'un espace adapté à la nature des données.

**L'espace initial.** Un grand nombre de traitements sont effectués dans la base où les données sont acquises. Cette manière de procéder présente un double avantage. D'une part, elle garantit l'intégrité des données : il n'y a aucune déformation ou perte d'information avant les traitements. D'autre part, les espaces d'acquisition ne présentant généralement pas de spécificité particulière, les traitements proposés dans ce contexte pourront être facilement transposables à d'autres situations multicomposantes.

**Réduction de la dimension de l'espace initial des données.** La transformation de l'espace initial avec réduction de la dimension se fait par deux méthodes : décorrélation des données ou transposition dans un espace de dimension 1. Le principe de base de la première transformation repose sur l'hypothèse de l'existence d'une corrélation inter-composante des données. Cette hypothèse est souvent vérifiée dans l'espace d'acquisition des images multicomposantes. On cherche alors une transformation orthogonale permettant de définir une base dans laquelle les nouvelles composantes sont décorrélées. Si la corrélation est importante, on constate alors qu'un petit nombre des nouvelles composantes concentrent à elles seules la majorité de l'information contenue dans les données initiales. On trouve dans la littérature de nombreux travaux fondés sur cette approche, que ce soit avec des objectifs de compression [Lee91] ou de traitement [Dev00], [CTB+00] .... La transposition dans un espace de dimension 1 est une solution plus radicale. On se retrouve alors dans un contexte mono-composante où l'on peut mettre en œuvre les traitements scalaires. Ce type de transposition introduit dans la plupart du temps des distorsions, ou des altérations des données provoquant alors une perte d'informations.

**Analyse en composantes indépendantes :** Ce principe, inspiré des techniques de séparation des sources utilisées en traitement du signal, commence à être utilisé en traitement d'images [NM00]. L'image observée est considérée comme la superposition de plusieurs images, ou sources, indépendantes. L'identification des différentes sources facilite alors les étapes de segmentation ou d'analyse. Ces approches reposent sur une connaissance a priori importante (nature, nombre des sources, ...).

Le principe n'est pas spécifique aux images multicomposantes, mais le fait d'avoir une information plus riche sur chaque source permet une meilleure estimation du processus de mélange des différentes sources.

**Mise en évidence de l'information utile.** Cette approche est essentiellement utilisée dans le cas particulier des images couleur. En effet, ces images sont classiquement obtenues dans l'espace Rouge-Vert-Bleu qui n'est pas forcément le mieux adapté à l'élaboration des traitements :

- les composantes y sont fortement corrélées,
- la notion de couleur, telle que l'œil humain la perçoit, n'est pas directement accessible,
- l'appréciation par l'œil humain des écarts de couleur ne coïncide pas avec les distances calculées avec les mesures classiques (norme $L2$).

## 1.7 Hypergraphe de Voisinage d'une image couleur

### 1.7.1 Approches marginale et vectorielle

Deux approches sont possibles pour appliquer la représentation par hypergraphe de voisinage aux images couleurs : l'approche marginale ou l'approche vectorielle.

**Approche marginale.** La représentation par hypergraphe de voisinage consiste dans ce cas à traiter séparément chacune des composantes. Pour construire l'hypergraphe de voisinage, on associe à chaque composante couleur un hypergraphe scalaire.

Supposons que l'image couleur soit définie par ces trois composantes $I_{C_1}$, $I_{C_2}$, et $I_{C_3}$, alors l'Hypergraphe de Voisinage SpatioColorimétrique Marginal, noté HVSC-M est composé de trois hypergraphes scalaires $H^{C_1}_{\alpha,\beta}$, $H^{C_2}_{\alpha,\beta}$, et $H^{C_3}_{\alpha,\beta}$.

**Approche vectorielle :** La définition de la représentation par hypergraphe de voisinage d'une image reste la même pour une image à niveaux de gris ou couleur. Pour les images à niveaux de gris, on manipule des intensités scalaires, tandis que pour les images couleur, on manipule des vecteurs 3D. Donc, à toute image couleur $I$ munie d'une distance colorimétrique dans un espace couleur donné, on peut associer un hypergraphe de voisinage nommé cette fois Hypergraphe de Voisinage SpatioColorimétrique Vectoriel HVSC-V$_{\alpha,\beta}$ (Fig.1.13). La représentation est définie par :

$$H = (X, (\{x\} \cup \Gamma_{\alpha,\beta}(x))_{x \in X}) \tag{1.32}$$

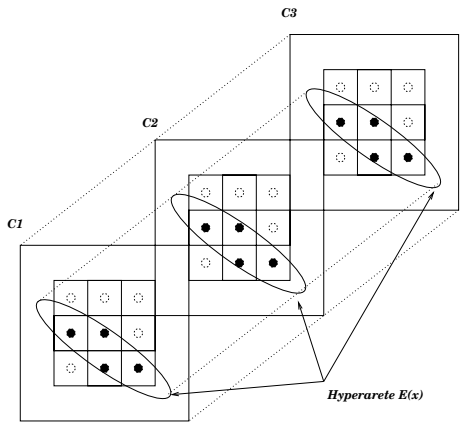



FIG. *1.13: Exemple d'un hypergraphe de voisinage spatiocolorimétrique (HVSC) vectoriel d'une image couleur.*

Notre représentation emploie une distance colorimétrique dans une grille donnée afin de générer une hyperarête. Plusieurs espaces de représentation de la couleur existent. Comment déterminer quel espace de couleur est le mieux adapté au problème posé ? Ceci nous paraît être la principale difficulté lorsque l'on parle de choix d'un espace de couleur : ce choix doit-il se faire indépendamment du problème à résoudre ou doit-il tenir compte de la représentation HVSC ? . Pour générer la représentation par hypergraphe de voisinage spatiocolorimétrique, il paraît toutefois difficile de séparer le choix de l'espace couleur du problème à résoudre. Nous avons donc axé notre réflexion en ce sens : le choix de l'espace couleur est lié à la représentation HVSC.

### 1.7.2 Distances colorimétriques

Pour un certain nombre d'applications, la mesure de distance entre couleur, ou plus généralement entre vecteurs s'est avérée très importante. La relation entre vecteur et un autre ou, entre vecteur et un groupe de vecteurs a été au cœur des plusieurs applications, filtrage des images, indexation [APV98], ... Ainsi, différentes mesures de similarité entre vecteurs ont été développées et implémentées.

*La distance de Minkowski.* En général, la mesure la plus souvent utilisée pour quantifier une distance entre deux vecteurs est la distance de Minkowski (la norme $L_p$) définie comme :

$$d_p(i,j) = \left( \sum_{k=1}^{m} |(I_{i,k} - I_{j,k})|^p \right)^{1/p} \tag{1.33}$$

où $m$ représente la dimension du vecteur $\vec{I_i}$ et $I_{i,k}$ est le $k^{\text{ème}}$ élément du vecteur $\vec{I_i}$. Trois cas de la norme $L_p$ ont un intérêt particulier à savoir :

1. La distance de *City-Block* ou la distance de *Manhatton* (norme $L_1$) correspond à $p = 1$ :

$$d_1(\vec{I_i}, \vec{I_j}) = \left( \sum_{k=1}^{m} |I_{i,k} - I_{j,k}| \right) \tag{1.34}$$

2. La distance euclidienne (norme $L_2$) correspond à $p = 2$ :

$$d_2(\vec{I_i}, \vec{I_j}) = \sqrt{\left( \sum_{k=1}^{m} (I_{i,k} - I_{j,k})^2 \right)} \tag{1.35}$$

3. La distance de *Chessboard* (norme $L_\infty$) correspond à $p = \infty$ :

$$d_\infty(\vec{I_i}, \vec{I_j}) = max\{|I_{i1} - I_{j1}|, |I_{i2} - I_{j2}|, ..., |I_{ik} - I_{jk}|\} \tag{1.36}$$

*La distance de Canberra* définie comme :

$$d_c(I_i, I_j) = \sum_{k=1}^{m} \frac{|I_{i,k} - I_{j,k}|}{|I_{i,k} + I_{j,k}|} \tag{1.37}$$

où $m$ représente la dimension du vecteur $\vec{I_i}$ et $I_{ik}$ est le $k^{\text{ème}}$ élément du vecteur $\vec{I_i}$. $d_c(I_i, I_j)$ est nulle si les deux $I_{i,k}$ et $I_{j,k}$ sont nulles [PAV95]. Cette mesure est appliquée aux vecteurs de $M$ dimension qui ne prennent pas de valeurs négatives, comme la couleur.

**Distances spécifiques**

Depuis de nombreuses années, les experts en colorimétrie s'attachent à trouver une formule permettant d'effectuer le calcul de différence entre deux couleurs. Même si la distance euclidienne reste la mesure de distance la plus simple, elle n'en est pas moins inadaptée dans les espaces couleur non linéaires. L'espace colorimétrique $L^*a^*b^*$ a été introduit par la CIE en 1976 pour permettre l'évaluation des différences colorimétriques appliquées au monde industriel. En effet, non seulement la mesure de la couleur est importante dans le monde de l'industrie (textile, carrosserie...), mais aussi la mesure de la déviation colorimétrique. Les couleurs utilisées ont généralement une tolérance à la déviation très faible ; plus cette tolérance est faible, et plus le processus de fabrication du produit est complexe. La mesure de la différence colorimétrique permet d'avoir une échelle numérique représentant la différence entre deux couleurs définies par leurs triplets.

Ce calcul de différence est associé à l'amplitude visuelle perçue de la différence colorimétrique entre deux échantillons.

Afin de normaliser son modèle de mesure de différence couleur, la CIE a adopté un certain nombre de conditions de référence. Ces conditions de référence sont les suivantes :

- Source de lumière : illuminant $D_{65}$ d'excitance [3] 1000 Lux ;
- Fond : gris neutre avec L * = 50 ;
- échantillons : placés les uns au contact des autres ; la taille totale de l'échantillon ne doit pas être supérieure à quatre degrés d'angle visuel.

Il est important de rappeler que tout changement de ces conditions peut affecter les performances du modèle proposé par la CIE.

La différence totale euclidienne $\Delta E^*_{ab}$ est calculée dans l'espace $L^*a^*b^*$ à l'aide de la formule suivante :

$$\Delta E^*_{ab} = \sqrt{\Delta L^{*2} + \Delta a^{*2} + \Delta b^{*2}} \tag{1.38}$$

En utilisant le fait que la distance euclidienne dans le système rectangulaire ($\Delta L^*$, $\Delta a^*$, $\Delta b^*$) est identique à la distance euclidienne dans le système cylindrique ($\Delta L^*$ , $\Delta C^*_{ab}$, $\Delta H^*_{ab}$ ) nous obtenons facilement la formule de calcul permettant de mesurer la différence de teinte $\Delta H^*_{ab}$ :

$$\Delta H^*_{ab} = \sqrt{\Delta E^{*2} - \Delta L^{*2} - \Delta C^{*2}_{ab}} \tag{1.39}$$

En 1994, la CIE propose une nouvelle formule du calcul de la différence colorimétrique entre deux couleurs. En effet, nous savons que le système visuel humain possède une plus grande sensibilité envers la grandeur de luminance que de chrominance. Or, dans l'équation 1.38, nous remarquons que cette importante caractéristique n'est pas prise en compte car les trois axes possèdent la même pondération. Pour pallier cela, la CIE a donc mis en place une distance euclidienne pondérée utilisant le triplet ($\Delta L^*$, $\Delta C^*_{ab}$, $\Delta H^*_{ab}$) :

$$\Delta E^*_{ab} = \sqrt{(\frac{\Delta L^*_{ab}}{k_L S_L})^2 + (\frac{\Delta C^*_{ab}}{k_C S_C})^2 + (\frac{\Delta H^*_{ab}}{k_H S_H})^2} \tag{1.40}$$

où

- $S_L$, $S_C$, $S_H$ sont des fonctions de pondérations qui permettent un ajustement du calcul de la différence colorimétrique, de manière à prendre en compte la variation de l'amplitude de la différence perçue par rapport à la variation de couleur dans l'espace $L^*a^*b^*$.

[3] on dit aussi émittance ou irradiance, même si ce dernier n'est plus guère en usage aujourd'hui.

- $k_L$, $k_C$, $k_H$ sont des facteurs paramétriques qui permettent de compenser les variations des composantes de différence de couleurs perçues, en fonction des variations dues aux conditions expérimentales.

En 1996, une distance S-CIELab a été développée par Wandell d'après les études de Poirson & Wandell [PW93] [PW96]. L'espace CIElab permet la mesure des différences de couleur entre des plages uniformes assez étendues (quelques degrés), l'objectif de S-CIELab est de donner une mesure des différences de couleur pour des images structurées. En général, plus la structure spatiale devient fine (la fréquence spatiale s'élève), plus les différences de couleur deviennent difficiles à percevoir, particulièrement dans la direction bleu-jaune. C'est pourquoi S-CIELab ajoute au calcul $\Delta E_{ab}$ conventionnel une étape de traitement spatio-chromatique préliminaire. L'architecture du calcul maintient le traitement spatial et le traitement coloré séparés. Comme illustré dans la figure 1.14, l'image originale est d'abord décomposée en :

- Une image achromatique où seules les variations de luminance sont présentes ;
- Et deux images chromatiques contenant exclusivement,
  pour l'une, l'information Rouge/Vert
  pour l'autre, l'information Bleu/Jaune.

Chaque image est ensuite filtrée spatialement, c'est à dire nettoyée de ses harmoniques spatiales élevées, par une fonction gaussienne dont la fréquence de coupure est respectivement élevée. Les trois images filtrées sont alors recombinées et le résultat transformé dans l'espace $XYZ$, prêt à être traité selon la procédure CIELab conventionnelle.

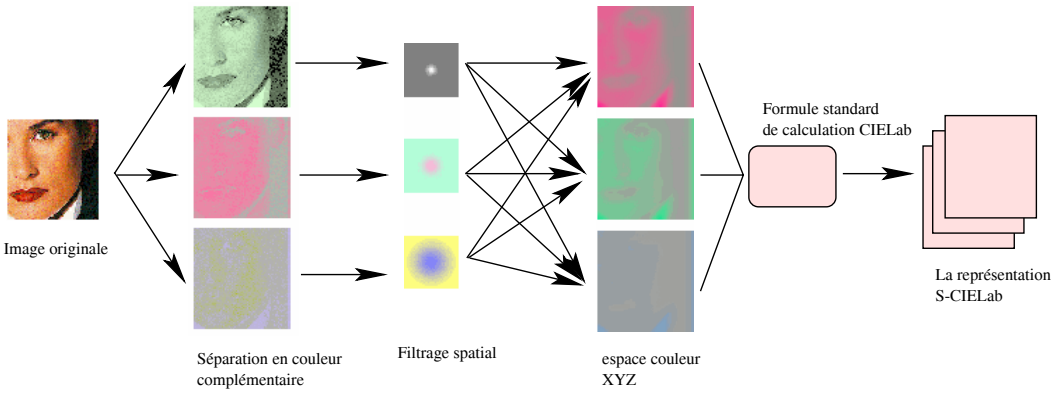



FIG. *1.14: Procédure de calcul de la métrique S-CIELab.*

Même si l'expression du calcul de la distance entre couleur peut et doit normalement s'exprimer dans un espace couleur autre que l'espace RVB, il ne faut pas oublier que ces formules ne sont valables que pour des petites distances, et qu'a priori, elles n'ont pas été validées pour de grandes distances entre couleur. Néanmoins, ces formules restent les seules à avoir été définies de cette façon et sont donc les meilleures garantes d'une différence colorimétrique significative d'un point de vue visuel. Qui plus est, ces formules n'ont de sens que dans un contexte d'étude très particulier à savoir des échantillons de couleurs placés les uns au contact des autres, le tout sur un fond gris. L'utilisation de ces formules dans un autre contexte que celui pour lequel elles ont été définies n'est pas garantie par la CIE.

On peut cependant s'interroger sur la légitimité d'utiliser de tels espaces pour la comparaison de la couleur de deux pixels d'une image numérique. D'une part, les conditions de mesures précisées par la CIE lors de la définition de ces espaces, sont très précises et imposent aux utilisateurs, de se placer dans ces conditions particulières pour obtenir une bonne évaluation des

différences de couleur. Dans le cadre des images numériques, nous sommes loin des quatre degrés d'angle visuel et du fond gris uniforme ! D'autre part, soulignons encore que ces espaces couleurs sont conçus pour évaluer des écarts de couleurs très faibles.

## 1.8 Conclusion

Dans ce chapitre, nous nous sommes attachés, premièrement, à définir un graphe et un hypergraphe. Nous avons ensuite présenté les notions de base de l'image numérique. Ceci nous a permis d'introduire la notion d'hypergraphe de voisinage d'une image.

La représentation proposée dépend de la distance colorimétrique $d'(I(x), I(y))$ , du seuil colorimétrique $\alpha$ et de l'ordre de voisinage $\beta$. Selon les deux premiers paramètres, nous avons présenté trois représentations d'hypergraphe de voisinage différentes, une représentation avec un seuil colorimétrique global, une représentation avec un seuil colorimétrique dynamique et une représentation avec un seuil global utilisant des fonctions de similarité.

L'extension de la représentation par hypergraphe de voisinage à l'image couleur commence par une étude préalable sur le traitement des images multicomposantes : espace de représentation, stratégie de traitement. En effet, selon cette brève étude, deux extensions sont possibles : l'approche marginale ou l'approche vectorielle. Nous nous sommes intéressés, particulièrement à l'approche vectorielle de la représentation, car cette dernière, est vectorielle par nature et son extension vectorielle est immédiate.

De la même façon que la représentation scalaire, la représentation vectorielle emploie une distance colorimétrique, un seuil spatiocolorimétrique et un seuil spatial. Le plus important de ces paramètres pour la construction de l'hypergraphe de voisinage vectoriel de l'image couleur est la distance colorimétrique. Pour cette raison, nous avons donné une brève présentation des distances colorimétriques utilisées dans le traitement des images couleur.

# Chapitre 2

# APPLICATION : SUPPRESSION DE BRUIT

**Résumé :**

*Dans ce chapitre, nous nous intéressons plus particulièrement à la détection et à l'estimation conditionnelle de bruit dans les images numériques. Nous décrivons tout d'abord les principaux modèles de bruits utilisés dans la littérature, ainsi que quelques techniques, de suppression de bruit dans les images numériques. Nous décrivons ensuite les modèles de bruit structurels sur l'hypergraphe de voisinage associé à l'image et proposons un algorithme de débruitage basé sur cette modélisation. Cette étude est menée successivement pour des images à niveaux de gris et des images couleurs. Ensuite, nous comparons nos résultats en utilisant des mesures objectives de qualité avec quelques algorithmes de la littérature.*

## 2.1 Introduction

De nombreuses applications en traitement numérique d'image nécessitent une étape de suppression de bruit. Cette étape consiste à préserver, ou restituer au mieux, l'information utile de l'image, tout en réduisant au maximum ce qui vient la parasiter, c'est à dire tout ce que l'on qualifie de "bruit d'image". Suivant les applications, la partie de l'information considérée comme étant utile peut varier, et le terme de bruit recouvre donc, suivant les cas, des réalités très diverses :

- Il peut s'agir d'une dégradation effective de l'image : capteur défectueux lors de l'acquisition, bruit de quantification lors de la numérisation, pertes de la transmission, etc. Dans ce cas, l'objectif de la suppression de bruit d'image est alors de restaurer l'image initiale, c'est à dire d'en donner la meilleure estimation possible.
- Il peut également s'agir d'une dégradation subjective de l'image par rapport à un objectif donné. Par exemple, on rehaussera le contraste d'une image prise dans la pénombre ; ou encore, on cherchera à simplifier une image au contenu informationnel trop riche, afin de pouvoir la compresser efficacement et la transmettre rapidement.

L'objectif très général de la suppression de bruit d'image est de restituer au mieux les régions pertinentes de l'image, au sens de l'application traitée, et de supprimer les autres. Cela doit se faire avec le double impératif suivant :

- trouver la meilleure estimation possible de la valeur associée à chaque pixel,
- préserver au mieux les frontières entre chaque région.

Deux grandes méthodologies sont utilisées pour résoudre le problème de la suppression de bruit dans une image : l'approche dite statistique et l'approche structurelle. L'approche statistique est fondée sur l'étude statistique des mesures que l'on a effectuées sur l'image observée. Ces méthodes s'appuient en général sur des hypothèses (explicites ou non) concernant la description statistique du bruit dans l'image à traiter. On distingue d'une façon générale deux grands types de méthodes : les méthodes dites non paramétriques et les méthodes dites paramétriques. Les méthodes paramétriques ou bayésiennes, où l'on se donne un modèle de distribution de chaque classe de bruit et où l'on cherche la classe à laquelle le bruit a la probabilité la plus grande d'appartenir. Ces méthodes utilisent les densités de probabilités (ddp) de bruit. On cherche à estimer au mieux à l'aide d'une fonction de coût l'image originale. Les méthodes non paramétriques, où l'on cherche à définir les frontières des classes de bruit, de façon à pouvoir classer le bruit inconnu par approximation. Dans le cadre de cette méthodologie, deux stratégies de traitements peuvent être utilisées. La première consiste à estimer l'image bruitée dans sa globalité. Tandis que la deuxième stratégie commence tout d'abord par une étape de détection des zones bruitées puis une estimation des dégradations détectées. Si l'approche statistique permet de se placer dans un cadre mathématique solide et général, elle présente cependant le défaut d'oublier la nature spatiale de l'image.

Dans ce chapitre, nous décrivons les différentes façons d'éliminer le bruit en s'appuyant sur la représentation par hypergraphe de voisinage. Dans un premier temps, nous présentons les modèles statistiques de bruit rencontrés couramment dans le traitement d'image. Nous présentons ensuite quelques techniques de suppression de bruit telles que le filtre médian [GW81] et le filtre médian multiétage [Wan92] utilisés dans la cadre d'une stratégie d'estimation. L'algorithme de Tovar [TER$^+$00] et l'algorithme de Stevenson [SS92], ainsi que l'algorithme proposé pour les trois types de représentations mentionnés dans le premier chapitre utilisent quant à eux une stratégie de détection suivi d'une estimation conditionnelle. Nous comparons les résultats de l'algorithme proposé en utilisant ces trois représentations à l'aide des mesures objectives de qualité afin de trouver la "meilleure" représentation compatible avec l'application proposée pour des images à niveaux de gris. Ensuite, nous donnons les résultats expérimentaux de comparaison avec les méthodes citées précédemment.

En dernier lieu, nous étendons l'algorithme proposé aux images couleurs en employant deux approches : marginale et vectorielle.

Nos expérimentations dans cette dernière partie commencent d'abord par une comparaison des deux approches marginale et vectorielle, ensuite une comparaison des espaces couleur RVB et CIELab. La comparaison entre l'algorithme vectoriel proposé de suppression de bruit dans l'espace couleur CIELab et les filtres VMF [AHN90] et BVDF [TKV96][TV93] est présentée dans le dernier paragraphe.

## 2.2 Modèles de bruit

Historiquement, l'hypothèse de bruit gaussien a longtemps été dominante pour caractériser le bruit d'image. Néanmoins, dans la pratique (Image naturelle, Image radar, Image médicale) cette hypothèse est désavouée. Les bruits observés sont non gaussiens et ont un comportement impulsionnel. Ceci a mené aux développements de plusieurs modèles de bruit non gaussien. Nous ne dressons pas une bibliographie exhaustive du sujet. Nous nous contentons de rappeler quelques modèles utilisés fréquemment.

### 2.2.1 Bruit gaussien généralisé [Kas88]

La densité de probabilité du bruit gaussien généralisé est caractérisée par deux paramètres, la variance $\sigma^2$, et le taux d'atténuation exponentiel $k > 0$. Elle est symétrique et unimodale. Elle est définie par [Kas88] :

$$f_k(x) = \frac{k}{2A(k)\Gamma(1/k)} e^{[-|x|/A(k)]^k} \tag{2.1}$$

avec

$$A(k) = \left[\sigma^2 \frac{\Gamma(1/k)}{\Gamma(3/k)}\right] \quad et \quad \Gamma(\alpha) := \int_0^\infty x^{\alpha-1} e^{-x} dx. \tag{2.2}$$

Certaines valeurs particulières de la constante $k$ de l'équation 2.1 caractérisent la nature plus ou moins impulsionnelle du bruit : lorsque $k$ vaut 2, le bruit est gaussien, lorsque $k$ vaut 1, il est laplacien. Pour $k < 2$, les queues de la distribution sont plus prononcées que celles de la gaussienne. Ceci permet de décrire la nature plus ou moins impulsionnelle du bruit.

### 2.2.2 Le modèle de Middleton de classe A [Mid79]

Le modèle de Middleton de classe A est basé sur des considérations d'une modélisation physique plutôt que sur un ajustement aux données observées.

Les processus de bruit observés sont supposés avoir deux composantes indépendantes :

$$X(t) = X_G(t) + X_P(t) \tag{2.3}$$

avec, $X_G(t)$ qui représente une composante de bruit gaussien stationnaire et $X_P$ la composante impulsionnelle définie par :

$$X_P(t) = \sum_i U_i(t, \theta) \tag{2.4}$$

où $U_i$ dénote la ième onde de la source et $\theta$ représente une série des paramètres aléatoires qui décrivent l'échelle et la structure de l'onde. Le temps d'arrivée de ces événements impulsionnels indépendants est supposé avoir une distribution de Poisson. Sous certaines hypothèses [MW98], la ddp du bruit est donnée par :

$$f_A(x) = e^{-A} \sum_{m=0}^{\infty} \frac{A^m}{m!\sqrt{2\pi\sigma_m^2}} e^{-x^2/2\sigma_m^2} \tag{2.5}$$

avec

$$\sigma_m^2 = \frac{(m/A) + \Gamma'}{1 + \Gamma'} \tag{2.6}$$

Le paramètre $A$, nommé index impulsionnel, détermine la proportion du bruit impulsionnel. Une valeur faible de $A$ implique une interférence fortement impulsive (bien que pour $A = 0$ $x(t)$ dégénère un bruit purement gaussien). Le paramètre $\Gamma'$ est le rapport entre la puissance du bruit dans la composante gaussienne et la puissance dans le processus de Poisson d'interférence. Le terme correspondant à $m = 0$ représente la composante gaussienne du bruit. Les autres termes correspondent à la composante impulsive du bruit. Ce modèle s'est avéré fournir de très bons ajustements à une grande variété de mesures de bruit et d'interférence en traitement du signal [Kas88].

### 2.2.3 La distribution alpha-stable [SN93]

La distribution alpha-stable est un autre modèle utile de distribution de bruit. Ce modèle est décrit par sa fonction caractéristique. Pour une distribution symétrique, la fonction caractéristique est donnée par :

$$\varphi(t) = e^{\{jat - \gamma|t|^{\alpha}\}} \tag{2.7}$$

où $a$ est le paramètre de localisation ($-\infty < a < +\infty$) définissant le centre de la distribution et $\gamma(>0)$ mesure la dispersion de la distribution autour du centre $a$, $\alpha$ est la caractéristique exponentielle ($0 < \alpha \leq 2$) : lorsque $\alpha$ vaut 2 la distribution est gaussienne, lorsque $\alpha$ vaut 1, il s'agit de la distribution de Cauchy. Pour $\alpha \longrightarrow 0$, la distribution est fortement impulsionnelle.

## 2.3 Modèles de perturbations

Trois modèles de génération de bruit dans une image sont généralement admis : le plus commun est le bruit additif. En considérant le bruit comme un vecteur aléatoire $\eta$, il affecte une image $I$ sous la forme $\eta + I$. Le bruit additif est le plus usuellement utilisé dans le sens où c'est lui qui est le plus souvent rencontré dans des situations réelles. Un autre modèle de bruit est le bruit multiplicatif, c'est le bruit qui est présent sur un film radiographique par exemple. En reprenant les notations précédentes, son action induit une modification multiplicative de l'information utile de l'image : $\eta \times I$. Citons enfin le bruit dit convolutif $\eta * I$ caractérisant les flous de bougés, les mauvaises mises au point et qui agit convolutivement sur l'image.

## 2.4 Suppression de bruit dans une image

L'objectif des paragraphes suivants est de donner les principes élémentaires de fonctionnement de quelques méthodes de suppression de bruit couramment utilisées.

### 2.4.1 Approche estimation

Le traitement d'images (scalaires ou vectorielles) suppose l'exécution, par un bloc fonctionnel, d'une opération de type "Image d'entrée, Image de sortie" [Cas96]. Dans la littérature, on distingue deux types d'estimation, linéaire et non linéaire.

Les estimateurs linéaires sont les plus simples et les plus faciles à implanter. La justification de ces estimateurs repose en particulier sur l'hypothèse d'un modèle stationnaire de l'image. De nombreux opérateurs linéaires, classiquement de type passe-bas pour le lissage, ont ainsi été proposés dans la littérature. Même si la structure et les propriétés mathématiques des structures linéaires de l'estimation sont simples, elles présentent le désavantage d'une faible performance en présence du bruit non additif ou non gaussien. A cause de ces limitations, les estimateurs non linéaires se sont imposés [PV90] soit par adaptation des estimateurs linéaires, soit à travers le développement de traitements intrinsèquement non linéaires. Le terme "adaptatif" porte sur la capacité de la méthode d'estimation à modifier son comportement en fonction de certains critères. Par le processus d'adaptation, en chaque nouveau point traité, la structure de l'approche est différente, et dépend des caractéristiques locales du pixel courant. Il existe deux variantes d'adaptation : la modification de la forme de la fenêtre de la méthode d'estimation et la modification des coefficients de la fenêtre d'estimation de forme fixe [Ver99].

L'exemple le plus connu des méthodes d'estimation non linéaires est la famille d'estimateurs avec ordonnancement selon le rang [PV90], qui inclut les estimateurs d'ordre, les L-filtres et une partie des estimateurs morphologiques [TKP98] [KP96]. Tous ces estimateurs sont fondés sur

une opération d'ordre (ordre croissant des valeurs des pixels extraits par la fenêtre d'estimation) [Ver99].

#### 2.4.1.1 Le filtre médian.

Soit $X = (I_1, I_2, ..., I_N)$, $N = 2n + 1$, le vecteur des intensités des pixels dans la fenêtre d'analyse $W$. Soit $T = (I_{(1)}, I_{(2)}, ..., I_{(N)})$, le vecteur des statistiques d'ordre associées à ces valeurs et telles que :

$$I_{(1)} \leq ... \leq I_{(i)} \leq ... \leq I_{(N)} \tag{2.8}$$

La sortie $y$ d'un filtre médian est définie par la relation suivante :

$$y = Med(I_1, I_2, ...I_N) = I_{n+1} \tag{2.9}$$

Le filtre médian est optimal au sens du critère de Maximum de Vraisemblance (MV) dans le cas d'un bruit exponentiel [M.T99]. Il permet à la fois de conserver les transitions monotones entre régions de largeur supérieure à la moitié de la taille du filtre et aussi de supprimer les impulsions parasites. Par contre, ses performances se détériorent en présence de bruit à distribution bornée ou concentrée [CP95]. De plus, il a tendance à favoriser le phénomène de stries, c'est à dire, la création en sortie du filtre de zones formées de pixels ayant exactement la même intensité. Ces zones peuvent être à l'origine de production de fausses régions dans une image. A contrario, il supprime les détails fins qui constituent souvent des informations utiles. Pour améliorer les performances du filtre médian, plusieurs techniques de filtrage ont été proposées. On peut citer les filtres médian pondéré [Bro84], médian à étages [NN88], médian récursif [Mou89], médian récursif à étages [Raj92], médian généralisé [Rop95]. Cependant, les performances obtenues grâce à ces améliorations restent moyennes, notamment en présence de bruit à distribution concentrée.

#### 2.4.1.2 Le filtre médian multiétage

Les filtres multiétages comportent généralement deux étages. Le premier étage est constitué de filtres agissant directement sur les données de l'image. Le deuxième et les suivants traitent les sorties des filtres de l'étage précédent. Pour le filtre médian multiétage (*MultiLevel Median Filter MLMF*) [Wan92], le premier étage est formé de filtres médians $1D$ agissant dans les directions horizontale, verticale et diagonales. Le deuxième étage sélectionne le Min et le Max des valeurs obtenues. Le filtre MLMF est simple et effectif. Il est capable de supprimer le bruit impulsionnel et de préserver les divers types de contours, ligne, structure mince, ou autres [Wan92] [Jiu95].

Supposons que $\{I(.,.)\}$ est une séquence discrète à deux dimensions et $W$ une fenêtre carrée de taille $(2N+1) \times (2N+1)$ centrée sur le pixel $(i, j)$. Quatre sous-ensembles de W peuvent être définis comme :

$$\begin{aligned} W_1 &= \{I(i, j+k); \quad -N \leq k \leq N\} \\ W_2 &= \{I(i+k, j+k); \quad -N \leq k \leq N\} \\ W_3 &= \{I(i+k, j); \quad -N \leq k \leq N\} \\ W_4 &= \{I(i+k, j-k); \quad -N \leq k \leq N\} \end{aligned}$$

Considérons respectivement que $Med_S(i, j)(S = 1, 2, 3, 4)$ sont les valeurs médianes des quatre sous-ensembles $W_1$, $W_2$, $W_3$ et $W_4$. Et considérons que :

$$\begin{aligned} y_{min}(i, j) &= MIN[Med_1(i, j), Med_2(i, j), Med_3(i, j), Med_4(i, j)] \\ y_{max}(i, j) &= MAX[Med_1(i, j), Med_2(i, j), Med_3(i, j), Med_4(i, j)] \end{aligned}$$

Alors, la sortie du filtre MLMF est définie comme :

$$y(i, j) = Med[y_{min}(i, j), y_{max}(i, j), I(i, j)]$$

### 2.4.2 Approche détection et estimation

Dans la littérature, les estimateurs linéaires et non linéaires ont démontré de bonnes performances en suppression de bruit mais malheureusement, ils tendent également à modifier les pixels non corrompus par le bruit. Ceci a conduit à l'apparition des algorithmes conditionnels. Seul le pixel décidé corrompu par un bruit est estimé. Leurs principes se résument en deux étapes.

- Détection de pixels bruit,
- Ensuite, une estimation.

L'idée développée ici par les algorithmes de Tovar [TER+00], Stevenson [SS92] et l'algorithme proposé est de réduire le bruit par une méthode de classification des pixels en pixels bruités et non bruités et dès lors, de modifier leur couleur (intensité pour les images à niveaux de gris). Une telle méthode présente l'avantage de préserver les frontières entre les régions adjacentes.

#### 2.4.2.1 Algorithme de Tovar

L'algorithme de Tovar [TER+00] est un algorithme qui utilise une méthode sélective afin de détecter les pixels bruités ou non bruités. En effet, un pixel est détecté comme bruit, si la différence d'intensité avec son voisinage est supérieure à un seuil donné. Dans une fenêtre $W$ (8-adjacents) centrée sur le pixel $n$, on construit un vecteur $\Psi(n)$ où les éléments sont ordonnés selon leurs valeurs d'intensité. Il est défini comme :

$$\Psi(n) = [I_{(1)} \leq I_{(2)} \leq \ldots \leq I_{(8)}] \tag{2.10}$$

Finalement, la valeur médiane du vecteur $\Psi(n)$ est définie par

$$Med(n) = \frac{I_{(4)} + I_{(5)}}{2} \tag{2.11}$$

Un paramètre $\eta$ classe les fenêtres d'image en $M$ classes différentes dépendant de la variable d'état définie par $s(n) = \eta(I(n), \Psi(n))$. Cette variable décrit le degré de perturbation de chaque classe et par conséquent contrôle l'estimation donnée par :

$$y(n) = F(I(n), \Psi(n), s(n)) \equiv \upsilon_{s(n)} I(n) + (1 - \upsilon_{s(n)}) Med(n) \tag{2.12}$$

où $\upsilon_i$ pour $i = 1, ..., M$, représente le coefficient de chaque état possible dans lequel une fenêtre peut être classifiée.

La classification de l'image en $M$ classes se base sur le paramètre $\eta$. Ce paramètre doit produire une fonction d'état $s(n)$ qui est un indicateur probabiliste décrivant le degré de corruption du pixel par le bruit. Le classificateur $\eta$ opère sur la différence entre les pixels. Les différences sont définies par :

$$d_k(n) = \begin{cases} \Psi_k(n) - I(n) & I(n) \leq Med(n) \\ I(n) - \Psi_{9-k}(n) & I(n) > Med(n) \end{cases} \qquad k = 1, ..., 4 \tag{2.13}$$

où $\Psi_k(n)$ correspond à la valeur de $I$ à la position $k$ du vecteur $\Psi(n)$. Par exemple, $\Psi_1(n)$ correspond à $I_{(1)}(n)$.

Le cas le plus simple de l'algorithme de Tovar correspond à $M = 2$. Dans ce cas, il travaille seulement avec deux états $s(n) = 1$ et $s(n) = 2$. L'état s(n)=1 correspond à un pixel bruité tandis que s(n)=2 à un pixel non bruité. Pour distinguer ces deux états, il compare la distance $d_k(n)$ $k = 1, ..., 4$ calculée pour chaque pixel avec quatre seuils $T_1 < T_2 < T_3 < T_4$. Un pixel

est décidé bruité (s(n)=1) si l'une des inégalités $d_k(n) > T_k, \quad k = 1, ..., 4$ est vraie. Autrement, l'algorithme suppose que le pixel n'est pas corrompu par le bruit et assigne s(n)=2. Les auteurs recommandent les seuils suivants :

$$T_1 \leq 15, \qquad 15 \leq T_2 \leq 25, \qquad 30 \leq T_3 \leq 50, \qquad 40 \leq T_4 \leq 60. \tag{2.14}$$

ces seuils ont été déterminés expérimentalement pour $M = 2$.

Après avoir classifier les fenêtres $\Psi(n)$ et par conséquent les pixels $I(n)$, il assigne pour chaque $\Psi(n)$ une valeur de $\upsilon$. Dans le cas de $M = 2$, la valeur de $\upsilon$ prend deux valeurs 0 ou 1. La classe $s(n) = 1$ correspondant à un pixel détecté bruité, la valeur de $\upsilon$ est égale à 0 et la valeur de $I(n)$ est remplacée par $Med(n)$. Dans le cas contraire $s(n) = 2$, la valeur de $\upsilon$ est égale à 1 et la valeur de $I(n)$ reste inchangée. La figure 2.1 résume les grandes étapes de l'algorithme de Tovar.

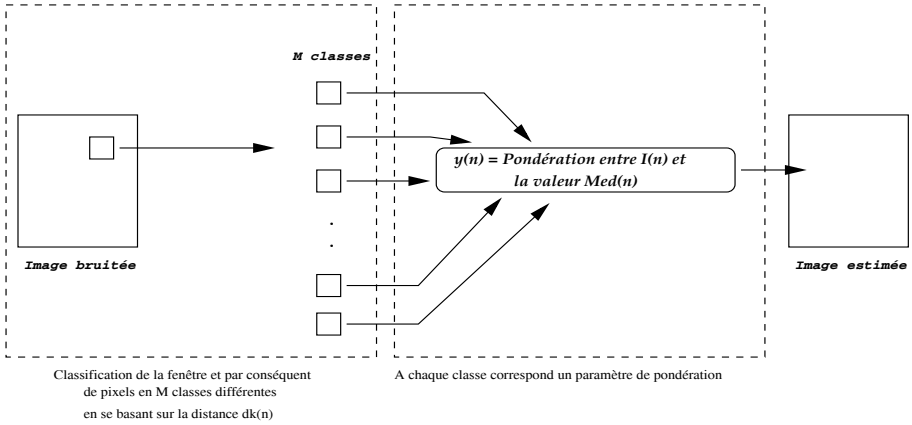



FIG. *2.1: Illustration graphique de l'algorithme de Tovar.*

#### 2.4.2.2 Algorithme de Stevenson

L'algorithme de Stevenson [SS92] est conçu pour la suppression de bruit dans des images à niveaux de gris corrompues par un bruit mixte : bruit gaussien additif et bruit impulsionnel. La détection du bruit impulsionnel emploie des opérations d'ordre statistique. Cependant, la suppression du bruit gaussien utilise le critère de maximum a posteriori (*Maximum A Posteriori* MAP).

L'observation $y$ est constituée de l'image originale $I$ (de taille $N \times N$) corrompue par un bruit additif $\eta$.

$$y = I + \eta \tag{2.15}$$

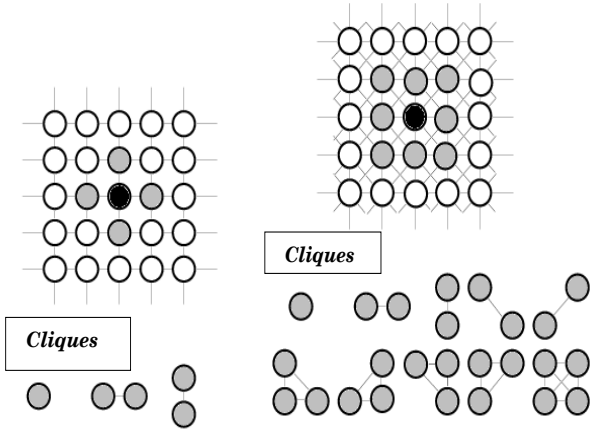
On cherche à estimer $I$ au sens du maximum a posteriori (MAP). Le modèle de bruit utilisé est un $\epsilon$-gaussien. Autrement dit, la perturbation est gaussienne sauf pour une proportion $\epsilon$ de pixels où la perturbation est impulsionnelle. La répartition spatiale des impulsions est uniforme.

L'image est modélisée par un Champ de Markov (Markov Random Field - MRF). Une distribution de *Gibbs* est employée pour écrire explicitement les distributions des Champs de Markov. Toute distribution qui peut être exprimée sous la forme suivante est une distribution de Gibbs [GG84] :

$$Pr(x) = \frac{1}{Z} e^{-\sum_{c \in C} V_c(x)} \tag{2.16}$$

où $Z$ est une constante de normalisation, $V_c(.)$ est une fonction potentielle d'un groupe local de cliques $c$ et $C$ est l'ensemble de tout le groupe local. Notons que le modèle gaussien est un cas

particulier de MRF. Rappelons la définition d'une clique : on se donne un ensemble de sites $S$, et une relation de voisinage sur $S$. On utilise très souvent la relation des 4 plus proches voisins, ou celle des 8 plus proches voisins, comme décrit dans la figure 2.2. Une fois fixée la relation de voisinage, un sous-ensemble $c \subseteq S$ est une clique si chaque paire de sites dans $c$ est voisine. $C$ note l'ensemble des cliques et $deg(C) = max_{c \in C}|c|$.




(a) (b)

FIG. *2.2: (a). Système de voisinage d'ordre 1. (b) Système de voisinage d'ordre 2.*

Si le modèle de MRF est utilisé pour modéliser une source donnée, le résultat de l'estimation de maximum a posteriori (MAP) en présence de bruit gaussien blanc additif d'écart-type $\sigma$ se ramène à un problème d'optimisation donné par :

$$\hat{I} = arg \;\; min_{\tilde{I} \in R^N} \left\{ \sum_{c \in C_s} V_c(\tilde{I}) + \frac{1}{2\sigma^2}||y - \tilde{I}||^2 \right\} \tag{2.17}$$

où $C_s$ dénote l'ensemble de cliques qui correspondent au système de voisinage choisi. La difficulté majeure en employant un tel modèle dans l'estimation du MAP est qu'il est très facile de construire un modèle statistique pour lequel le résultat d'optimisation du problème est non convexe. C'est particulièrement vrai quand le modèle essaie d'incorporer le fait que des discontinuités sont présentes dans le signal source. Des algorithmes ont été développés pour résoudre des problèmes d'optimisation non convexe. Cette optimisation peut être obtenue par des méthodes stochastiques ou déterministes. Les algorithmes stochastiques assurent la convergence asymptotique vers un minimum global au prix d'un temps d'exécution très élevé [Bes86] [GG84] [KGV83]. Cependant les méthodes déterministes sont moins coûteuses [AWJ90] [Mar92] [BFB93] [Ces96] [BZ87] [DC98].

L'algorithme de Stevenson emploie une distribution particulière de Gibbs.

$$Pr(x) = \frac{1}{Z} e^{\frac{1}{\lambda} \sum_{c \in C} \rho_T(d_c^t x)} \tag{2.18}$$

où $\lambda$ est un scalaire constant supérieur à 0, $d_c^t$ est une collection d'opérateurs linéaires, et $\rho_T(.)$ est donné par :

$$\rho_T(x) = \begin{cases} x^2, & x \leq T, \\ T^2 + 2T|x - T|, & x > T; \end{cases} \tag{2.19}$$

$\rho(.)$ est convexe, cette forme particulière d'énergie a comme conséquence la définition d'une optimisation d'un problème convexe une fois utilisée dans la formulation d'estimation MAP dans 2.17. La fonction $\rho(.)$ est connue comme fonction minmax de Huber [Hub81]. Ce modèle statistique se réfère à un modèle de Champ aléatoire de Markov de Huber (*Huber-Markov random filed, HMRF*).

- **Estimation.** La stratégie d'estimation consiste à ne pas incorporer l'information portée par le bruit impulsionnel dans l'estimateur car celui-ci n'apporte aucune information sur l'image originale. Si l'image est corrompue avec un bruit mixte, la distribution conditionnelle est donnée par :

$$Pr(y|I) = \frac{1}{R^{\epsilon N^2}\sqrt{(2\pi\sigma^2)^{(1-\epsilon)N^2}}} e^{-\frac{(y-I)^t I_I (y-I)}{2\sigma^2}} \tag{2.20}$$

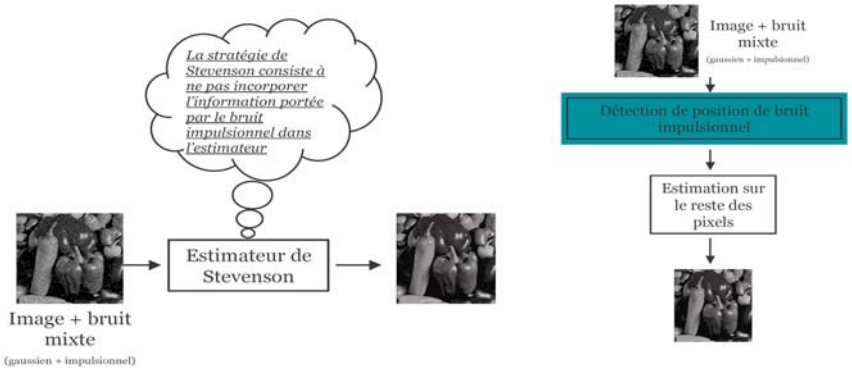
où, $\epsilon$ représente le pourcentage de bruit impulsionnel. $I_I$ est une matrice diagonale. La diagonale est égale à 1 si le pixel est corrompu par un bruit gaussien et à 0 s'il est corrompu par un bruit impulsionnel. En introduisant la fonction $\rho(.)$, la fonction 2.20 devient :

$$Pr(y|I) = \frac{1}{Z} e^{-1/2\sigma^2 \sum_{1\leq i,j\leq N} \rho_T(y_{i,j}-I_{i,j})} \tag{2.21}$$

En utilisant ce modèle pour le bruit et le même modèle pour l'image, l'estimation MAP est définie par :

$$\hat{I} = arg \;\; min_{\tilde{I}\in R^N} \left\{ \sum_{1\leq i,j\leq N} \sum_{k,l\in N_{i,j}} \rho(\tilde{I}_{i,j} - \tilde{I}_{k,l}) + \frac{\lambda}{2\sigma^2} \sum_{(i,j)\notin P} (y_{i,j} - \tilde{I}_{i,j})^2 \right\} \tag{2.22}$$

où, $P$ dénote les pixels $(i, j)$ corrompus par un bruit impulsionnel et $N_{i,j}$ dénote les 8 pixels voisins du pixel situé à $(i, j)$. L'estimateur rejette les points qui n'apportent aucune information sur l'image originale (Figure 2.3). D'un point de vue pratique, le problème qu'il reste à résoudre est de déterminer quels sont les pixels corrompus par un bruit impulsionnel à partir de la seule observation. Pour ce faire, R. Stevenson propose une méthode de détection en aval des pixels corrompus par le bruit impulsionnel afin de localiser ses positions.




(a) objectif (b) Solution

FIG. *2.3: Le principe de l'estimateur de Stevenson.*

- **Détection de bruit impulsionnel.** Afin d'effectuer la détection d'une impulsion, l'idée simple est de comparer le pixel en question à une version filtrée. Si la différence entre le pixel original et le pixel filtré est supérieure à un seuil donné $D$, le pixel est marqué bruité.

Autrement dit : soit $H_0$ l'hypothèse que le bruit est impulsionnel et $H_1$ l'hypothèse que le bruit est gaussien alors :

$$|y_{m,n} - F(y)_{m,n}| \underset{<H_0}{\overset{>H_0}{}} D \tag{2.23}$$

L'image filtrée $F(y)$ peut être obtenue par plusieurs algorithmes linéaires ou non linéaires. Les performances de la détection des pixels bruités avec un bruit impulsionnel dépendent du filtre utilisé.

Afin de comparer les diverses solutions possibles, R. Stevenson estime les performances en terme de probabilité de détection et fausse alarme. Rappelons que la probabilité de détection correspond à une bonne décision (à savoir décider que le bruit est impulsionnel) sachant que cette hypothèse est vraie. La probabilité de fausse alarme correspond quant à elle à une décision erronée (à savoir décider que le bruit est impulsionnel alors que dans la réalité il est gaussien).

Pour chacun des filtres étudiés, il détermine la valeur du seuil correspondant à une probabilité de décision égale à 0.98 et il estime la probabilité de fausse alarme. Le meilleur filtre est celui qui conduit à la plus faible valeur de probabilité de fausse alarme. Le meilleur filtre trouvé est le filtre médian multiétage du troisième ordre $[7 \times 7]$.

- **Suppression de bruit mixte.** Pour une image corrompue avec un bruit mixte, le filtre médian multiétage est utilisé pour localiser les pixels bruités. Une fois que la localisation de bruit impulsionnel est faite, on estime l'image de sortie par l'estimation MAP selon l'équation 2.22. Dans ce cas, tous les pixels sont considérés.

#### 2.4.2.3 Suppression de bruit en utilisant la représentation par hypergraphe de voisinage

### Principe

La stratégie que nous adoptons est une stratégie de décision binaire / estimation. Cette démarche s'apparente à celle utilisée dans la théorie de la décision et de l'estimation bayésienne. Elle en diffère néanmoins car les hypothèses testées et les connaissances a priori utilisées sont de nature différente. Dans la stratégie bayésienne, on utilise les distributions de l'observation sous chacune des hypothèses considérées. Classiquement, les problèmes traités concernent la discrimination entre un bruit et un signal utile. Lorsque le bruit est détecté, un algorithme d'estimation est utilisé pour approximer le signal. L'estimation est dans ce cas conditionnée à la détection du bruit. Le modèle de bruit proposé permet de distinguer un bruit d'une quelconque hypothèse alternative (contour, région). Les connaissances utilisées concernent l'organisation spatiale des pixels - à travers la notion de voisinage sur la grille - $\beta$ et la distribution d'intensité - à travers le paramètre $\alpha$.

L'algorithme de débruitage proposé exploite la notion d'inhomogénéité qui caractérise le bruit. Il opère en trois étapes :

1. Construction de l'Hypergraphe de Voisinage d'une Image HVI ;
2. Classification binaire des hyperarêtes de l'image :
   - $H_0$ hyperarête de bruit
   - $H_1$ hyperarête de donnée non bruitée
3. Filtrage des zones bruitées.

### Définition des hyperarêtes de bruit

L'hypergraphe de voisinage d'une image vise à obtenir un modèle de la forme et des caractéristiques géométriques importantes des objets d'intérêt. La réduction du bruit est un processus de comparaison des propriétés géométriques décrivant la forme de bruit dans une image à un modèle. Il est donc de toute importance de bien décrire la structure des hyperarêtes liée au bruit dans l'hypergraphe de voisinage d'une image. Le bruit traduit une inhomogénéité dans une image. Les hyperarêtes isolées peuvent être utilisées pour modéliser une inhomogénéité dans une image. Comme son nom l'indique, l'hyperarête isolée est une hyperarête qui n'a aucune information partagée avec son voisinage ouvert dans l'image. Nous appelons voisinage ouvert de l'ensemble $S$ noté $\Gamma^o(S)$, l'ensemble $\Gamma(S)\backslash S$. En utilisant cette propriété, nous proposons donc le modèle qui suit.

**Modèle de bruit 1** *Une hyperarête $E_{\alpha,\beta}$ est une hyperarête de bruit si elle vérifie l'une des deux conditions :*

1. *La cardinalité de $E_{\alpha,\beta}$ est égale à un ;*
2. *$E_{\alpha,\beta}(x)$ est une hyperarête isolée et pour tout y élément du voisinage ouvert sur la grille de $E_{\alpha,\beta}(x)$, il existe au moins une hyperarête $E_{\alpha,\beta}(y)$ isolée.*

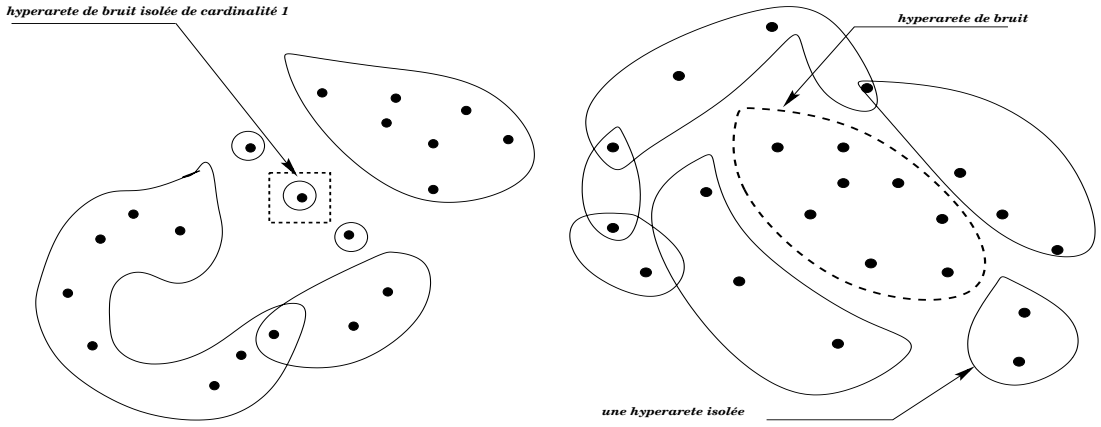



FIG. *2.4: Illustration graphique du modèle 1.*

Ce modèle 1 met en évidence les hyperarêtes qui sont incohérentes avec leur environnement. Elle permet de traduire l'inhomogénéité de l'image et donc de détecter le bruit. La figure 2.4 illustre le modèle d'une hyperarête de bruit. Le lemme 2.4.1 montre qu'une hyperarête de bruit est nécessairement isolée.

**Lemme 2.4.1** *Si la cardinalité d'une hyperarête est égale à un, alors cette hyperarête est isolée.*

**Preuve 2.4.1** *Soit $E_{\alpha,\beta}(x)$ une hyperarête de cardinalité égale à un, supposons qu'elle ne soit pas isolée. Elle est donc contenue dans une hyperarête $E_{\alpha,\beta}(y)$, ainsi la cardinalité de $E_{\alpha,\beta}(x)$ est supérieure à un, contradiction.*

Le modèle précédent permet de classer les hyperarêtes isolées soit de cardinalité égale à 1 soit$>1$. Les hyperarêtes isolées de cardinalité 1 décrivent l'inhomogénéité, mais cette propriété est aussi

caractéristique des contours. La différence essentielle tient dans la structuration spatiale des contours dans l'image. En général, les pixels contours forment des chaînes. Afin de distinguer le bruit des contours, on propose de classer les hyperarêtes de cardinalité 1 comme hyperarêtes de bruit si et seulement si elles ne forment pas une chaîne disjointe avec les hyperarêtes voisines. La figure 2.5 illustre le deuxième modèle de bruit. Le nouveau modèle des hyperarêtes de bruit est donc :

**Modèle de bruit 2** *Une hyperarête $E_{\alpha,\beta}$ est une hyperarête de bruit si elle vérifie l'une des deux conditions :*

1. *La cardinalité de $E_{\alpha,\beta}$ est égale à un ;* **et** $E_{\alpha,\beta}(x)$ **n'est pas incluse dans une chaîne disjointe possédant $\omega$ éléments au maximum**,
2. *$E_{\alpha,\beta}(x)$ est une hyperarête isolée et pour tout y élément du voisinage ouvert sur la grille de $E_{\alpha,\beta}(x)$, il existe au moins une hyperarête $E_{\alpha,\beta}(y)$ isolée.*

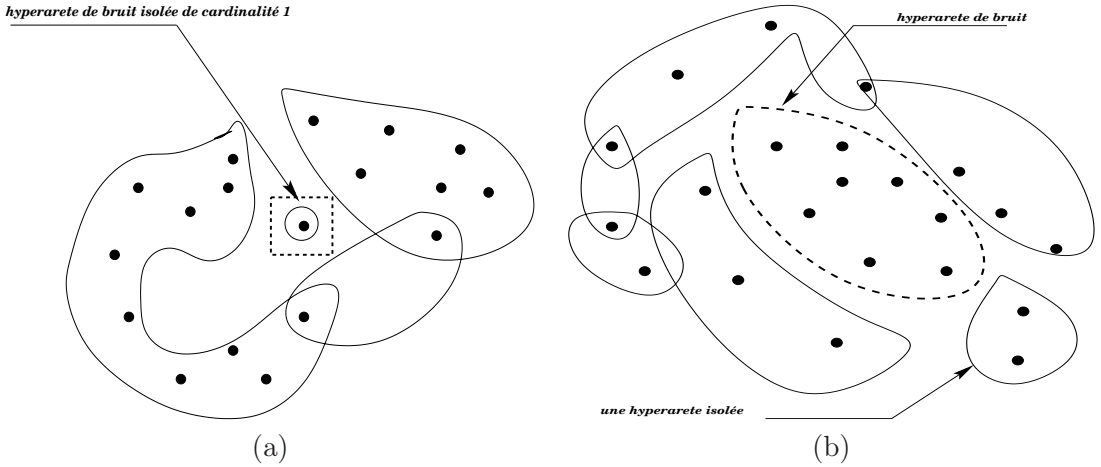



FIG. *2.5: Illustration graphique du modèle 2.*

## 2.5 Approche de suppression de bruit proposée

### 2.5.1 Algorithme

Dans cette section, nous présentons l'algorithme de débruitage appliqué aux images à niveaux de gris. Dans le chapitre 1, nous avons vu que la modélisation par hypergraphe de voisinage peut être générée soit en utilisant un seuil global sur toute l'image, soit en appliquant un seuil dynamique. Nous présentons ci-dessous un algorithme de débruitage.

- □ **Données.**
  Image à niveaux de gris $I$, ordre de voisinage $\beta$, longueur de la chaîne disjointe $\omega$ ;
- □ **Construction de l'hypergraphe de voisinage $H_{\alpha,\beta}$ sur $I$.**
  - ▷ hypergraphe de voisinage à seuil fixe :
    $\alpha$ fixe, $\in [0, 255]$ et $d'(I(x), I(y)) = |I(x) - I(y)| \leq \alpha$.

▷ hypergraphe de voisinage à seuil dynamique :
$\alpha = f(I) = k\hat{\sigma}$, avec $\hat{\sigma}$ est l'estimé de l'écart-type du voisinage 8-adjacents et k est une constante. $d'(I(x), I(y)) = |I(x) - I(y)| \leq \hat{\sigma}$.

▷ hypergraphe de voisinage utilisant une fonction de similarité comme distance spatiocolorimétrique :
$\alpha$ fixe, $\in [0, 1]$ et $d'(I(x), I(y)) = \mu(I(x), I(y)) \geq \alpha$. $\mu(.)$ est une fonction de similarité.

□ **Détermination des hyperarêtes isolées dans $H_{\alpha,\beta}$.** Pour chaque sommet $x$ de $X$

▷ $E' \longleftarrow \emptyset$

▷ $E' \longleftarrow \bigcup_{y \in E_{\alpha,\beta}(x)} E_{\alpha,\beta}(y)$

▷ si $E' = E_{\alpha,\beta}(x)$, ($E_{\alpha,\beta}(x)$ est une hyperarête isolée) alors ; $E^{is}_{\alpha,\beta}(x) \longleftarrow E_{\alpha,\beta}(x)$.

□ **Détection des hyperarêtes de bruit.** Pour chaque $E^{is}_{\alpha,\beta}(x)$ vérifiant le modèle 1 ou 2 alors $E^{b}_{\alpha,\beta}(x) \longleftarrow E^{is}_{\alpha,\beta}$ ;

□ **Estimation.** Pour chaque hyperarête de bruit $E^{b}_{\alpha,\beta}(x)$, remplacer l'intensité de $E^{b}_{\alpha,\beta}(x)$ par la valeur d'une fonctionnelle dépendant des intensités de $\Gamma^o(E^{b}_{\alpha,\beta}(x))$.

**Propriétés de l'algorithme : Convergence, Complexité et Unicité**

**Lemme 2.5.1** *Pour $\alpha$ et $\beta$ donnés, l'algorithme converge vers une solution unique. Sa complexité est en $O(n)$ (n désignant le nombre de pixels de l'image).*

**Preuve 2.5.1** *Une étude des étapes de l'algorithme permet de constater que la complexité de chacune de celles-ci est majorée par le nombre de pixels de l'image à un coefficient multiplicatif près. Ce coefficient multiplicatif est le nombre de pixels contenus dans une hyperarête , majoré lui-même par la cardinalité maximale des hyperarêtes. Par conséquent, la complexité de l'algorithme est en $O(n)$. De plus, pour un $\beta$ fixe et un seuil $\alpha$ dynamique ou fixe, il existe un hypergraphe unique $H_{\alpha,\beta}$ associé à une image donnée, d'où la convergence vers une classification unique et une solution unique.*

### 2.5.2 Simulations et résultats expérimentaux

Afin de nous assurer de l'efficacité de la représentation HVI et des modèles de bruit, nous avons effectué une série de mesures sur des images bruitées. Nous avons donc mené une série d'expérimentations dans le cas d'une perturbation de nature additive et d'une perturbation de nature multiplicative des niveaux de gris de l'image à restaurer. Dans les deux cas, une perturbation gaussienne de l'intensité des pixels a été simulée. Par ailleurs, une perturbation additive de distribution alpha-stable, a été simulée. La distribution spatiale de la position des perturbations est dans tous les cas uniformément distribuée sur la grille, ce qui permet d'affirmer l'équiprobabilité des positions des perturbations. Enfin, nous avons fait varier la densité de perturbations spatiales entre 5% et 35% des pixels de l'image, et ce, pour les mêmes proportions de pixels bruités spatialement.

L'algorithme proposé comporte deux grandes parties : détection et estimation. Ceci nous conduit à générer deux procédures de validation et comparaison : évaluation de la détection de bruit et ensuite évaluation de l'estimation des hyperarêtes de bruit.

Dans le chapitre 1, nous avons présenté trois modes de construction de l'hypergraphe de voisinage. Ces représentations sont obtenues par changement, soit du seuil colorimétrique $\alpha$, soit

| Nomenclature | Seuil $\alpha$ | Distance $d'(I(x), I(y))$ |
|---|---|---|
| Représentation A | fixe | $\|I(x) - I(y)\|$ |
| Représentation B | dynamique | $\|I(x) - I(y)\|$ |
| Représentation C | fixe | $\mu(I(x), I(y))$ |

TAB. *2.1: Nomenclature des hypergraphes de voisinage de l'image.*

par changement de la distance colorimétrique $d'(I(x), I(y))$ entre les pixels dans la fenêtre $W$ de voisinage 8-adjacents ($\beta = 1$). Afin de faciliter la lecture des résultats de simulation, nous utilisons les nomenclatures illustrées dans la table 2.1.

L'objectif de cette section est double. Dans un premier temps, il s'agit de déterminer quelle représentation d'hypergraphe est la plus adéquate pour le problème traité. Dans un second temps, nous comparons les performances de l'algorithme de débruitage proposé avec celles des solutions classiquement utilisées dans une stratégie d'estimation (Wiener, Médian, Médian multiétage) et celles proposées dans le cadre d'une stratégie détection et estimation (algorithme de Tovar, algorithme de Stevenson).

Dans la section 2.5.2.1, nous présentons les critères d'évaluation afin de mesurer le pouvoir de détection du bruit ainsi que la qualité de l'estimation des hyperarêtes de bruit.

Nous disposons de deux modèles d'hyperarête de bruit (modèle 1, modèle 2) et de trois représentations d'hypergraphe de voisinage : A, B et C.

Dans la section 2.5.2.2, l'évaluation de l'algorithme de débruitage utilisant la représentation A est double. Dans un premier temps, nous évaluons les performances de l'algorithme avec le modèle 1. Dans un second temps, nous comparons ces performances avec l'algorithme utilisant cette fois le modèle 2, ceci afin de déterminer le modèle adéquat de bruit. L'évaluation de l'algorithme utilisant les représentations B et C en terme de détection et d'estimation est présentée respectivement dans les sections 2.5.2.3 et 2.5.2.4. Les manipulations présentées dans les sections 2.5.2.2, 2.5.2.3, 2.5.2.4 ont pour objectif la comparaison des modèles A, B et C afin de trouver la bonne représentation par hypergraphe de voisinage compatible avec l'application proposée. Nous passons ensuite dans la section 2.5.3 à une comparaison de leurs performances cette fois avec les algorithmes de débruitage utilisant une stratégie de détection et d'estimation (Stevenson et Tovar) et ensuite avec les méthodes utilisant une stratégie d'estimation (Wiener, Médian, Médian multiétage).

#### 2.5.2.1 Mesures d'évaluations

**(a). Détection du bruit.** Afin de mesurer les performances de détection de bruit, nous avons généré différentes cartes de bruit de différentes densités. Ces cartes ont été appliquées sur des images naturelles pour chacun des types et natures de bruit. Nous avons ensuite estimé deux paramètres pour chacune des images bruitées ainsi générées à savoir des probabilités de détection $P_D$ et de fausse alarme $P_{FA}$ [MFC75] [CJD94].

Rappelons que pour une réalisation de bruit $i$, la probabilité de détection est estimée comme le quotient du nombre de pixels détectés par un algorithme comme pixels bruités par rapport au nombre total de pixels bruités de la carte de bruit. La probabilité de fausse alarme est quant à elle estimée par la proportion de pixels classifiés comme bruit par rapport au nombre total de pixels non bruités de la carte de bruit. Ces définitions sont résumées par les deux formules suivantes :

$$\hat{pd}_i = \frac{\#\{\text{pixels correctement classifiés comme bruit}\}}{\#\{\text{pixels bruités}\}} \tag{2.24}$$

$$\hat{pf}_i = \frac{\#\{\text{pixels incorrectement classifiés comme bruit}\}}{\#\{\text{pixels non-bruités}\}} \tag{2.25}$$

Pour estimer les probabilités $\hat{pd}$ et $\hat{pf}$, on génère en fait dix réalisations de bruit et on calcule la valeur moyenne des estimés individuelles ; ce qui peut se résumer par l'algorithme suivant :

- Pour $i = 1$ jusqu'à $i = 10$ (10 réalisations)
  - ▷ génération d'une carte de bruit différente de la réalisation précédente,
  - ▷ détection de bruit par l'algorithme proposé,
  - ▷ calcul de $\hat{pd}_i$ et $\hat{pf}_i$.
- Faire la moyenne : $\hat{pd} = \frac{1}{10}\sum_i pd_i$, $\hat{pf} = \frac{1}{10}\sum_i pf_i$

**(b). Estimation des hyperarêtes de bruit.** Le rapport signal sur bruit crête (Peak Signal to Noise Ratio) connu sous l'étiquette PSNR est la mesure classiquement utilisée pour évaluer les performances en suppression de bruit. Cette mesure permet de quantifier la distorsion qui existe entre deux images. Il est défini par :

$$PSNR = 10\log_{10}\left(\frac{255^2}{\sum_{x,y}\left(I(x,y) - \hat{I}(x,y)\right)^2}\right) \tag{2.26}$$

où $I(x,y)$ représente le pixel de coordonnées $(x,y)$ de l'image initiale et $\hat{I}(x,y)$ le pixel de coordonnées $(x,y)$ de l'image débruitée.

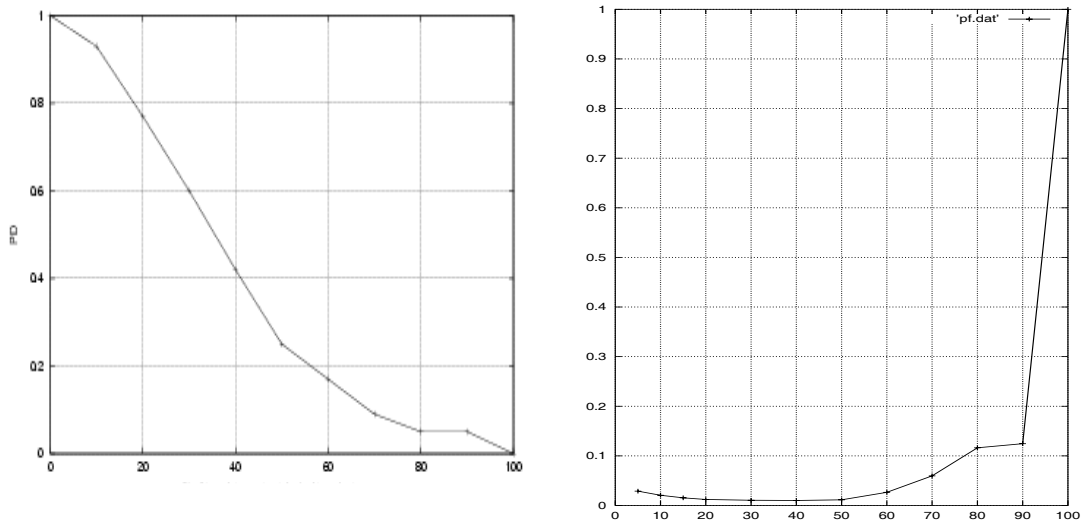
#### 2.5.2.2 Evaluation de la représentation A

Dans ce paragraphe, l'hypergraphe de voisinage de l'image est généré en utilisant un seuil fixe. Le paramètre $\alpha$ prend a priori sa valeur dans l'intervalle $[0; 255]$. Néanmoins, il est bien évident que certaines valeurs externes ne mèneront à aucun résultat intéressant : par exemple, prendre pour $\alpha$, la valeur 255, ne conduira qu'à l'obtention d'un hypergraphe correspondant à l'image tout entière. De même, de faibles valeurs de $\alpha$ conduiront à la construction d'un très grand nombre d'hyperarêtes, le pire des cas étant obtenu pour une valeur nulle de $\alpha$ où chaque pixel de l'image peut engendrer une hyperarête isolée. De plus, ces hyperarêtes isolées seront adjacentes les unes aux autres. Ainsi, le nombre d'hyperarêtes isolées adjacentes sera de l'ordre du nombre d'hyperarêtes elles-mêmes et l'image sera considérée comme trop bruitée pour pouvoir prétendre en faire quoi que ce soit. Enfin, les expériences ont été menées sur plusieurs images différentes de scènes naturelles issues de la base de données du GDR-ISIS [Gi].

**(a). Détection des hyperarêtes de bruit par le modèle 1**

**Evaluation de la détection.** Les principales conclusions que nous pouvons formuler à partir de l'étape de la détection du bruit (courbes opérationnelles) sont les suivantes :

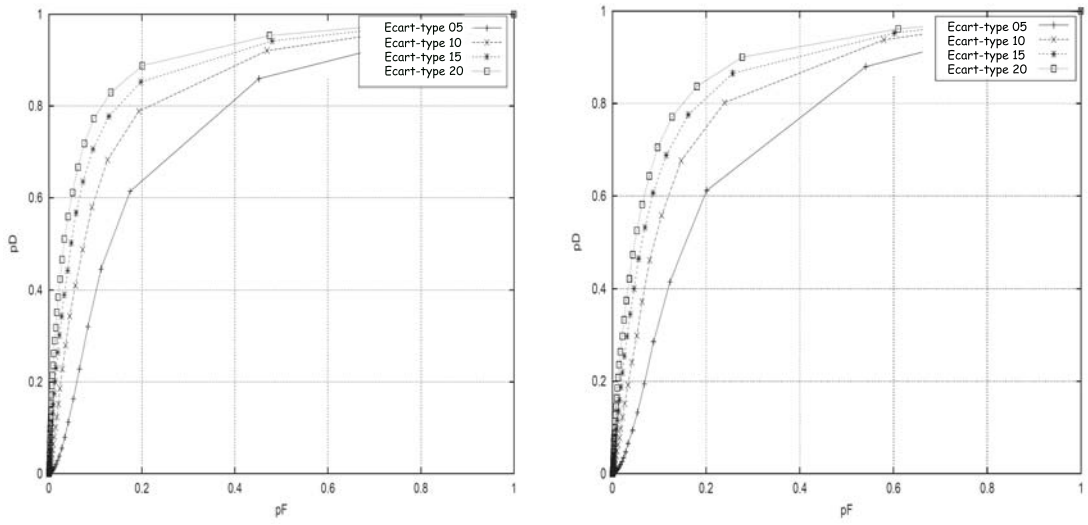
- L'efficacité de l'algorithme décroît si la proportion de pixels bruités dans l'algorithme croît. Ceci peut être expliqué par le fait que si l'inhomogénéité spatiale des données croît, le modèle 1 ne peut plus distinguer les hyperarêtes affectées par le bruit et celles dont les évaluations des cellules n'ont pas été modifiées (Fig. 2.6).




(a) (b)

FIG. *2.6: Evolution de (a) pD et (b) pF en fonction du pourcentage du bruit impulsionnel injecté dans l'image Poivron.*

- Plus la variance du bruit est élevée, plus le modèle 1 est apte à discerner les hyperarêtes de bruit. A titre illustratif, la figure 2.7 représente les courbes opérationnelles pour une perturbation gaussienne additive pour quatre valeurs d'écart-type (5, 10, 15 et 20) qui affecte 30% et 50% des pixels de l'image.
- Pour une même distribution de la perturbation, le modèle 1 est plus apte à détecter celle-ci lorsque cette perturbation s'exerce de manière additive plutôt que multiplicative comme le montre la figure 2.8.




(a) (b)

FIG. *2.7: Courbe expérimentale montrant la croissance de performance de l'algorithme avec la croissance de l'écart-type du bruit.*

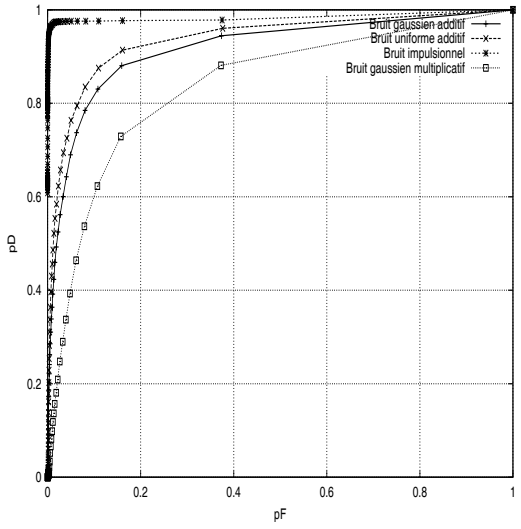



FIG. *2.8: Courbe expérimentale montrant la meilleure performance de l'algorithme pour les bruits de type additif par rapport à un type multiplicatif.*

**Evaluation de l'estimation.** Après avoir détecté les hyperarêtes de bruit par l'algorithme proposé, nous avons appliqué un estimateur sur le voisinage ouvert $\Gamma^o(E^{bruit}_{\alpha,\beta}(x))$. Dans l'évaluation de la procédure de l'estimation des hyperarêtes de bruit, nous avons traité les trois types de bruit : gaussien additif, alpha-stable additif et impulsionnel. Ces types de bruit ont été injectés d'abord dans une image de synthèse, ensuite dans des images naturelles telle que Maison et Bateau. Le choix de l'estimateur dépend du type de bruit injecté dans l'image. Deux filtres sont utilisés, filtre médian et filtre de Wiener d'ordre 1 (Voisinage 8-adjacents) : le filtre médian pour le bruit alpha-stable et impulsionnel, tandis que le filtre de Wiener pour le bruit gaussien.

Le rapport PSNR est utilisé pour donner une évaluation quantitative des effets de la suppression de bruit. Nous commençons d'abord dans ces évaluations par une série de manipulations sur une image de synthèse afin de caractériser le comportement de l'algorithme vis à vis des types de bruit traités. Ensuite, nous comparons les résultats de l'algorithme sur plusieurs types d'images naturelles, tels que Maison, et Bateau. Ces images sont illustrées dans la figure 2.9.

Synthèse Maison Bateau

FIG. *2.9: Les images de simulations.*

Dans les tables 2.2, 2.3 et 2.4, sont présentées respectivement les performances de l'estimation des bruits gaussien, alpha-stable et impulsionnel. En examinant les résultats obtenus sur ces types de bruit, on peut remarquer que l'algorithme est plus performant pour la suppression de bruit alpha-stable additif et impulsionnel par rapport au bruit gaussien. Cette observation est donnée par les valeurs de $\Delta$PSNR. Ce dernier est défini par :

$$\Delta PSNR = PSNR_{If} - PSNR_{Ib} \tag{2.27}$$

où les rapports $\text{PSNR}_{If}$ et $\text{PSNR}_{Ib}$ représentent respectivement les rapports de PSNR des images filtrée et bruitée.

| paramètre de bruit | gaussien additif | | |
|---|---|---|---|
| (%, $\sigma$) | $\text{PSNR}_{dB}$ $I_b$ | $\text{PSNR}_{dB}$ $I_f$ | $\Delta$PSNR |
| $(5, 15)$ | 37.98 | 51.02 | 13.04 |
| $(5, 20)$ | 35.74 | 45.81 | 10.07 |
| $(5, 30)$ | 32.39 | 38.89 | 6.50 |

TAB. *2.2: Evaluation de la suppression de bruit gaussien additif injecté dans l'image de synthèse par l'algorithme proposé ($\alpha = 5$, $\beta = 1$).*

Les tables 2.5, 2.6 et 2.7 présentent les résultats de suppression de bruit obtenus sur les images naturelles (Maison et Bateau). Afin de rendre la comparaison cohérente, les paramètres de détection $\alpha$ et $\beta$ et d'estimation (taille des filtres médian et Wiener) sont choisis égaux pour

| paramètre de bruit | | | Bruit à distribution alpha-stable | | |
|---|---|---|---|---|---|
| % | $\alpha$ | dispersion $\gamma$ | PSNR$_{dB}$ $I_b$ | PSNR$_{dB}$ $I_f$ | $\Delta$PSNR |
| 5 | 0.5 | 1 | 27.45 | 62.26 | 34.81 |
| 5 | 0.8 | 1 | 30.19 | 60.08 | 29.89 |
| 5 | 1.5 | 1 | 31.67 | 60.51 | 28.84 |

TAB. *2.3: Evaluation de la suppression de bruit de distribution alpha-stable injecté dans l'image de synthèse par l'algorithme proposé ($\alpha = 5$, $\beta = 1$).*

| paramètre de bruit | Bruit impulsionnel | | |
|---|---|---|---|
| % | PSNR$_{dB}$ $I_b$ | PSNR$_{dB}$ $I_f$ | $\Delta$PSNR |
| 5 | 17.67 | 47.44 | 29.77 |

TAB. *2.4: Evaluation de la suppression de bruit impulsionnel injecté dans l'image de synthèse par l'algorithme proposé ($\alpha = 5$, $\beta = 1$).*

les deux images. Les paramètres $\alpha$ et $\beta$ sont fixés respectivement à 5 et 1. La taille des filtres Médian et Wiener est de $3 \times 3$. Dans la table 2.5, nous comparons la suppression de bruit gaussien injecté dans les images Maison et Bateau. Les résultats de suppression de bruit alpha-stable sont présentés dans la table 2.6. La table 2.7 donne les résultats de suppression de bruit impulsionnel. En examinant ces résultats, et particulièrement les $\Delta$PSNR, on peut remarquer que la suppression de bruit dans l'image Bateau donne de moins bons résultats que l'image Maison. Cette observation est justifiée par le fait que l'image Bateau présente plus de détails que dans l'image Maison.

| bruit gaussien | Maison | | | Bateau | | |
|---|---|---|---|---|---|---|
| ($\%$, $\sigma$) | PSNR$_{dB}$ $I_b$ | PSNR$_{dB}$ $I_f$ | $\Delta$PSNR | PSNR$_{dB}$ $I_b$ | PSNR$_{dB}$ $I_f$ | $\Delta$PSNR |
| $(5, 15)$ | 31.13 | 36.29 | 5.16 | 37.71 | 40.08 | 2.37 |
| $(5, 20)$ | 31.02 | 36.26 | 5.24 | 35.29 | 39.09 | 3.80 |
| $(5, 30)$ | 31.08 | 36.27 | 5.19 | 31.87 | 36.98 | 5.11 |

TAB. *2.5: Evaluation de la suppression de bruit gaussien additif injecté dans les images Maison et Bateau par l'algorithme proposé ($\alpha = 5$, $\beta = 1$).*

Les figures 2.10, 2.11 et 2.12 donnent respectivement les résultats de suppression des bruits alpha-stable, gaussien additif et impulsionnel injectés dans l'image Maison.

**(b). Comparaison des modèles de bruit**

Afin de trouver le modèle de bruit adéquat au problème de suppression de bruit par hypergraphe de voisinage, nous comparons les résultats des deux modèles 1 et 2 de bruit utilisés par l'algorithme (représentation A).

En effet, pour comparer ces deux derniers, nous traçons les courbes opérationnelles de détection de bruit gaussien additif d'écart-type 20 et de pourcentage 5% injecté dans l'image Maison. La courbe opérationnelle représente l'évolution de la probabilité de détection $\hat{pd}$ en fonction de la probabilité de fausse alarme $\hat{pf}$ pour des valeurs de $\alpha$ variant de 0 à 255.

D'après la figure 2.13, nous constatons que la courbe représentative du modèle 2 est située au dessus de la courbe opérationnelle correspondante au modèle 1. Ceci signifie que pour n'importe

| bruit alpha-stable | Maison | | | Bateau | | |
|---|---|---|---|---|---|---|
| (%,$\alpha$,$\gamma$) | PSNR$_{dB}$ $I_b$ | PSNR$_{dB}$ $I_f$ | $\Delta$PSNR | PSNR$_{dB}$ $I_b$ | PSNR$_{dB}$ $I_f$ | $\Delta$PSNR |
| $(5, 0.5, 1)$ | 29.95 | 42.04 | 12.09 | 30.92 | 40.58 | 9.66 |
| $(5, 0.8, 1)$ | 34.69 | 43.19 | 8.50 | 35.89 | 41.99 | 6.10 |
| $(5, 1.5, 1)$ | 37.55 | 44.46 | 6.91 | 38.83 | 43.70 | 4.87 |

TAB. *2.6: Evaluation de la suppression de bruit de distribution alpha-stable dans les images Maison et Bateau par l'algorithme proposé ($\alpha = 5$, $\beta = 1$).*

| % | Maison | | | Bateau | | |
|---|---|---|---|---|---|---|
| % | PSNR$_{dB}$ $I_b$ | PSNR$_{dB}$ $I_f$ | $\Delta$PSNR | PSNR$_{dB}$ $I_b$ | PSNR$_{dB}$ $I_f$ | $\Delta$PSNR |
| 5 | 18.12 | 34.12 | 16 | 18.65 | 36.43 | 17.78 |
| 10 | 15.16 | 29.67 | 14.51 | 15.55 | 30.31 | 14.76 |

TAB. *2.7: Evaluation de la suppression de bruit impulsionnel injecté dans les images Maison et Bateau par l'algorithme proposé ($\alpha = 5$, $\beta = 1$).*

(a) (b)

FIG. *2.10: Bruit de distribution alpha-stable ($\alpha$=0.5, $\gamma = 1$, 5% ). (a) l'image bruitée. (b) l'image estimée par l'algorithme proposé avec les seuils $(\alpha, \beta) = (5, 1)$. Les hyper-arêtes de bruit sont estimées par un filtre médian $3 \times 3$.*

(a) (b)

FIG. 2.11: *Bruit gaussien additif ($\sigma = 20$, 5%). (a) L'image bruitée. L'écart-type du bruit a été choisi ici volontairement assez grand de façon à générer des possibilités de saturation du niveau de gris des pixels et donc à montrer visuellement l'efficacité de l'algorithme proposé. (b) L'image estimée par l'algorithme $(\alpha, \beta) = (5, 1)$.*

(a) (b)

FIG. 2.12: *Bruit impulsionnel (5%). (a). L'image bruitée. (b) L'image estimée par l'algorithme $(\alpha, \beta) = (5, 1)$.*

quelle valeur de $\alpha$, la probabilité de détection de bruit par le modèle 2 est supérieure à la probabilité de détection de bruit par le modèle 1. La même remarque est valable pour la probabilité de fausse alarme. Par conséquent, le modèle 2 conduit à des résultats de suppression de bruit bien meilleurs que le modèle 1. Ces résultats se confirment quelque soit la nature du bruit additif.

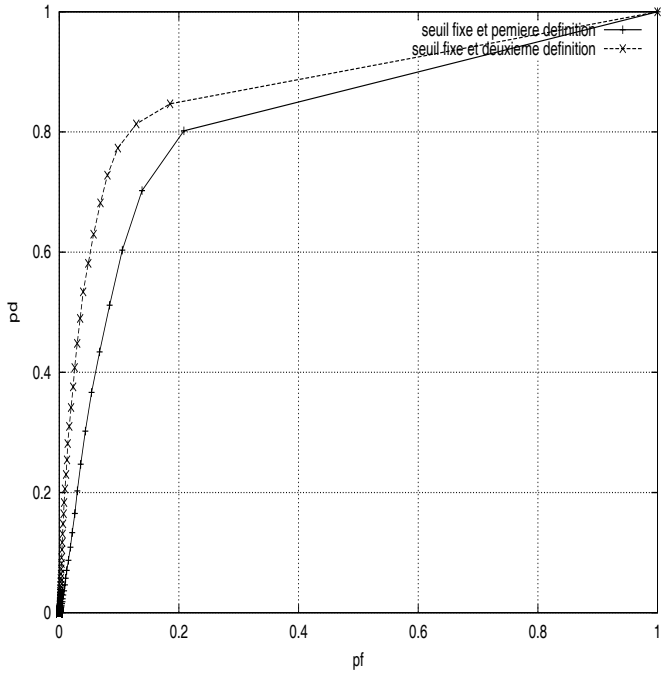
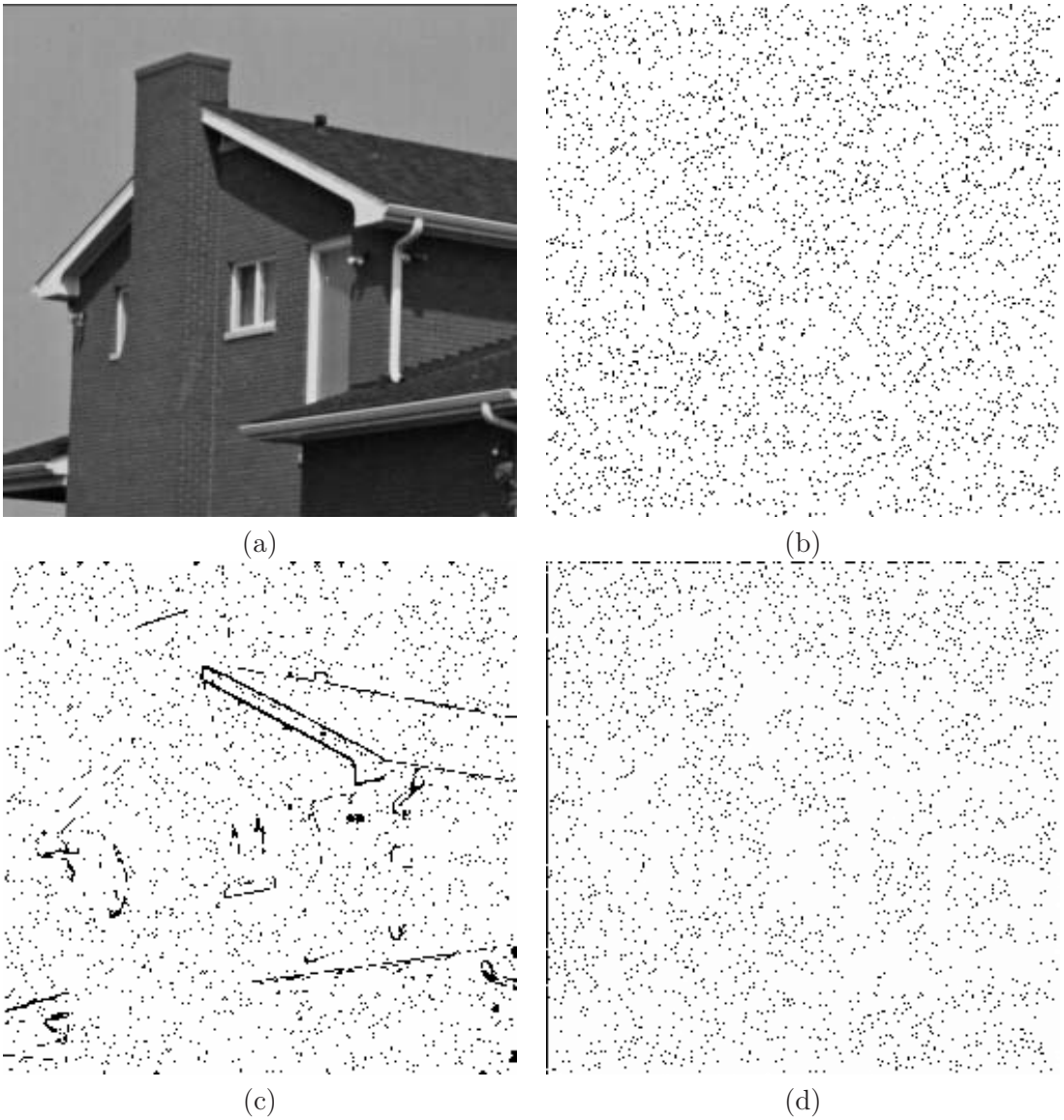



FIG. *2.13: Comparaison de la détection des hyperarêtes de bruit en utilisant les deux modèles de bruit 1 et 2.*

La courbe opérationnelle ne permet pas d'affirmer que les contours sont mieux préservés. Elle nous donne qu'une supériorité globale du deuxième modèle. Pour illustrer l'aptitude de ce modèle à distinguer contour et bruit, nous montrons visuellement les sorties de l'algorithme pour un bruit gaussien. La figure 2.14 illustre ces résultats. Les sorties de l'algorithme ont été générées pour la même valeur du seuil $\alpha = 5$. L'image originale est corrompue avec un bruit gaussien d'écart-type $\sigma = 20$ et de pourcentage $p = 5\%$. Il est clair d'après cette figure que l'objectif de la première condition du modèle 2 est atteint. D'après cette figure, le modèle 2 est capable de distinguer le bruit avec n'importe quelles informations de l'image traitée que ce soit les contours ou les régions homogènes.

Le modèle 2 emploie un paramètre $\omega$ indiquant la longueur de la chaîne disjointe. Le choix de paramètres $\omega$ est fixé expérimentalement et dépend de l'image traitée. Pour choisir la valeur de ce paramètre, nous avons réalisé plusieurs simulations sur différentes images et nous avons constaté que la longueur d'une chaîne disjointe égale à 5 conduit à des performances satisfaisantes dans la majorité des cas.

D'après la figure 2.13 et la figure 2.14, nous pouvons constater que le deuxième modèle des hyperarêtes de bruit détecte mieux le bruit dans l'image tout en conservant les contours. Dans la suite, nous utiliserons dorénavant, uniquement ce modèle pour détecter les hyperarêtes de bruit.



(a) (b) (c) (d)

FIG. *2.14: La sortie de l'algorithme avec un seuil $\alpha = 5$. (a) (a) Image originale "Maison". (b) Bruit injecté dans l'image. (c) Bruit détecté par la condition 1 du modèle 1. (d) Bruit détecté par la condition 1 du modèle 2.*

#### 2.5.2.3 Evaluation de la représentation B

Cette fois, l'algorithme proposé emploie une modélisation par hypergraphe de voisinage d'image à seuil dynamique $\alpha = f(I)$. La construction des hyperarêtes $E(x)$ de l'image dépend de l'estimé de l'écart-type de voisinage 8-adjacents du pixel $I_x$. Le seuil dynamique $\alpha$ est donné par : $\alpha = k\sigma$, où k est une constante positive. Durant ces expérimentations, la valeur de k est 0.5.

Nous commençons d'abord par une évaluation des performances de l'algorithme en fonction de la nature du bruit injecté dans l'image. Puis, nous évaluons l'évolution des performances en fonction du contenu des images traitées. La figure 2.15 présente les images de simulation par ordre croissant de complexité du contenu.

(a) (b) (c) (d)

FIG. *2.15: les images de simulation. (a) Synthèse, (b) Maison, (c) Poivron et (d) Bateau.*

Dans les tables 2.8, 2.9 et 2.10, sont présentées les performances de la détection des bruits gaussien, alpha-stable et impulsionnel. En examinant les résultats obtenus sur ces différentes natures de bruit, on peut remarquer que l'algorithme donne de bons résultats de suppression de bruit impulsionnel par rapport aux autres types de bruit. Les valeurs en moyenne de la probabilité de détection $\hat{pd}$ des bruits impulsionnel, alpha-stable et gaussien additif sont respectivement 0.97, 0.90 et 0.82, tandis que la valeur de la probabilité de fausse alarme est sensiblement égale. D'après le tableau, nous constatons que la probabilité de fausse alarme est de l'ordre de $10^{-3}$. La principale remarque que suggèrent ces résultats est que les performances évoluent dans le même sens que le caractère impulsif du bruit. Autrement dit, plus le bruit comporte un caractère impulsif plus l'algorithme est apte à le détecter.

| paramètres de bruit | gaussien additif | |
|---|---|---|
| (%, $\sigma$) | $\hat{pd}$ | $\hat{pf}$ |
| $(5, 15)$ | 0.8471 | 0.00787 |
| $(5, 20)$ | 0.8284 | 0.00785 |
| $(5, 30)$ | 0.8746 | 0.00788 |

TAB. *2.8: Evaluation de la détection de bruit gaussien additif injecté dans l'image Synthèse par l'algorithme proposé (représentation B) en utilisant $\hat{pd}$ et $\hat{pf}$.*

Les tests effectués sur des images naturelles montrent qu'il y a une différence notable entre les résultats obtenus par l'algorithme sur les images Maison, Poivron et Bateau. Les résultats de comparaison de l'algorithme sur ces images sont présentés dans les tables 2.11, 2.12 et 2.13. En examinant une des tables, par exemple la suppression de bruit impulsionnel (Tab. 2.13), nous constatons que les performances de détection en terme de probabilité de détection $\hat{pd}$ sont bien meilleures dans le cas des images Maison et Poivron que dans l'image Bateau. Par ailleurs, en comparant cette fois la probabilité de fausse alarme $\hat{pf}$ entre les images Maison et Poivron, nous

| paramètres de bruit | bruit alpha-stable | |
|---|---|---|
| (%,$\alpha$,$\gamma$) | $\hat{pd}$ | $\hat{pf}$ |
| $(5, 0.5, 1)$ | 0.9248 | 0.007693 |
| $(5, 0.8, 1)$ | 0.9080 | 0.007795 |
| $(5, 1.5, 1)$ | 0.8665 | 0.007873 |

TAB. *2.9: Evaluation de la détection de bruit alpha-stable injecté dans l'image Synthèse par l'algorithme proposé (représentation B) en utilisant $\hat{pd}$ et $\hat{pf}$.*

| paramètres de bruit | bruit impulsionnel | |
|---|---|---|
| % | $\hat{pd}$ | $\hat{pf}$ |
| 5 | 0.9741 | 0.0077 |

TAB. *2.10: Evaluation de la détection de bruit impulsionnel injecté dans l'image Synthèse par l'algorithme proposé (représentation B) en utilisant $\hat{pd}$ et $\hat{pf}$.*

remarquons que la fausse alarme est plus faible dans le cas d'image Maison. Pour les autres natures de bruit, il suffit de comparer les valeurs de $\hat{pd}$ pour remarquer que les images Maison et Poivron restent les images où les performances en terme de détection sont bonnes. Ceci peut en partie s'expliquer par le fait que l'image Bateau contient plus de détails. Autrement dit, plus l'image est "homogène" plus les performances augmentent.

| bruit gaussien | Maison | | Poivron | | Bateau | |
|---|---|---|---|---|---|---|
| (%, $\sigma$) | $\hat{pd}$ | $\hat{pf}$ | $\hat{pd}$ | $\hat{pf}$ | $\hat{pd}$ | $\hat{pf}$ |
| $(5, 20)$ | 0.7894 | 0.0617 | 0.6689 | 0.1276 | 0.6105 | 0.1150 |
| $(5, 30)$ | 0.8027 | 0.0651 | 0.6752 | 0.1277 | 0.6749 | 0.1065 |

TAB. *2.11: Evaluation de la détection de bruit gaussien additif injecté dans les images naturelles Maison, Poivron et Bateau.*

**Comparaison des représentations A et B.** L'objectif de cette section est de démontrer l'influence du paramètre $\alpha$ sur la détection des hyperarêtes de bruit. Pour cela, nous comparons les performances de l'algorithme de détection en utilisant les représentations A et B. Pour l'algorithme proposé avec la représentation A, nous traçons les $\hat{pd}$ et $\hat{pf}$ en fonction d'un seuil $\alpha$ variant de 0 à 255. Tandis que pour l'algorithme proposé avec la représentation B, nous traçons les points $(\hat{pd}\ ,\ \hat{pf})$ en fonction de la valeur de la constante $k$.

Globalement, nous avons constaté que la représentation B donne de meilleurs résultats que la représentation A. Ceci est vrai quelque soit la nature de bruit et le type d'image traités. Nous présentons à titre indicatif dans la figure 2.16, les courbes opérationnelles pour un bruit gaussien ($\sigma = 20, 5\%$) injecté dans l'image Maison. D'après cette figure, nous remarquons que l'algorithme avec la représentation B est plus performant que l'algorithme utilisant la représentation A. Cette observation est justifiée par la localisation de la courbe opérationnelle $\hat{pd} = f(\hat{pf})$ de l'algorithme avec la représentation B.

| bruit alpha-stable | Maison | | Poivron | | Bateau | |
|---|---|---|---|---|---|---|
| (%, $\alpha$,$\gamma$) | $\hat{pd}$ | $\hat{pf}$ | $\hat{pd}$ | $\hat{pf}$ | $\hat{pd}$ | $\hat{pf}$ |
| $(5, 0.8, 1)$ | 0.7749 | 0.1671 | 0.7221 | 0.112 | 0.6911 | 0.1573 |

TAB. *2.12: Evaluation de la détection de bruit alpha-stable injecté dans les images naturelles Maison, Poivron et Bateau.*

| bruit impulsionnel | Maison | | Poivron | | Bateau | |
|---|---|---|---|---|---|---|
| % | $\hat{pd}$ | $\hat{pf}$ | $\hat{pd}$ | $\hat{pf}$ | $\hat{pd}$ | $\hat{pf}$ |
| 5 | 0.9708 | 0.0511 | 0.9717 | 0.0808 | 0.9200 | 0.0967 |

TAB. *2.13: Evaluation de la détection de bruit impulsionnel injecté dans les images naturelles Maison, Poivron et Bateau.*

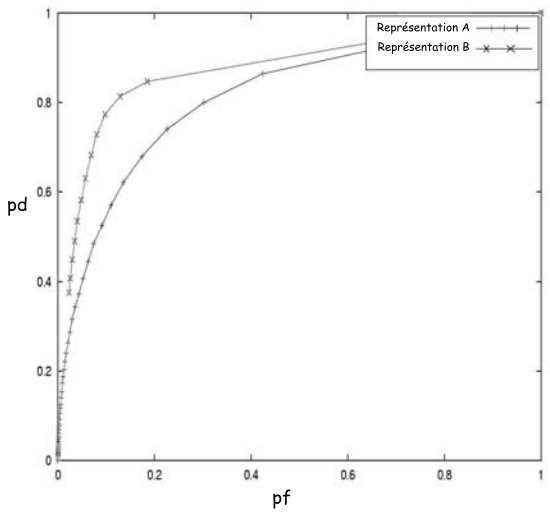



FIG. *2.16: Comparaison de la suppression de bruit gaussien additif ($\sigma = 20$, 5%) injecté dans l'image Maison par l'algorithme utilisant les représentations A et B.*

**(b). Evaluation de l'estimation des hyperarêtes de bruit**

Ce paragraphe présente les résultats de l'estimation des hyperarêtes de bruit effectuée sur l'image de synthèse corrompue par les bruits gaussien, alpha-stable et impulsionnel. Afin d'évaluer cette étape de l'algorithme, nous utilisons le PSNR. En examinant les résultats de la table 2.14, nous pouvons remarquer tout d'abord que les résultats de l'estimation corroborent les résultats de la détection. En effet, les étapes de détection et d'estimation sont toujours meilleures pour l'image de synthèse dans le cas de bruit impulsionnel. Le gain en PSNR pour l'image de synthèse corrompue par un bruit gaussien est de l'ordre de 14dB, 30.39dB dans le cas de bruit impulsionnel et de 27dB dans le cas du bruit alpha-stable. Cette supériorité est due aux valeurs faibles de la probabilité de fausse alarme dans le cas de la suppression de bruit impulsionnel. L'étape de l'estimation réagit de la même façon que l'étape de la détection dans le cas des simulations de tests sur différents types d'images (Maison, Bateau et Poivron). L'image Bateau reste toujours l'image qui pose le plus de difficulté à traiter. Les figures 2.17 et 2.18 permettent de comparer visuellement les performances de l'algorithme en terme de suppression de bruit pour l'image Poivron (bruit gaussien, bruit impulsionnel).

| image de synthèse | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| gaussien additif ($\sigma$,%) (20,5%) | | | alpha-stable (%,$\alpha$,$\gamma$) (5%,0.8,1) | | | impulsionnel (%) 5% | | |
| $\mathrm{PSNR}_{Ib}$ | $\mathrm{PSNR}_{If}$ | $\Delta$PSNR | $\mathrm{PSNR}_{Ib}$ | $\mathrm{PSNR}_{If}$ | $\Delta$PSNR | $\mathrm{PSNR}_{Ib}$ | $\mathrm{PSNR}_{If}$ | $\Delta$PSNR |
| 35.74 | 49.77 | 14.03 | 30.19 | 58.14 | 27.95 | 17.67 | 47.96 | 30.29 |

TAB. *2.14: Evaluation de l'étape de l'estimation des hyperarêtes de bruit.*

(a) (b)

FIG. *2.17: Suppression du bruit gaussien ($\sigma = 20$,5%) par l'algorithme proposé (représentation B). (a) l'image bruitée. (b) l'image estimée.*

(a) (b)

FIG. *2.18: Suppression du bruit impulsionnel ($\sigma = 20{,}5\%$) par l'algorithme proposé (représentation B). (a) l'image bruitée. (b) l'image estimée.*

#### 2.5.2.4 Evaluation de la représentation C

De la même façon que précédemment, l'algorithme utilisant les fonctions de similarité pour générer les hyperarêtes est testé sur un ensemble d'images à niveaux de gris de taille $256 \times 256$. Les images sont bruitées avec un pourcentage $p$. Nous avons utilisé un ensemble de fonctions de similarité satisfaisant les conditions citées dans l'équation 1.25 [SPC+02] :

$$\begin{aligned}
&\mu_1(x) = e^{-\gamma_1 x}, \gamma_1 \in (0;\infty) \\
&\mu_2(x) = \tfrac{2}{1+e^{\gamma_2 x}}, \gamma_2 \in (0;\infty) \\
&\mu_3(x) = \begin{cases} 1-\gamma_3 x & \text{si } x < 1/\gamma_3, \\ 0 & \text{si } x \geq 1/\gamma_3, \end{cases} \quad \gamma_3 \in (0;\infty).
\end{aligned}$$

Une mesure de similarité peut être obtenue par la conversion d'une mesure de dissimilarité en utilisant une transformation $T$. La fonction $\mu_3$ est obtenue par une transformation de la distance utilisée pour la génération de la représentation A si l'on considère $x = |I(x) - I(y)|$. La transformation est définie par : $\mu_3 = 1 - \gamma_3 d$. Le coefficient $\gamma_3$ est un facteur de normalisation.

L'objectif du premier test de comparaison est d'évaluer l'algorithme de suppression de bruit employant la représentation C d'abord par rapport aux résultats de l'algorithme cité dans la section 2.5.2.2, puis par rapport à ceux de l'algorithme mentionné dans le paragraphe 2.5.2.3 en utilisant les probabilités de détection et de fausse alarme.

Durant la suite des comparaisons avec les représentations A et B, nous utilisons les constantes des fonctions de similarité suivantes proposées par Smolka : $\gamma_1 = 5,04\ 10^{-3}, \gamma_2 = 9.90\ 10^{-3},\ et\ \gamma_3 = 3.92\ 10^{-3}$. Ces valeurs correspondent aux valeurs optimales obtenues expérimentalement au sens de la minimisation de l'erreur quadratique moyenne.

##### (a). Comparaison avec l'algorithme employant la représentation A

La comparaison entre l'algorithme à seuil fixe $\alpha$ et l'algorithme utilisant les fonctions de similarité $\mu(x)$ se base sur une comparaison de l'étape de la détection des hyperarêtes.

La figure 2.19 illustre un exemple de comparaison entre l'algorithme à seuil fixe et l'algorithme utilisant la fonction de similarité $\mu_1(x)$. Le bruit utilisé dans la comparaison est le bruit gaussien. L'image de simulation est l'image Bateau. D'après cette figure, on constate que la courbe opérationnelle de l'algorithme (représentation C) est située au dessus de celle qui correspond à l'algorithme avec un seuil $\alpha$ fixe. Ceci illustre la supériorité de la représentation C par rapport à A.

##### (b). Comparaison avec l'algorithme employant la représentation B

Après la démonstration de la supériorité de l'algorithme utilisant les fonctions de similarité par rapport à l'algorithme à seuil fixe, nous passons à la deuxième comparaison, cette fois avec l'algorithme à seuil dynamique. L'image traitée est Poivron, corrompue par un bruit impulsionnel de 5%.

Avant cette comparaison, illustrons d'abord l'influence des fonctions de similarité $\mu(x)$ et les normes $L_1$ et $L_2$ sur les performances de l'algorithme proposé (représentation C). Dans la table 2.15, nous présentons les probabilités de détection et de fausses alarmes dans le cas de l'utilisation des fonctions $\mu_1$, $\mu_2$ et $\mu_3$ avec les deux normes $L_1$ et $L_2$. Au vu des résultats, deux constatations s'imposent :

- A une norme donnée $L_1$ ou $L_2$, les trois fonctions de similarité ont sensiblement les mêmes résultats de détection de bruit que ce soit pour la probabilité de détection ou de fausse alarme.

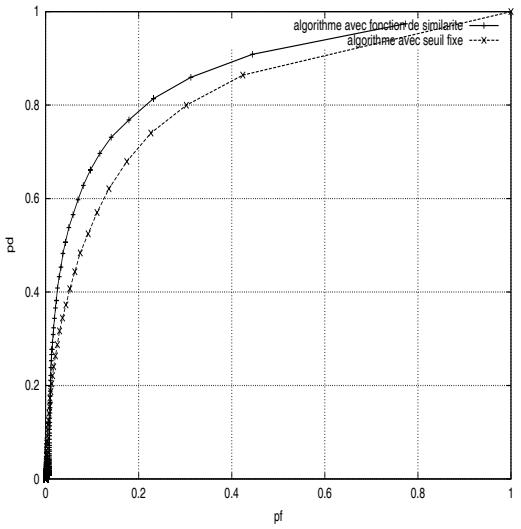



FIG. *2.19: Comparaison des courbes opérationnelles de l'algorithme à seuil fixe et l'algorithme utilisant une fonction de similarité $\mu_1(x)$. L'image traitée est Bateau corrompue par un bruit gaussien additif d'écart-type $\sigma = 30$ et de pourcentage 5%.*

Par exemple, en employant la norme $L_1$, la valeur de $\hat{pd}$ est presque égale à 0.86 pour les trois fonctions, tandis que la valeur de $pf$ est de 0.009.

- Si nous observons uniquement la probabilité de détection $\hat{pd}$, nous constatons que la norme $L_2$ est plus adaptée que $L_1$. Mais l'inconvénient est la valeur élevée aussi de la probabilité de fausse alarme $\hat{pf}$. Il est difficile de faire un choix entre ces deux distances car la première $L_1$ détecte moins de bruit et préserve l'information utile, tandis que la deuxième $L_2$ détecte presque tous les pixels de bruit, mais avec une probabilité de fausse alarme élevée. Pour cette raison, notre choix s'est porté sur la norme $L_1$. Premièrement, les valeurs de $\hat{pd}$ sont satisfaisantes et deuxièmement l'objectif de telles méthodes de suppression de bruit est principalement la conservation de l'information.

| - | Norme $L_1$ | | Norme $L_2$ | |
|---|---|---|---|---|
| - | $\hat{pd}$ | $\hat{pf}$ | $\hat{pd}$ | $\hat{pf}$ |
| $\mu_1(x)$ | 0.8711 | 0.0095 | 0.936 | 0.11 |
| $\mu_2(x)$ | 0.8788 | 0.0099 | 0.90 | 0.025 |
| $\mu_3(x)$ | 0.8588 | 0.0085 | 0.936 | 0.09 |

TAB. *2.15: Illustration des résultats de détection de bruit en utilisant les probabilités de détection et de fausse alarme en fonction des normes $L_1$ et $L_2$ et des fonctions de similarités $\mu(x)$.*

La table 2.16 illustre une comparaison de l'algorithme utilisant les fonctions de similarité et les normes $L_1$ et $L_2$ avec l'algorithme à seuil dynamique. L'image traitée est Poivron corrompue par le bruit impulsionnel avec un pourcentage 5%. Nous voyons bien dans ce tableau que l'utilisation des fonctions de similarité avec un seuil $\alpha = 0.85$ diminue la probabilité de fausse alarme, ce qui implique une "bonne" conservation des contours et une augmentation de la probabilité de décision signifiant une "bonne" détection des pixels bruités. La détection dans ce cas est supérieure aux résultats donnés par l'algorithme à seuil dynamique. Ces constations ne restent plus valables

lorsqu'on change $\alpha$. En effet, pour une valeur $\alpha$ de 0.50, nous constatons que la détection se dégrade que ce soit en probabilité de détection ou en probabilité de fausse alarme.

| – | $\hat{pd}$ | $\hat{pf}$ |
|---|---|---|
| Algo. ($\alpha$ seuil dynamique) | 0.872 | 0.07 |
| Algo. ($\alpha = 0.85$), la norme $L_1$ | | |
| $\mu_1(x)$ | 0.8711 | 0.0095 |
| $\mu_2(x)$ | 0.8788 | 0.0099 |
| $\mu_3(x)$ | 0.8588 | 0.0085 |
| Algo. ($\alpha = 0.50$), la norme $L_1$ | | |
| $\mu_1(x)$ | 0.36 | 0.107 |
| $\mu_2(x)$ | 0.50 | 0.117 |
| $\mu_3(x)$ | 0.42 | 0.098 |

TAB. *2.16: Illustration de l'avantage de fonctions de similarité pour la suppression du bruit impulsionnel. L'image bruitée est l'image Poivron avec un pourcentage de 5%.*

### 2.5.3 Résultats de comparaison avec les algorithmes de la littérature

Dans le but de comparer les performances de la méthode proposée avec les autres approches de suppression de bruit, nous présentons une série d'expérimentations et de comparaisons. Nous comparons les résultats de l'approche proposée avec ceux obtenus par : les algorithmes standards tels que le filtre médian, le filtre de Wiener, et le filtre médian multiétage, et les algorithmes de Stevenson [SS92] et Tovar [TER+00]. La comparaison est basée sur le PSNR, les probabilités de décision et de fausse alarme et le critère visuel. On ne teste à ce niveau que les représentations B et C.

#### 2.5.3.1 Comparaison avec les stratégies : Détection / Estimation

Dans cette section, nous comparons l'algorithme proposé avec les algorithmes de Stevenson et Tovar pour les deux étapes de détection et d'estimation.

Pour l'algorithme de Stevenson, nous comparons seulement la stratégie de détection dans le cas du bruit impulsionnel. En effet, comme il est indiqué dans la section 2.4.2.2, l'algorithme de Stevenson calcule la différence entre l'image bruitée et l'image filtrée par un filtre donné. Dans notre simulation, il s'agit du filtre médian multiétage. Si cette différence est supérieure à un seuil $D$, le pixel est détecté bruit sinon le pixel est non bruité.

**Comparaison de la détection.** Le tableau 2.17 illustre une comparaison de l'algorithme de détection utilisant les fonctions de similarité $\mu(x)$ avec les algorithmes de Stevenson et Tovar pour 5% de bruit impulsionnel ajouté à l'image Poivron. Pour l'algorithme proposé, le seuil spatial $\beta$ est égal à 1 et le seuil $\alpha$ a pour valeur 0.85. Pour les constantes des fonctions de similarité (Eq. 2.5.2.4), nous utilisons les valeurs $\gamma_i 10^{-3}\{\gamma_1 = 5,04, \gamma_2 = 9.90, \gamma_3 = 3.92\}$.

Il est clair que l'algorithme de Tovar détecte mieux le bruit, si on regarde que la valeur de $\hat{pd} = 0.96$. Mais l'inconvénient est la valeur élevée aussi de $\hat{pf} = 0.55$. Ceci va se traduire par un lissage de l'information utile de l'image. La valeur de la probabilité de détection de l'algorithme proposé avec la norme $L_1$ est identique à la valeur donnée par l'algorithme de Stevenson du $3^{\text{ème}}$ ordre. L'amélioration apportée dans ce cas par l'algorithme proposé est due à la probabilité de fausse alarme. Une faible probabilité $\hat{pf}$ signifie une conservation de contours.

Dans le cas de l'utilisation de la norme $L_2$, l'algorithme proposé donne plus de performance en terme de détection $pd \hat{=} 0.93$, et moins en terme de conservation de l'information $\hat{pf}$. Néanmoins, les performances restent satisfaisantes. Si nous utilisons la norme $L_1$, nous détectons mieux le bruit que l'algorithme de Stevenson parce que nous avons presque la même valeur de $\hat{pd}$, mais des $\hat{pf}$ différents. Si nous utilisons la norme $L_2$, la détection du bruit est également meilleure qu'avec l'algorithme de Stevenson car dans ce cas, nous avons une valeur de $\hat{pd}$ élevée. Les probabilités de fausse alarme sont sensiblement les mêmes. En terme de détection de bruit, il semble que l'algorithme proposé avec la représentation C est plus performant que les algorithmes de Stevenson et Tovar.

| Méthode | $\hat{pd}$ | $\hat{pf}$ |
|---|---|---|
| STV[$1^{rd}$] | 0.73 | 0.01 |
| STV[$2^{rd}$] | 0.85 | 0.03 |
| STV[$3^{rd}$] | 0.86 | 0.042 |
| TOV | 0.96 | 0.55 |
| Algo. ($\alpha = 0.85$), la norme $L_1$ | | |
| $\mu_1(x)$ | 0.87 | 0.0095 |
| $\mu_2(x)$ | 0.87 | 0.0099 |
| $\mu_3(x)$ | 0.85 | 0.0085 |
| Algo. ($\alpha = 0.85$), la norme $L_2$ | | |
| $\mu_1(x)$ | 0.936 | 0.11 |
| $\mu_2(x)$ | 0.90 | 0.025 |
| $\mu_3(x)$ | 0.936 | 0.09 |

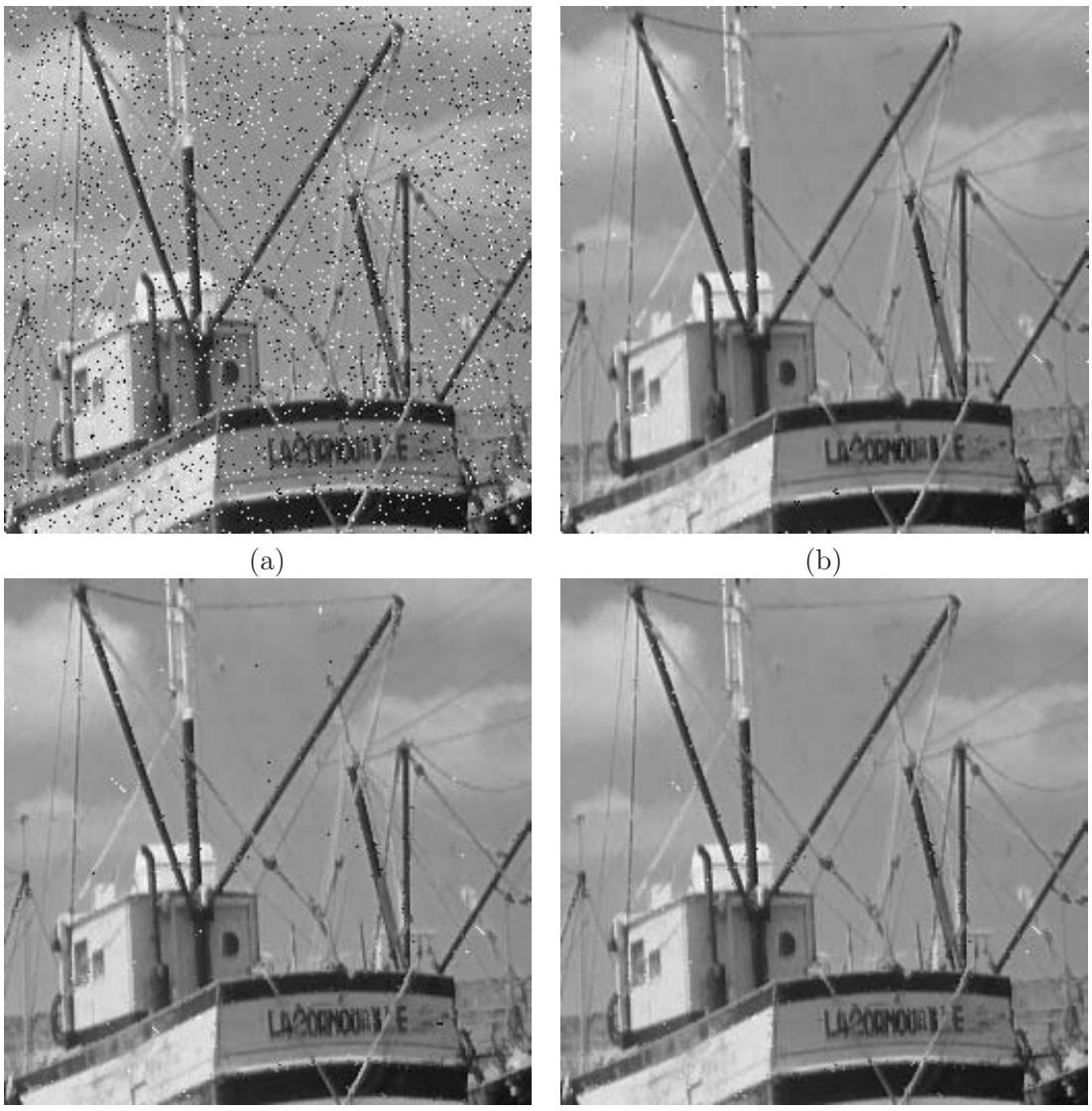
TAB. *2.17: Comparaison des performances en détection pour un bruit impulsionnel (5%) injecté dans l'image Poivron par l'algorithme proposé (représentation C) et les algorithmes de Stevenson et Tovar. Les paramètres de l'algorithme de Tovar sont ceux définis dans l'équation 2.14. Le seuil D de l'algorithme de Stevenson est égal à* 15.

**Comparaison de l'estimation** Les résultats de comparaison avec l'algorithme proposé à seuil $\alpha$ dynamique et les algorithmes de Tovar et Stevenson sont donnés dans le tableau 2.18. Les hyperarêtes de bruit après détection par l'algorithme proposé sont estimées par une valeur médiane sur leur voisinage ouvert. Si l'on considère les résultats des traitements obtenus lorsque les paramètres ont été bien ajustés afin d'obtenir le meilleur PSNR possible, nous remarquons une nette différence : d'où la supériorité du débruitage par hypergraphe de voisinage à seuil dynamique.

#### 2.5.3.2 Comparaison avec les stratégies : Estimation

Nous avons mené une série de comparaison en utilisant le critère visuel. En effet, nous avons comparé l'algorithme (représentation C) avec le filtre médian standard et le filtre médian multiétage MLMF. Les figures 2.21, 2.20 illustrent cette comparaison. D'après ces figures, nous constatons que l'approche proposée préserve mieux les contours par rapport au filtre médian et au filtre MLMF.

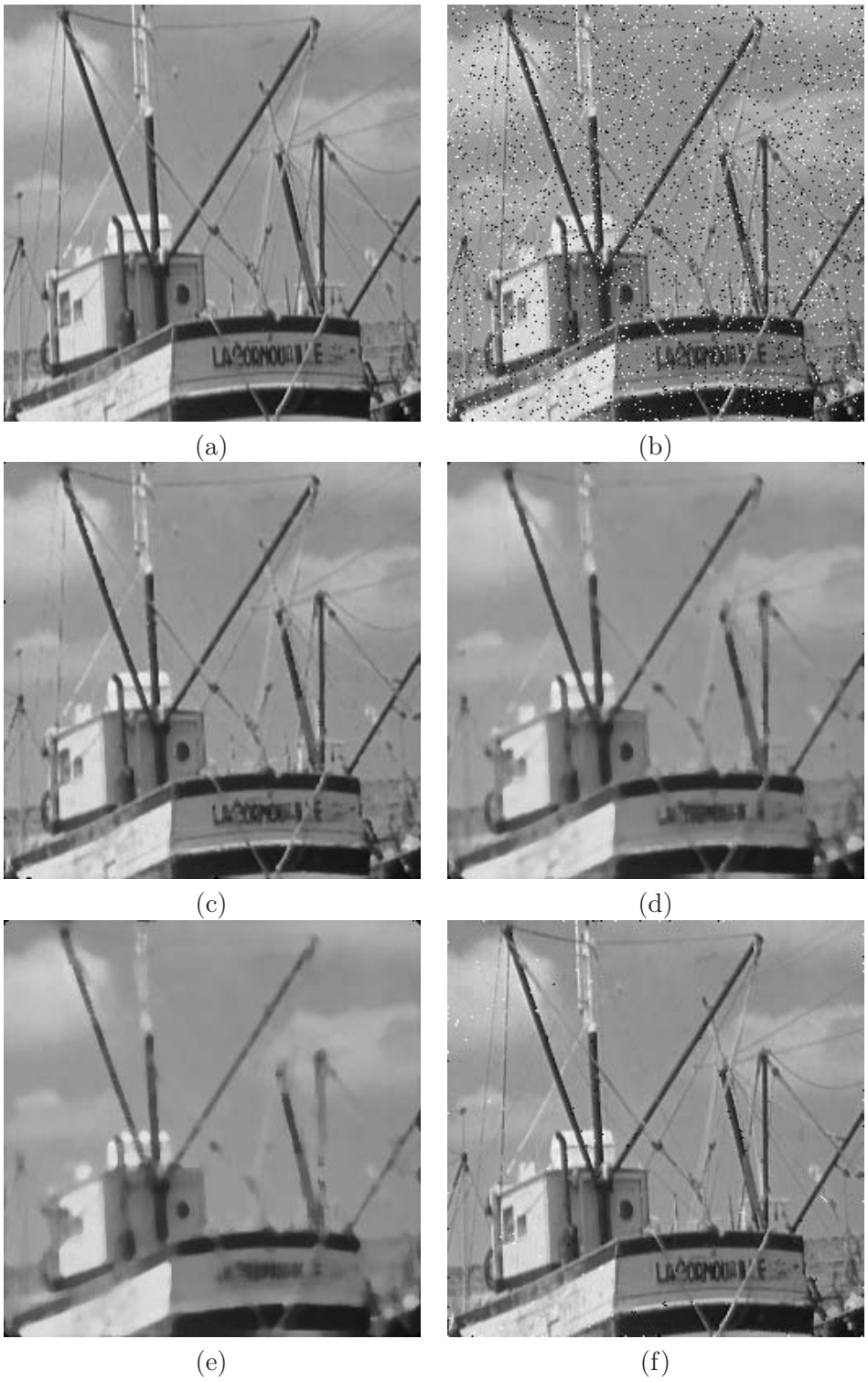
En plus de la suppression du bruit impulsionnel, l'approche proposée permet aussi une amélioration de la qualité de l'image lorsqu'il s'agit d'une perturbation de type bruit gaussien. Le tableau 2.19 résume les différentes simulations de comparaison visuelle entre l'algorithme proposé avec un seuil dynamique $\alpha = k\sigma$ et le filtre de Wiener. Les hyperarêtes de bruit sont détectées



(a) (b)

(c) (d)

FIG. *2.20: Comparaison avec le filtre médian multiétage. (a) image bateau (bruit impulsionnel 5 %), (b) Image de sortie de l'algorithme (représentation B), Images filtrées par MLMF avec une taille de fenêtre de : (c)* $5 \times 5$*, (d)* $7 \times 7$*.*



(a) (b) (c) (d) (e) (f)

FIG. *2.21: Comparaison avec le filtre médian. (a) l'image originale, (b), (c) et (d) représentent respectivement les images filtrées avec une fenêtre de* $[3 \times 3]$, $[5 \times 5]$ *et* $[7 \times 7]$. *(f) l'image filtrée par l'algorithme proposé (représentation B).*

| Image | Alg. | % injecté | PSNR de $I_b$ (dB) | PSNR de $I_f$ (dB) |
|---|---|---|---|---|
| Poivron | algo. (représentation B) | 5 | 18.08 | 33.38 |
| – | algo. (représentation B) | 10 | 15.06 | 27.29 |
| – | STV[$1^{rd}$] | 5 | 18.08 | 23.45 |
| – | STV[$2^{rd}$] | 5 | 18.08 | 25.08 |
| – | STV[$3^{rd}$] | 5 | 18.08 | 25.95 |
| – | STV[$1^{rd}$] | 10 | 15.06 | 19.09 |
| – | STV[$2^{rd}$] | 10 | 15.06 | 22.86 |
| – | STV[$3^{rd}$] | 10 | 15.06 | 24.24 |
| – | algo. de Tovar | 5 | 18.08 | 25.88 |
| – | – | 10 | 15.06 | 25.55 |

$I_b$ : Image bruitée,
$I_f$ : Image de sortie,
$1^{rd}$, $2^{rd}$ et $3^{rd}$ représentent respectivement une fenêtre de taille $3 \times 3$, $5 \times 5$ et $7 \times 7$.

TAB. *2.18: Comparaison des résultats de suppression du bruit impulsionnel par le biais du rapport PSNR entre l'algorithme proposé et les algorithmes de Stevenson et Tovar. Les paramètres de l'algorithme de Tovar sont ceux définis dans l'équation 2.14. Le seuil D de l'algorithme de Stevenson est égal à* 15.

par le modèle 2 et elles sont estimées par le filtre Wiener sur leurs voisinages ouverts.

| bruit | écart-type | pourcentage % | résultats |
|---|---|---|---|
| gaussien additif | 15 | 5 | Fig. 2.22 |
| | 20 | 5 | Fig. 2.23 |

TAB. *2.19: Résumé des différentes figures des résultats de comparaison avec le filtre Wiener.*

Pour une bonne préservation de contours, il est clair que le recours à des filtres globaux, qui s'appliquent à l'image entière, n'est pas la meilleure solution. Les zones intactes des images sont souvent de haute définition et il serait souhaitable de ne pas filtrer ces régions. Il est à noter d'après les figures 2.20, 2.21, 2.22 et 2.23 que les filtres globaux médian et de Wiener lissent les contours.

(a) (b)

(c) (d)

FIG. *2.22: Comparaison avec le filtre de Wiener (bruit gaussien additif $\sigma = 15$, 5%). (a) l'image originale, (b) l'image bruitée, (c) l'image de sortie du filtre de Wiener. (d) l'image filtrée par l'algorithme (représentation B).*

(a) (b)

(c) (d)

FIG. *2.23: Comparaison avec le filtre de Wiener (bruit gaussien additif $\sigma = 20$, 5%). (a) l'image originale, (b) l'image bruitée, (c) image de sortie du filtre de Wiener. (d) l'image filtrée par l'algorithme (représentation B).*

## 2.6 Suppression de bruit dans les images couleurs en utilisant la représentation HVSC

De manière similaire à son "ancêtre" analogique, l'imagerie numérique s'est tout d'abord focalisée sur les images à niveaux de gris (monochromes, ou monocomposantes). Même s'il reste encore beaucoup à faire dans de nombreux domaines, les techniques correspondantes sont aujourd'hui relativement bien définies et maîtrisées. Ainsi, grâce aux avancées réalisées en imagerie monochrome, mais également grâce aux progrès des moyens techniques disponibles, les recherches se sont progressivement orientées vers des images plus complexes, de dimensions plus élevées, et ce, afin de répondre à des applications de plus en plus complexes et exigeantes. Il semble intéressant de pouvoir disposer, pour ces données multi-dimensionnelles, des mêmes outils que pour les images à niveaux de gris. C'est dans ce cadre que s'inscrit l'objectif de cette section. Nous nous sommes en effet intéressés aux différentes approches possibles permettant d'effectuer des traitements bas-niveaux sur des images multicomposantes, particulièrement des images couleur, si possible en tirant profit des travaux déjà effectués et validés en niveaux de gris.

### 2.6.1 Problème des "fausses couleurs"

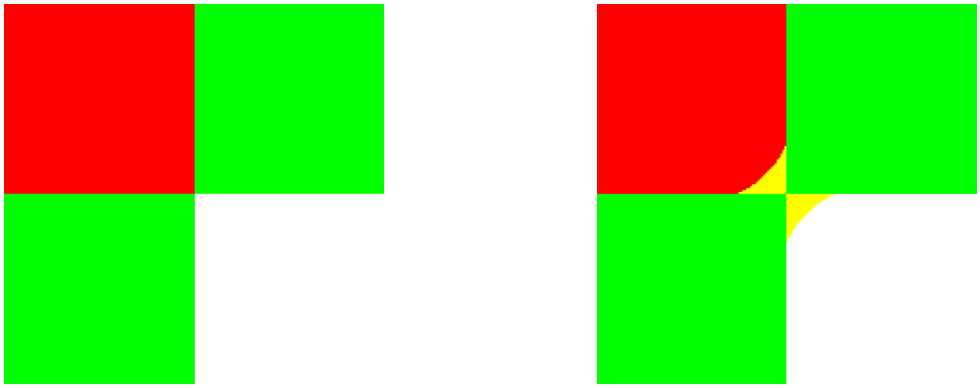
En imagerie monochrome, l'estimation scalaire crée souvent de nouveaux niveaux de gris qui "homogénéisent" l'image. Ce phénomène est très fréquent avec le filtre moyenneur qui peut introduire des niveaux de gris intermédiaires qui ne modifient pas notablement la perception visuelle de l'image. Dans le cas des images multicomposantes, cette situation peut prendre un caractère plus gênant. Les estimateurs peuvent en effet générer de nouveaux vecteurs qui, du point de vue de la perception visuelle de l'image, n'ont aucune ressemblance avec les vecteurs d'entrée de l'estimateur, ne provoquant donc pas l'effet d'homogénéisation recherché. Cette situation particulière que nous appellerons par la suite le problème des "*fausses couleurs*", apparaît de manière récurrente dans les stratégies de filtrage évoquées dans la littérature.

Traitons par exemple le cas du filtre médian (mais le même problème se retrouve avec la plupart des opérateurs). Pour une fenêtre de filtrage $W$ de $N$ vecteurs $X_j : W = X_j, j = 1, ...N$, chaque vecteur $X_j$ ayant $M$ composantes $X_j(i)$ , la sortie du filtre médian marginal est le vecteur $X_{med}$ dont les $M$ composantes $X_{med}(i)$ sont respectivement les valeurs médianes des M ensembles $\{X_j(i), j = 1, \dots N\}$ :

$$\begin{aligned} & X_{med} = med\{X_j, j = 1, \dots N\} \\ \iff & X_{med}(i) = med\{X_j(i), j = 1, \dots, N\}, \forall i \in 1, \dots, M \end{aligned}$$

Or, il n'y a aucune raison pour que $X_{med}$ soit l'un des vecteurs d'entrée : $X_{med} \notin W$. On assiste donc à l'apparition de nouveaux vecteurs en sortie du filtre, ce qui est gênant dans de nombreuses applications. Par exemple, dans le cas d'images couleur, l'apparition de nouvelles couleurs peut détériorer la qualité visuelle d'une image ; elle peut également perturber un système de reconnaissance qui serait basé sur les attributs chromatiques des objets. De même, en télédétection, cela peut être néfaste pour l'interprétation et la segmentation d'images multicomposantes par classification de pixels. Cette classification (i.e. l'affectation de chaque pixel à une classe : forêt, marécage, culture etc.) s'effectue en étudiant la signature vectorielle de chaque pixel (i.e. l'intensité de ses émissions dans différentes bandes spectrales). L'apparition de nouveaux vecteurs lors du filtrage ou de la simplification des données pourrait nuire à l'interprétation des images. D'une manière générale, dans toute la suite, on qualifiera de "fausse couleur" toute apparition, lors d'un traitement, d'un vecteur qui ne faisait pas partie de la fenêtre de filtrage initiale (le terme de "couleur" sera abusivement conservé quel que soit le type de données multicomposantes traité).

Ce phénomène est décrit par un exemple sur la figure suivante. La figure 2.24 présente une image couleur synthétique originale (-a-) et l'image obtenue après filtrage médian marginal de taille 51( Cette taille a été choisie volontairement grande pour faciliter la visualisation du phénomène. Il faut cependant souligner que même de petites tailles de filtre auraient présenté cet inconvénient). Nous constatons une déformation des contours, typique du filtrage médian, au niveau de l'angle. Mais on observe également l'apparition de la couleur jaune qui était absente de l'image originale. [Cha97] donne une description numérique de cet exemple.

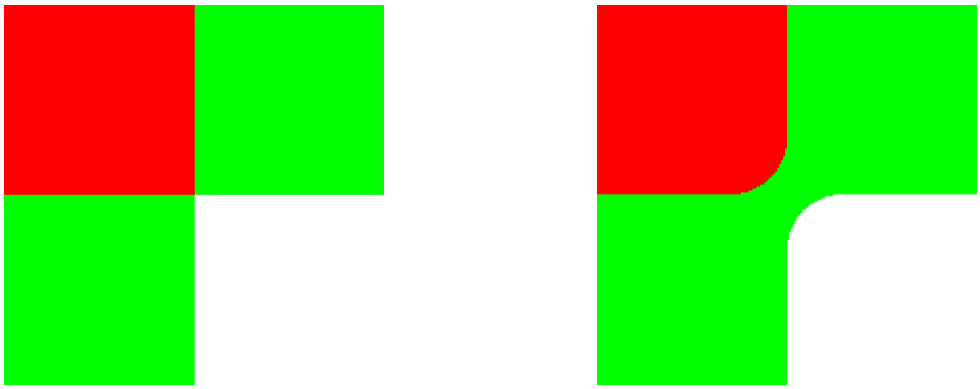


(a) (b)

FIG. *2.24: Filtre médian marginal (b) d'une image couleur (a)*

L'approche vectorielle permet d'imposer à la sortie du filtre de faire partie de la fenêtre de filtrage initiale : ainsi l'apparition de nouveaux vecteurs lors du traitement est évitée.

La figure 2.25 reprend l'exemple de la figure 2.24 : la même image couleur originale (-a-) est maintenant filtrée avec l'approche vectorielle standard du filtre médian [AHN90]. Nous constatons qu'il n'y a plus de fausses couleurs sur l'image filtrée (-b-). Nous conservons bien sûr la déformation des angles due au filtrage médian, mais nous notons que la couleur de sortie est maintenant systématiquement l'une des couleurs d'entrée (par exemple le vert au lieu du rouge quand il est le plus représentatif de la fenêtre de filtrage).



(a) (b)

FIG. *2.25: Filtre médian vectoriel standard (b) d'une image couleur (a).*

Bien sûr, pour ce type d'approche, il faut rendre le filtre que l'on désire utiliser, compatible avec la structure désormais vectorielle des pixels : il faut réaliser une extension vectorielle du

filtre scalaire, et ce, tout en préservant ses propriétés. C'est avec cet objectif que la présente étude a été menée. Toutefois, cette extension ne peut pas toujours être réalisée directement.

### 2.6.2 Techniques de suppression de bruit dans des images couleurs

La famille des filtres non linéaires est très vaste [PV90]. Elle regroupe des opérateurs très divers, chacun correspond à un domaine privilégié d'application. On peut classifier ces techniques selon qu'elles utilisent ou non une notion d'ordre. Pour la plupart des classes utilisées d'opérateurs non linéaires, la notion d'ordre joue un rôle capital. L'ordre croissant des valeurs d'un ensemble est le fondement des opérateurs d'estimation dits de rang (filtre d'ordre). Les valeurs ordonnées des échantillons statistiques d'ordre de la fenêtre de l'estimation sont nommées statistiques d'ordre. Dans le cas des images scalaires, la relation d'ordre ne pose aucun problème ; l'extension des filtres de rang aux images couleur (ou, en général, vectorielles) n'est pas simple.

#### 2.6.2.1 Méthodes utilisant l'ordre vectoriel

Les premiers travaux portant sur le problème du tri de données vectorielles sont présentés dans [Barnett-76]. Quatre méthodes de tri y sont définies : le tri marginal, le tri conditionnel, le tri partiel, et le tri réduit. De nombreux travaux ([HA91] [PT91] [NP96] [KP97] etc.) ont repris ou adopté ces différentes approches et les ont appliquées au filtrage d'image. Notons cependant que la plupart des filtres obtenus reposent sur des préordres totaux.

**Le tri marginal.** L'ordre marginal [Bar76] est, selon son nom, un ordre qui se fait selon les échantillons marginaux des vecteurs considérés. Cela revient à ordonner de manière indépendante les valeurs des échantillons de chaque composante selon un modèle de la pile de composantes que nous pouvons traiter séparément. Ce traitement indépendant des composantes ne prend pas en compte la corrélation qui existe entre les composantes. Parmi les opérateurs utilisant un tri marginal, nous trouvons l'approche de filtrage médian marginal à décorrélation.

L'approche par décorrélation pour le filtrage médian des images couleur a été introduite dans [VMZB96] et propose la réalisation du filtrage des composantes après leur décorrélation. La décorrélation est réalisée par la transformée de Karhunen-Loève (KL). L'avantage de cette approche est l'indépendance vis-à-vis du système de représentation utilisé pour la représentation des couleurs.

**Le tri conditionnel.** L'ordre conditionnel est une manière d'établir des rangs (ou un ordre) [Bar76], [PV90] pour les vecteurs d'un ensemble, conditionnés par l'ordre d'une de leurs composantes. La composante à traiter doit être la plus affectée par le bruit. La solution est de choisir adaptativement la composante pour chaque pixel de l'image ; la composante la plus active (au sens d'une mesure de non-uniformité importante -intervalle de variation, intervalle de variation normé à la moyenne, variance, rapport de contraste, valeurs propres). Cette approche a été introduite dans [VMZB96] comme TMMF -Truncated Marginal Median Filter.

**Le tri partiel.** Pour le tri partiel, les données sont regroupées en sous-ensembles pour former des bulles convexes minimales. La première bulle est une hypersurface convexe qui regroupe un minimum de points. La deuxième bulle est formée de la même façon en prenant les données restantes, et ainsi de suite. Les données sont ensuite classées en comparant le numéro de leur "couche" d'appartenance. Contrairement aux tris marginal et conditionnel ou au tri scalaire des données en imagerie monocomposante, les "plus petits" vecteurs (au sens du tri partiel) sont ici les vecteurs les plus typiques : ils sont situés au cœur du nuage des données dont ils

fournissent la meilleure estimation. La figure 2.26 illustre cette méthode pour un nuage de points bidimensionnel.

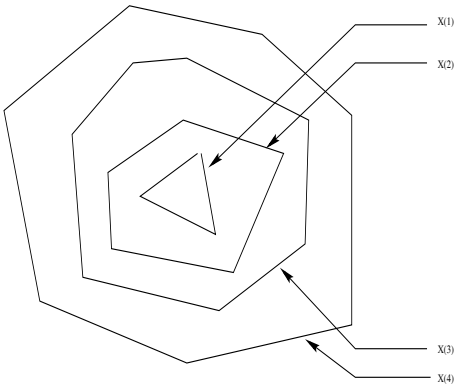



FIG. *2.26: Tri partiel et bulles convexes.*

**Le tri réduit.** L'ordre réduit est fondé sur la réduction de chaque observation vectorielle à une valeur unique par une combinaison de ses composantes. Les scalaires ainsi obtenus sont ordonnés ; cet ordre définit celui des vecteurs.

- Ordre réduit par distances. Le plus souvent, les scalaires $s_i$ associés aux vecteurs $X_i$ sont déduits par une distance généralisée à un point fixe spécifié $p$, en utilisant la forme quadratique $s_i = (X_i - p)^T A^{-1}(X_i - p)$. La matrice carrée de dimension $p \times p$, $A$, peut être n'importe quelle matrice semi-définie positive. En général, on préfère utiliser une matrice unitaire qui génère la distance euclidienne, mais on peut utiliser aussi la matrice de covariance des observations (qui génère la distance Mahalanobis) ou une matrice diagonale (qui génère une distance euclidienne pondérée). Les points fixes sont en général des vecteurs obtenus par des traitements marginaux (moyenne, médian).

  Une autre variante pour construire le scalaire à base des distances est de sommer les distances entre le vecteur courant et tous les autres vecteurs de l'ensemble à ordonner (distance par agrégation [Bar76]).

  $$s_i = \sum_{j=1}^{n} (x_j - x_i)^T A^{-1}(x_j - x_i). \tag{2.28}$$

  Selon [AHN90], [PV90] le médian d'un ensemble de données est caractérisé par une distance par agrégation minimale (dans le cas scalaire ou vectoriel), $X_{VMF} = argmin_{i=1,n}\{s_i\}$. Le filtre qui utilise ce principe est le VMF - Vector Median Filter [AHN90] ; son introduction a constitué le commencement des "vrais" traitements en multicomposantes (qui utilisent de manière intrinsèque la corrélation entre les composantes de l'image).

- Ordre réduit par direction (distances angulaires). Les critères utilisés pour l'ordre réduit ont employé seulement l'aspect "module" des vecteurs ; l'aspect "orientation" (angle) est resté inaperçu jusqu'au moment de l'introduction des traitements directionnels. Ce type de filtres introduit dans [TV93] utilise comme scalaire $s_i$ la distance angulaire par agrégation (somme des angles entre un vecteur et tous les autres). Tout comme pour le VMF [AHN90], le filtre directionnel de base BVDF -Basic Vector Directional Filter produit à la sortie le vecteur dont

la distance angulaire par agrégation est minimale. Pour ce filtre, le scalaire est donc :

$$s_i = \sum_{j=1}^{n} \widehat{x_i x_j} = \sum_{j=1}^{n} arccos\left(\frac{< x_i, x_j >}{||x_i||||x_j||}\right) \tag{2.29}$$

- Ordre réduit par distances et direction. Les comportements relativement complémentaires des deux types essentiels de filtrages vectoriels (ordre réduit selon les distances et selon la direction) en présence de différents types de bruit ont suggéré de combiner ces deux traitements dans une approche mixte. La première étape a été introduite dans [TKV96] par un traitement séquentiel : sélection selon la direction des vecteurs suivie par un traitement par distances.

#### 2.6.2.2 Méthodes indépendantes de l'ordre

Si $X_j$ est l'ensemble des $n$ vecteurs de la fenêtre courante de filtrage, la sortie du filtre pour la position donnée est la combinaison linéaire pondérée de ces valeurs.

$$y = \sum_{j=1}^{Card(W)} \omega_j X_j, \qquad X_j \in W. \tag{2.30}$$

Les méthodes d'estimation non linéaire indépendantes de l'ordre, sont obtenues par adaptation des coefficients de pondération $\omega_i$.

- Méthodes dépendant de la distance euclidienne entre vecteurs. En général, le coefficient de pondération est une fonction qui dépend d'un scalaire $d_j$, de type scalaire d'ordonnancement $(s_j)$. Les fonctions polynomiales ont été appliquées avec succès par de nombreux chercheurs, [BP94],[EF93],[FE95],[PAV96] ;
- Méthodes dépendant de la distance angulaire entre vecteurs. Exemple, le FVDF - Fuzzy Vector Directional Filter [PAV95], dont chaque coefficient de pondération des vecteurs est donné par une formule qui associe les approches polynomiale et exponentielle.

$$a_j = \frac{1}{1 + \exp(d_j^r)} \tag{2.31}$$

Pour des résultats optimaux, le paramètre de contrôle $r$ a les valeurs 1 ou 2. Comme dans le cas de l'ordre des vecteurs, on peut considérer des approches mixtes distance - angle (l'intégration dans un seul scalaire $d_j^r$ - par un produit - des distances agrégées angulaires et euclidiennes).

- Méthodes à voisinage adaptatif. La construction de la fenêtre de filtrage selon la spécificité locale du point à traiter (et, implicitement, le calcul d'une fenêtre de filtrage pour chaque pixel de l'image) a été nommée filtrage à voisinage adaptatif [RCF98], [CRZB98]. L'essentiel de cette méthode est le calcul, pour chaque pixel, d'une zone relative uniforme, connexe, par une technique de croissance de régions [Jai89] (usuellement utilisée pour la segmentation orientée région des images). L'avantage de cette méthode est double : d'une part, on élimine les valeurs aberrantes (pixels corrompus par le bruit impulsionnel), d'autre part, le filtre de lissage ainsi réalisé, préserve extrêmement bien le contraste des contours de l'image (parce que les régions uniformes ne contiennent pas de pixels situés d'une part et de l'autre des frontières). Le filtrage lui-même est une moyenne [CRZB98] ou une moyenne pondérée par des coefficients adaptatifs [RCF98]. Le problème principal de cette approche est la complexité du calcul.

- Méthodes intégrant la logique floue dans l'adaptation. Un ensemble flou est une fonction qui associe à chaque élément de l'univers un nombre positif sous-unitaire. La modélisation floue et l'utilisation de la logique floue ne se réduisent pas à l'utilisation des pondérations de l'intervalle [0 ; 1] pour chaque univers. Il existe quand-même des cas, où des filtres adaptatifs ont été nommés flous seulement en raison de cette caractéristique de leurs coefficients (par exemple FVDF -Fuzzy Vector Directional/ Filter [PAV95]). Une méthodologie fondée sur la logique floue est développée dans [PAV97] : le calcul de nouveaux filtres par la combinaison de plusieurs filtres (concept qui provient, naturellement, de l'intégration des traitements directionnels et orientés amplitude pour les images couleur).

  Une telle approche utilise plusieurs critères (non-optimaux) pour l'évaluation des poids des vecteurs de la fenêtre de filtrage dans le vecteur de sortie du filtre et l'agrégation non linéaire de ces poids par un principe flou (un agrégateur flou). Dans [VMB97], une extension du filtre médian scalaire flou introduit par [YT95] et nommé AFMMF -Adaptive Fuzzy Multilevel Median Filter est proposée. Une autre approche du filtrage médian flou par des méthodes de partitionnement flou a été proposée dans [VMB97] comme extension du filtre médian flou scalaire [DR96].

#### 2.6.2.3 Approches morphologiques

L'approche suggérée par la définition des filtres médians vectoriels est d'utiliser les relations d'ordre partiel, et, en conséquence, de relâcher le cadre théorique des définitions des opérations morphologiques de base. L'ordre marginal des vecteurs [Bar76] revient à traiter indépendamment chaque plan de couleur (composante) de l'image. L'ordre réduit a été suggéré dans [VBP96] et [BV95], conduisant à des transformations semblables aux transformations morphologiques (morphologic-like), par l'utilisation des distances aux points fixes. Dans ce cas, les définitions modifiées des opérations de dilatation et d'érosion sont : le résultat de la dilatation d'un ensemble de vecteurs est le vecteur le plus approché d'un point extrémal de maximum (le maximum marginal de l'ensemble de vecteurs ou le maximum absolu de l'espace des vecteurs) ; le résultat de l'érosion d'un ensemble de vecteurs est le vecteur le plus approché d'un point extrémal de minimum (le minimum marginal de l'ensemble de vecteurs ou le minimum absolu de l'espace des vecteurs).

### 2.6.3 Suppression du bruit dans une image couleur en utilisant l'hypergraphe de voisinage spatiocolorimétrique

#### 2.6.3.1 Présentation

L'algorithme est basé sur les propriétés de la géométrie et la topologie de la représentation par Hypergraphe de Voisinage SpatioColorimétrique HVSC. Cette application représente une extension de l'algorithme de débruitage appliqué à l'image à niveaux de gris, cité dans la section 2.5. L'algorithme est basé sur le principe suivant :

1. Représentation de l'image couleur par HVSC ;
2. Classification binaire des hyperarêtes de l'image ($H_0$ hyperarête de bruit et $H_1$ hyperarête de donnée non bruitée) ;
3. Estimation des zones bruitées.

Dans le chapitre 1, nous avons évoqué les différentes approches d'extension de la représentation d'image couleur par hypergraphe de voisinage. En effet, l'algorithme développé dans cette section, peut utiliser soit une approche marginale, qui consiste à appliquer le même algorithme

de débruitage de la section 2.5 sur chaque composante de l'image couleur, soit, nous pouvons construire un algorithme vectoriel qui traite les trois composantes couleurs en même temps.

Le modèle des hyperarêtes de bruit est indépendant de la stratégie de traitement d'image couleur. Ces modèles 1 et 2 restent valables pour l'image à niveaux de gris et l'image couleur. Pour modéliser le bruit dans une image couleur, nous utilisons le modèle 2. Ce modèle met en évidence les hyperarêtes qui sont incohérentes avec leur environnement. Il permet de traduire l'inhomogénéité de l'image couleur et donc de détecter le bruit.

#### 2.6.3.2 Algorithme

Pour chaque pixel $x$ de l'image $I$, nous procédons de l'étape 1 à l'étape 5 :

1. Données : image $I$, le seuil colorimétrique $\alpha$ , l'ordre de voisinage $\beta$ ;
2. Construction de la représentation HVSC : $H_{\alpha,\beta}$ de l'image $I$ selon une stratégie marginale ou vectorielle ;
3. Détermination des hyperarêtes isolées ;
4. Détection des hyperarêtes de bruit $E^b_{\alpha,\beta}(x)$ ;
5. Estimation des hyperarêtes de bruit.

Notons que $E^b_{\alpha,\beta}(x)$ est une hyperarête vérifiant le modèle 2. L'hyperarête $E_{\alpha,\beta}(x)$ centrée sur le pixel $x$ est localisée comme bruit, si $E_{\alpha,\beta}(x) \in E^b_{\alpha,\beta}(x)$.

#### 2.6.3.3 Suppression de bruit en utilisant une stratégie marginale

L'extension de la représentation par hypergraphe de voisinage aux images couleurs selon une stratégie marginale conduit à l'apparition de trois hypergraphe scalaires : $H^{C1}_{\alpha,\beta}$, $H^{C2}_{\alpha,\beta}$ et $H^{C3}_{\alpha,\beta}$. Les hypergraphes de voisinage scalaires de la représentation HVSC-M de l'image couleur se génèrent de la même façon que l'hypergraphe de voisinage d'image à niveaux de gris. En effet, la génération d'hypergraphe scalaire peut se faire selon les trois représentations : représentation à seuil fixe, représentation à seuil dynamique ou représentation utilisant les fonctions de similarité comme distance marginale.

La figure 2.27 illustre le principe de fonctionnement de cette approche marginale de suppression de bruit pour une image couleur dans l'espace RVB : d'abord une décomposition de l'image couleur en image monocomposante R, V et B ; ensuite une génération de la représentation par hypergraphe de voisinage scalaire, suivie d'une étape de détection et estimation des hyperarêtes de bruit. L'image couleur résultante est la composition des composantes traitées ($R_{\text{estimée}}$, $V_{\text{estimée}}$ et $B_{\text{estimée}}$.

#### 2.6.3.4 Suppression de bruit en utilisant une stratégie vectorielle

L'extension de l'hypergraphe de voisinage aux images couleur par approche vectorielle conduit à l'apparition d'un seul hypergraphe vectoriel : $H_{\alpha,\beta}$.

L'hypergraphe de voisinage vectoriel dépend de trois paramètres : la distance sur la grille $\beta$, le seuil spatiocolorimétrique $\alpha$ et la distance colorimétrique entre les vecteurs couleur $d'(I(x), I(y))$. Le seuil $\alpha$ peut être global, local ou dynamique.

L'algorithme de suppression de bruit vectoriel se déroule en deux étapes : (1) d'abord une détection des hyperarêtes de bruit selon le modèle 2, ensuite une estimation vectorielle de bruit. La figure 2.28 représente le bloc diagramme de l'algorithme proposé.

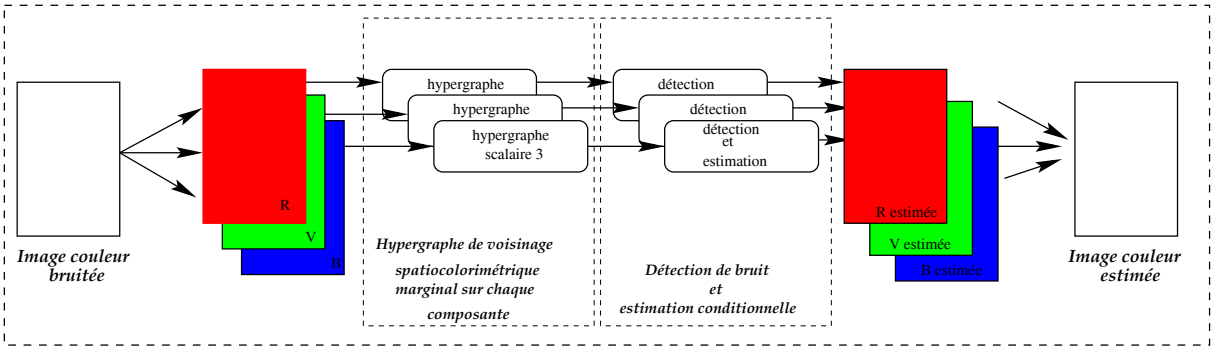



FIG. *2.27: Bloc Diagramme de l'algorithme de débruitage par approche marginale.*

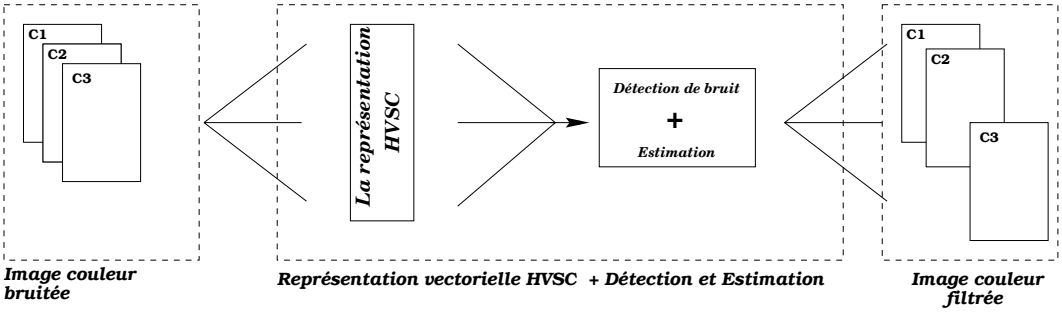



FIG. *2.28: Suppression de bruit par approche vectorielle en utilisant la représentation HVSC-V.*

### 2.6.4 Résultats Expérimentaux

Dans cette section, nous présentons un ensemble d'expériences afin d'évaluer l'algorithme proposé. Nous limitons nos simulations aux traitements des images couleur corrompues par un bruit impulsionnel. Notre but, dans les premières expériences, est de comparer les deux approches marginale et vectorielle dans l'espace RVB, puis de comparer les espaces couleurs RVB et CIELab dans l'approche vectorielle.

Le deuxième objectif de simulation est d'étudier les performances de l'algorithme après estimation des hyperarêtes de bruit par rapport aux filtres non linéaires appliqués à la suppression du bruit impulsionnel comme le filtre VMF proposé par Astola [AHN90] et le filtre BVDF [PV00].

Nous construisons deux types de représentation, marginale en utilisant la représentation HVSC-M, et vectorielle utilisant HVSC-V. La représentation HVSC-M est appliquée dans l'espace couleur RVB. De la même manière, la représentation HVSC-V est appliquée dans deux espaces RVB et CIELab. L'approche vectorielle et tout particulièrement la représentation HVSC utilise une distance colorimétrique de type euclidien pour générer les hyperarêtes. Elle est définie respectivement dans les espaces de couleurs RVB et CIELab par l'équation 2.32 et 2.33.

$$\Delta d_{RVB} = \sqrt{(\Delta R)^2 + (\Delta V)^2 + (\Delta B)^2} \tag{2.32}$$

$$\Delta E^*_{Lab} = \sqrt{(\Delta L^*)^2 + (\Delta a^*)^2 + (\Delta b^*)^2} \tag{2.33}$$

Dans ces simulations, nous utilisons un seuil $\alpha$ global pour générer les deux représentations HVSC-M et HVSC-V. Le seuil choisi dans nos simulations dépend de deux paramètres $D_m$ et

$D_\delta$ définis par [YJH+99] :

$$\alpha \quad = \quad D_m + D_\delta \tag{2.34}$$

où

$$\begin{array}{lcl} D_m & = & \sum_{i=1}^{M} \sum_{j=1}^{N} \frac{d_{i,j}}{M \times N} \\ D_\delta & = & \sum_{i=1}^{M} \sum_{j=1}^{N} \frac{|d_{i,j} - D_m|}{M \times N} \\ avec & : & \\ d_{i,j} & = & \frac{1}{8} \sum_{x=i-1}^{i+1} \sum_{y=j-1}^{j+1} \Delta I_{x,y}, \quad sachant \quad que \quad \Delta I_{x,y} = |I_{x,y} - I_{i,j}| \end{array} \tag{2.35}$$

où $(M, N)$ représente le nombre de lignes et colonnes de l'image. $d_{i,j}$ est la différence moyenne entre le pixel et ses voisins (voisinage 8-adjacents). Dans le cas de seuillage de l'image couleur par une approche vectorielle, la valeur $\Delta I_{x,y}$ est une distance entre deux vecteurs. Tandis que dans le cas de l'approche marginale, la valeur est une distance entre deux scalaires.

Contrairement à l'approche vectorielle, la représentation HVSC-M a besoin pour chaque composante d'un seuil $\alpha$ différent calculé de la même façon que pour l'approche vectorielle en utilisant l'équation 2.34. Pour une image couleur donnée, on calcule le seuil $\alpha = (\alpha_{C1}, \alpha_{C2}, \alpha_{C3})$.

#### 2.6.4.1 Evaluation de la suppression de bruit

Après une brève description de nos simulations, nous expliquons maintenant les différents critères de comparaison utilisés dans les simulations. Commençons d'abord par la façon dont nous bruitons les images naturelles. En effet, pour une image couleur donnée, chaque composante est bruitée avec un bruit impulsionnel de pourcentage $p$.

**Critères d'évaluation.** Pour l'évaluation de la détection et de l'estimation de bruit, nous utilisons trois critères :

- les probabilités de détection et de fausse alarme pour évaluer l'étape de la détection (cf. § 2.5.2.1) ;
- le rapport PSNR pour évaluer l'étape de l'estimation. Ce dernier est défini par :

$$PSNR = 10 \log_{10} \frac{3 \times M \times N \times 255^2}{\sum_{i=1}^{M} \sum_{j=1}^{N} ||I_{i,j} - \hat{I}_{i,j}||_{L_2}^2} \tag{2.36}$$

  où $I$ et $\hat{I}$ représentent respectivement les images couleur originales et estimées.
- La différence de couleur normalisée NCD (Normalized Color Difference) qui correspond à une mesure d'erreur perceptuelle dans un espace uniforme. Le NCD est calculé entre deux images dans l'espace couleur uniforme $CIELab$ [EF95]. Elle est définie par :

$$NCD = \frac{\sum_{i=1}^{M} \sum_{j=1}^{N} ||\Delta E^*_{Lab}||}{\sum_{i=1}^{M} \sum_{j=1}^{N} ||E^*_{Lab}||} \tag{2.37}$$

  avec $\Delta E^*_{Lab} = \sqrt{(\Delta L^*)^2 + (\Delta a^*)^2 + (\Delta b^*)^2}$ et $E^*_{Lab} = \sqrt{(L^*)^2 + (a^*)^2 + (b^*)^2}$.

**(a). Evaluation de la détection des hyperarêtes bruit**

**Comparaison des approches marginale et vectorielle.** Nos simulations débutent avec une comparaison de détection des hyperarêtes de bruit entre les approches marginale et vectorielle dans l'espace RVB en utilisant les courbes opérationnelles ($\hat{pd} = f(\hat{pf})$). En effet, pour tracer

ces courbes, nous calculons pour chaque seuil $\alpha$ compris dans l'intervalle $[0, 255]$ les probabilités de détection et de fausse alarme. Les images Poivron et Tiffany sont corrompues par un bruit impulsionnel de 2% sur chaque composante de l'image couleur. Les résultats de cette comparaison sont illustrés dans la figure 2.29. L'approche marginale est présentée par la courbe (a), tandis que l'approche vectorielle est présentée par la courbe (b). L'approche qui détecte le mieux les pixels de bruit devrait avoir une probabilité de détection $\hat{pd}$ aussi élevée que possible, et une probabilité de fausse alarme aussi faible que possible, c'est à dire une courbe opérationnelle qui cintre loin de la ligne diagonale autant que possible. En examinant ces deux courbes, on remarque une supériorité de l'approche vectorielle dans l'espace de couleur RVB par rapport à l'approche marginale dans le même espace. Ceci est justifié par la prise en compte de la corrélation entre les composantes couleurs par l'approche vectorielle.

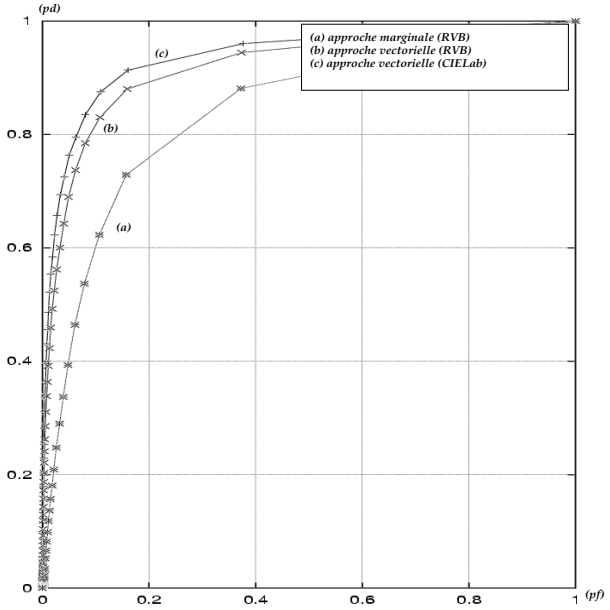



FIG. *2.29: Les courbes opérationnelles de détection de bruit par l'algorithme proposé en utilisant les représentations HVSC-V dans l'espace couleur CIELab (courbe (c)), HVSC-V dans l'espace couleur RVB (courbe (b)) et HVSC-M dans l'espace couleur RVB (courbe (a)).*

**Comparaison de l'espace couleur pour l'approche vectorielle** Dans la même figure 2.29, nous avons tracé la courbe opérationnelle ($\hat{pd} = f(\hat{pf})$) correspondante à l'approche vectorielle dans l'espace couleur CIELab. Cette approche est présentée par la courbe (c). De cette figure, nous pouvons remarquer que l'algorithme proposé utilisant la représentation HVSC vectorielle dans l'espace CIELab donne des résultats significatifs. En effet, sa courbe opérationnelle est située au-dessus de l'algorithme utilisant les représentations vectorielles HVSC dans l'espace RVB. Ces résultats démontrent l'amélioration présentée en terme de détection, en utilisant une distance euclidienne $\Delta E^*_{ab}$ pour la génération de la représentation HVSC-V et à la supériorité de l'espace de CIELab en conséquence.

Au vu de ces deux comparaisons, deux constatations s'imposent :

- L'approche vectorielle est plus performante que l'approche marginale. Cette dernière ignore totalement la dépendance pouvant exister entre les différentes composantes, délaissant ainsi une information pouvant participer à l'amélioration des performances de l'algorithme.

- L'approche vectorielle dans l'espace couleur CIELab est meilleure que la même approche dans l'espace couleur RVB. Cette supériorité est justifiée par l'inadéquation de la distance euclidienne de l'approche vectorielle dans l'espace RVB.

### (b). Evaluation de l'estimation des hyperarêtes de bruit

Après une comparaison des approches vectorielle et marginale et une comparaison des espaces couleur en terme de détection de bruit, nous évaluons maintenant dans la suite les performances de la méthode proposée après avoir estimé les hyperarêtes de bruit. Les hyperarêtes de bruit détectées par l'approche marginale sont estimées par le filtre médian, tandis que les hyperarêtes de bruit détectées par l'approche vectorielle sont estimées par le filtre VMF.

De la même manière, nous présentons dans la première étape, une comparaison des approches marginale et vectorielle, puis une comparaison des espaces couleur. La table 2.20 illustre les résultats de comparaison entre les approches marginale et vectorielle et les espaces RVB et CIELab.

Le rapport PSNR est utilisé pour donner une évaluation quantitative des effets d'estimation. En plus du PSNR, nous utilisons la mesure NCD (Normalized Color Difference) pour mesurer l'erreur perceptuelle dans l'espace CIELab. La méthode la plus performante en terme d'estimation doit avoir une valeur élevée de PSNR et une valeur faible de NCD.

Ces résultats sont calculés avec un seuil global de l'image selon l'équation 2.34 (Tab. 2.21). En examinant ce tableau, nous remarquons que l'approche vectorielle dans l'espace CIELab est meilleure que les autres représentations.

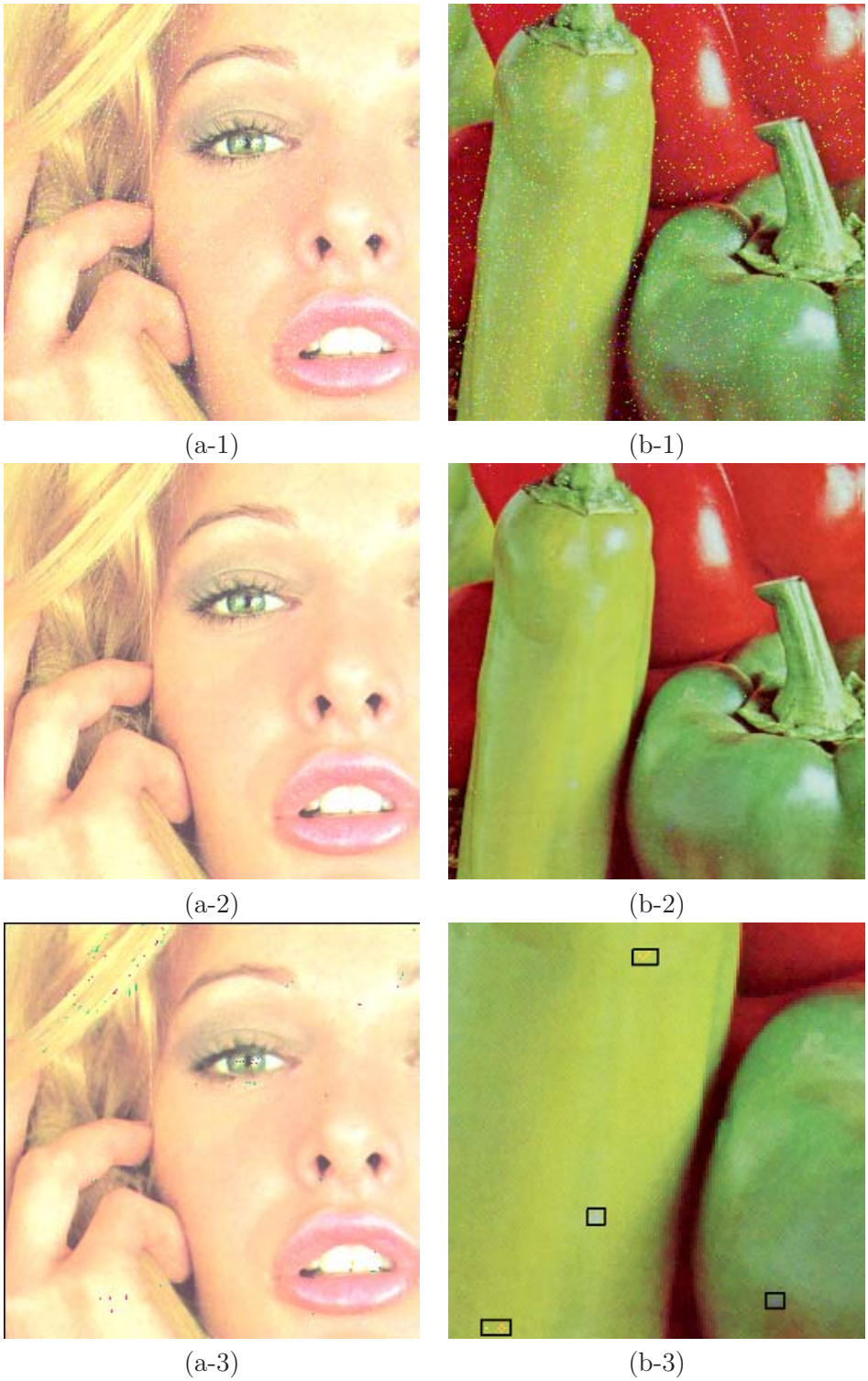
| | RVB | | | | CIELab | |
|---|---|---|---|---|---|---|
| Stratégie de traitement | Marginale | | Vectorielle | | Vectorielle | |
| Image couleur | PSNR | NCD | PSNR | NCD | PSNR | NCD |
| Tiffany | 28.62 | 0.0061 | 31.63 | 0.0033 | 32.12 | 0.0008 |
| Poivron | 24.29 | 0.0174 | 26.49 | 0.0058 | 31.80 | 0.0023 |

TAB. *2.20: Comparaison des approches marginale et vectorielle dans les espaces couleur CIELab et RVB.*

| | RVB | | CIELab |
|---|---|---|---|
| | Marginale | Vectorielle | Vectorielle |
| Tiffany | (8,13,11) | 26 | 15 |
| Poivron | (18,22,27) | 49 | 27 |

TAB. *2.21: Les seuils colorimétriques $\alpha$ globaux calculés selon l'équation 2.34.*

Les résultats de comparaison de l'estimation entre les deux applications, marginale et vectorielle dans les espaces couleur CIELab et RVB sont montrés dans la figure 2.30. Au vu de ces résultats, on peut conclure que la représentation HVSC utilisée par l'algorithme dans l'espace de CIELab est meilleure que la méthode marginale HVSC-M (RVB). Les résultats de l'algorithme utilisant HVSC-M (RVB) sont de mauvaise qualité. Ils contiennent de fausses couleurs, puisque la corrélation entre les composantes est ignorée.



(a-1) (b-1)

(a-2) (b-2)

(a-3) (b-3)

FIG. 2.30: *Résultats de l'estimation de bruit impulsionnel injecté dans les images Tiffany et Poivron. (a-1, b-1) Image bruitée par 2% de bruit impulsionnel. Images (a-2,b-2), (a-3,b-3) sont les sorties de l'algorithme proposé utilisant respectivement les représentations HVSC-V (CIELab) et HVSC-M (RVB).*

### (c). Comparaison avec les filtres VMF et BVDF

Dans cette section, nous présentons une comparaison de l'algorithme proposé dans l'espace couleur CIELab par rapport aux filtres VMF (Vector Median Filter) et BVDF (Based Vector Directional Filter). Dans ce cas, la comparaison est limitée à l'étape de l'estimation, puisque les deux filtres entrent dans le cadre des stratégies d'estimation. Après détection des hyperarêtes de bruit, nous les remplaçons par la valeur médiane vectorielle calculée sur le voisinage ouvert de $E^b_{\alpha,\beta}(x)$.

Dans la figure 2.31, nous présentons les courbes de PSNR et NCD en fonction du pourcentage de bruit impulsionnel injecté dans l'image originale, pour l'approche marginale à base de la représentation HVSC-M et l'approche vectorielle utilisant HVSC-V dans les espaces couleur RVB et CIELab ainsi que les deux filtres VMF et BVDF.

Les hyperarêtes de bruit sont estimées par le filtre médian pour l'approche marginale, et le filtre VMF pour l'approche vectorielle. Le résultat de la figure 2.31 montre que les images estimées par l'algorithme vectoriel dans l'espace CIELab ont une valeur de PSNR très élevée et une valeur de NCD faible, ce qui implique une supériorité de l'extension vectorielle de l'algorithme proposé dans l'espace CIELab. Par exemple, pour l'image Poivron corrompue par 2% de bruit impulsionnel, la sortie de l'algorithme proposé a une valeur PSNR de 31.80dB et une valeur NCD de 0.0023, tandis que le filtre VMF a pour valeur de PSNR 22.65 dB et NCD de 0.0162. Les valeurs élevées de PSNR et NCD de l'algorithme proposé dans l'espace couleur CIELab expliquent l'intérêt de l'estimation conditionnelle en utilisant les modèles de bruit, ainsi que la distance euclidienne dans l'espace uniforme CIELab. Néanmoins, l'algorithme proposé présente une incapacité notoire lorsque le pourcentage de bruit augmente.

| Image "TIFFANY" Taille = 256 × 256 | | | | | | |
|---|---|---|---|---|---|---|
| % injecté | 1% | 2% | 3% | 4% | 5% | 10% |
| HVSC-V (CIELab) | 11 | 15 | 17 | 19 | 22 | 31 |
| HVSC-V (RVB) | 22 | 26 | 30 | 34 | 37 | 40 |
| HVSC-M (RVB) | 8,13,11 | 8,14,15 | 8,16,18 | 9,17,21 | 9,18,23 | 10,24,35 |
| Image "POIVRON" Taille = 256 × 256 | | | | | | |
| HVSC-V (CIELab) | 19 | 27 | 33 | 39 | 46 | 67 |
| HVSC-V (RVB) | 36 | 49 | 59 | 69 | 79 | 116 |
| HVSC-M (RVB) | 14,17,19 | 18,22,27 | 22,26,33 | 25,30,39 | 29,34,44 | 41,50,69 |

TAB. *2.22: Les seuils colorimétriques $\alpha$ calculés selon l'équation 2.34.*

Nous présentons dans les figures 2.32, 2.33 et 2.34 les résultats de l'estimation de l'algorithme proposé, le filtre VMF et le filtre BVDF appliqués sur les images bruitées Tiffany, Poivron et Fruit.

Les figures 2.32(b), 2.33(b) et 2.34(b) montrent les images Poivron, Tiffany et Fruit bruitées par 2% de bruit impulsionnel. Les images estimées par l'algorithme vectoriel dans l'espace couleur CIELab, le filtre VMF et le filtre BVDF sont montrées dans les figures 2.32(a-1,a-2,a-3), 2.33(a-1,a-2,a-3) et 2.34(a-1,a-2,a-3) respectivement. La différence absolue entre l'image originale et l'image estimée pour les trois algorithmes est illustrée dans les figures 2.32(b-1,b-2,b-3), 2.33(b-1,b-2,b-3) et 2.34(b-1,b-2,b-3).

En examinant ces figures, on constate que l'algorithme employant la représentation vectorielle préserve mieux l'information utile après estimation.

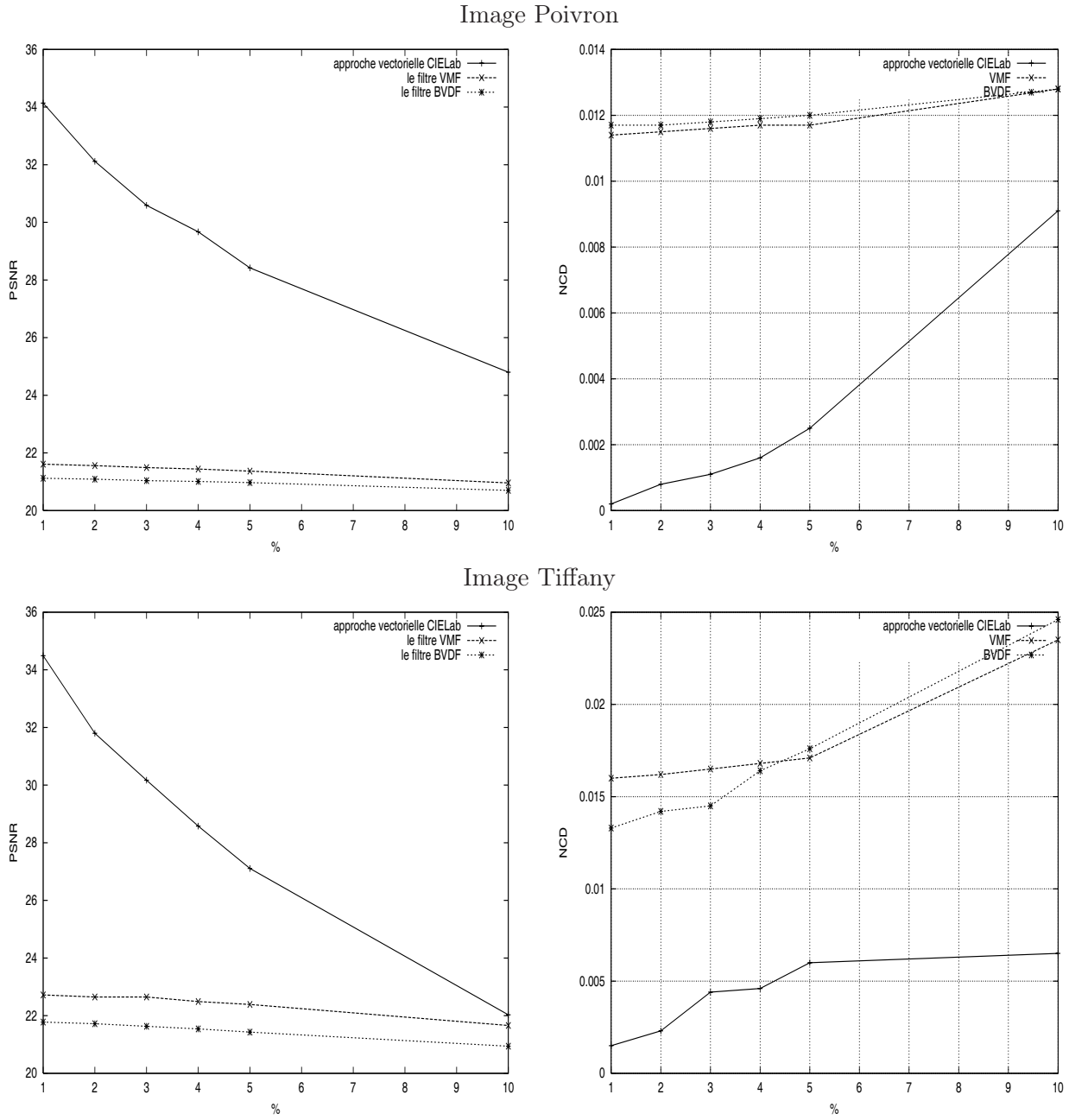
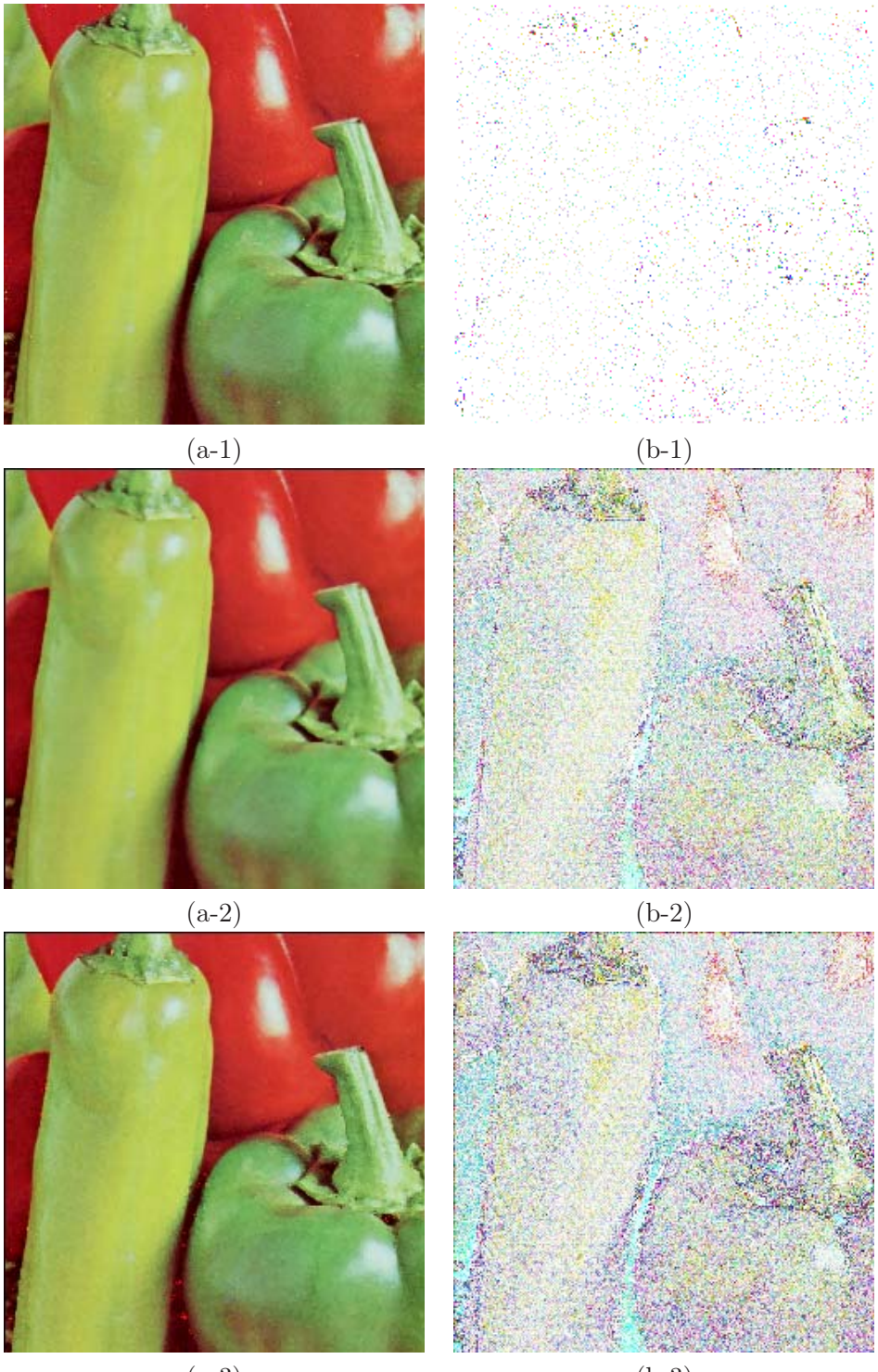



FIG. *2.31: Evolutions de PSNR et NCD en fonction du pourcentage de bruit impulsionnel injecté pour l'approche vectorielle dans l'espace couleur CIELab et les filtres VMF et BVDF. Les seuils colorimétriques utilisés sont montrés dans la table 2.22.*

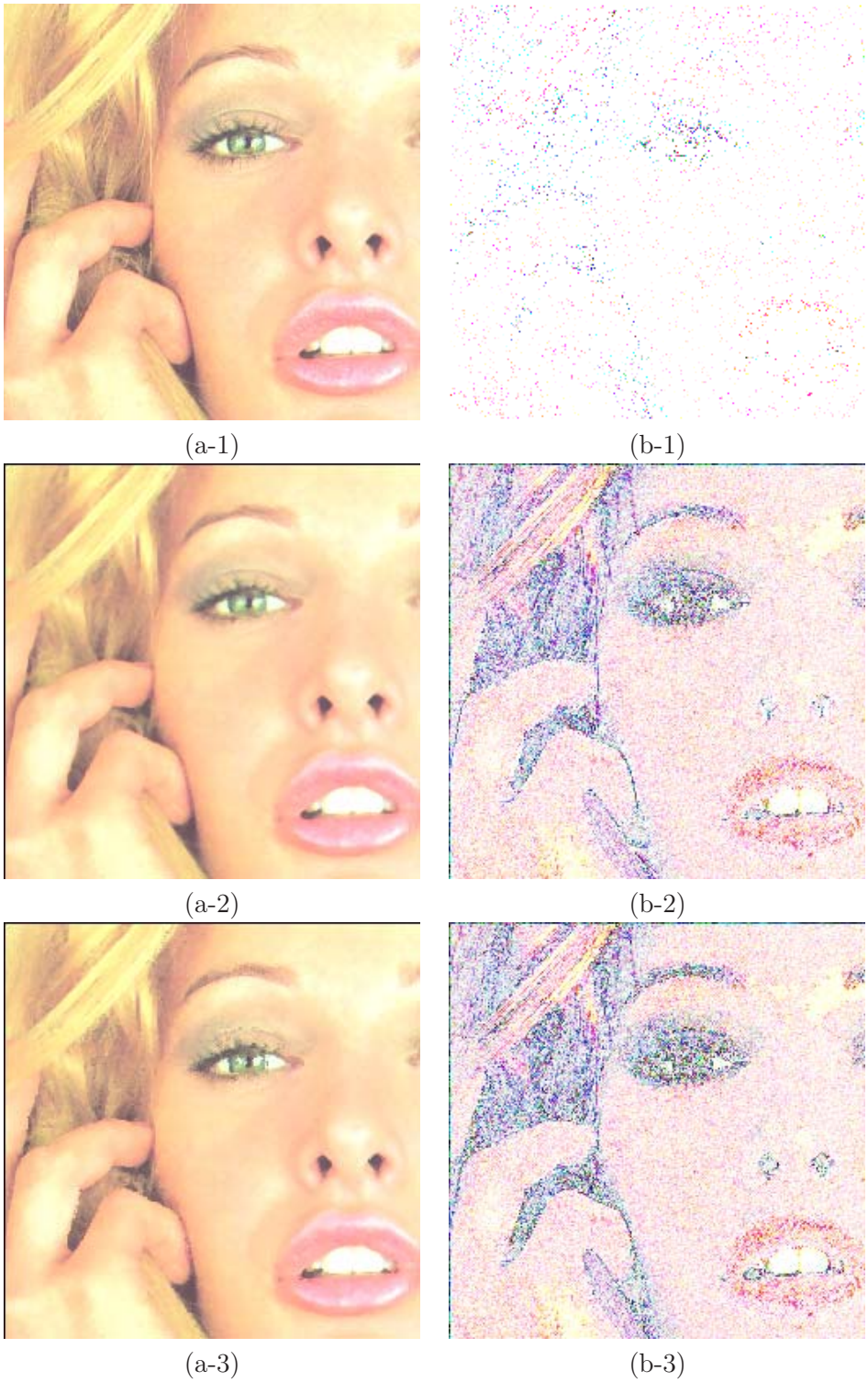


(a-1) (b-1)

(a-2) (b-2)

(a-3) (b-3)

FIG. *2.32: Comparaison des résultats avec VMF et BVDF pour l'image Poivron bruitée par 2% de bruit impulsionnel. (a-1) la sortie de l'algorithme proposé dans l'espace couleur CIELab, (a-2),(a-3) VMF et BVDF respectivement, (b-1),(b-2) et (b-3) la différence absolue entre l'image originale et l'image filtrée (les valeurs de l'espace couleur sont multipliées par un facteur 10).*

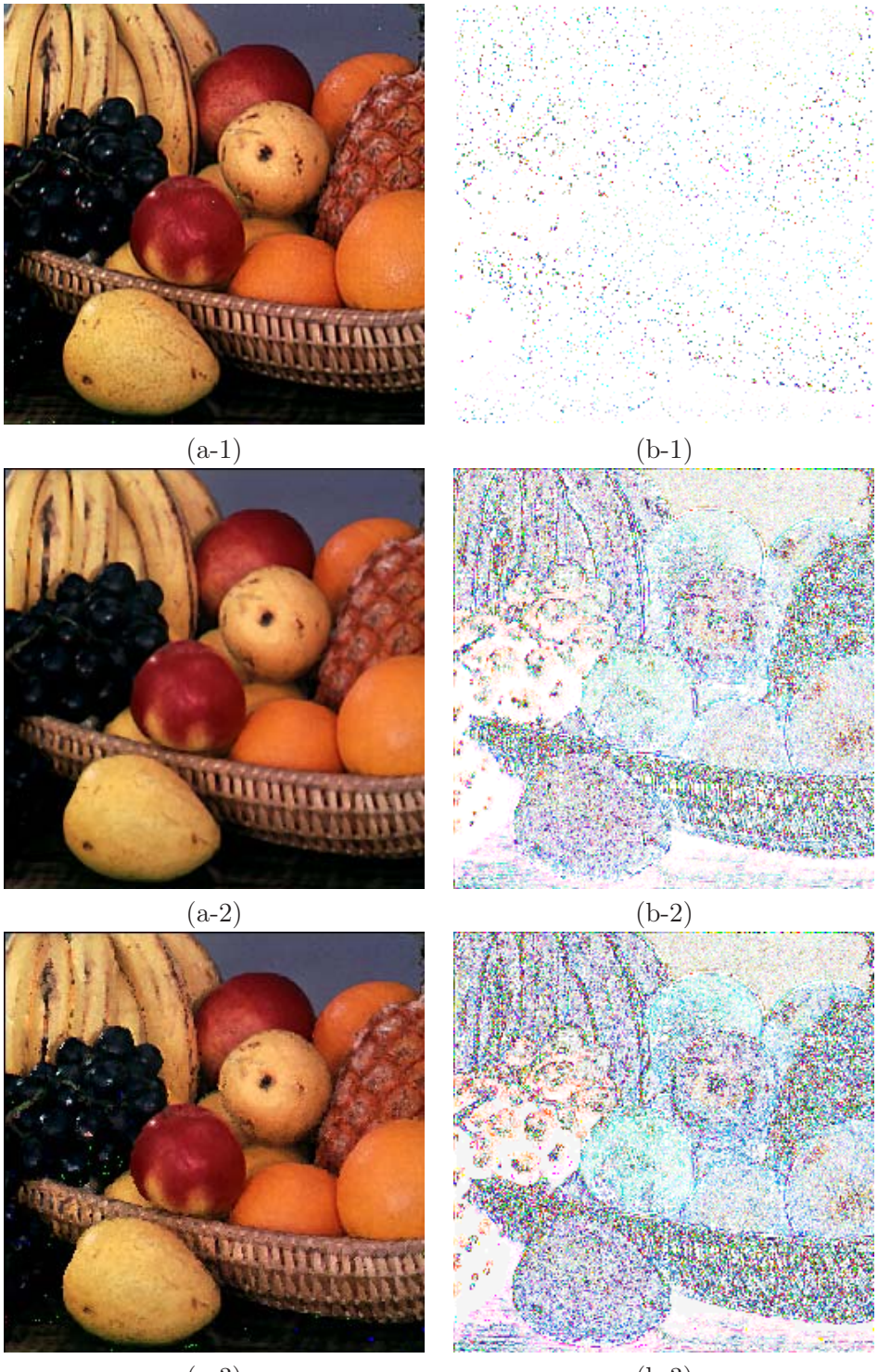


(a-1) (b-1)

(a-2) (b-2)

(a-3) (b-3)

FIG. 2.33: *Comparaison des résultats avec VMF et BVDF pour l'image Tiffany bruitée par 2% de bruit impulsionnel. (a-1) la sortie de l'algorithme proposé dans l'espace couleur CIELab, (a-2),(a-3) VMF et BVDF respectivement, (b-1),(b-2) et (b-3) la différence absolue entre l'image originale et l'image filtrée (les valeurs de l'espace couleur sont multipliées par un facteur 10).*



(a-1) (b-1)

(a-2) (b-2)

(a-3) (b-3)

FIG. *2.34: Comparaison des résultats avec VMF et BVDF pour l'image Fruit bruitée par 2% de bruit impulsionnel. (a-1) la sortie de l'algorithme proposé dans l'espace couleur CIELab, (a-2),(a-3) VMF et BVDF respectivement, (b-1),(b-2) et (b-3) la différence absolue entre l'image originale et l'image filtrée (les valeurs de l'espace couleur sont multipliées par un facteur 10).*

## 2.7 Conclusion

Dans une première partie de ce chapitre, nous nous sommes attachés à présenter les modèles de bruit et les modèles de perturbations. En effet, nous avons présenté trois modèles de bruit, gaussien généralisé, Middleton de classe A et alpha-stable. Trois modèles de génération de bruit dans une image sont définis dans ce chapitre : additif, multiplicatif et convolutif. Ensuite, nous avons présenté quelques techniques de suppression de bruit dans une image numérique. Nous les avons regroupées en deux catégories : estimation et détection/estimation. L'algorithme proposé dans ce chapitre entre dans le cadre de la deuxième catégorie. Le principe de base de cette dernière est d'abord la détection des pixels de bruit ensuite une estimation.

Il repose à la fois sur une représentation souple et un mécanisme de classification utilisant des modèles des hyperarêtes de bruit. Le modèle de bruit permet de distinguer un bruit et n'importe quelle information utile, région homogène ou contour : ceci en se basant sur la notion d'homogénéité et d'inhomogénéité. L'inhomogénéité correspond au bruit, tandis que l'homogénéité traduit une région homogène. Une zone plus ou moins homogène correspond aux contours.

En effet, après une représentation de l'image par hypergraphe, nous déterminons si un pixel est bruité ou non. Nous avons défini deux modèles de bruit en se basant sur la notion d'inhomogénéité. Le premier modèle permet effectivement de classer les hyperarêtes isolées soit de cardinalité égale à 1 soit de cardinalité supérieure à 1. Les hyperarêtes de cardinalité 1 décrivent l'inhomogénéité, mais cette propriété est aussi caractéristique des contours. Pour différencier ces deux entités, nous avons ajouté une autre condition sur la structure spatiale des hyperarêtes de bruit.

Dans les premières expérimentations de ce chapitre, nous avons commencé d'abord par une comparaison de ces deux modèles afin de trouver le meilleur d'entre eux, capable de supprimer le bruit tout en conservant les informations utiles de l'image et particulièrement les contours. Les comparaisons sont menées sur la représentation A. En examinant ces résultats, nous avons constaté que le deuxième modèle est plus performant en terme de détection et préservation des contours par rapport au premier modèle. Ensuite, nous avons comparé les résultats de la détection de bruit et de l'estimation de l'algorithme proposé en utilisant les trois représentations par hypergraphe de voisinage de l'image, avec l'objectif de trouver la représentation adéquate pour l'application proposée dans ce chapitre. Après plusieurs simulations, nous avons constaté que les deux représentations B et C sont plus adaptées pour la suppression de bruit.

A cette étape, nous avons seulement évalué l'approche proposée pour trouver le bon modèle de bruit et la meilleure représentation d'image par hypergraphe de voisinage. Une fois évalué l'algorithme de suppression de bruit, nous sommes passés à une comparaison avec les algorithmes de la littérature. Cette comparaison est menée avec deux classes d'algorithmes : la classe détection/estimation, où l'on trouve les algorithmes de Stevenson et Tovar, et la classe estimation où l'on trouve les filtres linéaire et non linéaire, tels que le filtre médian, médian multiétage et Wiener. En examinant ces résultats de comparaison, nous avons remarqué que l'algorithme de suppression de bruit développé permet d'obtenir une meilleure image estimée à la fois en suppression de bruit et en conservation de contours. Cette dernière propriété a fait défaut à un bon nombre d'approches.

Dans la deuxième partie de ce chapitre, nous avons présenté l'algorithme de suppression de bruit dans l'image couleur. Cet algorithme s'appuie uniquement sur la représentation SpatioColorimétrique HVSC puisque le modèle de bruit est resté le même.

Pour générer la représentation HVSC, on a fait appel à des distances colorimétriques. Ces dernières changent en fonction de l'espace couleur utilisé. Dans notre cas, nous avons utilisé la distance euclidienne dans l'espace couleur.

Dans les expérimentations, nous avons comparé tout d'abord l'algorithme proposé utilisant les stratégies marginale et vectorielle. En examinant ces résultats, nous avons constaté que l'approche vectorielle est plus performante que l'approche marginale, car cette dernière ignore la corrélation entre les composantes. Ensuite, nous avons comparé les espaces couleur RVB et CIELab. Les résultats de cette comparaison ont montré que l'espace CIELab est meilleur que le RVB. En effet, cette supériorité est justifiée par l'utilisation de la distance euclidienne dans un espace uniforme.

Dans la dernière section de cette deuxième partie, nous avons présenté un ensemble d'expérimentations. L'objectif de ces expérimentations est de comparer l'algorithme proposé avec les filtres VMF et BVDF. Pour une bonne préservation de contours, il est clair que le recours à des filtres globaux, qui s'appliquent à l'image entière, n'est pas la meilleure solution. Les zones intactes des images sont souvent de haute définition et il serait souhaitable de ne pas filtrer ces régions.

# Chapitre 3

# APPLICATION : DETECTION DE CONTOURS

**Résumé :**

*La détermination des limites des objets dans une scène présente un intérêt primordial pour le traitement de l'image. La détection de contours est ainsi un sujet de recherche très important en traitement d'images, et les contours ont été et sont toujours beaucoup utilisés comme primitives en analyse d'images et en vision par ordinateur. La qualité et la précision des contours détectés jouent donc un rôle très important chaque fois que l'on doit mettre en correspondance des primitives robustes issues d'images différentes. Citons par exemple : la stéréovision, l'analyse du mouvement, etc. Dans le présent chapitre, nous nous intéressons à la détection des contours. Dans la première partie, nous décrivons tout d'abord quelques méthodes de détection des contours des images à niveaux de gris. Nous décrivons ensuite le modèle de contour structurel sur l'hypergraphe de voisinage associé à l'image et proposons un algorithme de détection des contours des images à niveaux de gris basé sur cette modélisation. Les résultats de la détection seront comparés avec les algorithmes de détection couramment utilisés, tels que les algorithmes de Canny [Can86], SUSAN [SB97], Deriche [Der87a] et Shen et Castan [SC92]. Dans la deuxième partie de ce chapitre, nous présentons l'extension de cet algorithme à des images couleur. Nous commençons d'abord par une présentation de quelques méthodes de détection des contours dans les images couleur, ensuite nous donnons l'algorithme proposé. Les résultats de la détection seront comparés avec le détecteur de Canny marginal et l'algorithme de Cumani [Cum89]. Dans la troisième partie de ce chapitre, nous décrivons le principe de l'algorithme de classification de l'image en bruit, contours et régions homogènes. Cet algorithme sera appliqué en premier lieu à des images à niveaux de gris et ensuite à des images couleur.*

## 3.1 Introduction

Dans une image, les contours se situent entre les pixels appartenant à des régions ayant des intensités moyennes différentes ; il s'agit de contours de type "saut d'amplitude". Un contour peut également correspondre à une variation locale d'intensité présentant un maximum ou un minimum, il s'agit de contour "en toit".

Les approches dérivatives sont les plus immédiates pour détecter et localiser les variations du signal. Elles sont fondées sur la recherche des points de l'image présentant un fort gradient, ou de dérivée seconde nulle. Dans les approches surfacique et morphologique, l'image des intensités est considérée comme une surface. Dans les méthodes surfaciques, la transition entre deux régions est modélisée par un gabarit utilisable à deux fins :

- ▷ un contour est présent quand la mise en correspondance entre le gabarit et une zone de l'image est bonne ;
- ▷ l'approximation par facettes de la surface fournit une équation analytique locale qui permet de calculer de manière très précise (sub-pixel) la position du contour et ses caractéristiques.

Les méthodes morphologiques travaillent sur les maximums et les minimums des intensités du voisinage de chaque pixel. Un contour de type "saut d'amplitude" sera détecté si la différence entre le maximum et le minimum est importante ou, pour des contours "en toit" par extraction des lignes de crêtes entre bassins versants. Notons que d'une manière générale, chaque méthode de détection est justifiée par la définition du type de contour recherché. La plupart des méthodes de détection de contours sont basées sur la recherche des modèles de contours de type "marche" additionné à un bruit blanc.

L'objectif de ce chapitre est d'utiliser la structure d'hypergraphe de voisinage d'images afin de caractériser les zones de transition entre objets, c'est à dire typiquement ce que nous dénommons hyperarête de contours. Dans la première section de ce chapitre, nous présentons les algorithmes de Canny, Deriche, Shen et Castan et SUSAN. Nous procédons ensuite à la définition de ce que nous appelons pixels appartenant à une hyperarête de contours d'objet, puis, nous proposons un algorithme permettant de les déterminer au sein d'une image numérique à niveaux de gris. Après avoir donné quelques résultats expérimentaux, nous procédons à une comparaison de ces derniers avec les algorithmes les plus couramment utilisés en traitement d'images. Des résultats sur plusieurs types d'images sont présentés et des études comparatives permettent alors de tirer quelques conclusions. Dans la deuxième partie de ce chapitre, nous présentons d'abord une extension aux images couleur de l'algorithme de détection de contours. Ensuite, nous comparons les cartes de contours de l'algorithme proposé avec le détecteur de Canny marginal et l'algorithme de Cumani. Dans la troisième et la dernière partie, nous donnons un algorithme de classification de l'image à niveaux de gris et couleur en hyperarêtes de contours et bruit. En utilisant cette classification, nous présentons les résultats de l'algorithme de détection des contours dans des images bruitées. Les résultats de cet algorithme seront comparés avec les algorithmes de Canny, Deriche, Shen et Castan et SUSAN.

Nous allons maintenant décrire quelques exemples typiques de détection de contours. Le lecteur pourra trouver une description beaucoup plus complète des techniques de détection de contours d'images dans [CP95].

## 3.2 Approche contour classique

L'approche de détection de contours consiste à convoluer une fenêtre de l'image par un opérateur. Ce dernier est appliqué sur les pixels de la fenêtre afin d'estimer s'il y a une transition significative au niveau de l'attribut choisi. A partir des pixels susceptibles d'appartenir à un contour, il faut ensuite extraire des contours fermés. Pour une image sans texture, un pixel contour est souvent défini comme un maximum local du module du gradient dans la direction du gradient, ou encore comme un passage par zéro de la dérivée seconde dans cette même direction. Comme les opérateurs de dérivation sont très sensibles au bruit, des images bruitées doivent être préalablement lissées. Un grand nombre d'opérateurs gradient ont été proposés. Ils se distinguent entre eux principalement par le choix du filtre de lissage. Le lissage et la dérivation sont en pratique réunis dans un seul filtre.

Soit $I(x, y)$ la valeur du pixel $(x, y)$ d'une image brute ou lissée. Les approximations discrètes des premières dérivées partielles continues sont les différences suivantes :

$$\frac{\partial}{\partial x} I(x, y) \backsimeq \Delta_x * I(x, y) = I(x + 1, y) - I(x, y) \tag{3.1}$$

$$\frac{\partial}{\partial y} I(x,y) \backsimeq \Delta_y \star I(x,y) = I(x,y+1) - I(x,y) \tag{3.2}$$

où $*$ et $\star$ signifient convolution dans les directions horizontale et verticale, respectivement. Les deuxièmes dérivées partielles dans ces mêmes directions sont approchées par :

$$\frac{\partial^2}{\partial^2 x} I(x,y) \backsimeq \Delta_x^2 * I(x,y) = I(x+1,y) - 2I(x,y) + I(x-1,y) \tag{3.3}$$

$$\frac{\partial^2}{\partial^2 y} I(x,y) \backsimeq \Delta_y^2 \star I(x,y) = I(x,y+1) - 2I(x,y) + I(x,y-1) \tag{3.4}$$

Les masques de convolution correspondants sont présentés dans la figure 3.1

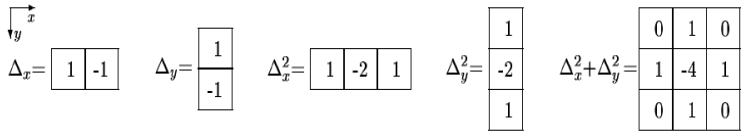



FIG. *3.1: Masques de convolution réalisant les approximations discrètes des dérivées continues.*

### 3.2.1 Opérateurs différentiels du premier ordre

Ces méthodes consistent à estimer le module du gradient en chaque pixel de l'image :

$$|\nabla I(x,y)| = \sqrt{\left(\frac{\partial}{\partial x} I(x,y)\right)^2 + \left(\frac{\partial}{\partial y} I(x,y)\right)^2} \tag{3.5}$$

à l'aide de deux masques de convolution correspondant aux directions verticale et horizontale. Pour atténuer le bruit, les masques comportent souvent un certain lissage de l'image.

**Opérateurs simples.** Plusieurs opérateurs gradient simples basés sur les approximations discrètes des dérivées de premier ordre données par 3.3 et 3.4 ont été proposés. Quelques masques classiques sont illustrés dans la figure 3.2.

- ☐ Roberts : les masques de Roberts sont des versions de $\Delta_x$ et $\Delta_y$ ayant subi une rotation de -45 degrés. Ce sont de simples différences de niveaux de gris, sans aucun lissage.
- ☐ Prewitt : à un facteur constant près, la méthode de Prewitt [Pre70] consiste à calculer la différence des valeurs moyennes de chaque côté du pixel central dans les directions verticale et horizontale. Le moyennage rend la méthode plus robuste au bruit.
- ☐ Sobel : l'opérateur de Sobel [Sob90] se distingue de celui de Prewitt en donnant plus d'influence au plus proche pixel de chaque côté du pixel central, ce qui a pour effet de réduire un peu le lissage, mais aussi d'améliorer légèrement la précision de localisation.

Ces opérateurs peuvent donner des résultats acceptables sur des images peu bruitées, mais il existe des méthodes beaucoup plus performantes.

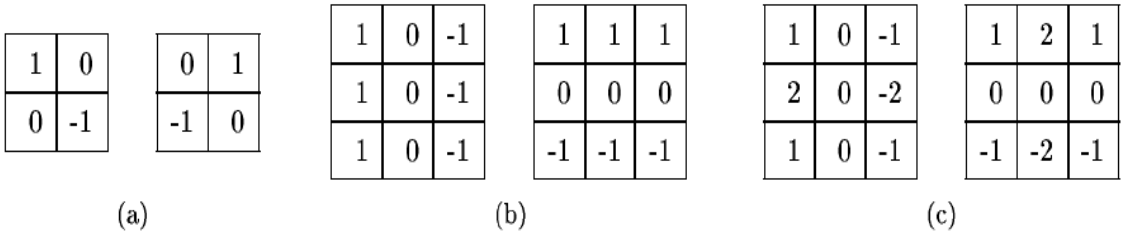



FIG. *3.2: Masques de convolutions des opérateurs de (a) Roberts, (b) Prewitt et (c) Sobel.*

**Opérateurs optimaux.** L'opérateur de Sobel introduit une pondération au niveau des moyennes locales calculées de chaque côté du pixel central. Plusieurs travaux ont cherché à optimiser cette pondération. Canny, en 1983, a postulé trois critères [Can83] :

1. Bonne détection de contours ;
2. Bonne localisation de contours ;
3. Faible multiplicité des maxima dus au bruit.

En supposant qu'un contour peut se modéliser comme un échelon perturbé par un bruit blanc gaussien additif, Canny a traduit chaque critère en termes mathématiques précis, utilisés ensuite pour optimiser le filtre de lissage $f(x)$ et le filtre différentiel correspondant $g(x) = df(x)/dx$.

- ☐ Canny s'est limité aux filtres de réponse impulsionnelle finie. L'optimisation de la robustesse au bruit et de la localisation, sous la contrainte d'une distance moyenne minimale entre les réponses multiples, a donné le différentiel $g$ qui peut être approximé par la première dérivée d'une gaussienne [Can86].
- ☐ Deriche a généralisé les idées de Canny en permettant que le filtre soit de réponse impulsionnelle infinie. Il a ainsi obtenu en 1987 le filtre de lissage et le filtre différentiel suivants [Der87a][Der90] :

$$f(x) = b_1(1 + \tau|x|)e^{-\tau|x|} \tag{3.6}$$

$$g(x) = -b_2 x e^{-\tau|x|} \tag{3.7}$$

  où $b_1$ et $b_2$ sont des facteurs de normalisation, et $\tau$ est le paramètre de forme qui, entre autre, détermine la capacité de lissage.

- ☐ Shen et Castan ont proposé de façon indépendante une approche assez voisine de celle proposée par Canny en présentant un critère qui combine la bonne détection et la bonne localisation. L'optimisation de ce critère les conduit à proposer une fonction de régularisation continue qui est ensuite dérivée pour aboutir aux filtres optimaux de réponse impulsionnelle suivants [SC92] [SC93] :

$$f(x) = a_1 e^{-\gamma|x|} \tag{3.8}$$

$$g(x) = \begin{cases} a_2 e^{-\gamma|x|} & pour \quad x < 0 \\ -a_2 e^{-\gamma|x|} & pour \quad x > 0 \end{cases} \tag{3.9}$$

  où $a_1$ et $a_2$ sont les facteurs de normalisation, et $\gamma$ est le paramètre qui contrôle la pente de la fonction exponentielle, donc le lissage.

Les réponses impulsionnelles des filtres de lissage optimaux de Deriche et de Shen et Castan sont présentées par la figure 3.3 pour respectivement $\tau = 0.50$ et $\gamma = 0.25$. Les filtres différentiels correspondants sont représentés par la figure 3.4. Les versions discrètes de ces filtres peuvent être implantées récursivement, ce qui rend ces méthodes très rapides [Der87b]. Soient $G_x$ et $G_y$ les composantes horizontale et verticale du gradient de l'image $I$ lissée par $f(x,y) = f(x)f(y)$. Compte tenu des règles de dérivation de l'opération de convolution, nous pouvons écrire :

$$G_x(x,y) = \frac{\partial}{\partial x}(f(x,y) \otimes I(x,y)) = g(x) * (f(x,y) \star I(x,y)) \tag{3.10}$$

$$G_y(x,y) = \frac{\partial}{\partial y}(f(x,y) \otimes I(x,y)) = g(x) \star (f(x,y) * I(x,y)) \tag{3.11}$$

où $\otimes$ signifie convolution bidimensionnelle. Rappelons que $*$ et $\star$ représentent des convolutions unidimensionnelles effectuées respectivement ligne par ligne et colonne par colonne. On utilise par ailleurs la notion de séparabilité de la réponse impulsionnelle du filtre.

Rappelons que l'optimisation des filtres selon les trois critères postulés par Canny est basée sur l'hypothèse d'un contour isolé. Une propriété intéressante du filtre de lissage de Shen et Castan est qu'il est également optimal sous un modèle multicontour stochastique [SC92]. Cette méthode tient donc compte du risque que plusieurs contours soient simultanément présents à l'intérieur de la fenêtre d'analyse. Il est particulièrement important de prendre en compte cette éventualité pour des filtres de réponse impulsionnelle infinie.

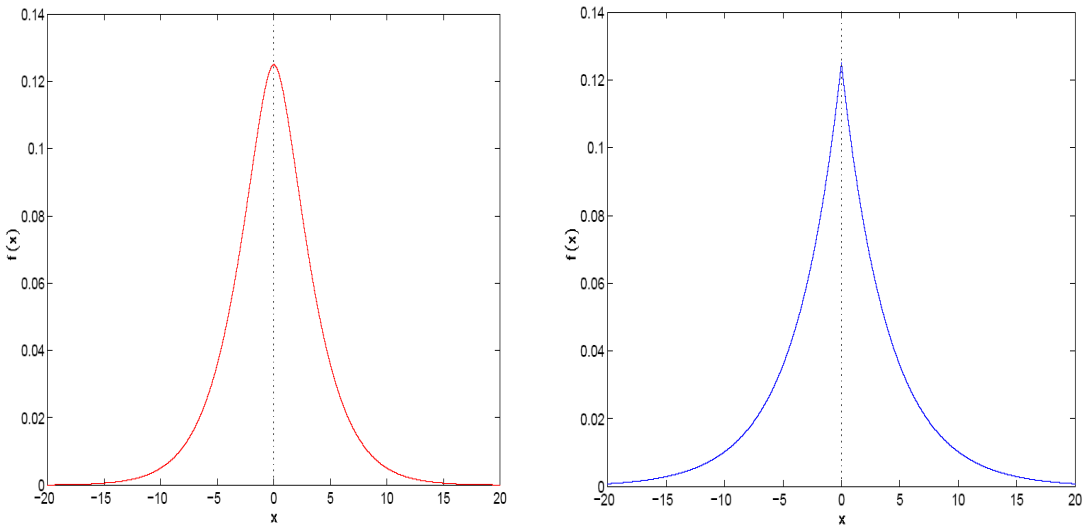
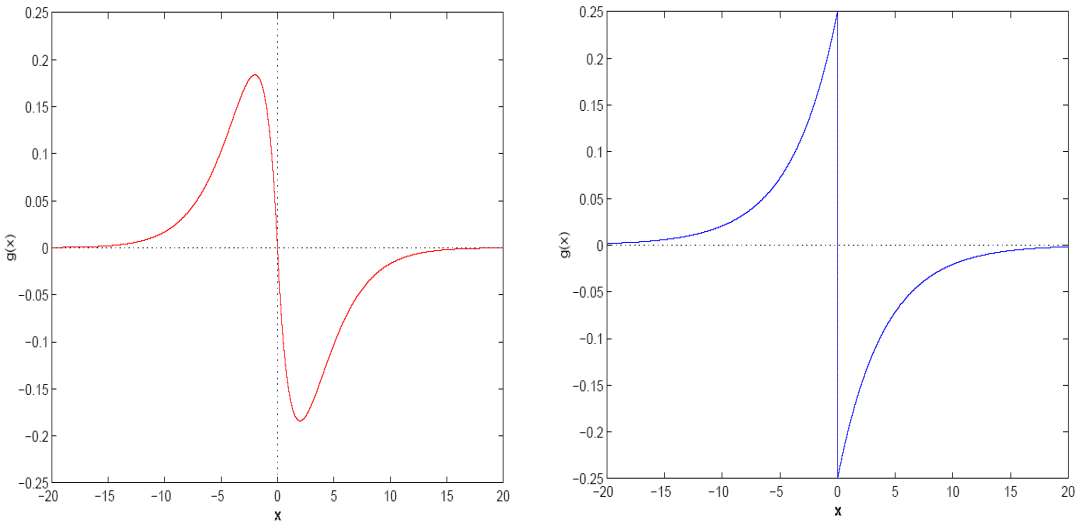



FIG. *3.3: Réponses impulsionnelles des filtres de lissage (a) de Deriche et (b) de Shen et Castan.*

**Mise en œuvre.** La démarche générale de la recherche des points de contours à partir du gradient se décompose comme suit :

- ☐ Calculer le gradient en chaque point de l'image ;
- ☐ Créer l'image de la norme du gradient ;
- ☐ Extraire les maxima locaux dans la direction du gradient




(a) (b)

FIG. *3.4: Réponses impulsionnelles des filtres différentiels (a) de Deriche et (b) de Shen et Castan.*

☐ Effectuer un seuillage à hystérésis de l'image des maxima locaux. Le seuillage à effet hystérésis consiste à ne conserver que :
- Les points dont la norme du gradient est supérieure à un seuil haut ($s_h$).
- Les points dont la norme du gradient est supérieure à un seuil bas ($s_b$ avec $s_b < s_h$) et appartenant à un bout de contour dont au moins un point possède une norme du gradient supérieure à $s_h$.

Les méthodes de détection de contours dérivatifs permettent d'obtenir très rapidement (de l'ordre de la seconde) un ensemble de contours qui serviront de base à des algorithmes de fermeture de contours. En raison de leur caractère local, ces méthodes doivent diminuer le bruit en appliquant un filtre passe-bas comme la gaussienne ou les filtres de Deriche et Shen. L'utilisation de ces filtres pose le problème du réglage des paramètres. Si l'on filtre trop, on risque d'exclure des bouts de contours importants alors que si on ne filtre pas assez, on risque de se retrouver avec une quantité de contours non significatifs.

## 3.3 Approche utilisant une modélisation surfacique

Il est possible de définir un modèle surfacique de transition, par exemple : un échelon bidimensionnel sur un voisinage donné. Un contour sera détecté en un point s'il y a une bonne corrélation entre le modèle et la fenêtre de l'image centrée en ce point.

Huertas et Médioni [HM86] utilisent l'image résultant de l'application du laplacien d'une gaussienne sur l'image initiale. Au voisinage d'un passage par zéro, l'image est approximée par un polynôme cubique. Les passages par zéro identifiant les contours sont détectés de manière analytique par interpolation à partir de l'équation du polynôme.

Une autre méthode consiste à approcher la surface représentant une transition par un polynôme dont l'équation aura pour objectif d'obtenir l'orientation du contour.

## 3.4 Algorithme SUSAN

L'algorithme de Smith [SB97] est conçu pour un traitement d'image bas-niveau, particulièrement la détection de contours, détection des coins et réduction de bruit en conservant les détails de l'image traitée. Dans cette section, nous nous intéressons seulement à la description de la méthode de détection de contours de l'algorithme de Smith. La figure 3.5 illustre un rectangle (région homogène 1) dans un arrière plan blanc (région homogène 2). Un masque circulaire (avec un pixel de centre) est placé sur cinq positions (a, b, c, d et e). La comparaison de l'intensité de chaque pixel situé à l'intérieur du masque circulaire avec l'intensité du pixel central du masque permet de définir une zone où les pixels ont la même intensité que le centre du masque. Cette zone est nommée USAN (*Univalue Segment Assimilating Nucleus*).

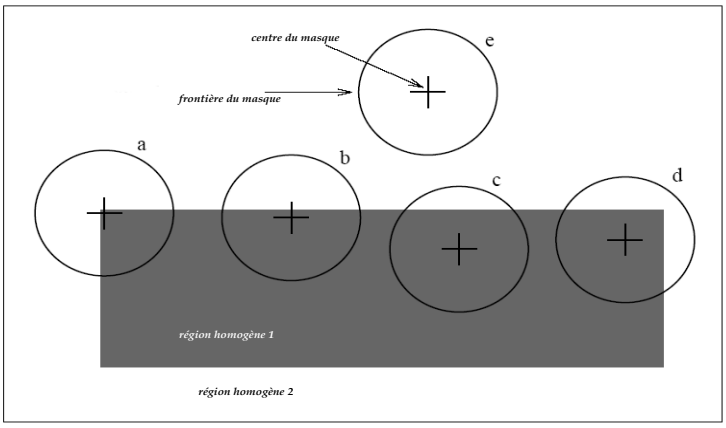



FIG. *3.5: Quatre masques circulaires pour différentes positions dans une image simple.*

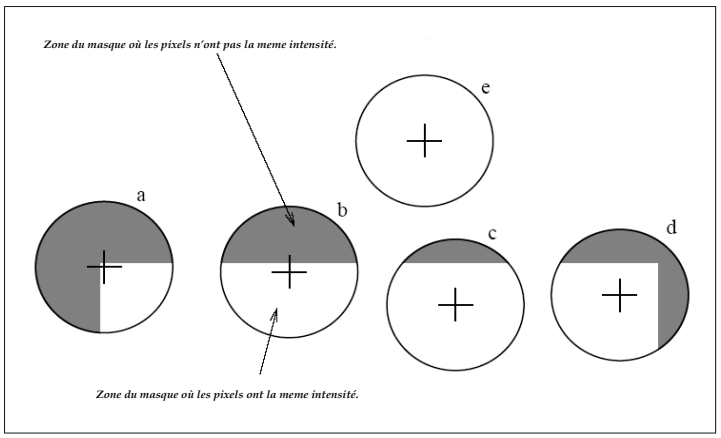



FIG. *3.6: Les cinqs zones USAN de la figure 3.5 pour les cinq positions des masques circulaires.*

La zone USAN de chaque pixel contient une information importante de la structure de l'image. Pour chaque masque dans la figure 3.6, la zone USAN est montrée en blanc. D'après les figures 3.5 et 3.6, la zone USAN est maximale lorsque le masque circulaire est situé dans une région homogène. Les contours correspondent à une USAN qui occupe 1/2 du masque. Les coins cor-

respondent à une zone qui occupe le 1/4 du masque. Ce sont ces propriétés qui sont utilisées par Smith pour classifier les entités de l'image traitée, régions homogènes, contours et coins. Dans ce qui suit, nous donnons l'algorithme de Smith :

1. Placer un masque circulaire autour de chaque pixel ;
2. Construire les USANs : utiliser l'équation 3.12, pour calculer le nombre de pixels situés à l'intérieur du masque circulaire qui ont une intensité similaire à celle du pixel central. Le nombre de pixels dans un USAN est défini par :

$$n(\vec{r_0}) = \sum_{\vec{r}} c(\vec{r}, \vec{r_0}) \tag{3.12}$$

avec :

$$c(\vec{r}, \vec{r_0}) = \begin{cases} 1 & si \quad |I(\vec{r} - I(\vec{r_0}))| \leq t \\ 0 & si \quad |I(\vec{r} - I(\vec{r_0}))| > t, \end{cases} \tag{3.13}$$

$$c(\vec{r}, \vec{r_0}) = e^{-\left(\frac{I(\vec{r}) - I(\vec{r_0})}{t}\right)^6} \tag{3.14}$$

$\vec{r}$ et $\vec{r_0}$ représentent respectivement les vecteurs du centre du masque et à l'intérieur du masque. $t$ est un seuil sur l'intensité. La fonction 3.14 est plus adéquate que la fonction 3.13 [SB97].

3. Calculer la réponse initiale des contours $R(\vec{r_0})$ en utilisant l'équation définie par :

$$R(\vec{r_0}) = \begin{cases} g - n(\vec{r_0}) & si \quad n(\vec{r_0}) < g \\ 0 & ailleurs \end{cases} \tag{3.15}$$

ou $g$ est un seuil géométrique.

4. Déterminer la direction des contours en calculant les moments dans les USAN. Cette direction est estimée par :

$$\overline{(x - x_0)^2}(\vec{r_0}) = \sum_{\vec{r}} (x - x_0)^2 c(\vec{r}, \vec{r_0}) \tag{3.16}$$

$$\overline{(y - y_0)^2}(\vec{r_0}) = \sum_{\vec{r}} (y - y_0)^2 c(\vec{r}, \vec{r_0}) \tag{3.17}$$

et

$$\overline{(x - x_0)(y - y_0)}(\vec{r_0}) = \sum_{\vec{r}} (x - x_0)(y - y_0) c(\vec{r}, \vec{r_0}) \tag{3.18}$$

Le rapport $\frac{\overline{(y-y_0)^2}}{\overline{(x-x_0)^2}}$ est utilisé pour déterminer l'orientation du contour ; le signe de $\overline{(x - x_0)(y - y_0)}$ est utilisé pour déterminer si le contour diagonal a un gradient négatif ou positif.

5. Suppression des non maxima locaux, amincissement et éventuellement estimation sub-pixel.

## 3.5 Autres approches de détection de contours

Dans les méthodes de détection de contours présentées jusqu'à maintenant, le but était de modéliser le contour à détecter. Ce qui n'est pas le cas d'autres catégories de méthodes, parmi lesquelles on trouve : l'approche variationnelle [KWT87], l'approche statistique [Sou83][KLL96], l'approche basée sur la programmation dynamique [TGD92], l'approche morphologique[AM00] [GA95], l'approche multirésolution [LB02] etc.

## 3.6 Détection des contours en utilisant la représentation par hypergraphe de voisinage

### 3.6.1 Modèle de contour

Nous avons vu dans le chapitre 2 qu'il était possible de caractériser les zones de bruit d'une image donnée. Nous avions en effet défini celles-ci comme l'ensemble des pixels regroupés dans des hyperarêtes isolées vérifiant le modèle du bruit proposé. Rappelons le premier modèle que nous avions donné d'une hyperarête de bruit. Une hyperarête est considérée comme hyperarête de bruit si : (1) elle est isolée de cardinalité égale à 1 ou (2) elle est isolée de cardinalité supérieure à 1 et en plus dans son voisinage ouvert, il existe au moins une hyperarête isolée.

D'après les simulations expérimentales, nous avons constaté que ce modèle peut ne pas distinguer les contours et le bruit. En effet, la carte de bruit de sortie porte une information sur les contours de l'image traitée. Pour pallier à cette difficulté, nous avons ajouté la définition d'une chaîne disjointe (cf. §1.2). Et par conséquent, une hyperarête est dite hyperarête de bruit, si : (1) elle est isolée de cardinalité égale à 1 et ne forme pas une chaîne disjointe avec son voisinage ouvert et (2) elle est isolée de cardinalité supérieure à 1 et en plus dans son voisinage ouvert, il existe au moins une hyperarête isolée.

En nous penchant sur l'étude des propriétés des hyperarêtes, nous avons constaté qu'il était possible de caractériser les pixels appartenant à un contour d'objet.

L'étude du bruit détecté par le premier modèle, nous a permis de constater qu'on peut définir l'hyperarête de contour à l'aide des hyperarêtes non isolées formant une chaîne. En effet, une hyperarête non isolée, signifie, une relation avec le voisinage ouvert. Ceci nous permet de dire, que les hyperarêtes non isolées peuvent être la clé pour faire la détection de contours. En plus des hyperarêtes non isolées, nous ajoutons au modèle de contours les hyperarêtes résultantes d'une soustraction entre les deux modèles de bruit 1 et 2, c'est à dire les hyperarêtes isolées de cardinalité 1 formant une chaîne disjointe. Cette propriété est issue de l'étude du bruit. En effet, d'après les simulations du deuxième chapitre, nous avons constaté qu'effectivement, les hyperarêtes isolées formant une chaîne disjointe décrivent une partie des contours. Néanmoins cette deuxième modélisation ne donne pas la totalité des zones de contours. Nous donnons ci-dessous le modèle de contours. La figure 3.7 représente une illustration graphique de ce modèle.

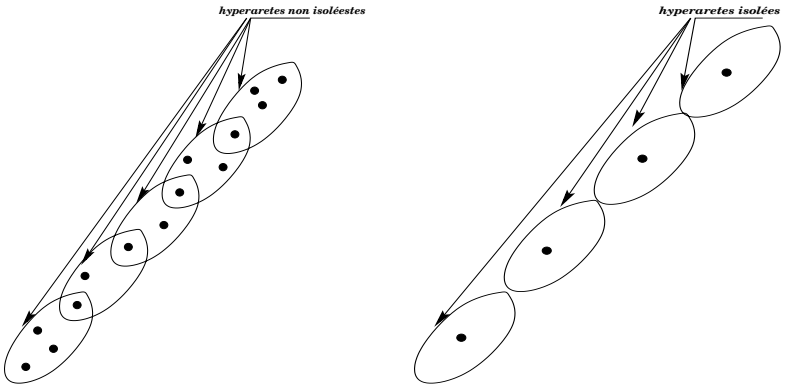
**Modèle de contour :** Pour chaque $E_{\alpha,\beta}(x)$.
Si cette hyperarête vérifie l'une des deux conditions :
- $E_{\alpha,\beta}(x)$ est une hyperarête non isolée, incluse, dans une chaîne de plus de $\omega$ éléments ;
- $E_{\alpha,\beta}(x)$ est une hyperarête isolée de cardinalité égale à 1, incluse dans une chaîne disjointe de plus de $\omega$ éléments ;

alors : $E^{contour}_{\alpha,\beta}(x) \longleftarrow E_{\alpha,\beta}(x)$.

Le modèle de contour exploite le critère de l'homogénéité. Nous considérons que l'homogénéité globale caractérise les régions, l'homogénéité locale caractérise les contours. La notion de l'homogénéité a été exploitée dans [BACL97] pour mettre en œuvre un algorithme de segmentation (Approche région en utilisant la représentation A pour la construction de l'hypergraphe).

### 3.6.2 Algorithme de détection de contours

L'algorithme de détection de contour est basé en premier lieu sur une représentation d'image en utilisant l'hypergraphe d'image décrit dans le premier chapitre, en deuxième lieu sur une classification des hyperarêtes de l'image, contour ou non contour, en utilisant le modèle décrit




(a) chaîne formée par des hyperarêtes non isolées

(b) chaîne formée par des hyperarêtes isolées de cardinalité 1.

FIG. *3.7: Illustration graphique du modèle de contour.*

précédemment. De la même manière, l'algorithme de détection de contours peut utiliser soit la représentation A, B ou C ; c'est à dire : changement des paramètres de la représentation par hypergraphe de voisinage de l'image.

L'algorithme de classification de pixels appartenant à une zone de contours est peu différent de l'algorithme de débruitage. Le voici néanmoins explicitement exprimé :

**Algorithme :**

- ☐ **Données.**
  Image à niveaux de gris $I$, ordre de voisinage $\beta$, longueur de la chaîne $\omega$ ;
- ☐ **Construction de l'hypergraphe de voisinage $H_{\alpha,\beta}$ sur $I$.**
  - ▷ hypergraphe de voisinage à seuil fixe (représentation A) :
    $\alpha$ fixe, $\in [0, 255]$ et $d'(I(x), I(y)) = |I(x) - I(y)| \leq \alpha$.
  - ▷ ou, hypergraphe de voisinage à seuil dynamique (représentation B) :
    $\alpha = f(I) = k\hat{\sigma}$, avec $\hat{\sigma}$ est l'estimé de l'écart-type du voisinage $8 - adjacents$ et k varié de 0 à 1. $d'(I(x), I(y)) = |I(x) - I(y)| \leq k\hat{\sigma}$.
  - ▷ ou, hypergraphe de voisinage utilisant une mesure de similarité (représentation C) :
    $\alpha$ fixe, $\in [0, 1]$ et $\mu(I(x), I(y)) \geq \alpha$. $\mu(.)$ est une fonction de similarité.
- ☐ **Détermination des hyperarêtes non isolées :** Pour chaque sommet $x$ de $X$
  - ▷ $E' \longleftarrow \emptyset$
  - ▷ $E' \longleftarrow \bigcup_{y \in E_{\alpha,\beta}(x)} E_{\alpha,\beta}(y)$
  - ▷ si $E' \neq E_{\alpha,\beta}(x)$, alors $E_{\alpha,\beta}(x)$ est une hyperarête non isolée.
- ☐ **Détermination des hyperarêtes isolées de cardinalité 1 :**
- ☐ **Hyperarêtes de contours.**
  Pour chaque $E_{\alpha,\beta}(x)$ isolée ou non isolée vérifiant le modèle de contour alors $E^{contour}_{\alpha,\beta}(x) \longleftarrow E_{\alpha,\beta}(x)$ ;

**Fin de l'algorithme**

### 3.6.3 Simulations et résultats expérimentaux

#### 3.6.3.1 Mesures d'évaluations

Afin de nous assurer de l'efficacité du modèle de contour, nous avons effectué une série de mesures sur des images numériques : d'abord synthétiques, ensuite réelles. L'algorithme de détection de contours proposé utilise les trois représentations par hypergraphe de voisinage d'image citées précédemment : représentations A, B et C.

Pour l'évaluation de la détection de contours, deux critères sont utilisés :

- ☐ les probabilités de détection et de fausse alarme. La probabilité de détection désigne ici la probabilité de décider qu'un pixel appartient à un contour sachant que c'est effectivement la cas. La probabilité de fausse alarme mesure quant à elle la probabilité des faux contours (pixel décidé contour alors qu'il appartient à une zone homogène). Pour les calculer, il faut d'abord disposer d'une carte de contours initiale qui permet de connaître les positions des pixels contours (carte de contours d'épaisseur 2) ; ensuite après détection par l'algorithme proposé, c'est à dire génération de la carte de sortie, nous comparons les deux cartes : initiale et après détection. Ce critère ne sera utilisé que pour les images de synthèse.
- ☐ le critère visuel. Dans ce cas, nous évaluons la sortie de l'algorithme visuellement en regardant les contours significatifs de l'image ainsi que les faux contours. Un bon détecteur de contours doit avoir plus de contours significatifs et moins de faux contours.

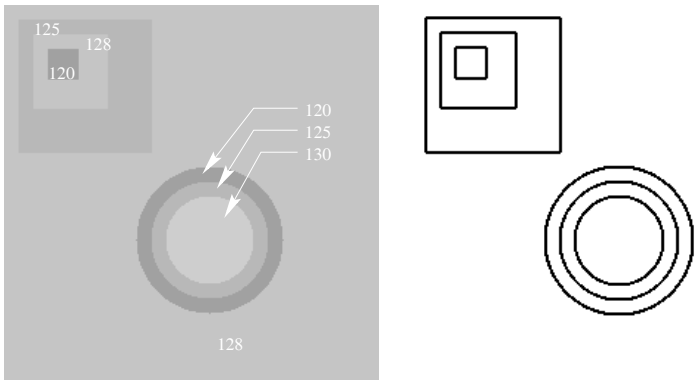
L'évaluation de l'algorithme de détection de contours par hypergraphe de voisinage se déroule en deux étapes : premièrement évaluation sur des images de synthèse, ensuite sur des images réelles. Dans le premier cas, nous commençons d'abord par une évaluation des représentations A, B et C. Ensuite, nous évaluons les performances en fonction de la longueur de la chaîne des hyperarêtes isolées ou non isolées. Puis une comparaison avec les algorithmes existants dans la littérature : Canny [4], Deriche, Shen et Castan et Susan. Finalement, dans le cas des images réelles, nous évaluons la représentation la plus adéquate avec les algorithmes cités précédemment.

#### 3.6.3.2 Evaluation sur des images de synthèse

Dans cette section, nous commençons tout d'abord par une évaluation des représentations A, B et C afin de trouver la représentation la plus adéquate à la détection de contours. Pour atteindre cet objectif, nous comparons les courbes opérationnelles de décision des trois représentations. La courbe opérationnelle illustre l'évolution de la probabilité de détection en fonction de la probabilité de fausse alarme. En effet, pour les représentations A, et B à une longueur de chaîne donnée $\omega$, nous faisons varier le seuil $\alpha$ et nous calculons les $\hat{pd}$ et $\hat{pf}$, puis nous traçons la courbe opérationnelle. Pour la représentation B, nous faisons varier le facteur $k$. Afin de tracer ces trois courbes, une image de test de dimension $256 \times 256$ a été synthétisée. La figure 3.8 représente l'image de synthèse utilisée ainsi que la carte de contours associée.

La figure 3.9 illustre les courbes opérationnelles de l'image de synthèse de la figure 3.8. Les courbes opérationnelles représentent l'évaluation de la probabilité de détection $\hat{pd}$ en fonction de la probabilité de fausse alarme $\hat{pf}$ de l'algorithme proposé en utilisant les trois représentations A, B et C avec une longueur de chaîne donnée $\omega = 5$ et un ordre de voisinage $\beta = 1$. La représentation C utilise la fonction de similarité $\mu_2(.)$ qui s'est avérée être efficace pour la détection de bruit.

[4] La solution trouvée par Canny pour aboutir à la réalisation des trois critères : bonne détection, bonne localisation et faible multiplicité des maxima de bruit n'étant pas simple à mettre en œuvre, il l'approxime par la fonction dérivée première d'une gaussienne. Durant nos simulations, l'algorithme de Canny est implémenté suivant cette approximation.




(a) (b)

FIG. *3.8: Image de synthèse (a) et carte de contours associée "épaisseur 2" (b).*

D'après cette figure, nous constatons tout d'abord que la courbe opérationnelle peut se diviser en 3 zones :

- ☐ Zone 1 : courbe opérationnelle possédant des valeurs de $\hat{pd}$ et $\hat{pf}$ faibles approximativement nulles (moins de contours significatifs et moins de faux contours).
- ☐ Zone 2 : courbe opérationnelle possédant des valeurs de pd élevées et pf faibles (plus de contours significatifs et moins de faux contours).
- ☐ Zone 3 : courbe opérationnelle possédant des valeurs de pd élevées et pf élevées (plus de contours significatifs et plus de faux contours).

La zone la plus intéressante dans l'évaluation des représentations est la zone 2. Nous constatons que la courbe opérationnelle associée à la représentation B se situe au dessus de celle associée aux autres représentations. Ceci nous amène à conclure que la représentation B est la plus performante. Il faut néanmoins nuancer cette conclusion. En effet, considérons les trois représentations pour une probabilité de détection fixe égale à $\hat{pd} = 0.73$. Les fausses alarmes associées respectivement aux représentations A, B et C sont 0.00078, 0.000641 et 0.00071. Selon ces valeurs, nous constatons que la représentation B introduit moins de faux contours que les représentations A et C dans la carte de contours. Les représentations A et C introduisent respectivement $21,68\%$ et $10.76\%$ de faux contours en plus que la représentation B. De la même façon, nous fixons la probabilité de fausse alarme par, exemple à $\hat{pf} = 0.000683$ et nous évaluons les probabilités de détection $\hat{pd}$. Les valeurs de détection de $\hat{pd}$ associées respectivement aux représentations A, B et C sont 0.7039, 0.7683 et 0.7283. Selon ces valeurs, la représentation B détecte plus de contours significatifs dans l'image de synthèse. Les représentations A et C perdent respectivement $8.38\%$ et $5.20\%$ de contours significatifs en les comparant avec la représentation B. Les performances des trois représentations sont donc relativement proches.

Après évaluation des représentations, nous illustrons maintenant l'influence du paramètre $\omega$ sur les performances de l'algorithme de détection de contours utilisant la représentation B. La table 3.1 représente les valeurs des probabilités de détection et de fausse alarme obtenues pour différentes valeurs de $\omega$ sur l'image de synthèse. L'algorithme utilise la représentation B avec un facteur $k = 0.5$ et un ordre de voisinage $\beta = 1$. Pour simplifier la lecture de la table 3.1, nous traçons les courbes des probabilités de détection et de fausse alarme en fonction de $\omega$. Ces courbes sont présentées dans la figure 3.10. D'après ces figures, nous pouvons diviser les courbes en deux zones. Pour la probabilité de détection (courbe de la figure 3.10(a)), la zone 1 correspond à une pd élevée avec des $\omega$ de 3, 4 et 5, tandis que la zone 2 correspond à une probabilité moins élevée avec des $\omega$ de 6, 7 et 10. Pour la probabilité de fausse alarme (courbe de la figure 3.10(b)),

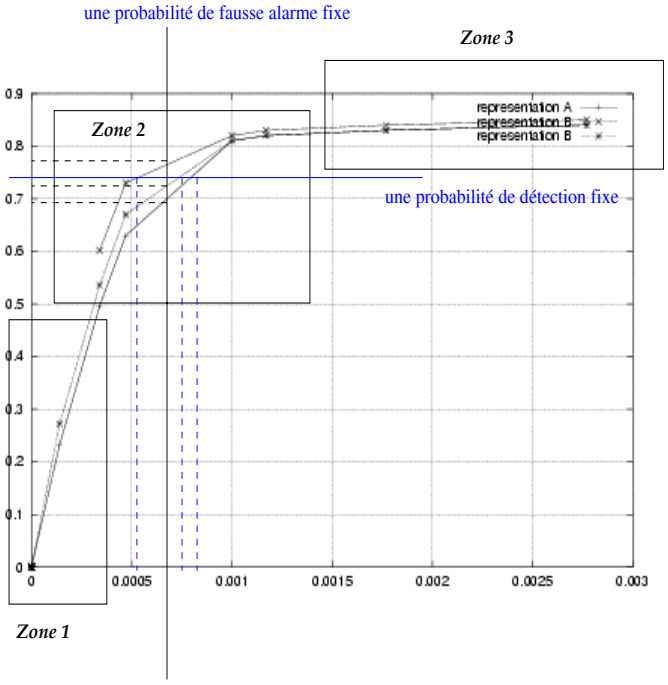



FIG. *3.9: Courbe opérationnelle de détection de contours par l'algorithme proposé utilisant les trois représentations A, B et C pour une longueur de chaîne donnée $\omega = 5$. Le seuil $\alpha$ pour la représentation A varie de 0 à 255, de 0 à 1 pour la représentation C. Le facteur $k$ varie de 1 à 1/7 pour la représentation B.*

la zone 1 correspond à une pf faible avec des $\omega$ de 3, 4 et 5, tandis que la zone 2 correspond à une probabilité très faible avec des $\omega$ de 6, 7 et 10. Les $\hat{pd}$ et $\hat{pf}$ d'une bonne valeur de $\omega$ doivent être respectivement dans la zone 1 de la courbe 3.10(a) et dans la zone 2 de la courbe 3.10(b). La seule valeur qui peut vérifier cette condition est la valeur 5. Par conséquent, la valeur de $\omega = 5$ correspond à un bon compromis pour l'image de synthèse.

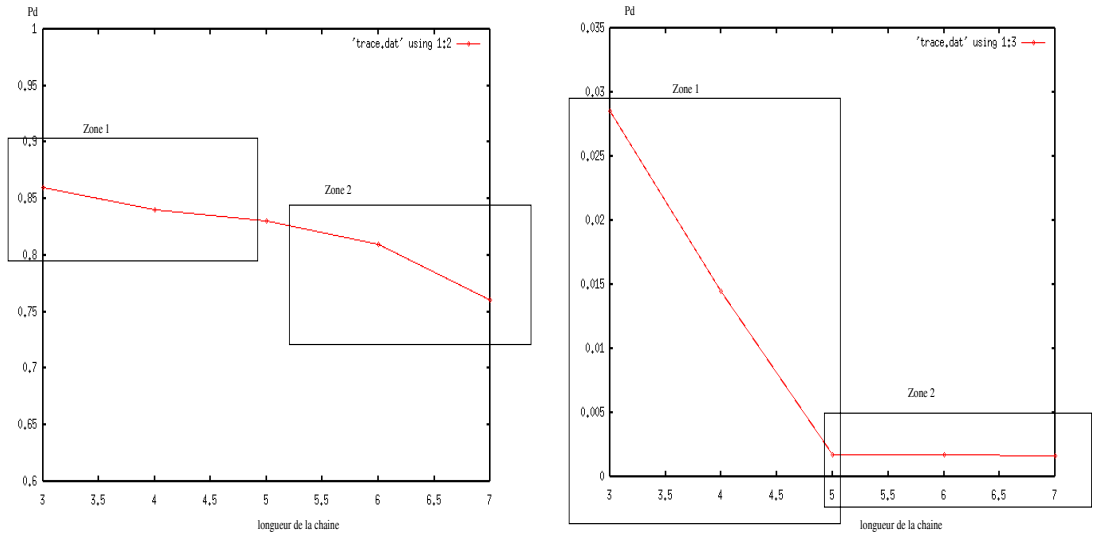
En conclusion, nous remarquons que les performances de détection de contours dépendent de la valeur de $\omega$ utilisée. En effet, plus $\omega$ diminue, plus nous détectons les faux contours de longueur faible et par conséquent plus la probabilité de fausse alarme augmente. Plus $\omega$ augmente, plus nous perdons les contours significatifs, et par conséquent plus la probabilité de détection diminue. L'algorithme de détection de contours en utilisant la représentation A ou C présente le même comportement vis à vis du changement de la longueur $\omega$ de la chaîne.

D'après l'évaluation des représentations, nous constatons que la représentation B est plus adéquate à la détection de contours. Cette représentation est plus performante en terme de détection de contours et d'aptitude à ne pas introduire de faux contours pour l'image de synthèse pour un facteur $k = 0.5$ et une longueur de chaîne $\omega = 5$. Dans la suite, nous allons comparer ces résultats de détection de contours utilisant ces paramètres "optimaux" avec les algorithmes de Canny, SUSAN, Deriche et Shen et Castan. Pour rendre la comparaison des résultats compatible entre l'algorithme proposé et les autres approches, nous cherchons tout d'abord les paramètres donnant les meilleurs résultats en terme de $\hat{pd}$ et $\hat{pf}$ pour chaque algorithme : Canny, SUSAN, Deriche et Shen et Castan.

Les résultats des comparaisons sont présentés dans la table 3.2. Les valeurs des paramètres des filtres associés sont présentées dans la table 3.3. Si nous regardons uniquement la probabilité de détection, nous remarquons que les algorithmes de Canny et proposé offrent une bonne valeur

| Représentation B à seuil $\alpha$ dynamique ($k = 0.5, \beta = 1$) | | |
|---|---|---|
| $\omega$ | $\hat{pd}$ | $\hat{pf}$ |
| 3 | 0.86 | 0.0285 |
| 4 | 0.84 | 0.0145 |
| 5 | 0.83 | 0.0017 |
| 6 | 0.81 | 0.0017 |
| 7 | 0.76 | 0.0015 |
| 10 | 0.57 | 0.0012 |

TAB. *3.1: Probabilités de détection et de fausse alarme en fonction de la longueur de la chaîne $\omega$.*




(a) $\hat{pd} = f(\omega)$ (b) $\hat{pf} = f(\omega)$

FIG. *3.10: Les probabilités de détection et de fausse alarme en fonction de la longueur de la chaîne $\omega$.*

de détection $\hat{pd}$ suivis par l'algorithme de Susan, Deriche et Shen et Castan. En regardant cette fois la probabilité de fausse alarme, nous constatons que le meilleur algorithme est celui de Shen et Castan. En effet, l'algorithme de Shen et Castan et aussi celui de Deriche détectent moins les contours significatifs mais avec moins de faux contours. Selon ces deux probabilités, nous pouvons regrouper ces algorithmes en deux catégories. La première catégorie contient les algorithmes de Canny, SUSAN. La deuxième catégorie, regroupe Deriche et Shen et Castan. En conclusion, les algorithmes de la première catégorie détectent plus de contours significatifs avec plus de faux contours, ce qui augmente la probabilité de fausse alarme. Tandis que les algorithmes de la deuxième catégorie détectent moins de contours significatifs mais avec moins aussi de faux contours. L'algorithme proposé est situé dans la première catégorie des algorithmes de Canny et SUSAN en terme de $\hat{pd}$ et dans la deuxième catégorie des algorithmes de Deriche et Shen et Castan en terme de $\hat{pf}$. Autrement dit, il est performant en terme de détection de contours sans pour autant introduire plus de faux contours que les méthodes les plus efficaces en ce sens.

| Algorithme | $\hat{pd}$ | $\hat{pf}$ |
|---|---|---|
| proposé (représentation B) | 0.83 | 0.0017 |
| Canny | 0.83 | 0.0965 |
| SUSAN | 0.74 | 0.0417 |
| Deriche | 0.66 | 0.0016 |
| Shen et Castan | 0.61 | 0.0006 |

TAB. *3.2: Probabilités de détection et de fausse alarme des algorithmes de Canny, Susan, Deriche, Shen et Castan et l'algorithme proposé avec la représentation B. Les paramètres des algorithmes sont illustrés dans la table 3.3.*

| Algorithme | Paramètres |
|---|---|
| proposé (représentation B) | $\omega = 5$, $k = 0.5$ |
| Canny | $\sigma = 1.20$, $t_b = 0.20$ $t_h = 0.80$ |
| SUSAN | $BT = 2$, $DT = 4$ |
| Deriche | $\tau = 1.0$, $t_b = 50$ $t_h = 200$ |
| Shen et Castan | $\gamma = 1.5$, $t_b = 25$ $t_h = 200$ |

TAB. *3.3: Paramètres des algorithmes de détection de contours.*

#### 3.6.3.3 Evaluation sur des images réelles

Dans [HSSB97] Heath et al. ont proposé une stratégie de comparaison des performances des algorithmes de détection de contours. La méthode proposée dans cet article se base sur une comparaison visuelle. Les auteurs proposent d'évaluer les détecteurs de contours à l'aide d'expérimentations psychovisuelles pour une tâche de reconnaissance d'objets dans une scène. Autrement dit, les performances d'un détecteur de contours sont liées à l'aptitude pour un expérimentateur à reconnaître l'objet dans la scène.

Les performances de l'algorithme de détection de contours dépendent de plusieurs facteurs. Les plus importants de ces facteurs sont :

1. **Le type d'image utilisé.** Heath et al. ont collecté une base d'image à niveaux de gris de taille $512 \times 512$. Ces images contiennent des objets qui peuvent être reconnus et nommés par les observateurs. Dans la figure 3.11, nous présentons trois images de cette base qui

seront utilisées aussi dans notre comparaison des performances de l'algorithme proposé et des algorithmes existants (Canny, Deriche, Shen et Castan et SUSAN).

Pneu œuf Panier

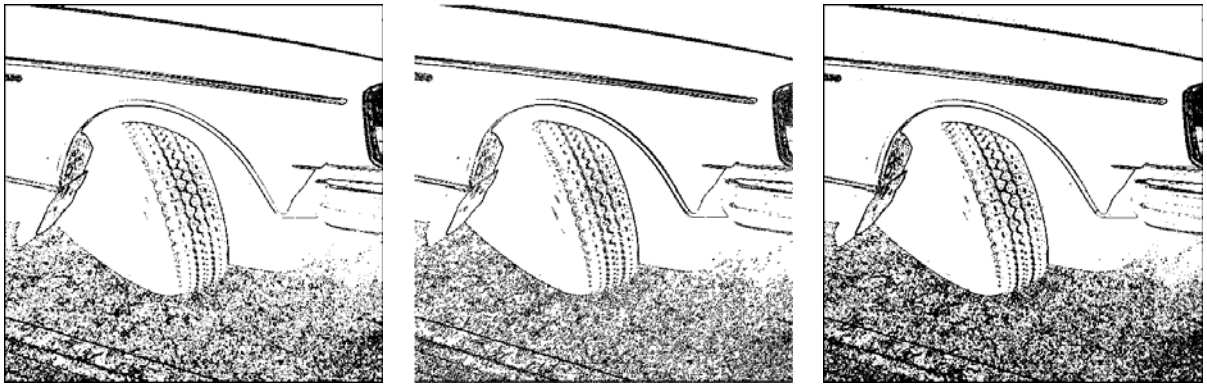
FIG. *3.11: Images de Simulation.*

2. **Choix des paramètres de chaque algorithme de détection de contours.** La sélection des paramètres d'entrées de chaque algorithme est une étape très importante dans l'évaluation des performances, parce que les résultats de la qualité de la détection varient avec le choix de ces paramètres. Dans [HSSB97], Heath et al. ont choisi premièrement 64 jeux de paramètres. Ces paramètres sont choisis de façon à produire de bonnes cartes de contours. Un observateur visualise ces cartes de contours afin de choisir 12 cartes. L'objectif de l'expérimentation est de trouver un seul jeu de paramètre pour chaque algorithme. Afin d'atteindre cet objectif, Heath et al. ont mené une expérimentation d'évaluation visuelle sur ces 12 jeux de paramètres pour chacun des détecteurs étudiés. Neuf étudiants volontaires d'une classe de traitements d'images ont participé à une expérimentation qui consistent à attribuer une note à chacune des cartes de contours en disposant de l'image réelle.

En utilisant ces images réelles et cette méthodologie des algorithmes de détection de contours de Canny, Nalwa [NB86], Inverson [IZ95], Bergholm [Ber87c] et Rothwell [RMHN95] , Heath et al. ont montré que l'algorithme de Canny est plus performant en détection de contours pour la tâche de reconnaissance.

Avant de comparer l'algorithme proposé avec les algorithmes de Canny, Deriche, Shen et Castan et SUSAN, nous commençons tout d'abord par une évaluation des représentations A, B et C sur des images réelles. Les images de simulations sont présentées dans la figure 3.11. Rappelons que ces trois représentations ont donné à une différence près les mêmes résultats de détection de contours sur l'image de synthèse en terme de probabilités de détection et de fausse alarme. Dans la figure 3.12, nous présentons les cartes de contours de l'image Pneu obtenues par l'algorithme proposé en utilisant les trois représentations. Les cartes sont générées par un ordre de voisinage $\beta = 1$ et une longueur de chaîne $\omega = 5$. Le seuil $\alpha$ des représentations A et C est choisi de façon à obtenir une 'bonne' carte de contours avec moins de faux contours et plus de contours significatifs. En examinant les cartes de contours, nous constatons que les trois représentations A, B et C donnent approximativement les mêmes cartes de contours. En effet, ceci confirme les faibles différences en terme de probabilités de détection $\hat{pd}$ et de fausse alarme $\hat{pf}$ observées sur l'image de synthèse.

Nous montrons dans les figures 3.13 3.14 et 3.15 les cartes de contours des algorithmes de Canny, Shen et Castan, SUSAN, Deriche et l'algorithme proposé. Ce dernier utilise la représentation B. Les sorties de l'algorithme de Canny sont générées avec les paramètres optimaux donnés



(a) ($\alpha = 20$, $\omega = 5$, $\beta = 1$) (b) ($\alpha = 0.5\hat{\sigma}$, $\omega = 5$, $\beta = 1$) (c) ($\alpha = 0.92$, $\omega = 5$, $\beta = 1$)

FIG. *3.12: Cartes de contours de l'algorithme proposé en utilisant les représentations A (a), B (b) et C (c).*

dans [HSSB97]. Pour les autres algorithmes, nous avons mené une série d'expérimentations afin de trouver les meilleurs paramètres de chaque algorithme. Ces paramètres correspondent aux meilleures cartes de contours avec plus de contours significatifs et moins de faux contours. La table 3.4 illustre les paramètres utilisés pour les différents algorithmes.

D'après les cartes de contours des figures 3.13 3.14 et 3.15, nous constatons que les cinq algorithmes produisent des cartes de contour significatives. Ils permettent tous de détecter les objets présents dans les images. N'importe quel observateur peut reconnaître ces objets, Pneu d'une voiture, œuf et Panier.

Dans la figure 3.13, nous voyons que l'algorithme de Canny détecte plus de contours significatifs avec plus de faux contours. La même remarque est valable aussi pour l'algorithme SUSAN, mais avec moins de faux contours. Tandis que les algorithmes de Deriche, Shen et Castan, détectent moins de faux contours mais avec moins aussi de contours significatifs. Pour l'algorithme proposé, nous remarquons qu'il détecte plus de contours significatifs avec moins de faux contours par rapport aux autres algorithmes. Pour l'image 3.13, le meilleur détecteur de contours est l'algorithme proposé. Tandis que pour l'image de la figure 3.14, l'algorithme proposé et l'algorithme de Deriche et Shen et Castan et SUSAN donnent approximativement la même carte de contours. La figure 3.15, contient des régions homogènes et texturées. D'après les résultats de ces cartes de contours de différents algorithmes, nous constatons que les algorithmes de Deriche et Shen et Castan sont les moins sensibles aux textures que les algorithmes de Canny, Susan et proposé. Néanmoins, les contours présentés par ces algorithmes dans le panier sont moins détectés. Les algorithmes Susan et proposé ont pu détecter les contours dans le panier de l'image.

Il est difficile de comparer ces algorithmes visuellement, puisque les cinq algorithmes arrivent à détecter l'objet de l'image. Cette détection a nécessité une étape de prétraitement qui correspond aux choix des paramètres de chaque algorithme. En examinant ces paramètres dans la table 3.4, nous constatons que l'algorithme proposé avec les mêmes paramètres a détecté les objets avec la même qualité que les autres algorithmes. Et par conséquent, l'algorithme proposé ne nécessite par une présélection des paramètres de détection. Autrement dit, cet algorithme est robuste vis à vis de la variabilité des images.

Nous avons testé l'algorithme proposé sur différents types d'images, et dans chaque cas, nous obtenons des résultats satisfaisants de détection de contours. Un autre algorithme est proposé, c'est l'algorithme de détection de contours dans une image couleur. Cet algorithme fait l'objet du paragraphe suivant et a beaucoup de points communs avec l'algorithme précédemment décrit.

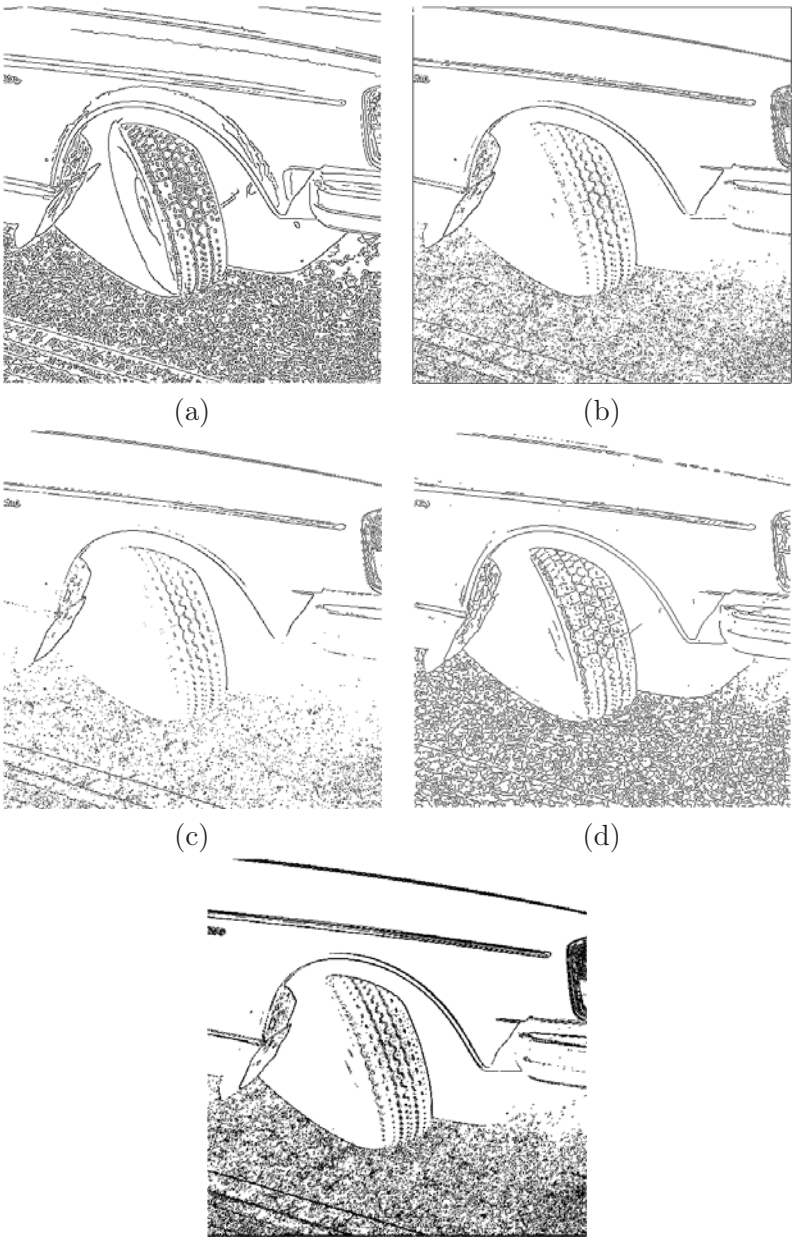



FIG. *3.13: Carte de contours des algorithmes : (a) Canny, (b) Deriche, (c) Shen et Castan, (d) SUSAN et (e) Proposé pour l'image Pneu.*

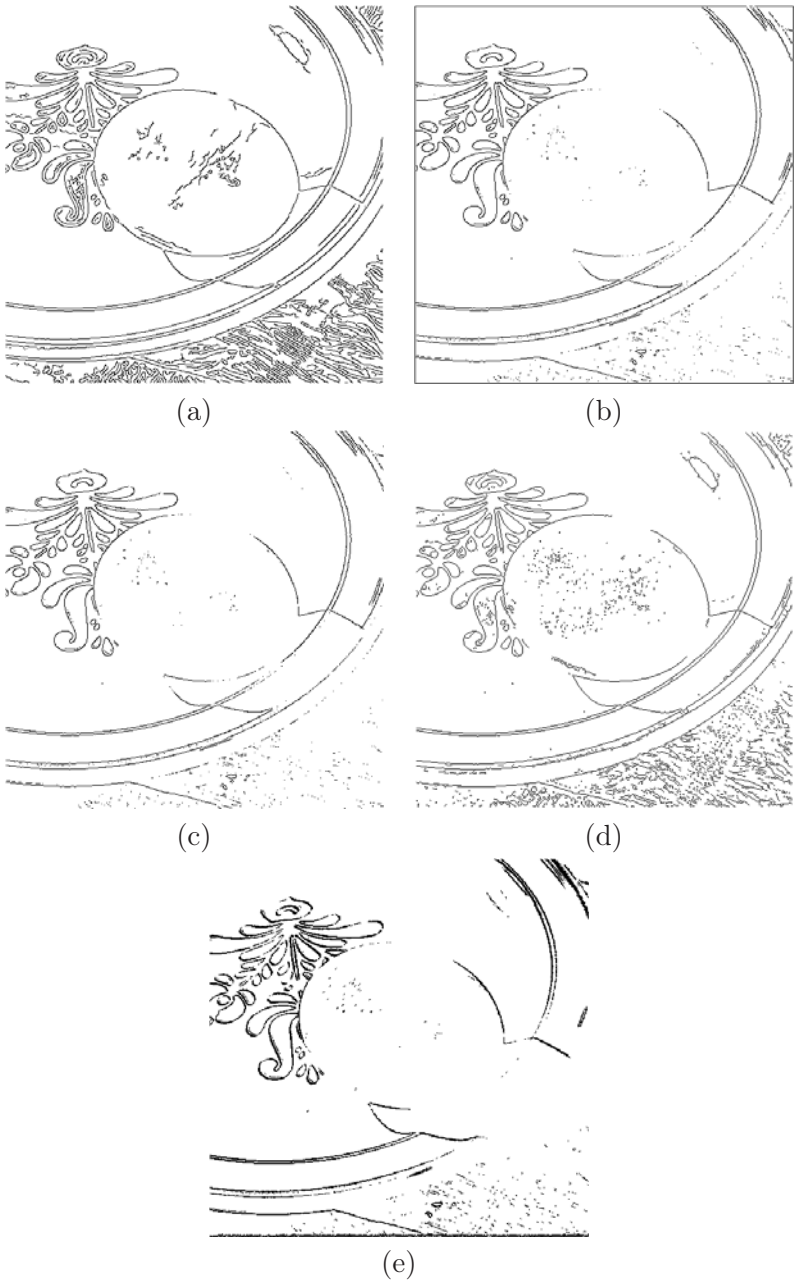



FIG. *3.14: Carte de contours des algorithmes : (a) Canny, (b) Deriche, (c) Shen et Castan, (d) SUSAN et (e) Proposé pour l'image œuf .*

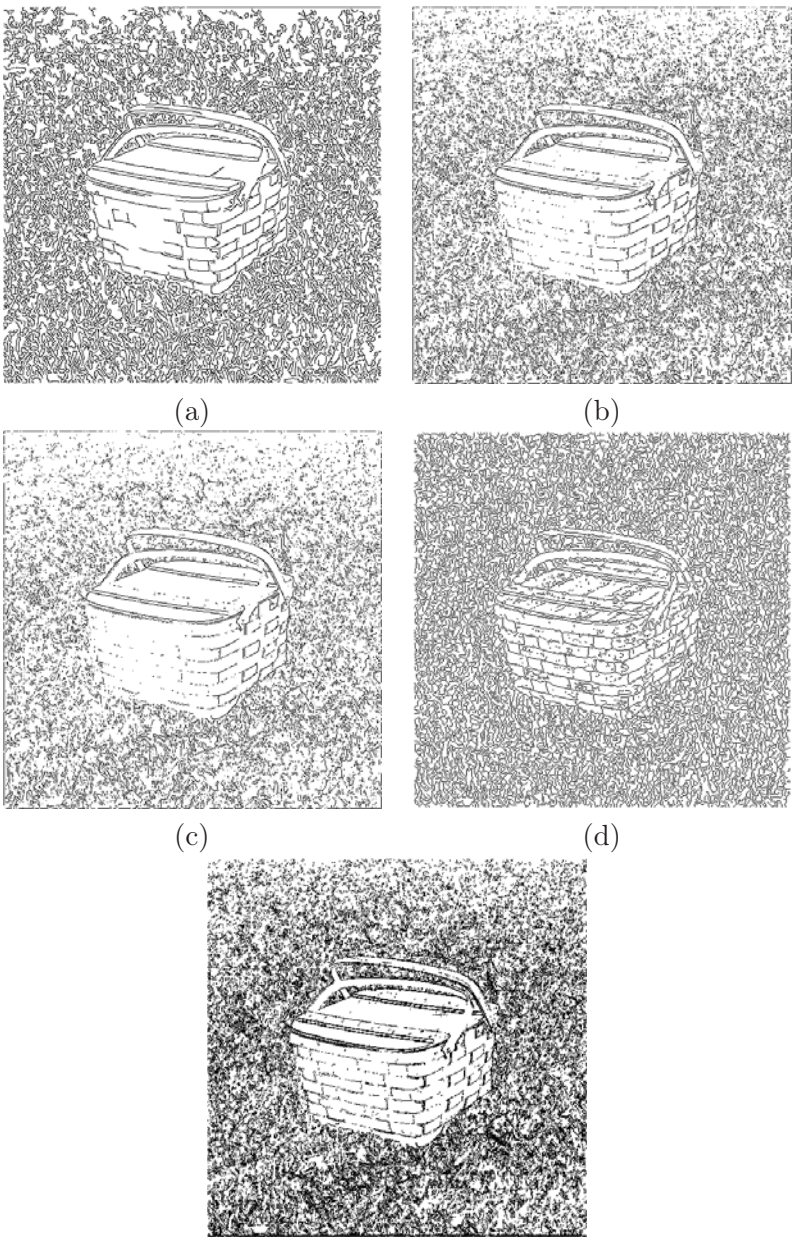



FIG. *3.15: Carte de contours des algorithmes : (a) Canny, (b) Deriche, (c) Shen et Castan, (d) SUSAN et (e) Proposé pour l'image Panier.*

| - | Algorithme de détection de contours | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Image | Canny | | | Deriche | | | Shen et Castan | | | SUSAN | | Proposé | |
| - | $\sigma$ | $t_b$ | $t_h$ | $\tau$ | $t_b$ | $t_h$ | $\gamma$ | $t_b$ | $t_h$ | BT | DT | $\omega$ | k |
| Pneu | 1.20 | 0.20 | 0.60 | 10 | 25 | 200 | 3 | 35 | 250 | 10 | 4 | 5 | 0.5 |
| œuf | 1.20 | 0.40 | 0.80 | 5 | 15 | 250 | 5 | 20 | 250 | 20 | 4 | 5 | 0.5 |
| Panier | 1.20 | 0.40 | 0.60 | 3 | 20 | 150 | 10 | 30 | 150 | 15 | 4 | 5 | 0.5 |

TAB. *3.4: Paramètres des algorithmes de détection de contours de Canny, Susan, Deriche, Shen et Castan et l'algorithme proposé avec la représentation B pour les trois images de comparaison.*

## 3.7 Détection de contours dans une image couleur

En imagerie mono-composante, il existe de nombreuses manières d'aborder le problème de la détermination des contours d'une image. Dans beaucoup de cas, on retrouve un schéma générique de détection marginale des contours qui fait apparaître deux étapes. La première étape cherche à mettre en relief les zones correspondant aux contours recherchés et fournit pour chaque pixel une information d'amplitude du contraste (a), traduisant l'importance de la transition au voisinage de la frontière, et une information de direction (b) correspondant à l'orientation de la frontière. La deuxième étape consiste à exploiter ces deux informations pour détecter les contours binaires de l'image, c'est-à-dire des lignes d'une épaisseur d'un pixel. La figure 3.16 présente ce schéma général :

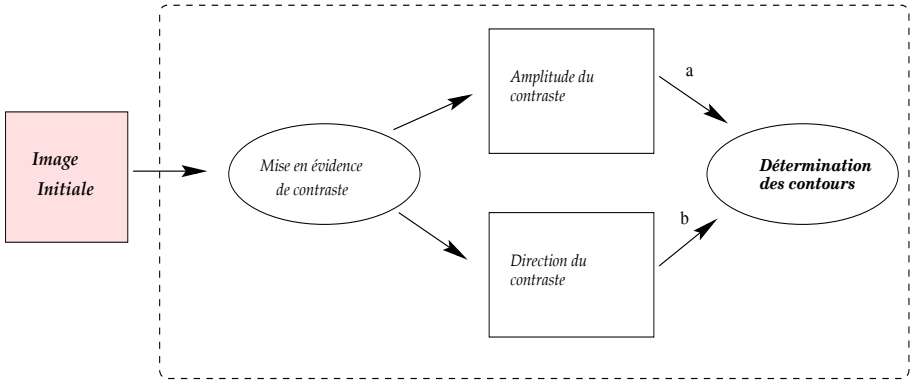



FIG. *3.16: Schéma de principe du détecteur de contours marginal.*

Dans le cas des images couleur, beaucoup des méthodes proposées reprennent cette stratégie en deux temps. Toutefois, selon l'endroit où va être introduite la fusion entre les différentes composantes, différentes structures sont envisageables.

### 3.7.1 Approche marginale

La première stratégie marginale de détection de contours d'une image couleur consiste à repousser l'étape de fusion le plus loin possible, c'est-à-dire après avoir obtenu une carte des contours sur chaque composante (Fig. 3.17).

La deuxième stratégie utilise encore une détection de contraste marginale. Mais la fusion est réalisée sur les informations de contraste, et le détecteur de contours opère sur le gradient multicomposante (Fig. 3.18).

Une image est considérée comme la discrétisation d'une fonction $f$ de $\mathbb{R}^2$ dans $\mathbb{R}^3$. Nous avons alors :

$$f\left(\begin{array}{lcl} \mathbb{R}^2 & \rightarrow & \mathbb{R}^3 \\ (x,y) & \mapsto & (f_1(x,y), f_2(x,y), f_3(x,y)) \end{array}\right) \tag{3.19}$$

Le gradient multi-composantes est la composition des gradients de chaque plan. Une manière simple de composer les gradients marginaux consiste à ajouter leur module carré.

$$|G_{f_1}(x,y)| = \sqrt{(G_x^{f1})^2 + (G_y^{f1})^2} \tag{3.20}$$

$$|G_{f_2}(x,y)| = \sqrt{(G_x^{f2})^2 + (G_y^{f2})^2} \tag{3.21}$$

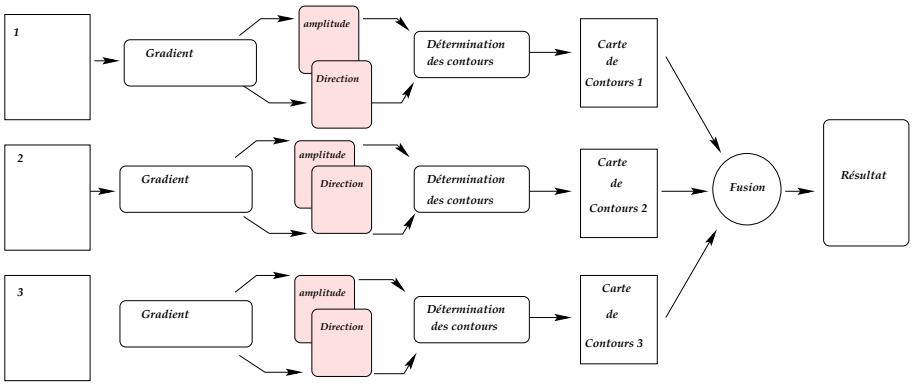



FIG. *3.17: Approche marginale par gradient + fusion sur les cartes de contours de chaque composante.*

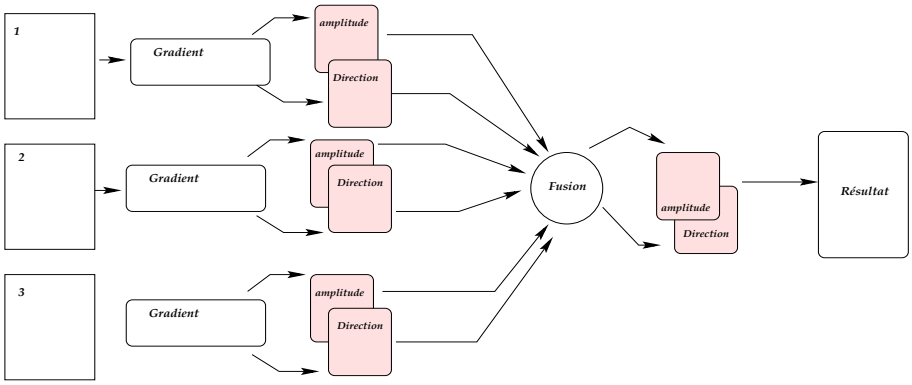



FIG. *3.18: Approche marginale par gradient + fusion des contours à partir du gradient multi-composante.*

$$|G_{f_3}(x,y)| = \sqrt{(G_x^{f3})^2 + (G_y^{f3})^2} \tag{3.22}$$

$$|G| = \sqrt{G_{f1}^2(x,y) + G_{f2}^2(x,y)G_{f3}^2(x,y)} \tag{3.23}$$

$\theta$ est la direction du plus fort gradient mesurée à $\pi$ près.

On applique la stratégie de recherche des contours à l'image gradient obtenue selon le schéma mono-composante. D'autres approches, relevant plus de la segmentation, consistent à garder la stratégie marginale jusqu'à l'obtention des contours, et à fusionner, de manière judicieuse, les différentes cartes de contours obtenues marginalement (Fig. 3.17).

### 3.7.2 Approche vectorielle

La différentielle première de $f$ en $p$ dans la direction $n$ s'écrit alors :

$$Df(p).(n_x,n_y) = \begin{pmatrix} \frac{\partial f_1}{\partial x}(p).n_x + \frac{\partial f_1}{\partial y}(p).n_y \\ \frac{\partial f_2}{\partial x}(p).n_x + \frac{\partial f_2}{\partial y}(p).n_y \\ \frac{\partial f_3}{\partial x}(p).n_x + \frac{\partial f_3}{\partial y}(p).n_y \end{pmatrix} \tag{3.24}$$

Comme dans le cas mono-dimensionnel, nous recherchons les fortes valeurs de la différentielle première. Cette différentielle étant cette fois-ci un vecteur de $\mathbb{R}^3$, nous calculons le carré de sa norme :

$$S(p,n) = \|Df(p).(n_x,n_y)\|^2 = En_x^2 + 2Fn_xn_y + Gn_y^2 \tag{3.25}$$

avec :

$$\begin{aligned} E &= \sum_{i=1}^{3} \left(\frac{\partial f_i}{\partial x}\right)^2 \\ F &= \sum_{i=1}^{3} \frac{\partial f_i}{\partial x}\frac{\partial f_i}{\partial y} \\ G &= \sum_{i=1}^{3} \left(\frac{\partial f_i}{\partial y}\right)^2 \end{aligned} \tag{3.26}$$

L'équation 3.25 nous donne la norme de la différentielle première dans la direction $n$. L'existence d'un contour en $p$ indépendamment de $n$ s'évalue en cherchant le maximum de l'équation 3.25 pour $n$ appartenant à la boule unité (seule la direction de $n$ nous intéresse). Les extrema de (Eq. 3.25) sont les valeurs propres de la matrice $\Gamma$ définie par :

$$\Gamma = \begin{pmatrix} E & F \\ F & G \end{pmatrix} \tag{3.27}$$

Ces extrema sont atteints lorsque $n = (cos(\theta_0), sin(\theta_0))$ représente la valeur propre correspondante suivante :

$$\theta_0 = \frac{1}{2}\arctan\left(\frac{2F}{E-G}\right) \tag{3.28}$$

Le maximum correspondant est égal à :

$$\lambda(x,y) = \frac{E+G+\sqrt{(E-G)^2+4F^2}}{2} = S(p,(\cos(\theta_0),\sin(\theta_0))) \tag{3.29}$$

La valeur $\lambda$ peut être vue comme l'extension du concept de gradient aux images couleurs. En effet, on constate facilement que dans le cas mono-dimensionnel, $\lambda$ est égal au carré de la norme du gradient. Il est donc naturel de considérer les passages par zéro de la différentielle de la fonction $S$, au point $p$, dans la direction $n$. Cette dernière valeur est donnée par l'expression suivante :

$$D_S(p).n = E_x(p)n_x^3 + (2F_x(p) + E_y(p))n_x^2 n_y + (G_x(p) + 2F_y(p))n_x n_y^2 + G_(p)n_y^3 \qquad (3.30)$$

où $E_x, E_y, F_x, F_y$ et $G_x, G_y$ représentent les dérivées partielles de $E, F$ et $G$ par rapport à $x$ et $y$. La valeur $D_S(p).n$ représente l'extension de l'opérateur laplacien aux images multi-dimensionnelles.

L'ensemble des techniques que nous venons de présenter semblent avoir été découvertes en parallèle par Di Zenzo [Zen86] et A. Cumani [Cum89] [Cum91] [CGG91]. Di Zenzo approxime l'image en chaque pixel par trois fonctions linéaires (une pour chaque composante).Ces approximations lui permettent de calculer les dérivées partielles et d'en déduire la valeur de $\lambda$ et l'angle pour lequel le maximum est réalisé. A. Cumani donne un ensemble d'outils théoriques permettant d'exploiter les passages par zéro de la fonction $D_S(p)$.

#### 3.7.2.1 Modélisation des hyperarêtes de contours dans une image couleur

Dans cette section, nous développons l'algorithme de détection de contours d'une image couleur. De la même façon que l'algorithme de débruitage, l'algorithme de détection de contours d'une image couleur peut être appliqué en approches marginale ou vectorielle. L'approche marginale utilise souvent une fusion après détection des contours sur chaque composante. Dans cette section, nous présentons la méthode proposée de détection de contours utilisant une approche vectorielle.

En effet, l'algorithme de détection des contours en utilisant la représentation vectorielle par hypergraphe de voisinage spatiocolorimétrique ainsi que le modèle des hyperarêtes de contours comporte trois étapes :

- ☐ génération de la représentation vectorielle par hypergraphe de voisinage spatiocolorimétrique.
- ☐ détection des hyperarêtes non isolées et isolées.
- ☐ classification de ces hyperarêtes, en contours ou non contours.

Les hyperarêtes de contours sont des hyperarêtes non isolées ou isolées formant une chaîne avec le voisinage ouvert. Le bloc diagramme ci-dessous illustre les grandes étapes de détection des contours en utilisant une représentation vectorielle HVSC.

#### 3.7.2.2 Résultats expérimentaux

Afin de valider l'algorithme de détection des contours, nous ne nous sommes pas fixés un type d'images précis. Nous avons considéré des images provenant de diverses banques d'images et en particulier celles de GdR-PRC ISIS. Ces images sont de taille $256 \times 256$ ou $512 \times 512$ pixels. Les résultats sont obtenus aussi bien sur des images réelles que sur des images synthétiques. Avant de présenter les résultats, nous présentons dans la figure 3.20 les images de simulation. Les images 3.20(a) et (b) représentent les images de synthèse. Tandis que les images 3.20(c) et (d) représentent les images réelles. L'objectif des images synthétiques est de tester la robustesse de l'algorithme vis-à-vis de la détection des petits contours tels que jonctions, ligne et marche.

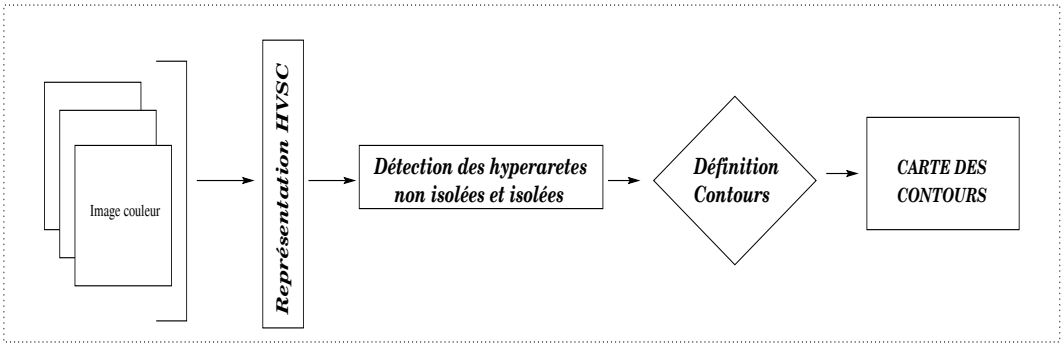
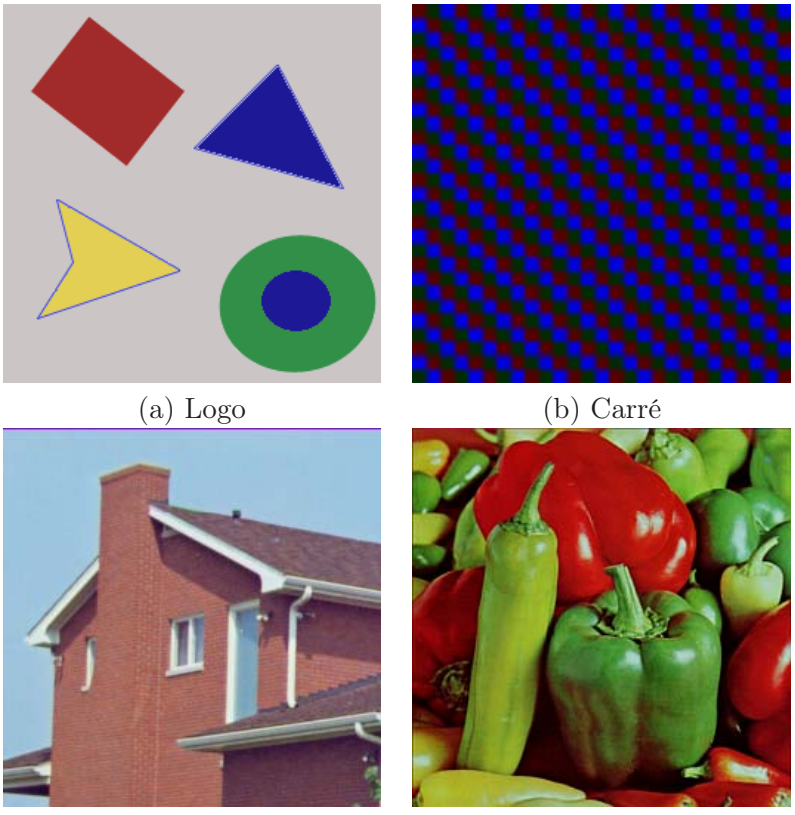



FIG. *3.19: Bloc diagramme de détection de contours par l'approche proposée.*



(a) Logo (b) Carré

(c) Maison (d) Poivron

FIG. *3.20: Images de simulation.*

**Comparaison des espaces couleur RVB et CIELab.** Comme ci-déjà mentionné dans les chapitres précédents, l'approche proposée peut être appliquée dans n'importe quel espace de couleur que ce soit RVB, CIELab ou autres, à condition que l'espace de représentation de la couleur possède une distance colorimétrique bien définie. Afin de trouver l'espace couleur adéquat entre RVB et CIELab pour l'approche vectorielle, nous donnons dans la figure 3.21, les sorties de l'algorithme associé à ces deux espaces couleur. La comparaison est réalisée en utilisant la représentation vectorielle par hypergraphe de voisinage à seuil spatiocolorimétrique global $\alpha$ (représentation A). La longueur de la chaîne est $\omega = 5$ et l'ordre de voisinage est $\beta = 1$. L'image couleur utilisée est l'image Poivron.

D'après cette figure, nous constatons que la valeur $\alpha = 40$ dans l'espace couleur RVB donne plus de contours significatifs avec moins de faux contours par rapport aux autres cartes de contours dans le même espace. Ainsi une valeur de $\alpha = 30$ dans l'espace CIELab donne de bons résultats par rapport aux cartes dans le même espace. En prenant les meilleures cartes des deux espaces, c'est à dire $\alpha = 40$ pour l'espace RVB et $\alpha = 30$ pour l'espace CIELab, nous constatons que l'approche vectorielle dans l'espace couleur CIELab est plus performante que dans l'espace couleur RVB. Ces résultats sont justifiés par l'utilisation de la distance euclidienne dans un espace uniforme.

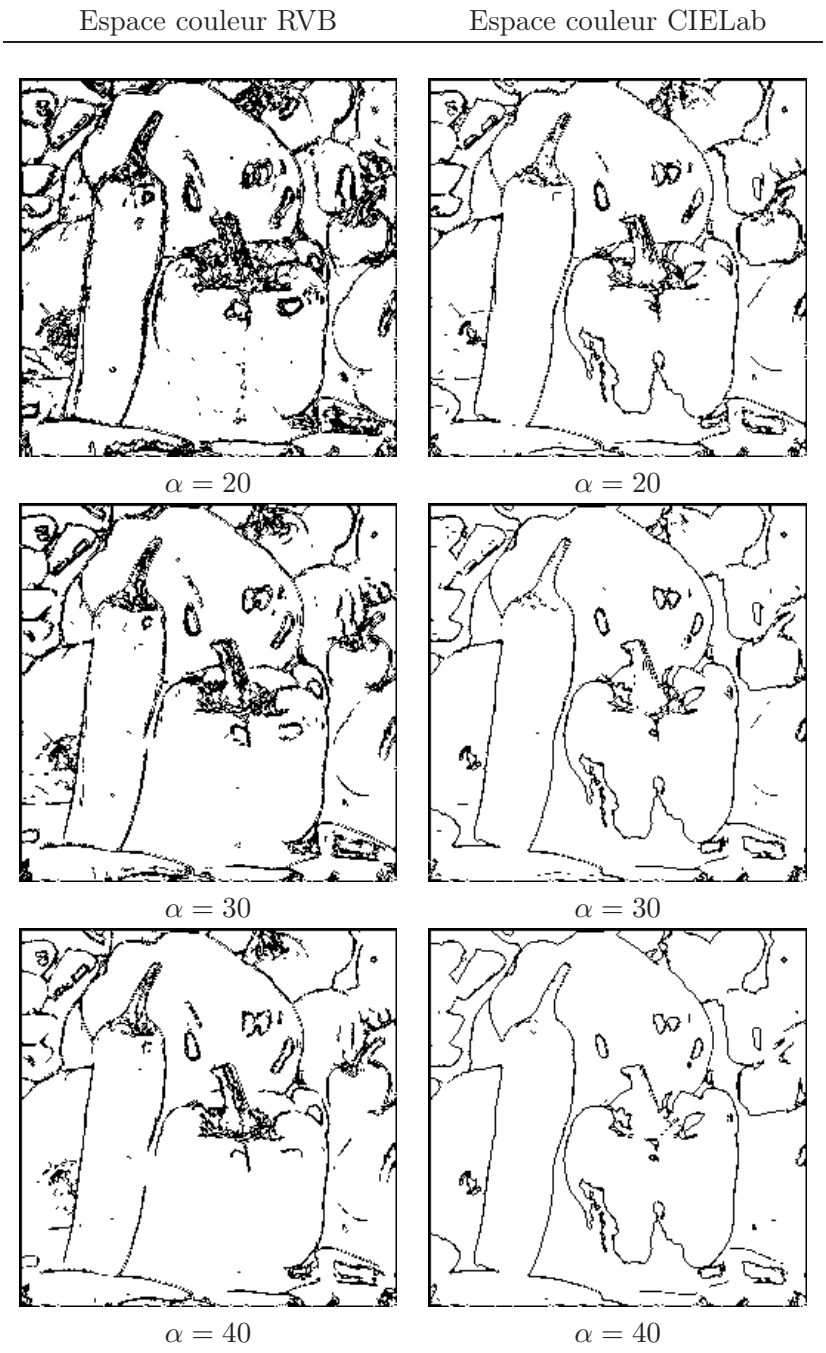



FIG. *3.21: Comparaison des résultats de la détection de contours par l'algorithme proposé (représentation A avec les paramètres $\omega = 5$ et $\beta = 1$) dans les espaces couleur RVB et CIELab.*

**Approche vectorielle de détection des contours dans l'espace couleur CIELab.** Les deux figures 3.22 et 3.23 représentent respectivement les résultats de détection de contours des images de synthèse et réelles. Nous pouvons remarquer que pour ces images, de bons résultats de détection de contours sont obtenus. La plupart des contours de l'image originale visibles à l'œil sont détectés de manière satisfaisante. Les contours finaux correspondent bien aux discontinuités visibles dans l'image originale, et la majorité d'entre eux sont extraits. On peut également noter leur bonne localisation et la préservation des structures fines.

Pour la plupart des images de simulation, on constate que la valeur de $\alpha$ conduisant à des résultats satisfaisants est compris dans l'intervalle $[20, 30]$ pour les images naturelles, tandis que le seuil $\alpha$ pour les images synthétiques est compris dans l'intervalle $[2, 10]$.

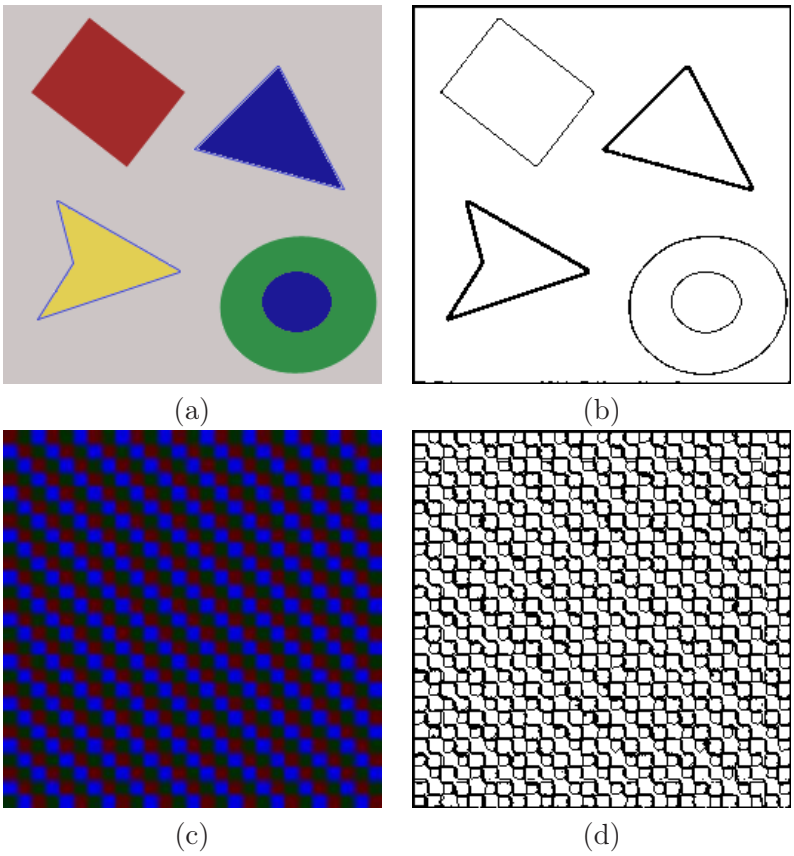
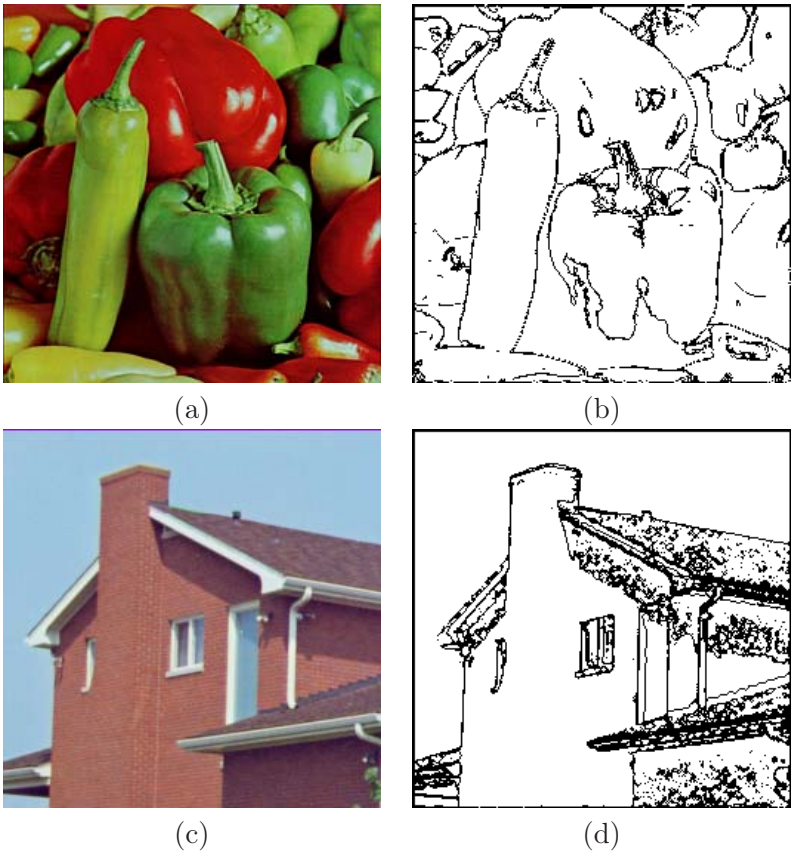



FIG. *3.22: Illustration des définitions des zones contours par l'algorithme vectoriel proposé dans l'espace CIELab en utilisant la représentation A ($\alpha = 2$, $\omega = 5$, $\beta = 1$) sur des images synthétiques.*

Jusqu'à présent, nous avons présenté les résultats de la détection des contours en appliquant un paramètre $\alpha$ global de la représentation HVSC. En effet, l'approche vectorielle dans l'espace couleur CIELab dépend de la valeur de $\alpha$. L'augmentation de la valeur $\alpha$ entraîne une perte des contours significatifs. Inversement, sa diminution, entraîne l'apparition de faux contours. Pour pallier aux problèmes de réglage de la valeur $\alpha$, nous utilisons la représentation A avec un seuil global estimé selon l'équation 2.34. Dans la figure 3.24, nous présentons les cartes de contours des images Poivron, Logo, Carré et Maison. Ces cartes de contours sont générées par l'algorithme proposé en utilisant la représentation A avec un seuil $\alpha$ estimé. D'après ces figures, nous constatons que l'algorithme avec un seuil estimé donne approximativement les mêmes résultats par rapport à un seuil $\alpha$ fixé expérimentalement.



(a) (b)

(c) (d)

FIG. *3.23: Illustration des définitions des zones contours par l'algorithme vectoriel proposé dans l'espace CIELab en utilisant la représentation A ($\alpha = 20$, $\omega = 5$, $\beta = 1$) sur des images réelles.*

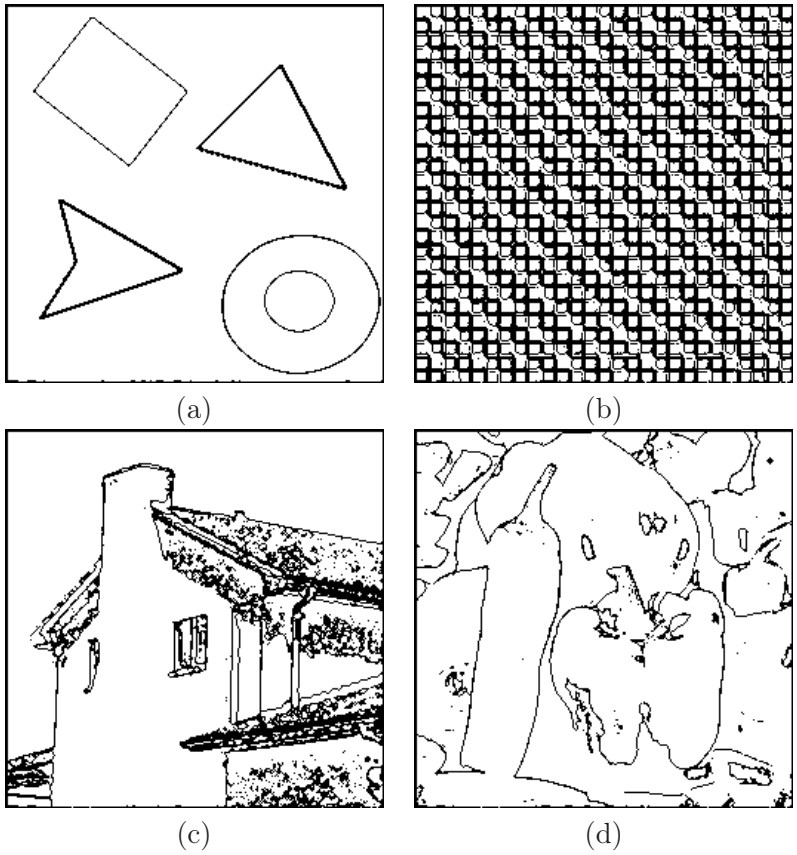



FIG. *3.24: Illustration des définitions des zones contours par l'algorithme vectoriel proposé dans l'espace CIELab en utilisant la représentation A avec un seuil $\alpha$ estimé selon l'équation 2.34.*

**Comparaison avec les algorithmes de Canny marginal et Cumani.** Dans cette section, nous utilisons la même méthodologie de comparaison que précédemment pour évaluer les détecteurs de contours dans des images couleur. Nous cherchons d'abord les paramètres des algorithmes de Canny marginal utilisant la première stratégie marginale décrite précédemment et Cumani vectoriel qui donnent visuellement les meilleures cartes de contours. Ensuite, nous comparons ces cartes avec celles obtenues avec l'algorithme proposé.

Dans les figures 3.25 et 3.26, nous présentons les sorties de l'algorithme proposé ainsi que les algorithmes de Canny marginal et Cumani. L'algorithme proposé dans ce cas utilise la représentation A dans l'espace couleur CIELab avec un seuil spatiocolorimétrique global estimé selon l'équation 2.34. L'ordre de voisinage est égal à 1. La longueur de la chaîne est égale à 5. Les cartes de contours sont générées à partir d'images de synthèse et d'images réelles.

Pour les images de synthèse, nous constatons que l'algorithme de Canny marginal, l'algorithme de Cumani et l'algorithme proposé détectent approximativement les mêmes contours significatifs. Néanmoins, les algorithmes de Canny marginal et Cumani ont donné quelques faux contours sur l'image de synthèse 3.25. Sur les deux images de synthèse, nous constatons que l'algorithme de Cumani ne détecte pas les jonctions de l'image. Ceci a entraîné une modification de la forme carrée des contours de l'image de synthèse 3.25.

Pour les images réelles, nous constatons que l'algorithme de Canny marginal introduit des faux contours sur les deux images Poivron et Maison. Ceci est valable pour l'algorithme de Cumani, mais seulement pour l'image Poivron. La carte de contours de l'image Maison est la meilleure carte des trois algorithmes. Contrairement à l'algorithme de Cumani, l'algorithme proposé a introduit des faux contours dans l'image Maison. En conclusion, l'approche marginale de détection de contour de Canny sur des images réelles, détecte plus de faux contours. Les approches vectorielles, proposée ou de Cumani, détectent approximativement les mêmes cartes de contours. La seule différence des deux derniers algorithmes est située au niveau de la détection des jonctions. Ces derniers sont bien détectés par l'algorithme proposé.

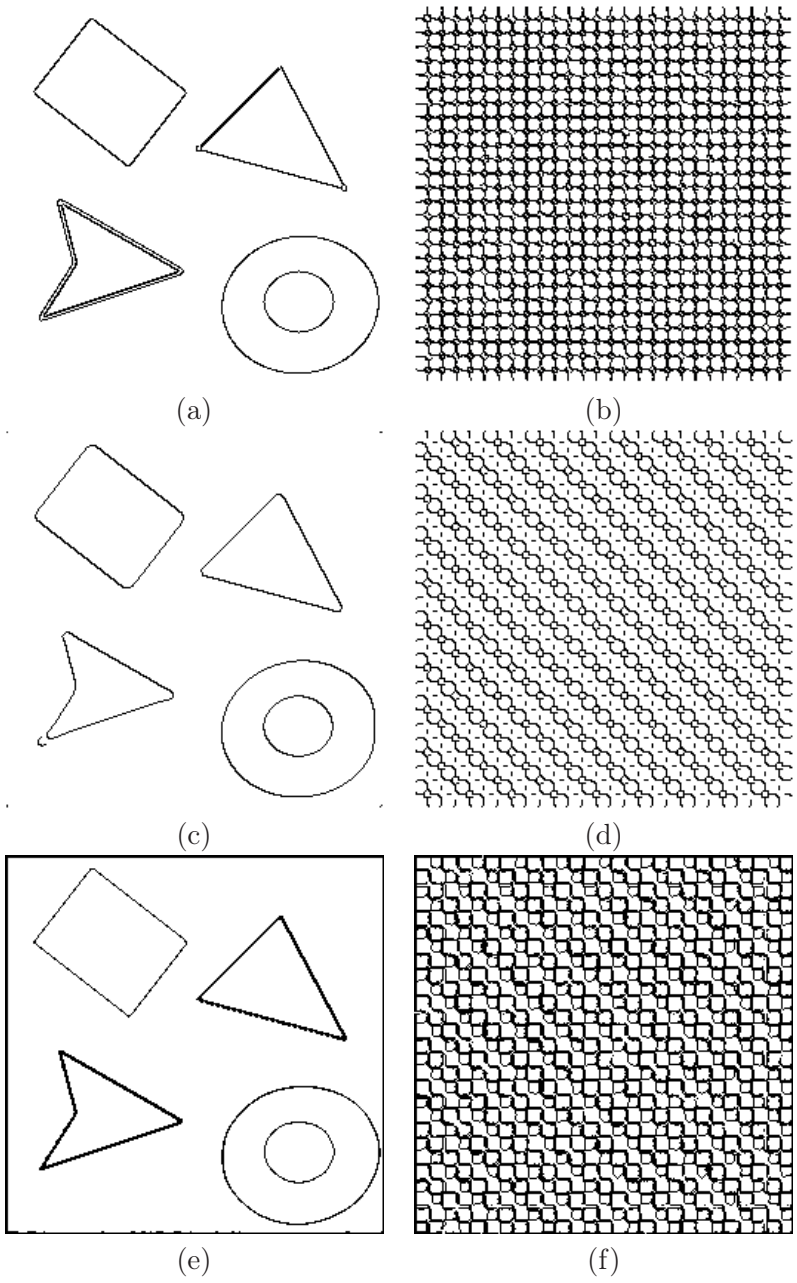



FIG. *3.25: Les zones de contours détectées par les algorithmes de Canny marginal (a,b), Cumani (c,d) et proposé (e,f). (a,b) avec les paramètres* $\sigma = 1$, $t_b = 0.30$ *et* $t_h = 0.60$. *(c) avec les paramètres* $\sigma = 3$ , *Seuil* $= 2$. *(d) avec les paramètres* $\sigma = 2$ , *Seuil* $= 10$. *(e,f) avec les paramètres* $\omega = 5$ $\beta = 1$.

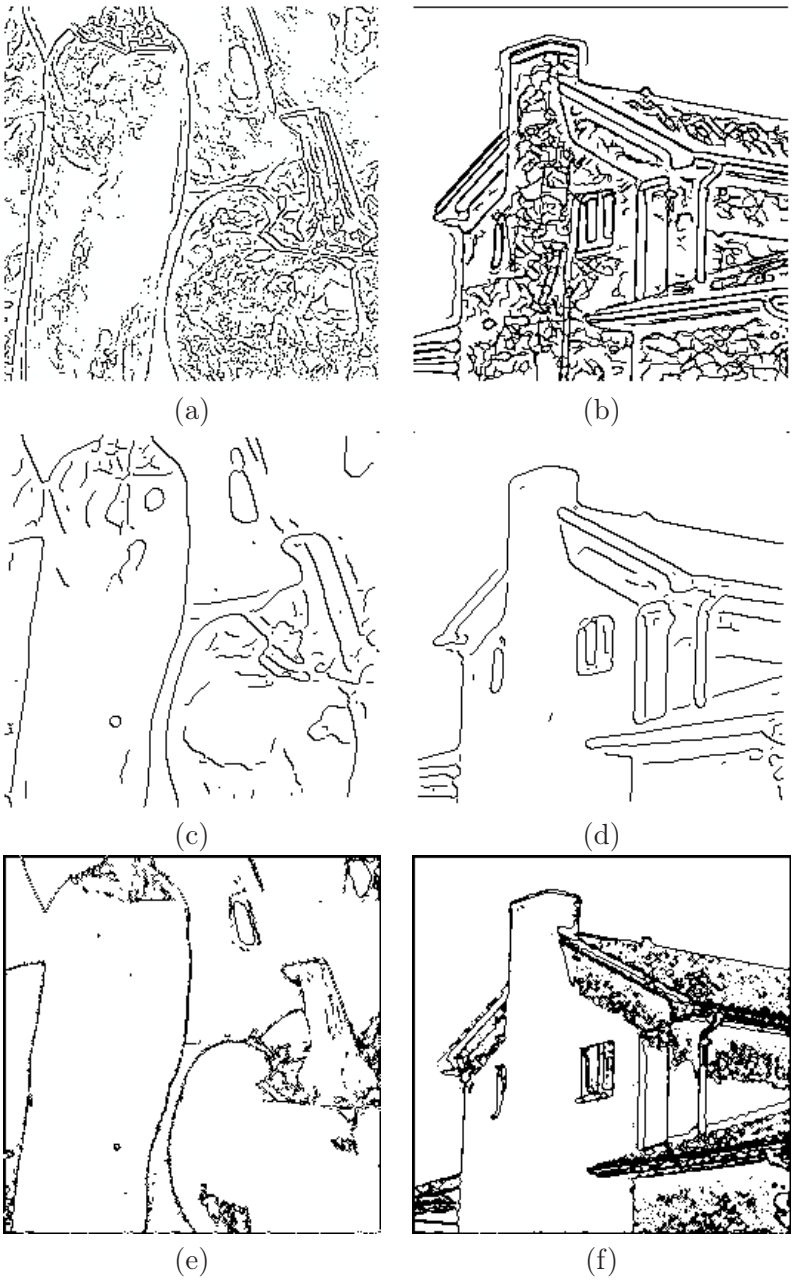



FIG. *3.26: Les zones de contours détectées par les algorithmes de Canny marginal (a,b), Cumani (b,c) et proposé (e,f). (a,b) avec les paramètres* $\sigma = 2$, $t_b = 0.30$ *et* $t_h = 0.60$. *(c) avec les paramètres* $\sigma = 3$ , *Seuil* $= 10$. *(d) avec les paramètres* $\sigma = 3$ , *Seuil* $= 5$. *(e,f) avec les paramètres* $\omega = 5$ $\beta = 1$.

## 3.8 Détection des contours dans une image bruitée

Dans cette section, nous présentons un algorithme capable de faire les deux applications simultanément, détection des contours et du bruit. L'idée de base est de regrouper les deux modèles correspondant aux bruits et aux contours. La stratégie que nous adoptons est une stratégie de décision, après une représentation de l'image à niveaux de gris par un hypergraphe de voisinage HVI ou Hypergraphe de Voisinage SpatioColorimétrique HVSC pour une image couleur, on passe à une étape de classification où une hyperarête est localisée hyperarête de bruit, contours.

Après représentation de l'image par hypergraphe de voisinage, on procède à une classification des hyperarêtes. Nous commençons d'abord par une détection des hyperarêtes isolées et non isolées. Ensuite, on traite ces deux entités. Si une hyperarête isolée vérifie le modèle 2 de bruit alors elle est classée comme bruit. Si une hyperarête non isolée ou isolée satisfait les conditions de modèle de contours alors elle est classée comme zone contours.

Ce regroupement des modèles permet de résoudre le problème de la détection des contours dans des images bruitées. Dans la figure 3.27, nous présentons le bloc diagramme de la classification de l'image.

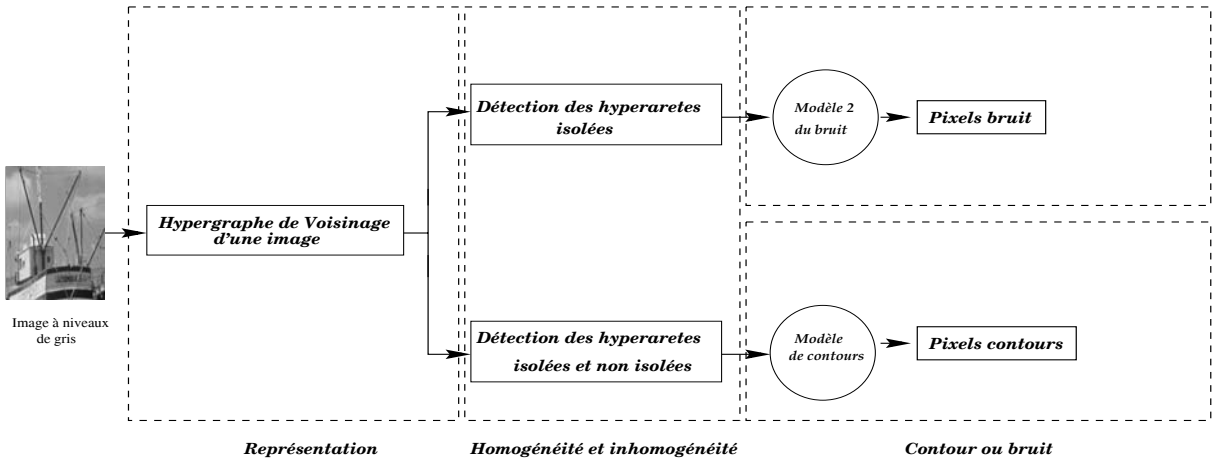



FIG. *3.27: Bloc diagramme de l'algorithme de détection des contours d'une image bruitée.*

Dans la section suivante, nous commençons par présenter l'algorithme appliqué à l'image à niveaux de gris. L'algorithme proposé pour l'image couleur utilise l'approche vectorielle dans l'espace couleur CIELab.

### 3.8.1 Cas des images à niveaux de gris

Dans la figure 3.28, nous présentons les images bruitées de simulations. Dans cette section, nous comparons les performances des algorithmes de Canny, Deriche, Shen et Castan, SUSAN et l'algorithme proposé sur des images bruitées avec un bruit impulsionnel. L'algorithme proposé utilise la représentation A avec un seuil $\alpha$ fixé expérimentalement, une longueur de chaîne $\omega = 5$ et un ordre de voisinage $\beta = 1$. Pour chacune des images traitées, la valeur du paramètres $\alpha$ a été choisi afin de donner les cartes de contours les plus significatives en terme de qualité visuelle.

Dans les figures 3.29 et 3.30 , nous présentons les résultats des cinq algorithmes.

Pour les images bruitées, Smith propose d'utiliser un stratégie de filtrage. Celle-ci consiste à appliquer un filtre moyenneur dans chaque USAN. Rappelons qu'une zone USAN représente

l'ensemble des pixels qui appartiennent à la même région que le pixel central du masque circulaire et qui ont approximativement la même intensité.

Après filtrage des images Poivron et Maison, l'algorithme SUSAN détecte mieux les contours des images Poivron et Maison. Néanmoins ce dernier a tendance à confondre le bruit impulsionnel et le contour. En effet, malgré l'étape de filtrage introduite dans l'algorithme SUSAN, nous constatons que les cartes de contours de ce dernier comportent des pixels bruit (Figures 3.29(c), 3.30(c)). Les algorithmes de Shen et Castan et de Canny se comportent de la même façon que l'algorithme SUSAN avec le bruit impulsionnel (Figures 3.29(a), 3.30(a), 3.29(b), 3.30(b)). Les seuls algorithmes capables de distinguer le bruit impulsionnel et les contours sont l'algorithme de Deriche et l'algorithme proposé. En effet, selon les figures 3.29(d), 3.30(d), 3.29(e) et 3.30(e), nous remarquons que les cartes de contours comportent moins de pixels bruit. En conclusion, les algorithmes de Deriche et proposé sont plus robustes au bruit impulsionnel.

Dans la figure 3.31, nous présentons les cartes de contours générées par l'algorithme proposé sur des images bruitées avec 10% de bruit impulsionnel. D'après ces résultats, nous constatons que : si nous augmentons le pourcentage des pixels de bruit dans les images Poivron et Maison, nous détectons quelques pixels de bruit impulsionnel comme pixels contours.

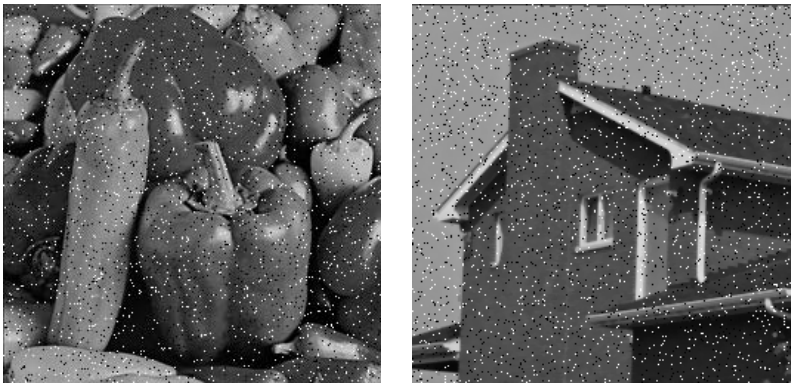


FIG. *3.28: Les images Poivron et Maison bruitées avec un bruit impulsionnel de 5%.*


FIG. *3.29: Cartes de contours de l'image Maison bruitée avec un bruit impulsionnel de 5% par les algorithmes : (a) Canny ($\sigma = 1.50$, $t_b = 0.30$ et $t_h = 0.80$), (b) Shen et Castan ($\gamma = 1$, $t_b = 30$ et $t_h = 60$), (c) SUSAN (BT=20 et DT=4), (d) Deriche ($\tau = 1$, $t_b = 10$ et $t_h = 50$), (e) Proposé ($\alpha = 30$, $\omega = 5$ et $\beta = 1$).*

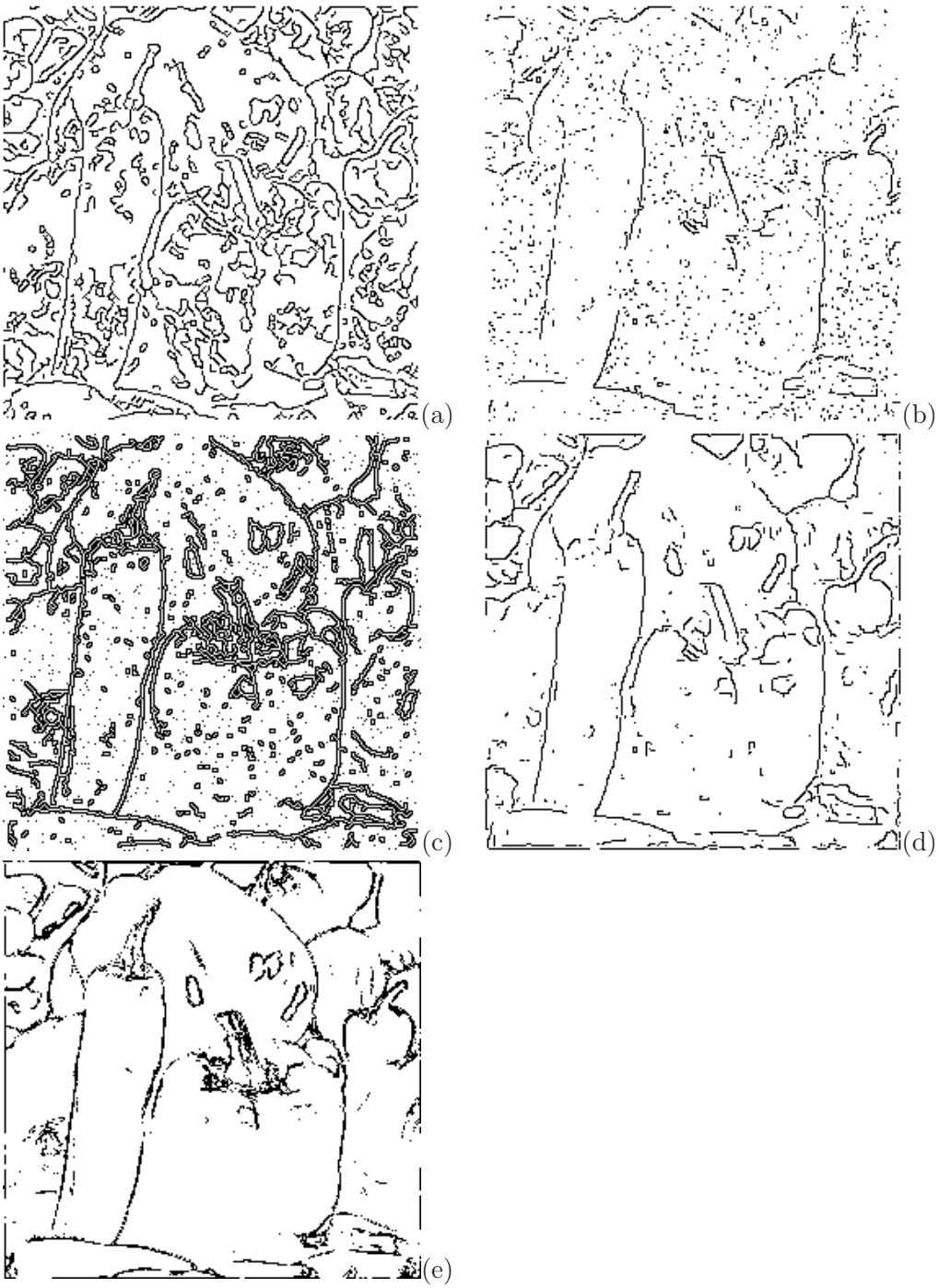
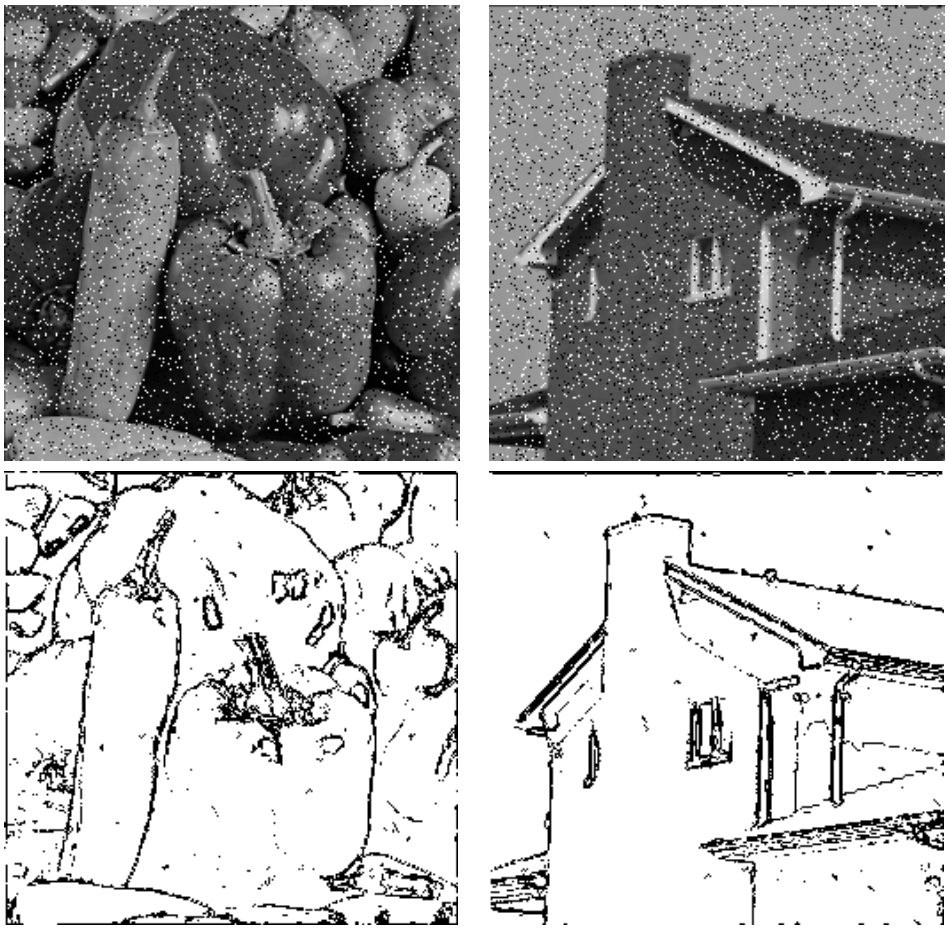



FIG. *3.30: Cartes de contours de l'image Poivron bruitée avec un bruit impulsionnel de 5% par les algorithmes : (a) Canny ($\sigma = 1.50$, $t_b = 0.30$ et $t_h = 0.60$), (b) Shen et Castan ($\gamma = 1$, $t_b = 25$ et $t_h = 60$), (c) SUSAN (BT=20 et DT=4), (d) Deriche ($\tau = 1$, $t_b = 8$ et $t_h = 45$), (e) Proposé ($\alpha = 25$, $\omega = 5$ et $\beta = 1$).*



(a)($\alpha = 35$, $\omega = 5$ et $\beta = 1$) (b) ($\alpha = 20$, $\omega = 5$ et $\beta = 1$)

FIG. *3.31: Cartes de contours de l'image Poivron bruitée avec un bruit impulsionnel de 10% par l'algorithme proposé.*

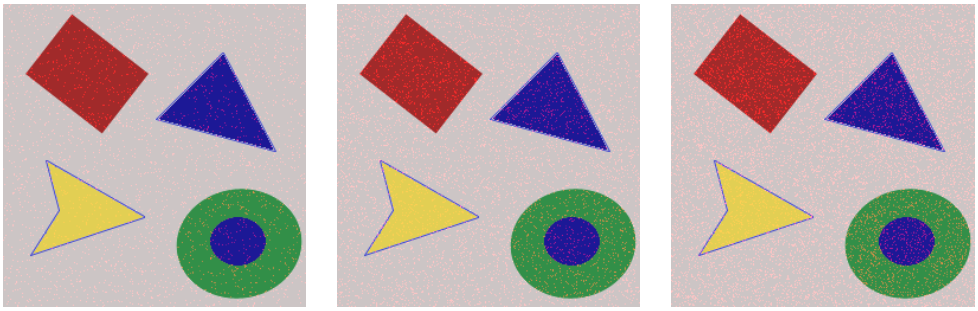
### 3.8.2 Cas des images couleurs

Dans cette section, nous présentons les résultats de détection de contours obtenus sur des images couleurs corrompues par un bruit gaussien additif d'écart-type 35 avec différents pourcentages de bruit (5%, 10% et 35% ).

La figure 3.32 représente les images de simulations. Le but de ces images de simulations, est de comparer la robustesse des algorithmes de Canny marginal, Cumani et l'algorithme proposé utilisant la représentation A dans l'espace couleur CIELab en détection de contours dans des images de couleur bruitées.

Les cartes de contours associées aux algorithmes sont présentées dans la figure 3.33. Ces cartes de contours sont générées avec les paramètres "optimaux" de chaque algorithme. Dans ces simulations, l'algorithme proposé utilise la représentation A avec un seuil $\alpha$ global estimé.

En examinant ces résultats, nous constatons que l'algorithme de Cumani est le plus robuste au bruit gaussien pour les trois pourcentages : 5%, 10% et 15%. En effet, pour un pourcentage de 5, l'algorithme de Cumani a bien distingué le bruit et le contour. L'algorithme de détection de contours proposé lui aussi a bien distingué le bruit et le contour. La seule différence entre les deux algorithmes est située au niveau de la détection des jonctions. D'après les résultats de détection de contour, nous voyons que l'algorithme proposé a bien détecté ces derniers. L'algorithme de Canny marginal détecte les jonctions et les contours de faible amplitude. Néanmoins, il confond le bruit gaussien et les contours. Ceci est valable pour les trois pourcentages de bruit.



(a-1) (a-2) (a-3)

FIG. *3.32: Les images logo bruitées avec un bruit gaussien d'écart-type 35. Les pourcentages de pixels de bruit sont respectivement 5%, 10% et 15% pour les images (a-1), (a-2) et (a-3).*

## 3.9 Conclusion

Dans ce chapitre, nous avons tout d'abord décrit quelques techniques de détection des contours : par exemple, le détecteur de Canny, de Deriche, de Shen et Castan et de l'algorithme SUSAN. Ensuite, nous avons défini un modèle de contour en utilisant les propriétés des hypergraphes. Ce modèle a été basé sur la notion de l'homogénéité locale. Cette dernière a été caractérisée par les hyperarêtes non isolées et isolées formant des chaînes. En utilisant ce modèle et la représentation par hypergraphe de voisinage de l'image, nous avons défini un algorithme de détection de contours. Cet algorithme a été appliqué en premier lieu à des images à niveaux de gris, puis à des images couleur.

Afin de nous assurer de l'efficacité du modèle de contour et par conséquent de l'algorithme proposé, nous avons effectué un série d'expérimentations sur des images numériques : d'abord

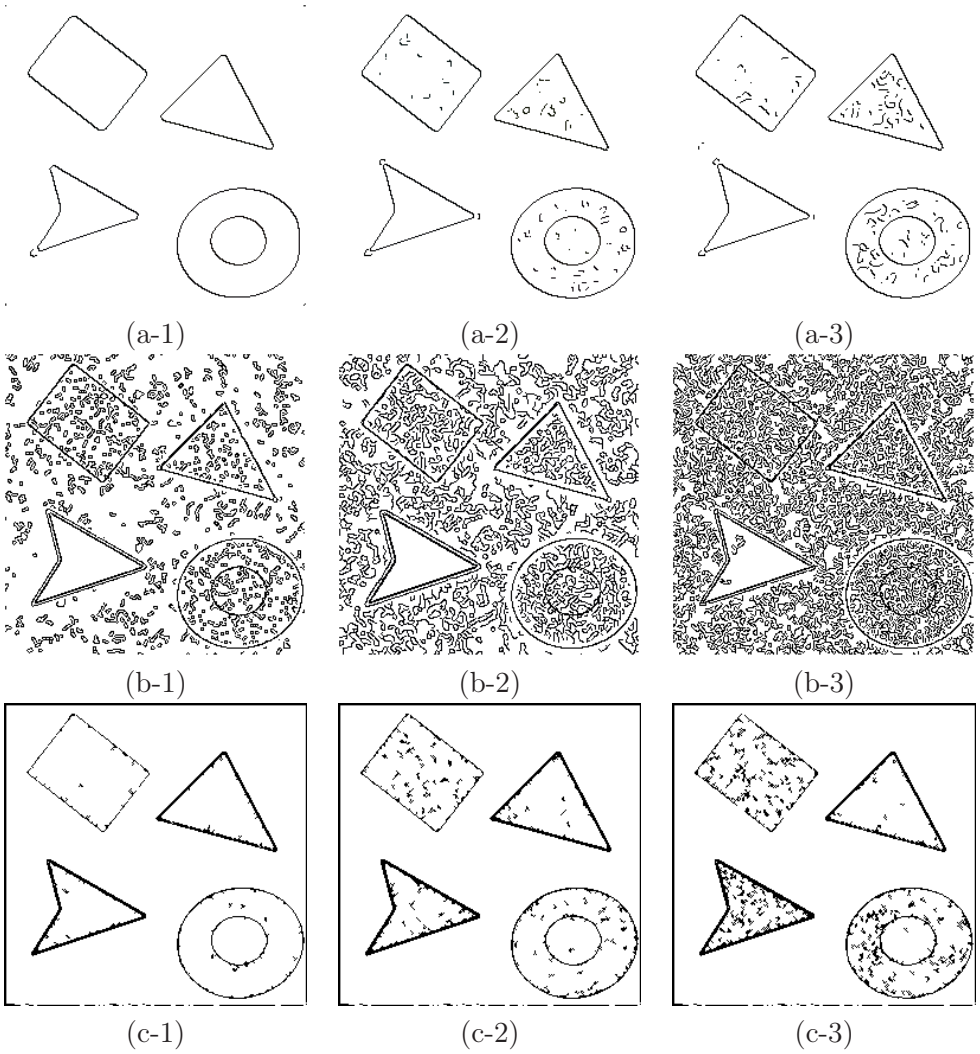



FIG. *3.33: Les résultats de l'algorithme proposé et du détecteur de Canny marginal sur une image corrompue par un bruit gaussien d'écart-type 35. Les cartes de contours (a-1) ($\sigma = 3$, seuil $= 10$) , (a-2) ($\sigma = 2$, seuil $= 15$) et (a-3) ($\sigma = 2$, seuil $= 15$) par l'algorithme de Cumani. Les cartes de contours (b-1) ($\sigma = 2$, $t_b = 0.30$, $t_h = 0.60$), (b-2) ($\sigma = 2$, $t_b = 0.30$, $t_h = 0.60$) et (b-3) ($\sigma = 2$, $t_b = 0.30$, $t_h = 0.60$) par l'algorithme de Canny marginal. Les cartes de contours (c-1), (c-2) et (c-3) par l'algorithme proposé dans l'espace couleur CIELab.*

synthétiques, ensuite réelles. L'algorithme de détection de contours utilise les trois représentations par hypergraphe de voisinage A, B et C. Pour l'évaluation de la détection de contours, nous avons utilisé deux critères : les probabilités de détection et de fausse alarme et le critère visuel. L'évaluation de l'algorithme de détection de contours s'est déroulée en deux étapes : d'abord, l'évaluation des représentations A, B et C suivie de la comparaison avec les algorithmes de Canny, Deriche, Shen et Castan et SUSAN.

En se basant sur les probabilités de détection et de fausse alarme, nous avons constaté que les performances de l'algorithme utilisant les représentations A, B ou C sont relativement proches. Néanmoins, la représentation B s'avère plus performante en terme de détection de contours et d'aptitude à ne pas introduire de faux contours. En utilisant cette représentation, nous avons comparé l'algorithme proposé avec celui de Canny, Deriche, Shen et Castan et Smith (SUSAN). La méthodologie de comparaison utilisée pour évaluer les algorithmes consiste à trouver d'abord les jeux de paramètres optimaux de chaque algorithme avant leur évaluation. Ces jeux de paramètres correspondent à des cartes de contours contenant plus de contours significatifs et moins de faux contours.

Sur une image de synthèse, et en utilisant les probabilités de détection et de fausse alarme, nous avons regroupé ces algorithmes en deux catégories. La première catégorie contient les algorithmes de Canny et SUSAN. La deuxième catégorie regroupe Deriche et Shen et Castan. Les algorithmes de la première catégorie détectent plus de contours significatifs avec plus de faux contours. Tandis que les algorithmes de la deuxième catégorie détectent moins de contours significatifs mais aussi moins de faux contours. L'algorithme proposé est situé dans la catégorie des algorithmes de Canny et SUSAN en terme de probabilité de détection et dans la catégorie des algorithmes de Deriche et Shen et Castan en terme de probabilité de fausse alarme. Autrement dit, il est performant en terme de détection de contours sans pour autant introduire plus de faux contours que les méthodes les plus efficaces en ce sens.

En conclusion, nous avons remarqué que l'algorithme proposé donne de bons résultats en terme de détection des contours. Néanmoins, ces performances dépendent de la longueur de la chaîne des hyperarêtes isolées ou non isolées. L'augmentation de la longueur de cette chaîne entraîne une sous-estimation des contours. La diminution de la longueur de cette chaîne entraîne une sur-estimation des contours.

Sur des images réelles, nous avons d'abord commencé par une comparaison des représentations A, B et C. En effet, les trois représentations ont donné approximativement les mêmes cartes de contours. Ensuite, nous avons comparé la représentation B avec les algorithmes de Canny, Deriche, Shen et Castan et SUSAN. En examinant les résultats de la détection de contours, nous avons remarqué que l'algorithme proposé a réagi de la même façon que sur les images de synthèse : bonne aptitude à détecter les contours et à ne pas introduire de faux contours, ce qu'aucun des filtres testés ne permet ; c'est à dire une bonne détection avec plus de contours significatifs et moins de faux contours.

Dans la deuxième partie de ce chapitre, nous avons évalué l'algorithme de détection de contours sur des images couleur en utilisant la représentation A. Dans ce cas, nous avons utilisé une stratégie vectorielle. Dans les premières expérimentations, nous avons comparé les deux espaces couleur RVB et CIELab. Ce dernier a donné de bons résultats en terme de détection de contours significatifs avec moins de faux contours. Ceci a justifié le "bon" choix de la distance colorimétrique dans l'espace uniforme CIELab. En utilisant la représentation A, nous avons constaté que l'algorithme proposé dépendait de la valeur de seuil colorimétrique. Dans le but de rendre cet algorithme non supervisé, nous avons estimé le seuil colorimétrique. Les résultats de cet algorithme avec ce nouveau seuil ont été évalués d'abord avec la représentation A à seuil fixe (algorithme supervisé), ensuite avec les algorithmes de Canny marginal et Cumani. Pour cette comparaison, nous avons utilisé des images de synthèse et des images réelles.

Sur des images de synthèse, nous avons constaté que l'algorithme de Canny marginal, l'algorithme de Cumani et l'algorithme proposé détectent approximativement les mêmes contours significatifs. Néanmoins, les algorithmes de Canny marginal et Cumani ont donné quelques faux contours sur l'image de synthèse. En plus, l'algorithme de Cumani n'a pas détecté les jonctions de l'image. Ceci a entraîné une modification de la forme des objets de l'image de synthèse.

Sur des images réelles, nous avons constaté que l'algorithme de Canny marginal introduit des faux contours. En conclusion, l'approche marginale de détection de contour de Canny sur des images réelles, détecte plus de faux contours. Les approches vectorielles, proposée ou de Cumani, ont détecté approximativement les mêmes cartes de contours. La seule différence des deux derniers algorithmes est située au niveau de la détection des jonctions.

Dans la troisième partie de ce chapitre, nous avons présenté un algorithme de détection de contours dans des images bruitées. Le principe de cet algorithme est basé sur les deux modèles de bruit et de contours. Afin d'évaluer cet algorithme, nous avons utilisé seulement la représentation A à seuil fixe. Les deux images à niveaux de gris et couleur ont été traitées. Pour les images à niveaux de gris, la carte de sortie de cet algorithme a été comparée avec les algorithmes de Canny, de Deriche, de Shen et Castan et SUSAN. Le bruit traité dans ce cas est le bruit impulsionnel. Sur des images à niveaux de gris, nous avons constaté que seuls l'algorithme proposé et celui de Deriche ont pu distinguer un contour d'un bruit. Les autres algorithmes ont confondu ces deux entités.

Pour les images couleur, la carte de sortie de l'algorithme proposé a été comparée avec les algorithmes de Canny marginal et de Cumani. Dans ce cas, le bruit injecté dans l'image couleur est le bruit gaussien. Sur des images couleur, nous avons remarqué que l'algorithme de Cumani et celui proposé ont détecté approximativement les mêmes cartes de contours. Par contre, l'algorithme de Cumani, ne détecte pas les jonctions.

# Conclusion et perspectives

Les travaux menés dans cette thèse ont conduit à l'obtention de plusieurs résultats importants. Les contributions de ce travail sont doubles puisqu'elles touchent à la fois la représentation d'image par hypergraphe de voisinage et l'application de celle-ci pour la suppression du bruit et la détection des contours. Nous avons ainsi :

- **étudié la représentation par hypergraphe de voisinage.** Nous avons repris celle définie par A. Bretto mais en approfondissant les propriétés. La représentation par hypergraphe de voisinage dépend de trois paramètres : le seuil spatial, le seuil spatiocolorimétrique et la distance colorimétrique. En étudiant ces paramètres, nous avons proposé trois représentations différentes. La première (représentation A) est basée sur un seuil spatiocolorimétrique global pour générer les hyperarêtes de l'hypergraphe de voisinage. La seconde (représentation B) utilise un seuil dynamique qui dépend des propriétés de l'image traitée. La dernière (représentation C) emploie des fonctions de similarité. Nous avons exploité ces trois représentations pour la modélisation des images à niveaux de gris et des images couleur. D'abord, nous avons commencé par la représentation par hypergraphe de voisinage de l'image à niveaux de gris. La génération de cette représentation n'a pas posé trop de difficultés, puisque l'image à niveaux de gris est une image monocomposante. Par conséquent, la distance colorimétrique est définie entre scalaires. La représentation par hypergraphe de voisinage spatiocolorimétrique emploie une distance colorimétrique entre vecteurs. C'est pour cette raison que nous avons d'abord commencé par décrire quelques espaces couleur ainsi que leurs distances colorimétriques ; ensuite nous avons mené une étude sur les stratégies de traitements des images couleur. En effet, selon cette brève étude, nous avons défini deux extensions : marginale et vectorielle. Nous nous sommes attachés à la dernière car la représentation par hypergraphe de voisinage est particulièrement adaptée aux signaux multicomposantes. Il suffit de définir une distance colorimétrique dans l'espace couleur donné pour générer l'hypergraphe de voisinage.
- **défini des modèles de bruit basés entièrement sur la représentation par hypergraphe de voisinage.** Il nous a fallu mettre en place un algorithme de suppression de bruit pour valider la représentation par hypergraphe. Nous avons exploité les trois représentations dans l'image à niveaux de gris et l'image couleur pour l'élaboration d'un algorithme de suppression de bruit. La stratégie de l'algorithme proposé commence tout d'abord par une étape de détection des zones bruitées puis une estimation des dégradations détectées. La suppression de bruit par hypergraphe de voisinage se base sur la notion d'inhomogénéité. Nous avons défini deux modèles de bruit en se basant sur la notion d'inhomogénéité. Le premier modèle permet effectivement de classer les hyperarêtes isolées soit de cardinalité égale à 1 soit de cardinalité supérieure à 1. Les hyperarêtes de cardinalité 1 décrivent l'inhomogénéité, mais cette propriété est aussi caractéristique des contours. Pour différencier ces deux entités, nous avons ajouté une autre condition sur la structure spatiale des hyperarêtes de bruit. Dans les premières expérimentations, nous avons commencé d'abord par une comparaison de ces deux modèles afin de trouver le meilleur d'entre eux, capable de supprimer le bruit tout en conservant les informations utiles de l'image et particulièrement les contours. Les comparaisons ont

été menées sur la représentation A. En examinant ces résultats, nous avons constaté que le deuxième modèle est plus performant en terme de détection et préservation des contours par rapport au premier modèle. Le choix du deuxième modèle de bruit, exploitant cette notion d'inhomogénéité sur lequel nous avons travaillé dans tout le document, nous a permis de proposer des solutions originales adaptées à ce problème. Ce modèle, tout en étant aussi simple que possible, prend simultanément en compte les deux aspects de l'estimation de l'image bruitée : détection des perturbations et préservation des contours.

Dans nos expérimentations de suppression de bruit dans des images à niveaux de gris, nous avons évalué les trois représentations afin de trouver la représentation la plus adéquate pour le problème proposé. En examinant les résultats, nous avons remarqué que les représentations par hypergraphe de voisinage B et C sont les plus compatibles avec le problème de suppression de bruit. Les exemples d'estimations, sur des images de synthèse et des images réelles de nature variées par l'algorithme proposé, montrent que l'algorithme permet d'obtenir des résultats de qualité correcte. Nous avons évalué l'algorithme proposé pour la suppression de plusieurs types de bruit, comme par exemple, les bruits gaussien, alpha-stable et impulsionnel. D'après les résultats de l'évaluation, nous avons constaté que l'algorithme est plus performant lorsqu'il s'agit d'un bruit qui comporte un caractère impulsif. Ensuite, nous avons comparé l'algorithme de suppression de bruit utilisant ces deux dernières représentations ainsi que le deuxième modèle de bruit avec différents algorithmes déjà existants. Il en résulte que notre approche a des performances équivalentes à celles des algorithmes existants.

L'algorithme de débruitage a été ensuite appliqué à l'image couleur. Les deux approches marginale et vectorielle ont été prises en considération. Cet algorithme s'appuyait uniquement sur la représentation SpatioColorimétrique HVSC puisque le modèle de bruit était le même.

Pour générer la représentation HVSC, nous avons fait appel à des distances colorimétriques. Ces dernières changent en fonction de l'espace couleur utilisé. Dans notre cas, nous avons utilisé la distance euclidienne dans les espaces couleur RVB et CIELab.

Dans les expérimentations, nous avons comparé tout d'abord l'algorithme proposé en utilisant les stratégies marginale et vectorielle. En examinant ces résultats, nous avons constaté que l'approche vectorielle est plus performante que l'approche marginale, car cette dernière ignore totalement la corrélation entre les composantes. Ensuite, nous avons comparé les espaces couleur RVB et CIELab. Les résultats de cette comparaison ont montré que l'espace CIELab est meilleur que le RVB. En effet, cette supériorité est justifiée par l'utilisation de la distance euclidienne dans un espace uniforme.

Nous avons ensuite comparé les résultats de suppression de bruit par l'algorithme vectoriel dans l'espace CIELab avec les filtres VMF et BVDF. Il en résulte que, pour une bonne préservation de contours, le recours à des filtres globaux s'appliquant à l'image entière, n'est pas la meilleure solution. Les zones intactes des images sont souvent de haute définition et il serait souhaitable de ne pas filtrer ces régions.

- **défini un modèle de contour dans des images à niveaux de gris et couleur.** Nous avons tout d'abord décrit quelques techniques de détection des contours : par exemple, le détecteur de Canny, de Deriche, de Shen et Castan et de l'algorithme SUSAN. Ensuite, nous avons défini un modèle de contour en utilisant les propriétés des hypergraphes. Ce modèle a été basé sur la notion de l'homogénéité locale. Cette dernière a été caractérisée par les hyperarêtes non isolées et isolées formant des chaînes. En utilisant ce modèle et la représentation par hypergraphe de voisinage de l'image, nous avons défini un algorithme de détection de contours. Cet algorithme a été appliqué en premier lieu à des images à niveaux de gris, puis à des images couleur.

  Afin de nous assurer de l'efficacité du modèle de contour et par conséquent de l'algorithme proposé, nous avons effectué un série d'expérimentations sur des images numériques : d'abord

synthétiques, ensuite réelles. L'algorithme de détection de contours utilise les trois représentations par hypergraphe de voisinage A, B et C. Pour l'évaluation de la détection de contours, nous avons utilisé deux critères : les probabilités de détection et de fausse alarme et le critère visuel. L'évaluation de l'algorithme de détection de contours s'est déroulée en deux étapes : D'abord, l'évaluation des représentations A, B et C suivie de la comparaison avec les algorithmes de Canny, Deriche, Shen et Castan et SUSAN.

En se basant sur les probabilités de détection et de fausse alarme, nous avons constaté que les performances de l'algorithme utilisant les représentations A, B ou C sont relativement proches. Néanmoins, la représentation B s'avère plus performante en terme de détection de contours et d'aptitude à ne pas introduire de faux contours. En utilisant cette représentation, nous avons comparé l'algorithme proposé avec celui de Canny, Deriche, Shen et Castan et Smith (SUSAN). La méthodologie de comparaison utilisée pour évaluer les algorithmes consiste à trouver d'abord les jeux de paramètres optimaux de chaque algorithme avant leur évaluation. Ces jeux de paramètres correspondent à des cartes de contours contenant plus de contours significatifs et moins de faux contours.

Sur une image de synthèse, et en utilisant les probabilités de détection et de fausse alarme, nous avons regroupé ces algorithmes en deux catégories. La première catégorie contient les algorithmes de Canny et SUSAN. La deuxième catégorie regroupe Deriche et Shen et Castan. Les algorithmes de la première catégorie détectent plus de contours significatifs avec plus de faux contours. Tandis que les algorithmes de la deuxième catégorie détectent moins de contours significatifs mais aussi moins de faux contours. L'algorithme proposé est situé dans la catégorie des algorithmes de Canny et SUSAN en terme de probabilité de détection et dans la catégorie des algorithmes de Deriche et Shen et Castan en terme de probabilité de fausse alarme. Autrement dit, il est performant en terme de détection de contours sans pour autant introduire plus de faux contours que les méthodes les plus efficaces en ce sens.

En conclusion, nous avons remarqué que l'algorithme proposé donne de bons résultats en terme de détection des contours. Néanmoins, ces performances dépendent de la longueur de la chaîne des hyperarêtes isolées ou non isolées. L'augmentation de la longueur de cette chaîne entraîne une sous-estimation des contours. La diminution de la longueur de cette chaîne entraîne une sur-estimation des contours.

Sur des images réelles, nous avons d'abord commencé par une comparaison des représentations A, B et C. En effet, les trois représentations ont donné approximativement les mêmes cartes de contours. Ensuite, nous avons comparé la représentation B avec les algorithmes de Canny, Deriche, Shen et Castan et SUSAN. En examinant les résultats de la détection de contours, nous avons remarqué que l'algorithme proposé a réagi de la même façon que sur les images de synthèse : bonne aptitude à détecter les contours et à ne pas introduire de faux contours, ce qu'aucun des filtres testés ne permet ; c'est à dire une bonne détection avec plus de contours significatifs et moins de faux contours.

Dans la deuxième partie de ce chapitre, nous avons évalué l'algorithme de détection de contours sur des images couleur en utilisant la représentation A. Dans ce cas, nous avons utilisé une stratégie vectorielle. Dans les premières expérimentations, nous avons comparé les deux espaces couleur RVB et CIELab. Ce dernier a donné de bons résultats en terme de détection de contours significatifs avec moins de faux contours. Ceci a justifié le "bon" choix de la distance colorimétrique dans l'espace uniforme CIELab. En utilisant la représentation A, nous avons constaté que l'algorithme proposé dépendait de la valeur de seuil colorimétrique. Dans le but de rendre cet algorithme non supervisé, nous avons estimé le seuil colorimétrique. Les résultats de cet algorithme avec ce nouveau seuil ont été évalués d'abord avec la représentation A à seuil fixe (algorithme supervisé), ensuite avec les algorithmes de Canny marginal et Cumani. Pour cette comparaison, nous avons utilisé des images de synthèse et des images

réelles.

Sur des images de synthèse, nous avons constaté que l'algorithme de Canny marginal, l'algorithme de Cumani et l'algorithme proposé détectent approximativement les mêmes contours significatifs. Néanmoins, les algorithmes de Canny marginal et Cumani ont donné quelques faux contours sur l'image de synthèse. En plus, l'algorithme de Cumani n'a pas détecté les jonctions de l'image. Ceci a entraîné une modification de la forme des objets de l'image de synthèse.

Sur des images réelles, nous avons constaté que l'algorithme de Canny marginal introduit des faux contours. En conclusion, l'approche marginale de détection de contour de Canny sur des images réelles, détecte plus de faux contours. Les approches vectorielles, proposée ou de Cumani, ont détecté approximativement les mêmes cartes de contours. La seule différence des deux derniers algorithmes est située au niveau de la détection des jonctions.

Dans la troisième partie de ce chapitre, nous avons présenté un algorithme de détection de contours dans des images bruitées. Le principe de cet algorithme est basé sur les deux modèles de bruit et de contours. Afin d'évaluer cet algorithme, nous avons utilisé seulement la représentation A à seuil fixe. Les deux images à niveaux de gris et couleur ont été traitées. Pour les images à niveaux de gris, la carte de sortie de cet algorithme a été comparée avec les algorithmes de Canny, de Deriche, de Shen et Castan et SUSAN. Le bruit traité dans ce cas est le bruit impulsionnel. Sur des images à niveaux de gris, nous avons constaté que seuls l'algorithme proposé et celui de Deriche ont pu distinguer un contour d'un bruit. Les autres algorithmes ont confondu ces deux entités.

Pour les images couleur, la carte de sortie de l'algorithme proposé a été comparée avec les algorithmes de Canny marginal et de Cumani. Dans ce cas, le bruit injecté dans l'image couleur est le bruit gaussien. Sur des images couleur, nous avons remarqué que l'algorithme de Cumani et celui proposé ont détecté approximativement les mêmes cartes de contours. Par contre, l'algorithme de Cumani, ne détecte pas les jonctions.

Tous ces travaux m'ont permis d'atteindre l'objectif initial de ma thèse, qui était de mettre au point un algorithme de classification de l'image en bruit, contour.

Nous pouvons envisager diverses extensions naturelles à ce travail :

- **Algorithme non supervisé.**
- **Influence du système de voisinage**
- **Autres mesures adaptatives.** En effet, la méthode proposée dépend de trois paramètres, et plus particulièrement du seuil spatiocolorimétrique. Ce seuil décrit la relation entre vecteurs dans un voisinage donné. Il serait très intéressant d'intégrer une nouvelle mesure adaptative pour la génération automatique des hypergraphes de voisinage.
- **Changement du domaine spatiocolorimétrique.** Durant cette thèse, nous avons travaillé que dans le domaine spatiocolorimétrique. En effet, il existe d'autres domaines où la méthode proposée peut être appliquée, citons comme exemple : le domaine fréquentiel.
- **Spatio-temporel (vidéo).** La représentation par hypergraphe de voisinage est particulièrement adaptée aux signaux multicomposantes. Il serait très intéressant de l'appliquer dans le domaine de la vidéo, où la troisième composante est le temps.
- **Algorithme itératif.** Dans les expérimentations, nous pouvons par exemple ne pas détecter quelques pixels bruités dans l'image. Pour pallier ce problème, il y a deux solutions, soit nous améliorons les modèles de bruit en prenant en considération la structure spatiale du bruit non détecté, soit tout simplement en exécutant l'algorithme encore une fois sur l'image estimée. Il serait très intéressant, dans ce cas d'utiliser la deuxième qui correspond à un algorithme itératif.

# Publications

## Articles

1. S. Rital, A. Bretto, H. Cherifi and D. Aboutajdine, A combinatorial based technique for impulsive noise removal in images , An International Journal, Image Processing and Communications, Vol. 1, Number 3-4, 2001.

## Conférences

1. S. Rital, H. Cherifi. A Combinatorial Color Edge Detector, International Conference on Image Analysis and Recognition ICIAR, Porto, Portugal 2004, to be published in the Springer Lecture Notes in Computer Science (LNCS) series.
2. S. Rital, H. Cherifi. Détection de contours d'images couleur par hypergraphe de voisinage spatio colorimétrique. Compression et Représentation des Signaux Audiovisuels. Lille - 25 et 26 mai 2004 à l'ENIC Télécom Lille 1. CORESA 2004.
3. S. Rital, H. Cherifi. Détection de contours d'images couleur bruitées par hypergraphe de voisinage spatio colorimétrique. 2nd International Symposium on Image/Video Communications over fixed and mobile networks. Brest, France 7 - 9 july 2004
4. S. Rital, H. Cherifi. Débruitage d'images couleur par hypergraphe de voisinage spatio colorimétrique. CORESA 2003, 16 janvier 2003, Lyon, France.
5. S. Rital, H. cherifi. Similarity hypergraph representation for impulsive noise reduction. 4th EURASIP Conference focused on Video/Image Processing and Multimedia Communications. EC-VIP-MC 2003. 25 july 2003. Zagreb, CROATIA.
6. S. Rital, H. cherifi. Similarity hypergraph representation for edge detection algorithm on noisy images 9th International Workshop on Systems, Signals and Image Processing IWSSIP'02. November 7-8, 2002 - Manchester, United Kingdom.
7. S. Rital, H. cherifi. Impulsive noise removal in color images using neighborhood hypergraph. ICCVG, International Conference on Computer Vision and Graphics. September 25-29 '2002 Zakopane, Poland
8. S. Rital, A. Bretto, H. Cherifi and D. Aboutajdine, A combinatorial edge detection algorithm on noisy images , 4th EURASIP-IEEE Region 8, International Symposium on Video / Image Processing and Multimedia Communications, 16-19 June 2002 Zadar, Croatia.
9. S. Rital, A. Bretto, H. Cherifi and D. Aboutajdine, Application of Adaptive Hypergraph Model to Impulsive Noise Detection, 9th International Conference on Computer Analysis of Images and Patterns CAIP'2001, 5-7 septembre 2001 Warsaw, Poland.
10. S. Rital, A. Bretto, H. Cherifi and D. Aboutajdine. Image Adaptive Modeling : application to impulsive Noise Cancellation, International Conference on Image and Signal Processing. ICISP'2001, 5-7 mai 2001, Agadir Maroc.

11. S. Rital, M. Elhassouni, A. Bretto, H. Cherifi and D. Aboutajdine, Image Modeling ny Hypergraph : Application to Noise Cancellation, International Conference on Vision Interface. VI'2000, 14-17 mai 2000 Montréal, Canada.
12. S. Rital, A. Bretto, H. Cherifi and D. Aboutajdine, La Modélisation d'Images par Hypergraphes de voisinage : Application au Débruitage. International Symposium on Image/Video Communications over Fixed and Mobile Networks, ISIVC'2000, 17-20 avril 2000, Rabat, Maroc.

# Index

# Bibliographie


[AHN90] J. Astola, P. Haavisto, and Y. Neuvo, *Vector median filters*, Proc. of the IEEE **78** (1990), no. 4, 678–689.

[AM00] T. Scott Acton and Dipti Prasad Mukherjee, *Area operators for edge detection*, Pattern Recognition Letters **21** (2000), no. 8, 771–777.

[APV98] D. Androutsos, K.N. Plataniotis, and A.N. Venetsanopoulos, *Distance measures for color image retrieval*, IEEE International Conference on Image Processing (1998).

[AWJ90] A. Amini, T.E. Weymounth, and R.C. Jain, *Using dynamic programming for solving variational problems in vision*, IEEE transactions on pattern analysis and machine intelligence **12** (1990), no. 9, 855–867.

[BACL97] A. Bretto, J. Azéma, H. Cherifi, and B. Laget, *Combinatorics and image processing*, Computer Vision Graphics and Image Processing **59** (1997), no. 5, 265–277.

[Bar76] V. Barnett, *The ordering of multivariate data*, J. of. Royal Stat. Soc. A **3** (1976), no. 139, 318–354.

[Ber73] C. Berge, *Introduction à la théorie des hypergraphes*, Les presses de l'Université de Montréal (1973).

[Ber79] ———, *The helly property*, Southeast Asian Math Soc. Bull **1** (1979), 16–19.

[Ber87a] ———, *Graphs*, North holland, 1987.

[Ber87b] ———, *Hypergraph*, North holland, 1987.

[Ber87c] F. Bergholm, *Edge focusing*, IEEE Trans. on Pattern Analysis and Machine Intelligence, PAMI **9** (1987), no. 6, 726–741.

[Bes86] J. Besag, *On the statistical analysis of dirty pictures*, Royal Statistical Society **48** (1986), no. 3, 259–302.

[BFB93] L. Blan-Féraud and M. Barlaud, *Restauration d'image bruitée par analyse multirésolution et champs de markov*, Traitement du signal **10** (1993), no. 2.

[Bol98] B. Bollobas, *Modern graph theory*, Springer-Verlag (1998).

[Bor84] G. Borgefors, *Distance trabsformation in arbitrary dimensions*, Computer Vision, Graphics, and Image Processing **27** (1984), no. 3, 321–345.

[Bor86] ———, *Distance transformation in digital images*, Computer Vision, Graphics, and Image Processing **34** (1986), no. 3, 344–371.

[BP94] A. Buchowicz and I. Pitas, *Multichannel distance filters*, n Proc. of IEEE Conference on Image Processing ICIF **2** (1994), 575–579.

[Bro84] D. R. K. Brownrigg, *Weighted median filters*, Comm. Assoc. Comput. Machinecy **27** (1984), no. 8, 807–818.

[BV95] V. Buzuloiu and C. Vertan, *Introducing mathematical morphology for vector signals*, In Proc. of the 3rd Conference on Applied and Industrial Mathematics ROMAI (1995), 17–19.

[BZ87] A. Blake and A. Zisserman, *Visual reconstruction*, MIT Press (1987).

[Can83] J.F. Canny, *Finding edges and lines in images*, Technical Report AI-TR-720, MIT Artificial Intelligence Laboratory, Cambridge, Massachusetts, USA (1983).

[Can86] ———, *A computational approach to edge detection*, IEEE Transactions on Pattern Analysis and Machine Intelligence **8** (1986), no. 6, 679–698.

[Cas96] K.R. Castleman, *Digital image processing*, Englewood Cliffs NJ, 1996.

[CD98] M.L. Cormer and E.J. Delp, *Morphological operations, chap 11 in the colour image processing handbook*, 1998.

[Cel95] M. Celenk, *Analysis of color image of natural scenes*, Journal of Electronic Imaging **4** (1995), no. 4, 382–392.

[Ces96] A. Cesare, *Algorithmes rapides de restauration des signaux : Application à l'imagerie médicale*, Thèse de doctorat de l'université de paris-sud, UFR Scientifique D'Orsay, 1996.

[CGG91] A. Cumani, P. Grattoni, and A. Guiducci, *An edge-based description of color images*, Graphical Models and Image Processing CVGIP **53** (1991), no. 4, 313–323.

[Cha97] J. Chanussot, *Ordres vectoriels, entrelacements de bits et "space filling cuves" pour le traitement d'images multi-composantes*, Rapport LAMII 4, Laboratoire de Méthodes Informatiques (LaMI), 1997.

[Cha98] C. Charrier, *Vers l'optimisation statistique et perceptuelle de la qualité pour la compression des images couleur par quantification vectorielle*, Ph.D. thesis, Université Jean Monnet, Novembre 1998.

[CJD94] C. Coroyer, C. Jorand, and P. Duvaut, *Roc curves of skeness and kurtosis statistical tests*, Proceedings of EUSIPCO Conference (1994).

[CM91] J. M. Chassery and A. Montanvert, *Géométrie discrète en analyse d'images (coll. traité des nouvelles technologies série images)*, Hermès, Paris, 1991.

[CP95] J.P. Cocquerez and S. Philipp, *Analyse d'images : Filtrage et segmentation. enseignement de la physique*, Masson, 1995.

[CRZB98] M. Ciuc, R. M. Rangayyan, T. Zaharia, and V. Buzuloiu, *Adaptive neighbourhood filters for color image filtering*, In Proc. of the SFIE Nonlinear Image Processing IX Conference **3304** (1998), 277–286.

[CTB+00] D. Coltuc, E. Trouvé, F. Bujor, N. Classeau, and J.P. Rudant, *ime-space filtering of multitemporal sar images*, Proc. of IEEE-IGARSS **7** (2000), 2909–2911.

[Cum89] A. Cumani, *A second order differential operator for multispectral edge detection*, In 5th Int. Conf. Image Anal. Process. (1989), 54–58.

[Cum91] ———, *Edge detection in multispectral images*, Computer Vision, Graphics, and Image Processing **8** (1991), no. 6, 40–51.

[DC98] M. Douimi and H. Cherifi, *Algorithme énumérative pour la minimisation de fonction d'énergie*, Revue Scientifique Française : Traitement du Signal **5** (1998), no. 1, 67–78.

[Der87a] R. Deriche, *Optimal edge detection using recursive filters*, In Proc. International Conference on Pattern Recognition (ICCV'87), London, United Kingdom (1987), 8–12.

[Der87b] ———, *Using canny's criteria to derive a recursively implemented optimal edge detector*, International Journal of Computer Vision **1** (1987), no. 2, 167–187.

[Der90] ———, *Fast algorithms for low-level vision*, IEEE Trans. Pattern Analysis and Machine Intelligence **12** (1990), no. 1, 78–87.

[Dev00] J. C. Devaux, *Segmentation par approche région des images couleurs : application aux images aériennes de milieux naturels*, Thèse de Doctorat, Université de Dijon (2000).

[Die97] R. Diestel, *Graph theory*, Springer-Verlag, 1997.

[DR96] M. Doroochi and A. M. Reza, *Fuzzy cluster filter*, In Proc. IEEE Conference on Image Processing ICIF **2** (1996), 939–942.

[EF93] G. Economou and S. Fotopoulos, *A family of adaptive nonlinear low complexity 1o'ilters*, In Proc. ECCTD '93 Circuit Theory and Design (1993), 521–524, Davos, Switzerland.

[EF95] A. M. Eskicioglu and P. S. Fisher, *Image quality measures and their performance*, IEEE Transactions on Communications **43** (1995), no. 12, 2959–2965.

[Fag83] R. Fagin, *Degrees of acyclicity for hypergraphs and relational database schemes*, J. Assoc. Comput. Mach. **30** (1983), 514–550.

[FE95] S. Fotopoulos and G. Economou, *Multichannel filters using composite distance metrics*, In Proc. of the IEEE Workshop on Nonlinear Signal and Image Processing **2** (1995), 503–506, Neos Marmaras, Greece.

[GA95] S. Godbole and A. Amin, *Mathematical morphology for edge and overlap detection for medical images*, Real-Time Imaging **1** (1995), no. 3, 191–201.

[GG84] S. Geman and D. Geman, *Stochastic relaxation, gibbs distributions and the bayesian restoration of images*, IEEE Trans. on Pattern Analysis and Machine Intelligence **6** (1984), no. 6, 721–741.

[Gi] Base d'images GDR-isis, *http ://www-isis.enst.fr/applications/baseimages/*.

[GM95] M. Gondran and M. Minoux, *Graphes et algorithmes*, Eyrolles, janvier 1995.

[GW81] N. C. Gallagher and G. L. Wise, *A theortical analysis of the properties of median filters*, IEEE Trans. on ASSP **29** (1981), no. 6, 1136–1141.

[HA91] R. Hardie and G. Arce, *Ranking in $r^p$ ans its use in multivariate image estimation*, IEEE Trans. on Circuits and systems for Video Technology **1** (1991), no. 2, 197–209.

[HM86] A. Huertas and G. Medioni, *Detection of intensity changes with suppixel accuracy using laplacian-gaussian masks*, IEEE Trans. on PAMI **8** (1986), no. 5, 651–661.

[HSSB97] D. Michael Heath, S. Sarkar, T. Sanocki, and W. Kevin Bowyer, *A robust visual method for assessing the relative performance of edge-detection algorithms*, IEEE Trans. On Pattern Analysis and Machine Intelligence **19** (1997), no. 12, 1338–1359.

[Hub81] P.J. Huber, *Robust statistics*, New York, 1981.

[IZ95] A. Lee Iverson and W. Steven Zucker, *Logical/linear operators for image curves*, IEEE Trans. on Pattern Analysis and Machine Intelligence **17** (1995), no. 10, 982–996.

[Jah95] B. Jahne, *Digital image processing*, Springer, 1995.

[Jai89] A. K. Jain, *Fundamentals of digital image processing*, Prentice Hall Intl., Englewood Cliffs NJ, 1989.

[Jiu95] Jing-Ying Jiu, *Multilevel median filter based on fuzzy decision*, IEEE Trans. Image Processing (1995), no. 4, 680–682.

[Kas88] S. A. Kassam, *Signal detection in non-gaussian noise*, Springer-Verlag : New York, 1988.

[KGV83] S. Kirpatrick, C.D. Gellatt, and M.P. Vecchi, *Optimization by simulated annealing*, Science Journal **220** (1983), no. 4598, 671–681.

[KLL96] H. Konik, V. Lozano, and B. Laget, *Color pyramids for image processing*, Journal of Imaging Science and Technology **40** (1996), no. 6, 535–542.

[KP96] C. Kotropoulos and I. Pitas, *Adaptive lms l-filters for noise suppression in images*, IEEE Trans. on Image Processing **5** (1996), no. 12, 1596–1609.

[KP97] _____, *Adaptive multichannel l-filters with structural constraints*, Proc. IEEE International Workshop on Non Linear Signal and Image Processing, Michigan, USA (1997).

[Kun93] M. Kunt, *Traitement numérique des images*, Press Polytechniques et Universitaires Romandes. Collections électricité **2** (1993).

[KWT87] M. Kass, A. Witkin, and D. Terzopoulos, *Snakes : Active contour models*, Int. Journal of Computer Vision 1 (1987), 321–331.

[Lam02] P. Lambert, *Etudes méthodologiques du filtrage et de segmentation d'images multi-composantes*, Hdr, Université de Savoie, 12 juille 2002.

[LB02] Z. Lei and P. Bao, *Edge detection by scale multiplication in wavelet domain*, Pattern Recognition Letters **23** (2002), no. 14, 1771–1784.

[Lee91] H. C. Lee, *Optimized quadtree for karhumen-lœeve transform in multispectral image coding*, IEEE Trans. on Image Processing **8** (1991), no. 4.

[Mac85] D. L. Macadam, *Color measurement, theme and variation*, vol. 27, Springer series in optical sciences, 1985.

[Mar92] R. March, *Visual reconstruction with discontinuities using variational methods*, Image and vision computing **10** (1992), no. 1, 30–38.

[MFC75] O. Macchi, C. Faure, and J. P. Caliste, *Probabilités d'erreur de détecteurs*, Revue CETHEDEC, Cah. NS (1975), 1–51.

[Mid79] D. Middleton, *Canonical non-gaussian noise models : their implications for measurement and for prediction of receiver performance.*, IEEE Trans. on Electromagnetic Compatibility **21** (1979), 209–220.

[Mou89] M. Mouhoub, *Filtres d'ordres, filtre médian récursif : Analyse et évaluation des performances en traitement d'image*, Thèse de doctorat informatique et automatique appliquée, Institut National des Sciences Appliquées de Lyon, 1989.

[M.T99] M.Tabiza, *Filtres lp : Etudes des propriétés et application en traitement d'images*, Thèse de doctorat, Université de Savoie, 1999.

[MW98] K. Vijay Madisetti and B. Douglas Williams, *The digital signal processing handbook*, Boca Raton, FL, USA : CRC Press, Inc., ISBN 0-8493-8572-5, 1998.

[NB86] V.S. Nalwa and T.O. Binford, *On detecting edges*, IEEE Trans. on Pattern Analysis and Machine Intelligence PAMI **8** (1986), no. 6, 699–714.

[NBBB99] A. Le Negrate, A. Beghdadi, and K. Boussaid-Belkacem, *Quelques traitements bas niveau basés sur une analyse du contraste local*, Vision Interface'99, Trois-Rivières, Canada (1999), 19–21.

[NM00] D. Nuzillard and J.M. Muzillard, *Application of blind source separation to 1-d and 2-d*, Astronomy and Astrophysics supplement series (2000), no. 147, 129–138.

[NN88] A. Nieminen and Y. Neuvo, *Comments on the theoretical analysis of the max/median filter*, IEEE Transactions on Acoustics, Speech and Signal Processing **ASSP-35** (1988), no. 5, 628–827.

[NP96] N. Nikolaidis and I. Pitas, *Adaptive multichannel l-filters based on reduced ordering*, Proc. of 8th EUSIPCO, Trieste, Italy (1996), 153–156.

[PAV95] K. N. Plataniotis, N. Androutsos, and A. N. Venetsanopoulos, *Color image processing using fuzzy vector directional filters*, In Proc. of the IEEE Workshop on Nonlinear Signal and Image Processing, Neos, Marmaras, Greece **2** (1995), 535–538.

[PAV96] K. N. Plataniotis, D. Androutsos, and A. N. Venetsanopoulos, *Multichannel filtering for color image processing*, In Proc. of IEEE Conference on Image Processing ICIP, Lausanne, Switzerland **1** (1996), 16–19.

[PAV97] K. N. Plataniotis, D. Androutsos, and A.N. Venetsanopoulos, *Multichannel filters for image processing*, Signal Processing : Image Communication **9** (1997), 143–158.

[Pet97] J. L. Peterson, *Petri nets*, Comput. Surv. **9** (1997), no. 3, 223–252.

[Pre70] J. M. S. Prewitt, *Object enhancement and extraction*, Academic Press, 1970.

[PT91] I. Pitas and Taskalides, *Multivariate ordering in color image filtering*, IEEE Trans. on Circuits and Systems for Video Technology **1** (1991), no. 3, 247–259.

[PV90] I. Pitas and A. N. Venetsanopoulos, *Nonlinear digital filters - principles and applications*, Kluwer Academic Publ., Norwell MA, 1990.

[PV00] K. N. Plataniotis and A. N. Venetsanopoulos, *Color image processing and applications*, Springer Verlag, Berlin Heidelberg New York, 2000.

[PW93] A. B. Poirson and B. A. Wandell, *The appearance of colored patterns : Pattern-color separability*, Journal of the Optical Society of America A **10** (1993), 2458–2471.

[PW96] ———, *Pattern-color separable pathways predict sensitivity to simple colored patterns.*, Vision Research **36** (1996), 515–526.

[Que97] A. Quesser, *Color spaces for inspection of natural objects*, Proc. of 4rd IEEE International Conference on Image Processing (ICIP'97) **3** (1997), 42–45.

[Raj92] A. Raji, *Filtres d'ordre récursifs symétriques : étude et évaluation des performances*, Thèse de doctorat, Université de Savoie, 1992.

[RCF98] R. M. Rangayyan, M. Ciuc, and F. Faghlh., *Adaptive-neighborhood filtering of images corrupted by signal dependent noise*, Applied Optics **37** (1998), no. 20, 4477–4487.

[RMHN95] C. A. Rothwell, J.L. Mundy, W. Hoffman, and V.-D. Nguyen, *Driving vision by topology*, In International Symposium on Computer Vision, Coral Gables, Florida (1995), 395–400.

[Rop95] M. Ropert, *Filtres médians et médians généralisés : application au traitement des images*, Thèse de doctorat, Université de Rennes I, 1995.

[Ros98] L. Paul Rosin, *Thresholding for change detection*, ICCV, International Conference on Scale-Space Theories in Computer Vision (1998), 274–279.

[RZ82] R. H. A. Rosenfeld and S.W. Zucker, *Scene labeling by relaxation operation*, IEEE Trans. on Systems Mans Cybernetics (1982).

[SB97] S. Smith and J. Brady, *Susan - a new approach to low level image processing*, International Journal of Computer Vision **23** (1997), no. 1, 45–78.

[SC92] J. Shen and S. Castan, *An optimal linear operator for step edge detection*, Computer vision, Graphics and Image Processing **54** (1992), no. 2, 112–133.

[SC93] J. Shen and C. Castan, *Towards unification of band-limited differential opertors for edge detection*, Signal Processing **31** (1993), no. 2, 103–119.

[SN93] M. Shao and C. L. Nikias, *Signal processing with fractional lower order moments : Stable processes and their applications*, Proceedings of the IEEE **81** (1993), no. 7, 986–1010.

[Sob90] I. Sobel, *An isotropic image gradient operator*, Academic Press, 1990.

[Sou83] P. De Souza, *Edge detection using sliding statistical tests*, CVGIP **23** (1983), 1–14.

[SPC+02] B. Smolka, K.N. Plataniotis, A. Chydzinski, M. Szczepanski, A.N. Venetsanopoulos, and K. Wojciechowski, *Self-adaptive algorithm of impulsive noise reduction in color images*, Pattern Recognition J. **35** (2002), 1771–1784.

[SS92] R. L. Stevenson and S. M. Schweizer, *Nonlinear filtering structure for image smoothing in mixed-noise environments*, Journal of Mathematical Imaging and Vision (1992), no. 2, 137–154.

[SS03] B. Sankur and M. Sezgin, *A survey over image thresholding techniques and quantitative performance evaluation*, Accepted in Journal of Electronic Imaging (2003).

[TER+00] R. Tovar, C. Esleves, R. Rustarnante, B. Psenicka, and S. K. Mitra, *Implementation of an impulse noise removal algorithm from images*, IASTED, Signal Processing and Communications - SPC (2000).

[TGD92] H. L. Tan, S. B. Gelfand, and E. J. Delp, *A cost minimization approach to edge detection using simulated annealing*, IEEE Trans. on Pettern Analysis and Machine Intelligence **14** (1992), 3–18.

[TKP98] S. Tsekeridou, C. Kotropoulos, and I. Pitas, *Adaptive order statistic filters for the removal of noise from corrupted images*, SPIE Optical Engineering **37** (1998), no. 10, 2798–2816.

[TKV96] P. E. Trahanias, D. Karakos, and A. N. Venetsanopoulos, *Directional processing of color images : Theory and experimental results*, IEEE Trans. on Image Processing **5** (1996), no. 6, 528–534.

[TLL95] A. Trémau, A. Lozano, and B. Laget, *How to optimize the use of the lhc color space in color image analysis processes*, Acta Sterologica Proceeding of the 9th ICS, Copenhagen **14** (1995), no. 2, 223–228.

[TV93] P. E. Trahanias and A. N. Venetsanopoulos, *Vector directional filters -a new class of multi-channel image processing filters*, IEEE Trans. on Image Processing **2** (1993), no. 4, 528–534.

[VBP96] C. Vertan, V. Buzuloiu, and V. Popescu, *Morphological like operators for color images*, In Proc. EUSIPCO'96, Trieste, Italy **1** (1996), 165–168.

[Ver99] Constantin Vertan, *Technique de traitement non linéaire des images en couleurs*, Ph.D. thesis, Université de Politehnica Buchresti, Faculté d'Electronique et Télécommunications, 1999.

[VMB97] C. Vertan, M. Malciu, and V. Buzuloiu, *Fuzzy developments of multichannel filters*, In Proc. of the first International Conference on Conventional and Knowledge-Based Intelligent Electronic Systems KES'97, Adelaide, Australia (1997), 21–23.

[VMZB96] C. Vertan, M. Malciu, T. Zaharia, and V. Buzuloiu, *Vector median filtering by iterative projections*, In. Proc. IEEE International Conference on Image Processing ICIP'96, Lausanne, Switzerland **1** (1996), 977–980.

[Wan92] X. Wang, *Adaptive multilevel median filter*, IEEE Trans.Signal Processing **4** (1992), no. 4.

[Wes] J. Weszka, *A survey of threshold selection techniques*, Comput. Graph. Image Process **7**, no. 2, 259–269.

[WR78] J. Weszka and R. Rosenfeld, *Threshold evaluation techniques*, IEEE Trans. **SMC-8** (1978), no. 8, 622–629.

[YJH+99] Mei Yu, Gang Yi Jilang, Dong Mun Ha, Tae Young Choi, and Yong Deak Kim, *New adaptive vector filter based on noise estimate*, IEICE Trans. Fundamentals **82** (1999), no. 6, 911–919.

[YT95] X. Yang and P.S. Toh, *Adaptive fuzzy multilevel median filter*, 1995IEEE Trans. on Image Processing **4** (1995), no. 5, 680–682.

[Zen86] S. Di Zenzo, *A note on the gradient of a multi-image*, CVGIP **33** (1986), no. 1, 116–125.