

Université Paris IX-Dauphine
U.F.R. Mathématiques de la Décision

SPÉCIALITÉ : Mathématiques

N° attribué par la bibliothèque

|-----|

THÈSE

Pour l'obtention du titre de
DOCTEUR ÈS SCIENCES
(arrêté du 30 mars 1992)
présentée et soutenue par
Pascal MONASSE

**Représentation morphologique
d'images numériques et
application au recalage**

JURY

M.	Vicent CASELLES	Rapporteur
Mme	Françoise DIBOS	Examineur
M.	Frédéric GUICHARD	Examineur
M.	Gary HEWER	Rapporteur
M.	Yves MEYER	Président
M.	Jean-Michel MOREL	Directeur
M.	Philippe SALEMBIER	Rapporteur
M.	Michel SCHMITT	Examineur

Présentée et soutenue publiquement le 30 juin 2000

à Alexandra

Remerciements

Je tiens tout d'abord à remercier Yves Meyer de m'avoir fait l'honneur de présider ce jury.

Je suis reconnaissant aux rapporteurs de cette thèse, Vicent Caselles, Gary Haver et Philippe Salembier, d'avoir accepté cette tâche, dont ils se sont acquittés avec le plus grand sérieux, et pour la pertinence de leurs remarques et suggestions. Je remercie doublement Vicent, ainsi que Coloma Ballester et Tomeu Coll, pour m'avoir permis de passer plusieurs séjours à l'Université des Iles Baléares, et pour de multiples discussions passionnantes sur la topologie des images.

Françoise Dibos, Frédéric Guichard et Michel Schmitt ont accepté de faire partie de mon jury, je les en remercie vivement. Frédéric sait à quel point ses idées, suggestions et remarques ont influencé cette thèse. Je lui exprime plus particulièrement ma gratitude et tiens à l'assurer de mon amitié.

Durant quatre années, Jean-Michel Morel a dirigé mes travaux de recherches. Cette thèse doit énormément à sa grande disponibilité, son dynamisme, 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.

Les échanges permanents et la très bonne ambiance au sein du groupe de traiteurs d'images réuni autour de Jean-Michel Morel sont, j'en suis persuadé, des éléments essentiels de cette réussite. Je remercie chaleureusement Andrès Almansa, Frédéric Cao, Agnès Desolneux, Jacques Froment, Yann Gousseau, Georges Koepfler, Saïd Ladjal, Jose-Luis Lisani, François Malgouyres, Simon Masnou et Lionel Moisan de rendre ce groupe si vivant et de maintenir un tel enthousiasme collectif. Je remercie également les personnes travaillant ou ayant travaillé au CEREMADE à l'Université Paris-Dauphine, ou au CMLA à l'ENS Cachan, et plus particulièrement Jérôme

Besnard, Pascal Bringas, Christophe Labourdette, Martin Lefébure, Jacques-Olivier Moussafir, Nicolas Pajor, Olivia Sanchez, Michel Vanbreugel, Frédéric Vautrain et Nicolas Vayatis.

Lenny Rudin et l'équipe de Cognitech m'ont révélé l'intérêt industriel de mes recherches, je leur en suis reconnaissant. De plus, Lenny m'a permis de découvrir des travaux en rapport du plus grand intérêt, en particulier un remarquable article de Kronrod, je le remercie vivement.

Bernard Rougé et Leonid Yaroslavsky m'ont permis de bien comprendre quelques problèmes qui se posent en traitement d'images, je tiens à les saluer.

Enfin, je remercie Alexandra, Léna et mes parents pour leur soutien constant durant ces années.

Table des matières

0.1	Considérations générales	13
0.2	Plan	14

I Représentation d'image invariante par changement de contraste 17

1	Introduction 19
1.1	Différentes représentations des images 19
1.2	La morphologie mathématique 21
1.2.1	Survol rapide 21
1.2.2	Ensembles de niveau, changement de contraste, lignes de niveau 21
1.2.3	Reconstruction 22
1.3	L'image et sa topologie 23
1.3.1	Descriptions topologiques des images 23
1.3.2	Topologie des représentations morphologiques 24
2	Arbre d'inclusion des formes 29
2.1	Motivation 29
2.2	Définitions de base 30
2.2.1	Cadre de travail 30
2.2.2	Des ensembles de niveau à leurs composantes 31
2.2.3	Au-delà des composantes d'ensembles de niveau 34
2.3	Arbre d'inclusion des formes 36
2.3.1	Définition axiomatique de la saturation, des trous et des formes 38
2.3.2	Saturation du complémentaire 39
2.3.3	Propriétés de la saturation 40
2.3.4	Décomposition d'une image en formes 46
2.3.5	Espaces unichérents 49

2.4	Applications	51
2.4.1	Image définie sur \mathbb{R}^n	51
2.4.2	Image définie sur un sous-ensemble borné de \mathbb{R}^n	52
2.5	Reconstruction	58
2.5.1	Cadre de travail	58
2.5.2	Reconstruction directe	62
2.5.3	Reconstruction indirecte	66
3	Fast Level Set Transform	71
3.1	Intérêt de l'algorithme	71
3.2	Domaine continu vs. domaine discret	71
3.2.1	Généralités	71
3.2.2	Notre modèle d'interpolation	72
3.2.3	Conséquence sur la connexité	74
3.2.4	Formes dans les images numériques	74
3.3	La FLST	86
3.3.1	Entrée et sortie de la FLST	86
3.3.2	Description de l'algorithme	87
3.3.3	Analyse et preuve de la FLST	99
3.3.4	Complexité	107
3.4	Prendre avantage de la structure d'arbre	113
3.4.1	Stockage des pixels des formes numériques	113
3.4.2	Calcul de caractéristiques additives de forme	115
3.5	Extensions	117
3.5.1	Changement de connexité	117
3.5.2	Dimension supérieure	119
4	Applications à des filtres morphologiques	121
4.1	Filtres morphologiques	121
4.2	Filtre de grain	124
4.2.1	Description	125
4.2.2	Lien entre opérateur ensembliste et opérateur fonctionnel	127
4.2.3	Propriétés	133
4.2.4	Expériences	142
4.3	Quantification adaptative	144
4.3.1	Description	144
4.3.2	Dégradés	152
4.3.3	Suppression d'un nœud	154

4.3.4	Expériences	156
II	Recalage d'images	163
5	Introduction	165
5.1	Généralités	165
5.2	Méthodes de corrélation	166
5.3	Correspondances de caractéristiques	168
5.4	Aperçu de la méthode	168
6	Recalage d'images contraste invariant	171
6.1	Correspondances	171
6.1.1	Choix des caractéristiques	171
6.1.2	Descripteurs	172
6.1.3	Recherche des correspondances	174
6.2	Votes	175
6.3	Précision	176
6.4	Complexité	178
6.5	Extensions	179
6.5.1	Réduction du nombre de correspondances	179
6.5.2	Autres déplacements globaux	185
6.5.3	Utilisation de l'information d'inclusion	185
6.5.4	Gestion des occlusions	186
7	Expériences de recalage	189
7.1	Estimation de pose	189
7.1.1	Quelle heure est-il?	189
7.1.2	Une autre expérience de montre	198
7.2	Similitude	202
7.3	Précision	205
8	Conclusion	209
8.1	Points forts de cette thèse	209
8.2	Extensions possibles	209
8.3	Problèmes ouverts	210

Notations

<u>Symbole</u>	<u>Page</u>	<u>Signification</u>
X		Un ensemble muni d'une topologie, généralement \mathbb{R}^2 , \mathbb{R}^n ou un sous-ensemble
$\mathcal{P}(X)$		Ensemble des parties de X
$\overset{\circ}{A}$		Intérieur de l'ensemble A
\bar{A}		Adhérence de l'ensemble A
∂A		Frontière topologique de l'ensemble A
λ, μ		Niveaux de gris (nombres réels)
$\mathcal{X}^{\geq \lambda} u$ ou $[u \geq \lambda]$	21	Ensemble de niveau supérieur de valeur λ , ensemble des points $\mathbf{x} \in X$ tels que $u(\mathbf{x}) \geq \lambda$
$\mathcal{X}^{< \mu} u$ ou $[u < \mu]$	21	Ensemble de niveau inférieur de valeur μ , ensemble des points $\mathbf{x} \in X$ tels que $u(\mathbf{x}) < \mu$
g	22	Un changement de contraste, c'est-à-dire une fonction croissante (éventuellement strictement) de \mathbb{R} dans \mathbb{R} et continue (ou semicontinue supérieurement)
$cc(A), cc(A, \mathbf{x})$		Une composante connexe de l'ensemble A , la composante connexe de A contenant \mathbf{x} (ou \emptyset si $\mathbf{x} \notin A$)
sat	38	Un opérateur de saturation (voir Définition 2.6)
Ext A	38	Extérieur de l'ensemble A (notion liée à celle de saturation)
\mathcal{H}_A	38	Ensemble des trous de l'ensemble A (notion liée à celle de saturation)
(λ, A)	59	Un niveau-forme, c'est-à-dire un niveau de gris λ associé la saturation d'une composante connexe de l'ensemble de niveau (inférieur ou supérieur) de valeur λ

<u>Symbole</u>	<u>Page</u>	<u>Signification</u>
$LS_{\mathbf{x}}$	59	Ensemble des niveaux-formes (λ, A) tels que $\mathbf{x} \in A$
$LS_{\mathbf{x}}^{\geq}$	59	Ensemble des niveaux-formes $(\lambda, A) \in LS_{\mathbf{x}}$, A étant une composante connexe de l'ensemble de niveau supérieur $[u \geq \lambda]$
$LS_{\mathbf{x}}^{<}$	59	Ensemble des niveaux-formes $(\lambda, A) \in LS_{\mathbf{x}}$, A étant une composante connexe de l'ensemble de niveau inférieur $[u < \lambda]$
\preceq	61	Relation d'ordre entre niveaux-formes
u_d	72	Une image discrète, une application d'un ensemble discret Ω_d du plan dans \mathbb{R}
$ P $	74	P étant un pixel (un élément de Ω_d), le pixel ouvert contenant P , un carré ouvert de côté 1 contenant P
T_t	126	Filtre de grain ensembliste à l'échelle t
T_t	126	Filtre de grain fonctionnel à l'échelle t
\mathcal{B}_t	126	Ensemble des éléments structurants du filtre de grain
m_{ij}	172	Moments d'un ensemble plan
μ_{ij}	172	Moments centrés d'un ensemble plan

Résumé de la thèse

0.1 Considérations générales

De l'observation que notre perception des images est à peu près insensible au changement de contraste, et que nous sommes capables de reconnaître sans difficulté un objet sous différents éclairages, nous déduisons qu'en vision artificielle nous avons besoin d'invariance par changement de contraste. Donc ses méthodes doivent traiter de caractéristiques invariantes par changement de contraste. Les ensembles de niveau de l'image, les caractéristiques de base manipulées dans le cadre de la morphologie mathématique, présentent cette propriété. Bien qu'un changement de contraste dans une image puisse modifier le niveau auquel nous voyons un objet, l'ensemble de niveau correspondant reste présent dans l'image, simplement à un autre niveau. La morphologie mathématique utilise des outils géométriques plutôt que des niveaux de gris.

Nous devons ajouter deux autres besoins à l'invariance par changement de contraste : les caractéristiques doivent être aussi locales que possible, et ni les objets sombres, ni les objets clairs ne doivent avoir la préférence sur l'autre. Ces conditions sont facilement remplies : la première en considérant les composantes connexes des ensembles de niveau, la seconde en utilisant aussi bien les ensembles de niveau supérieurs (ensembles des pixels de niveau de gris plus grand qu'un certain seuil) que les ensembles de niveau inférieurs (ensembles des pixels de niveau de gris plus petit qu'un certain seuil). La relation entre les ensembles de niveau supérieurs (ou les ensembles de niveau inférieurs), ou leurs composantes, est facile à établir : ils sont monotones, ce qui se traduit par une structure d'arbre pour les composantes. La propriété attrayante de ces caractéristiques est qu'elles représentent exactement l'image. Leur donnée est équivalente à celle de l'image elle-même, contrairement par exemple aux représentations basées sur des bords calculés. Néanmoins, elles donnent deux représentations pour la même image : les ensembles de niveau supérieur favorisent les objets clairs, tandis que les ensembles de niveau inférieur sélectionnent les

objets sombres. Le but de la première partie de cette thèse est de fusionner les deux en une unique structure, qui s'avère être un arbre reposant sur l'inclusion. Pour cela, nous utilisons l'idée élémentaire d'enlever des composantes connexes d'ensembles de niveau considérées comme les moins utiles, et de remplir les trous de celles restantes. Deux filtres morphologiques déduits de cette représentation sont proposés. Puisqu'ils reposent sur cette représentation, ils présentent la propriété intéressante d'être invariants par inversion du contraste : on les qualifie d'autoduaux dans le vocabulaire de la morphologie mathématique.

Nous utilisons ensuite cette représentation dans la deuxième partie pour traiter de l'un des problèmes les plus élémentaires en traitement de plusieurs images : le recalage. Bien que la littérature sur le sujet propose de nombreuses méthodes pour traiter ce problème depuis au moins vingt ans, aucune de celles-ci ne peut prétendre le résoudre : la plupart sont sensibles au contraste (en particulier la corrélation), et elles donnent des résultats erronés en présence de mouvements secondaires. Notre méthode repose sur les éléments de base de notre représentation, nommés *formes*, qui sont donc des caractéristiques stables pour effectuer le recalage. Nous cherchons les formes de chaque image dans l'autre image, établissant ainsi des correspondances entre formes d'une image et de l'autre image. Puis une procédure de vote des correspondances sélectionne le mouvement dominant. Nous montrons que cette méthode a le mérite de donner un recalage d'une précision subpixelique dans des conditions tout à fait défavorables.

0.2 Plan

Le premier chapitre rappelle les différentes représentations de la topologie de l'image qu'on trouve dans la littérature sur le sujet et le Chapitre 2 expose les conditions d'existence de l'arbre des formes pour des images définies sur un domaine continu. Nous proposons une définition axiomatique de la notion de trou et montrons qu'elle permet d'associer un arbre d'inclusion des formes à une image. Nous examinons également les conditions de reconstruction de l'image à partir de son arbre. Cela justifie le terme de « représentation » dans un contexte très général. Ce chapitre est principalement mathématique, le principal outil étant la topologie, et en particulier nous utilisons de façon intensive la notion de connexité.

Le troisième chapitre applique les résultats généraux du Chapitre 2 à des images numériques discrètes, et présente un algorithme, la Fast Level Set Transform (FLST en abrégé), pour extraire l'arbre d'inclusion d'une image, ainsi que la façon de reconstruire l'image à partir de l'arbre. Bien que l'algorithme en soi soit complexe, sa

sortie, l'arbre d'inclusion des formes, est très facile à manipuler.

Le Chapitre 4 présente deux filtres morphologiques reposant sur l'arbre. Nous examinons leurs propriétés, les principales étant leur invariance par transformation affine et leur autodualité. Ces propriétés sont prouvées, les rendant attractives en traitement d'image.

La deuxième partie de cette thèse traite du recalage de deux images, lorsqu'un mouvement global dépendant de peu de paramètres existe, par exemple une similitude. Le Chapitre 5 introduit le sujet et le Chapitre 6 décrit la méthode que nous proposons. Cette partie est plus orientée vers les applications.

Enfin, le Chapitre 7 montre les résultats de quelques expériences de recalage, dans des conditions réalistes. Nous obtenons la précision subpixellique même sous des conditions contraires.

Première partie

Représentation d'image invariante par changement de contraste

Chapitre 1

Introduction

1.1 Différentes représentations des images

Les représentations des images peuvent varier en fonction de leur utilisation. L'information brute, c'est-à-dire les valeurs des échantillons, ou pixels, est une représentation de trop bas niveau, et l'image doit être décrite suivant des modèles plus élaborés.

Pour de la restauration ou du débruitage, les représentations reposant sur la transformée de Fourier sont généralement les meilleures puisqu'elles s'appuient sur le processus de génération de l'image (théorie de Shannon), et/ou sur les modèles fréquentiels de dégradation, comme par exemple pour le bruit additif, ou un noyau de convolution spécifique. Pourtant, la transformation de Fourier est totalement orientée vers les fréquences et ne donne pas directement d'information spatiale. La théorie des ondelettes [54, 42], localise les fréquences, et, du fait de la structure linéaire des images à leurs plus petites échelles, la représentation par ondelettes est à ce jour la meilleure représentation de l'image pour la compression.

Néanmoins, du point de vue de l'analyse d'image, les représentations fréquentielles ne donnent pas l'information adéquate. En effet, la représentation de Fourier est non locale et la représentation par ondelettes est sensible à la translation, la rotation ou le zoom dans l'image, ne permettant pas de reconnaître les objets indépendamment du point de vue. De plus, ces deux représentations ont des échelles d'observation quantifiées.

La théorie des espaces-échelles (scale-space) et de la détection de bords proposent de représenter les images par des bords significatifs, où la notion de bord est définie de manière convenable. Les algorithmes fonctionnent en général en deux étapes (qui parfois fusionnent) : d'abord les images sont lissées (linéairement ou

non) [3, 86], puis un détecteur de bords est appliqué à l'image lissée. Les bords sont détectés en s'appuyant sur les dérivées secondes de l'image. La plus ancienne définition de bord est due à Marr et Hildreth [47] et une variante proposée par Canny [8]. L'échelle représente le montant du lissage antérieur à la détection de bords. Le premier espace-échelle à base de bords est le passage par zéro du laplacien dans la pyramide gaussienne, c'est-à-dire que le lissage est une convolution avec un noyau gaussien de variance variable. Selon Marr, ces passages par zéro représentent le « raw primal sketch » de l'image, la base sur laquelle les algorithmes de vision doivent s'appuyer, voir Marr [46] et Hummel [29].

En général, l'extraction des bords peut se formuler comme un problème variationnel, voir Nitzberg et Mumford [63], Morel et Solimini [60]. On calcule la meilleure approximation de l'image par fonction dans une classe de fonctions pour lesquelles la notion de bords est correctement définie : un exemple célèbre de telle classe est la famille des images constantes par morceaux, de longueur des courbes de discontinuité finie ; dans cette classe, les lignes de discontinuité de la fonction approximante sont interprétées comme les bords, voir Mumford et Shah [61]. Puis un équilibre entre la proximité et la complexité de l'approximation (dans l'exemple précédant, la complexité peut être la longueur des bords de discontinuité) définit une représentation à différentes échelles de l'image.

Malgré la généralité de l'approche variationnelle, celle-ci souffre du fait qu'il n'existe pas de théorie qui dit quel doit être le modèle. Ces représentations à base de bords ont deux problèmes majeurs, reconnus, voir Koenderink [30], Witkin [93] et Mallat [42], mais non résolus par la théorie des espaces-échelles. En premier lieu, la représentation géométrique des bords est incomplète : elle ne permet pas de reconstruire totalement l'image, donc de l'information est perdue dans le processus de détection de bords. En second lieu, la décomposition en échelles donne une représentation redondante.

Un autre problème de ces approches est lié au fait que le niveau de gris de l'image n'est pas une donnée absolue, puisque dans de nombreux cas, le contraste dépend de la caméra, l'optique de la caméra étant en général inconnue, et toujours difficile à mesurer. Ce problème peut être évité en travaillant dans le cadre morphologique.

1.2 La morphologie mathématique

1.2.1 Survol rapide

Dans les images naturelles, le contraste dépend du type de caméra, du processus de numérisation, à cause de la quantification des niveaux de gris, de l'éclairage, etc. Malgré la multiplicité des facteurs changeant le contraste, la perception des images doit rester identique, indépendante de l'écran sur lequel elles sont affichées. En d'autres termes, l'information de contraste est secondaire par rapport à l'information géométrique, et utile principalement pour le confort visuel.

L'invariance par changement de contraste fut établie en premier comme un principe « gestaltiste » par Wertheimer [92].

Matheron [50] et après lui Serra [83, 84] proposent une représentation « morphologique » des images par leurs ensembles de niveau. Cela donne une représentation complète et invariante par changement de contraste de l'image, ne dépendant d'aucun paramètre. Une variante de cette représentation est proposée par Caselles *et al.* dans [11], en considérant les frontières de ces ensembles, c'est-à-dire les lignes de niveau, ce qui forme la carte topographique.

1.2.2 Ensembles de niveau, changement de contraste, lignes de niveau

Une image (en niveaux de gris) est représentée par une fonction u d'un ensemble X dans \mathbb{R} . Les objets les plus élémentaires de la morphologie mathématique sont les ensembles de niveau. Nous appelons ensemble de niveau supérieur $\mathcal{X}^{\geq \lambda} u$ de valeur λ et ensemble de niveau inférieur $\mathcal{X}^{< \mu} u$ de valeur μ les sous-ensembles de X définis par les formules :

$$\mathcal{X}^{\geq \lambda} u = \{\mathbf{x} \in X, u(\mathbf{x}) \geq \lambda\} \quad \text{et} \quad \mathcal{X}^{< \mu} u = \{\mathbf{x} \in X, u(\mathbf{x}) < \mu\}. \quad (1.1)$$

La convention de prendre une inégalité stricte pour les ensembles inférieurs, et large pour les ensembles supérieurs, est là pour obtenir de la consistance entre elles, i.e., $X \setminus \mathcal{X}^{\geq \lambda} u = \mathcal{X}^{< \lambda} u$. Alors qu'elle n'est habituellement que de peu d'importance car on ne mélange pas les ensembles de niveau inférieurs et supérieurs, elle devient fondamentale quand on traite des deux simultanément.

Un changement de contraste global se modélise par une fonction $g : \mathbb{R} \rightarrow \mathbb{R}$, croissante. Parfois, g a besoin d'être *strictement* croissante, auquel cas nous parlons de changement de contraste strict. La fonction v se déduit de u par un changement

de contraste g si $v = g \circ u$. La morphologie mathématique traite de caractéristiques des images qui sont invariantes relativement à un changement de contraste. Cela signifie que si $v = g \circ u$, les caractéristiques de v et u doivent être les mêmes. En d'autres termes, la relation

$$u \mathcal{R} v \Leftrightarrow \exists g \text{ strictement croissante telle que } v = g \circ u \quad (1.2)$$

est une relation d'équivalence, et les caractéristiques morphologiques ne dépendent pas d'un représentant particulier dans une classe d'équivalence, tandis que les opérateurs morphologiques agissent sur ces classes d'équivalence.

La famille des ensembles de niveau est une caractéristique morphologique de l'image car

$$\mathcal{X}^{<g(\lambda)} g \circ u = \mathcal{X}^{<\lambda} u \quad \text{et} \quad \mathcal{X}^{\geq g(\lambda)} g \circ u = \mathcal{X}^{\geq \lambda} u \quad (1.3)$$

et donc la famille des ensembles de niveau inférieurs $\mathcal{X}^{< u}$ ne dépend pas de g , pas plus que la famille des ensembles de niveau supérieurs $\mathcal{X}^{\geq u}$. Seuls les indices des ensembles de niveau changent.

De plus, les caractéristiques topologiques extraites des ensembles de niveau sont aussi morphologiques. Un cas particulier est celui des composantes connexes des frontières des ensembles de niveau, appelées *lignes de niveau*. Un autre cas est celui où l'on considère les composantes connexes des ensembles de niveau, qui sont utilisées dans le chapitre suivant pour construire les « formes ».

1.2.3 Reconstruction

Notre confiance dans l'intérêt des ensembles de niveau vient également du fait qu'ils sont une *représentation* de l'image. Des ensembles de niveau inférieurs d'une image u , on peut retrouver u par la formule :

$$\forall \mathbf{x}, u(\mathbf{x}) = \inf \{ \lambda : \mathbf{x} \in \mathcal{X}^{<\lambda} u \} \quad (1.4)$$

et des ensembles de niveau supérieurs par la formule :

$$\forall \mathbf{x}, u(\mathbf{x}) = \sup \{ \lambda : \mathbf{x} \in \mathcal{X}^{\geq \lambda} u \}. \quad (1.5)$$

Dans le dernier cas, à cause de l'inégalité large, la borne supérieure est en réalité un maximum, puisque $\mathbf{x} \in \mathcal{X}^{\geq u(\mathbf{x})} u$.

De plus, partant d'une famille d'ensembles $(X_\lambda)_{\lambda \in \mathbb{R}}$, nous pouvons reconstruire

une image par la formule

$$\forall \mathbf{x} \in X, \quad u(\mathbf{x}) = \sup \{ \lambda : \mathbf{x} \in X_\lambda \}. \quad (1.6)$$

La reconstruction est exacte, dans le sens que les ensembles de niveau supérieurs de u sont précisément les X_λ , pourvu que :

$$\bigcap_{\lambda \in \mathbb{R}} X_\lambda = \emptyset; \quad (1.7)$$

$$\bigcup_{\lambda \in \mathbb{R}} X_\lambda = X; \quad (1.7')$$

$$\forall \lambda \in \mathbb{R}, \quad \bigcap_{\mu < \lambda} X_\mu = X_\lambda. \quad (1.7'')$$

Les deux premières conditions assurent que les valeurs prises par u sont finies, la troisième condition impliquant que les X_λ sont décroissants et exprimant une condition de semicontinuité sur les X_λ . Ces résultats sont exposés par Guichard et Morel dans [26]. Ils montrent également qu'une reconstruction approximative peut s'effectuer sous des hypothèses plus faibles : si les conditions (1.7)-(1.7'') sont remplacées par la condition plus faible

$$\lambda > \mu \quad \Rightarrow \quad X_\lambda \subset X_\mu,$$

c'est-à-dire celle de décroissance des $(X_\lambda)_{\lambda \in \mathbb{R}}$, alors $\mathcal{X}^{\geq \lambda} u = X_\lambda$ presque partout, et pour presque tout λ (relativement à la mesure de Lebesgue).

1.3 L'image et sa topologie

1.3.1 Descriptions topologiques des images

Une fois l'image segmentée, d'une manière ou d'une autre, la topologie résultante doit être décrite. La notion usuelle de segmentation correspond à une *partition* de l'image en régions connexes et les relations entre ces régions sont significatives. La première idée est de coder la relation d'adjacence : nous avons besoin de savoir quand deux régions ont une frontière commune. Le moyen classique de représenter cette relation est à travers un graphe, le graphe d'adjacence des régions : chaque région est représentée par un nœud et lorsque deux régions sont adjacentes, une arête relie leurs nœuds, voir Rosenfeld [69]. Toutefois, l'adjacence n'est pas la seule

relation entre les régions. Par exemple, si deux régions sont adjacentes, le nombre de composantes connexes de leur frontière commune n'est pas codé. La solution à ce problème serait d'ajouter le nombre correspondant d'arêtes entre les deux nœuds, donnant un multigraphe. Plus ennuyeux est le problème que la connaissance qu'une région est un trou dans une autre région n'est pas contenue dans le (multi)graphe. Gangnet *et al.* [25], reconnaissant que ces données manquent, proposent d'ajouter la structure d'inclusion des contours dans les graphes. Mais cela représente la topologie de l'image dans deux graphes, la rendant délicate à manipuler. Observant la difficulté de décrire les relations entre régions en termes de pixels uniquement, Kovalesky, dans [33], propose une représentation en liste de cellules, ajoutant les frontières entre les régions comme des éléments unidimensionnels et les points de jonction de ces frontières comme des éléments de dimension 0. Mais cette structure n'est pas un graphe, et ne code pas plus de données que le graphe d'adjacence des régions.

Suivant la voie ouverte par Kovalesky, Fiorio dans [21] utilise les mêmes éléments pour construire sa représentation comme carte combinatoire (voir Lienhardt [38]) et expose un algorithme de complexité linéaire pour construire sa représentation, le Graphe Topologique des Frontières. Fiorio insiste sur le fait que la représentation doit être consistante avec la topologie usuelle du plan, et que ce faisant, elle ne doit introduire qu'un nombre minimum d'éléments de dimension non maximale. Dans [22], il généralise en dimensions supérieures cette représentation, tandis que dans [23], il explique comment manipuler le Graphe Topologique des Frontières, en particulier comment mettre à jour la structure quand deux régions fusionnent et comment extraire le Graphe Topologique des Frontières d'une sous-image, pourvu que la sous-image ne coupe pas des régions. Malheureusement, ces opérations de base ne sont pas évidentes, du fait que la carte combinatoire est une représentation assez complexe.

1.3.2 Topologie des représentations morphologiques

Toutes ces représentations topologiques reposent sur une segmentation de l'image comprise comme une partition en régions connexes. Mais les éléments de base de la morphologie mathématique, les ensembles de niveau, ne forment pas une partition de l'image ; à la place, elles sont hiérarchiques, car ordonnées. Quand on parle d'ensembles de niveau dans leur globalité, cet ordre, la relation d'inclusion, est total, donnant une structure très élémentaire, une liste ordonnée. Néanmoins, il manque un trait important à cette représentation, la localité, ou le fait que les atomes de la représentation (les ensembles de niveau) ne sont pas connexes. D'où le besoin de

considérer plutôt les composantes connexes des ensembles de niveau.

Une approche fructueuse est proposée par Ballester, Caselles et Morel dans [5], où les atomes sont des composantes connexes de biniveaux, c'est-à-dire des ensembles de points dont les valeurs¹ sont comprises entre deux seuils donnés. Ils sont choisis de telle sorte que lorsque les seuils sont changés de manière à avoir un biniveau plus petit, la sous-partie de l'atome reste connexe. Ces atomes sont appelés les sections monotones maximales, et sont invariants par rapport à un changement de contraste. Leur étude vient d'un algorithme de rehaussement de contraste local préservant les formes, proposé par Caselles *et al.* dans [12] et [13]. Mais les relations entre ces structures ne sont pas complètement étudiées, et leur efficacité en terme de compacité de la représentation reste à démontrer.

Cox et Karron [16] explorent la structure de la famille des composantes connexes des ensembles de niveau supérieurs dans une image tridimensionnelle dans des objectifs de codage et de visualisation de données 3-D. Ils montrent que l'image peut être décrite par une structure discrète, l'arbre des singularités. Ils l'appellent la Théorie de Morse Numérique, car elle est analogue à la théorie de Morse pour les fonctions définies sur un domaine continu : une fonction de Morse, c'est-à-dire une fonction deux fois continûment différentiable, pour laquelle la matrice hessienne n'est pas dégénérée aux points critiques, peut être décrite par un arbre des singularités (voir Milnor [55]). De données discrètes, un tableau tridimensionnel de niveaux de gris, ils définissent des fonctions interpolées qui sont consistantes topologiquement avec les données discrètes, et montrent qu'elles partagent le même arbre des singularités. Bien qu'ils fassent remarquer qu'utiliser les notions discrètes de connexité (elles sont deux : 4- et 8-connexité en 2-D, 6- et 26-connexité en 3-D) sans référence à une fonction interpolée puisse donner des inconsistances lorsque l'on prend l'opposé de l'image, ils ne poussent pas la remarque à sa conclusion naturelle : les ensembles de niveau supérieurs ne suffisent pas à décrire topologiquement l'image, car ils sont adaptés aux objets clairs, alors que les objets sombres ne sont pas correctement représentés dans l'arbre de Morse numérique.

Dans une étude sur les fonctions numériques définies sur un rectangle de \mathbb{R}^2 , publiée en 1950, Kronrod [34] évite cet inconvénient. En fait, les atomes dans son travail sont les composantes connexes des ensembles isoniveaux, qui sont des continus (des compacts connexes). Etant donnée une telle composante K et un voisinage U de K , si nous appelons ouverts les composantes connexes d'ensembles isoniveaux contenues dans U , la famille de tous ces ensembles forme une topologie sur l'ensemble des

¹En fait, pas les valeurs aux points mais les intervalles limités par les limites inférieures et supérieures des valeurs à ces points.

composantes connexes des ensembles isoniveaux de l'image. La surjection naturelle, qui à un point du rectangle associe la composante connexe d'ensemble isoniveau le contenant, est continue. Puisque le rectangle est connexe, localement connexe et compact, il en est de même pour l'espace topologique des composantes connexes des ensembles isoniveaux. Il montre de plus que cet espace ne contient aucun sous-ensemble homéomorphe au cercle S^1 , concluant que cet espace est en fait un *arbre*, dans le sens topologique du terme. De plus, il montre que l'arbre a au plus un nombre dénombrable de feuilles et de points de ramification, et que les feuilles sont des composantes des ensembles isoniveaux ne séparant pas le rectangle (ils peuvent être des extrema régionaux, mais aussi ce qu'il appelle des singularités concentriques), alors que les points de ramifications sont celles qui séparent le rectangle en au moins trois parties. Il appelle cet arbre l'arbre unidimensionnel de la fonction et décrit les fonctions qui sont de la même famille qu'une fonction donnée : elles sont obtenues en fusionnant des parties de l'arbre. Sous de nombreux aspects, cette construction est remarquable : la famille des composantes connexes des ensembles isoniveau est globalement invariante sous un changement de contraste, mais aussi sous une inversion du contraste (le « négatif » de la fonction), ce qui était la propriété manquant à l'arbre de Morse numérique. Néanmoins, du point de vue de la représentation des images, cela présente deux inconvénients : les ensembles isoniveaux sont nombreux et ne représentent pas un objet dans l'image, et l'arbre n'est pas ordonné, ce qui signifie qu'il n'a pas de racine objective. Le premier inconvénient n'est pas imputable au travail de Kronrod, puisque son intérêt n'était pas l'analyse d'image, mais plutôt l'étude des fonctions, mais il ne résout le second que partiellement, bien qu'il n'insiste pas sur le problème : si nous fixons un point du rectangle, les composantes d'ensemble isoniveau ne contenant pas ce point peuvent être ordonnées relativement à ce point. Ce que cela revient à faire, c'est d'isoler une composante connexe d'ensemble isoniveau (celle contenant le point fixé), et d'ordonner les autres relativement à celle-ci, donnant un arbre avec racine. Du point de vue de l'analyse d'image, une telle construction n'est pas pertinente, puisque le point est choisi arbitrairement.

Sous de nombreux aspects, notre travail est étroitement lié à celui de Kronrod. Nous ne traitons pas d'ensembles isoniveaux, mais de composantes connexes d'ensembles de niveau supérieurs et inférieurs, dont nous bouchons les trous. La notion de trou n'est pas rigide, et nous développons une approche axiomatique des définitions adéquates de trou. Le fait de boucher les trous permet de mélanger les ensembles de niveau supérieurs et inférieurs dans la même structure, un arbre, qui est orienté par l'inclusion. De cette manière, l'arbre décrit directement la topologie de l'image. Ceci est lié à l'article de Kronrod dans le sens que les frontières de nos atomes sont (des

parties connexes) des composantes connexes des ensembles isoniveaux (du moins pour une fonction continue), et que boucher les trous d'une composante connexe d'ensemble de niveau supérieur est exactement la même chose que boucher les trous de sa frontière (voir Proposition 2.18 dans le prochain chapitre). De cette manière, nous précisons ce qu'est l'*intérieur* d'une composante connexe d'ensemble isoniveau, cet intérieur étant défini de façon non ambiguë, et cela ordonne les atomes par inclusion. Cela conserve les avantages de l'arbre de Kronrod, à savoir les propriétés d'invariance par changement et inversion de contraste, tout en étant adapté à l'analyse d'image, car de nombreux objets dans l'image sont probablement formés d'atomes de notre représentation. De plus, nous gagnons en généralité car nos résultats sont valides pour des images semicontinues.

Dans le chapitre qui suit, nous exposons la construction de l'arbre, le principal outil étant la topologie. Le chapitre suivant traduit les résultats théoriques au cas discret et expose un algorithme rapide permettant d'obtenir l'arbre associé à une image, lequel arbre est suffisant pour reconstruire l'image et non redondant. Pour ces raisons, nous disons que l'arbre est une représentation forte de la topologie de l'image. Finalement, nous montrons que l'arbre peut être utilisé pour analyser et calculer l'effet de quelques filtres morphologiques récents et nouveaux.

Chapitre 2

L'arbre des formes en tant que représentation d'image

2.1 Motivation

Dans ce chapitre, nous montrons que sous certaines conditions topologiques concernant les images et leur ensemble de définition, les « formes » ont une structure d'arbre. Cette notion d'arbre n'est pas la notion classique, dans le sens que ce n'est pas une structure discrète, puisqu'elle peut comporter un nombre infini (et éventuellement non dénombrable) de nœuds, pourtant c'est une notion consistante avec celle-ci : deux nœuds sont connectés, et il n'y a pas de boucle.

Les formes d'une image sont construites à partir des composantes connexes d'ensembles de niveau. Il est bien connu que les composantes connexes des ensembles de niveau ont une structure d'arbre. La différence est qu'ici nous considérons simultanément ensembles de niveau supérieurs *et* inférieurs, et les formes ainsi construites sont stockées dans une unique structure, sans redondance. Cela peut sembler paradoxal, puisque la donnée des composantes connexes d'ensembles de niveau inférieur uniquement, *ou* la donnée des composantes connexes d'ensembles de niveau supérieur, sont chacune suffisantes pour reconstruire l'image. L'explication de ce paradoxe est que les formes ne sont pas construites à partir de toutes les composantes connexes, mais à partir d'une sélection d'entre elles, cette sélection s'effectuant bien entendu indépendamment du contraste. De plus, cette sélection est compatible avec ce qu'on entend par « objets » dans l'image, et élimine le fond. Nous ne prétendons pas résoudre l'ambiguïté fond-forme en général, mais cette ambiguïté n'apparaissant que pour des régions rencontrant le bord de l'image, la plupart du temps le bon choix est fait.

L'arbre des formes est complet et sans redondance. Ce que signifient ces propriétés est que la donnée des formes est suffisante pour reconstruire l'image (complétude) and qu'elle est nécessaire pour cette opération (absence de redondance), dans le sens qu'enlever une partie de l'arbre ne permet pas de reconstruire l'image, ou bien donne une image différente. C'est en ce sens que l'arbre des formes est une *représentation* de l'image. De plus, nous pensons que cet arbre est une représentation adaptée à l'analyse d'images, son invariance par changement de contraste n'étant pas le moindre de ses avantages.

Finalement, pour les images discrètes, un algorithme rapide permet la décomposition, la reconstruction étant triviale. Nous exposons cela dans le prochain chapitre.

2.2 Définitions de base

2.2.1 Cadre de travail

Sauf contre-indication, X sera un espace topologique connexe. Bien que ce soit un espace métrique la plupart du temps, nous préférons prouver les résultats dans un cadre plus général quand c'est possible, de telle sorte à mettre en valeur les propriétés significatives de l'espace qui sont utilisées. Nous appelons image une application de X dans \mathbb{R} . X devra parfois être localement connexe. Nous rappelons la définition de la connexité locale (voir [27] et [62, IV.6, Théoreme 6.4]) :

Définition 2.1 *Un espace topologique X est dit localement connexe lorsque les propriétés équivalentes suivantes sont vérifiées :*

1. X a une base de voisinages connexes ;
2. les composantes connexes de n'importe quel ouvert de X sont des ouverts.

Notons que la connexité locale est une propriété totalement indépendante du fait que la topologie soit ou non métrique.

La notion de connexité utilisée est la notion classique :

Définition 2.2 (Connexité) *Un espace topologique X est dit connexe si toute partition de X en deux ensembles fermés résulte en l'un des deux étant \emptyset et l'autre X . Un sous-ensemble de X est dit connexe lorsqu'il est connexe en tant qu'espace topologique (pour la topologie induite par celle de X).*

Cette définition peut se formuler également avec des partitions en deux ouverts (il suffit de considérer les complémentaires), ou en disant que les seuls ensembles fermés et ouverts de X sont \emptyset et X , ou en une formulation alternative : les seuls sous-ensembles de X ayant une frontière nulle sont \emptyset et X . D'autres notions de

connexité existent, comme par exemple la connexité par arcs, ou la connexité forte, mais nous restreignons la discussion à la notion classique.

Les deux résultats de base les plus importants qui nous seront utiles sont :

- L'union d'une famille de sous-ensembles connexes de X ayant une intersection non vide est connexe.
- Si $C \subset X$ est connexe et $C \subset D \subset \bar{C}$, alors D est connexe.

Le premier point implique que tout espace topologique X peut être partitionné en une famille de sous-ensembles connexes maximaux, et que cette décomposition est unique. Ses éléments se nomment les composantes connexes. Le second point implique que si C est connexe, \bar{C} est connexe, et une conséquence facile en est que les composantes connexes d'un ensemble S sont des fermés dans S (mais pas nécessairement des ouverts, excepté lorsque S est localement connexe, d'où l'intérêt de cette notion de connexité locale).

Une image est une application $u : X \rightarrow \mathbb{R}$; les ensembles de niveau inférieurs et supérieurs de u sont définis comme dans (1.1).

Il est clair que la famille des ensembles de niveau supérieurs est décroissante, tandis que la famille des ensembles de niveau inférieurs est croissante :

$$\forall \lambda \leq \mu, \quad \mathcal{X}^{\geq \lambda} u \supset \mathcal{X}^{\geq \mu} u, \quad \mathcal{X}^{< \lambda} u \subset \mathcal{X}^{< \mu} u. \quad (2.1)$$

Comme expliqué dans la Section 1.2, chacune de ces familles est une représentation invariante par changement de contraste de l'image, permettant de reconstruire une image à partir des Equations (1.4) et (1.5).

2.2.2 Des ensembles de niveau à leurs composantes

Bien qu'invariants par changement de contraste, les ensembles de niveau ne sont pas assez compatibles avec notre perception visuelle pour avoir quelque espoir de représenter des « objets » visuels. Il semble bien vrai que l'œil soit le plus à l'aise dans la comparaison de deux intensités lumineuses (bien plus que par exemple dans la comparaison des teintes), pourtant ces comparaisons ne semblent pas globales : il est capable d'isoler parmi deux régions *adjacentes* la plus claire, mais pour des régions non adjacentes, la comparaison ne semble pas sûre (voir Figure 2.1).

La conséquence est que les comparaisons globales ne sont pas significatives, c'est-à-dire que seules deux régions adjacentes devraient être comparées. L'information restante est représentée dans la Figure 2.2. Les flèches dans cette figure représentent la relation « plus lumineux que ». Cette relation est transitive, mais observons qu'elle ne permet pas de comparer les niveaux de gris des deux carrés.

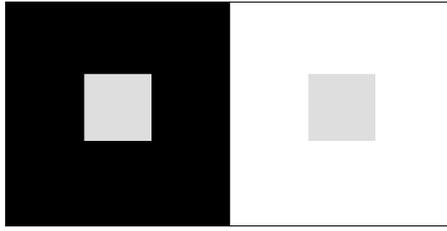


FIG. 2.1 – Dans la comparaison des deux petits carrés gris, l'œil n'ordonne pas facilement leurs niveaux de gris. Le petit carré de gauche pourrait apparaître plus clair que celui de droite, alors qu'ils ont objectivement la même luminosité.

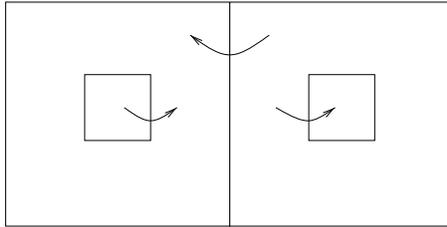


FIG. 2.2 – Information restante dans l'image de la Figure 2.1 lorsque seules les comparaisons locales sont valables. Les flèches représentent la relation d'ordre « plus lumineux que ».

De plus, toute région homogène apparaît comme un « objet », c'est-à-dire non coupé par l'œil. Cela nous amène à travailler sur des composantes connexes des ensembles de niveau plutôt que sur les ensembles de niveau complets. Le fait que deux régions soient des composantes connexes du même ensemble de niveau n'est pas une information significative, nous ne comparons pas leur niveau de gris. C'est le cas pour les petits carrés gris dans la Figure 2.1.

Notation : Etant donné un point \mathbf{x} dans $\mathcal{X}^{\geq \lambda} u$, notons $cc([u \geq \lambda], \mathbf{x})$ la composante connexe de $\mathcal{X}^{\geq \lambda} u$ contenant \mathbf{x} . Par convention, si $\mathbf{x} \notin \mathcal{X}^{\geq \lambda} u$, $cc([u \geq \lambda], \mathbf{x})$ est \emptyset . Une notion similaire s'applique à $cc([u < \mu], \mathbf{x})$.

Nous déduisons évidemment des Equations (1.4) et (1.5) les formules de reconstruction :

$$u(\mathbf{x}) = \inf \{ \mu / cc([u < \mu], \mathbf{x}) \neq \emptyset \} \quad (2.2)$$

$$u(\mathbf{x}) = \sup \{ \lambda / cc([u \geq \lambda], \mathbf{x}) \neq \emptyset \}. \quad (2.3)$$

La monotonie des ensembles de niveau se traduit en une structure d'arbre pour leurs composantes connexes. Puisque leur nombre n'est pas forcément fini, nous devons définir une notion plus générale d'arbre.

Définition 2.3 Soit \mathcal{E} une famille d'ensembles et \preceq une relation d'ordre partiel dans \mathcal{E} . Nous disons que \preceq induit une structure d'arbre dans \mathcal{E} si les deux conditions suivantes sont vérifiées :

1. $\exists R \in \mathcal{E}, \quad \forall E \in \mathcal{E}, E \preceq R;$
2. $\forall A, B, C \in \mathcal{E},$

$$\left. \begin{array}{l} A \preceq B \\ A \preceq C \end{array} \right\} \Rightarrow B \text{ et } C \text{ sont comparables.}$$

La première condition exprime la connexité de la structure, R étant la racine de l'arbre, et la seconde condition implique qu'il n'y a pas de boucle, parce que, étant donnés quatre ensembles $A, B, C, D \in \mathcal{E}$, la situation suivante est impossible :

$$\begin{array}{c} A \subset B \subset D \\ A \subset C \subset D \\ B \text{ et } C \text{ non comparables.} \end{array}$$

Un cas particulier est celui où la relation d'ordre l'inclusion entre ensembles, auquel cas nous parlons d'*arbre d'inclusion*.

Avec cette définition, nous prouvons la structure d'arbre des composantes connexes des ensembles de niveau.

Proposition 2.4 Soit u une image. Soit $A = \text{cc}([u \geq \lambda], \mathbf{x})$ (resp. $A = \text{cc}([u < \lambda], \mathbf{x})$) et $B = \text{cc}([u \geq \mu], \mathbf{y})$ (resp. $B = \text{cc}([u < \mu], \mathbf{y})$). Supposons que $A \cap B \neq \emptyset$. Alors soit $A \subset B$, soit $B \subset A$.

Preuve. Supposons, dans perte de généralité, que $\lambda \leq \mu$. Alors nous avons $[u \geq \mu] \subset [u \geq \lambda]$, donc $B \subset [u \geq \lambda]$. Soit $\mathbf{z} \in A \cap B$, alors clairement $A = \text{cc}([u \geq \lambda], \mathbf{z})$, et puisque B est connexe, contient \mathbf{z} and est contenu dans $[u \geq \lambda]$, nous déduisons que $B \subset A$.

Le cas des composantes connexes d'ensembles de niveau inférieurs se traite de la même manière. □

Ceci implique (et est plus fort que) la structure d'arbre d'inclusion :

Corollaire 2.5 Pour une image bornée u , l'ensemble des ensembles de niveau inférieurs $\mathcal{X}^< u$ et l'ensemble des ensembles de niveau supérieurs $\mathcal{X}^> u$ sont chacun des arbres d'inclusion.

Preuve. La racine est l'ensemble de définition de u . Si A , B et C sont des ensembles de niveau inférieurs, $A \subset B$ et $A \subset C$, nous obtenons $A \subset B \cap C$, ce qui prouve que $B \cap C \neq \emptyset$ et, en utilisant Proposition 2.4, que B et C sont comparables pour l'ordre d'inclusion. La preuve est similaire pour $\mathcal{X}^{\geq u}$. \square

2.2.3 Au-delà des composantes d'ensembles de niveau

Le résultat simple montré ci-dessus n'est qu'une petite extension de l'Equation (2.1). Pourtant, c'est une amélioration substantielle de ces formules car il représente plus fidèlement les objets dans l'image. Nous avons obtenu la localité, ce qui était l'une des principales motivations de ce travail. Dans ces deux arbres, nous nous attendons à trouver les objets significatifs perçus par l'œil. En ce sens, ces arbres semblent utiles pour l'analyse d'image.

Le problème avec leur usage est lié à la reconstruction. Il est reconnu que les arbres sont une information suffisante pour reconstruire l'image dont ils sont extraits, mais elle est redondante. Puisque chaque arbre représente exactement l'image, si nous voulons traiter aussi bien les ensembles de niveau supérieurs qu'inférieurs (ce qui est le cas), les manipulations de ces arbres posent problème. Par exemple, l'opération de base que nous voudrions effectuer sur un arbre est d'enlever un nœud. Puisque l'autre arbre n'est pas lié (excepté qu'il représente la même image initialement), il doit être réextrait de telle sorte à représenter à nouveau l'image du premier arbre. Il n'y a pas de solution rapide à cela ; nous devons reconstruire l'image à partir de l'arbre modifié puis extraire l'autre arbre. Cet inconvénient est dû à l'absence de lien entre les deux arbres. Alors que l'information d'inclusion est codée pour les composantes du même type d'ensemble de niveau dans leur arbre, il n'y a pas de telle information entre les composantes de différents types d'ensemble de niveau. Ce n'est pas une surprise car de telles composantes ne sont pas emboîtées, c'est-à-dire que nous ne pouvons pas conserver une structure d'arbre d'inclusion avec toutes les composantes des ensembles de niveau aussi bien supérieurs qu'inférieurs.

La Figure 2.3 illustre le fait que les deux arbres peuvent avoir des structures très différentes. Puisqu'aucune d'elle ne devrait être privilégiée, l'utilisation de leur structure d'arbre est un problème. Cet exemple suggère ce qui manque dans les deux arbres. Leur lien est lié à la notion de trous. Dans cette figure, D est un trou dans F , et c'est information est digne d'intérêt du point de vue de l'analyse d'image.

Puisque chaque arbre représente exactement l'image, la donnée des deux est à

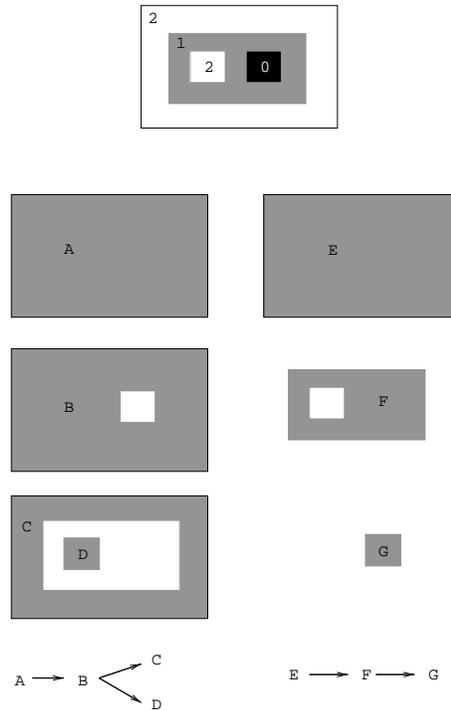


FIG. 2.3 – Haut : une image élémentaire avec trois « objets » : deux carrés et un rectangle. Colonne de gauche : les composantes connexes des ensembles de niveau supérieurs avec des seuils croissant de haut en bas. Colonne de droite : les composantes connexes d'ensembles de niveau inférieurs avec des seuils décroissant de haut en bas. Bas : les deux arbres associés, où les flèches représentent la relation « contient ». Les deux carrés, significatifs du point de vue de l'analyse d'image, sont de différents types et donc apparaissent dans des arbres différents, montrant que les deux arbres sont intéressants. Alors que le carré G est inclus dans le rectangle F , il n'y a pas de lien entre D et F .

la fois trop (puisque'il y a redondance) et pas assez puisque une information aussi importante que la relation d'être un « trou » dans un objet n'apparaît pas dans ces données.

Tous ces problèmes ont une solution commune : au lieu de considérer les composantes connexes des ensembles de niveau, nous travaillons avec les composantes connexes des ensembles de niveau *dont nous remplissons les trous*. Cette opération élémentaire donne ce que nous appelons des *formes*. Les formes gardent les mêmes propriétés que les composantes connexes d'ensembles de niveau : localité et insensibilité au changement de contraste. La relation entre composantes connexes de types différents « est un trou dans » se traduit dans ce cadre par la relation « est contenu dans ». Heureusement, cette opération reste consistante avec l'analyse d'image. Puisque nous vivons dans un monde dont de nombreux objets sont « pleins », un trou dans leur projection sur une image doit être dû à une occlusion, et représenter de telles projections sans leurs trous est fidèle au vrai objet.

La redondance entre les deux arbres est automatiquement enlevée. En prenant l'exemple de la Figure 2.3, les formes basées sur les composantes A , B , C et E sont les mêmes : l'image dans son intégralité. Les formes déduites de D et G sont D et G eux-mêmes, puisque ces composantes n'ont pas de trou. Au contraire, la composante F devient un rectangle plein F' et D , qui était un trou dans F , est un sous-ensemble de F' . Comme le montre la Figure 2.4, les formes ont une structure d'arbre d'inclusion.

Dans la section suivante, nous nous intéressons aux conditions sous lesquelles une image définie sur un domaine continu peut être représentée par un arbre de formes. Ceci impliquera la définition de la notion de trou et du concept de saturation.

2.3 Arbre d'inclusion des formes

Le but de cette section est de montrer que les formes extraites d'une image ont une structure d'arbre d'inclusion, et d'examiner les possibilités de reconstruction d'une image à partir de ses formes. Sous ces conditions, la décomposition d'une image en ses formes sera une représentation d'image puissante, bien adaptée à l'analyse d'image.

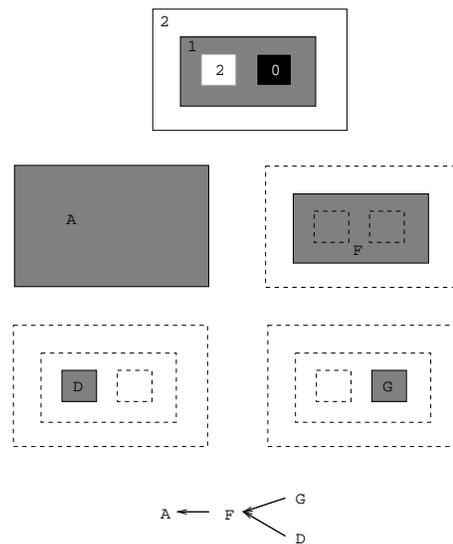


FIG. 2.4 – Formes de l'image élémentaire de la Figure 2.4. La composante F de la Figure 2.4 devient pleine ici ; D et G ne changent pas puisqu'elles ne comportent pas de trou, et toutes les autres composantes deviennent A , l'image entière. L'image est représentée par un unique arbre d'inclusion, où les ensembles de niveau inférieurs et supérieurs ont la même importance. Notons que l'inversion du contraste (prenant l'opposé de chaque niveau de gris) donnerait la même structure d'arbre.

2.3.1 Définition axiomatique de la saturation, des trous et des formes

Il s'avère que les principaux résultats de ce chapitre sont valides dans une grande variété de situations. La définition de trous n'a pas besoin d'être totalement figée. Cette liberté représente un avantage, car elle permet de définir des formes dans différentes situations, les deux plus importantes étant le cas d'une image définie sur \mathbb{R}^n et d'une image définie sur un sous-ensemble de \mathbb{R}^n . Les exigences pour la définition de trous sont incluses ci-dessous dans la Définition 2.6.

X sera un espace topologique connexe. Nous ne nous restreindrons pas au cas de \mathbb{R}^n , car la structure d'espace vectoriel sur X est inutile, bien que les cas pratiques concernent des sous-ensembles de \mathbb{R}^n . Nous appelons image une application de X dans \mathbb{R} .

La définition suivante expose les exigences pour un opérateur de saturation. L'opérateur de saturation est l'opérateur qui transforme les composantes connexes d'ensembles de niveau en « formes ». Cet opérateur remplit les trous des composantes connexes d'ensembles de niveau. La nécessité de ces conditions apparaîtra plus loin.

Définition 2.6 *Nous disons sat est un opérateur de saturation sur X si*

$$\text{sat} : \mathcal{P}(X) \rightarrow \mathcal{P}(X)$$

et

1. $\forall A \subset X$, $X \setminus \text{sat}(A)$ est soit \emptyset , soit une composante connexe de $X \setminus A$;
2. sat est croissant par rapport à l'inclusion des sous-ensembles de X ;
3. $\forall A \subset X$, $\text{sat}(X \setminus \text{sat}(A)) = X$ ou \emptyset ;
4. $\text{sat} \circ \text{sat} = \text{sat}$.

La définition d'une saturation nous permet de parler des trous et de l'extérieur d'un ensemble :

Définition 2.7 *Soit sat une saturation sur X et $A \subset X$. Nous appelons trous de A (par rapport à sat) les composantes connexes de $X \setminus A$ incluses dans $\text{sat}(A)$. Nous appelons extérieur de A (par rapport à sat) l'ensemble $X \setminus \text{sat}(A)$, que nous notons $\text{Ext } A$.*

Notation : Nous noterons par \mathcal{H}_A l'ensemble des trous de A , et $\text{Ext } A$ l'extérieur de A . Nous avons donc l'identité

$$\text{sat}(A) = A \cup \bigcup_{H \in \mathcal{H}_A} H,$$

où les unions sont disjointes.

Notons que les définitions de trous et d'extérieur dépendent de l'opérateur de saturation choisi sur X . Mais nous ne considérerons jamais plusieurs saturations en même temps, de telle sorte que le contexte sera suffisamment clair pour enlever l'ambiguïté.

Dans la Définition 2.6, la condition 1 exprime le fait que nous ajoutons à A toutes les composantes connexes de son complémentaire (nommées les trous), sauf éventuellement une (appelée l'extérieur). Le point 2 exprime le fait que l'opérateur de saturation est monotone et le point 3 est technique, signifiant que nous ne pouvons trouver une partition non triviale de X en deux formes. La dernière condition 4 signifie qu'une fois qu'un ensemble est plein, il ne reste plus de trou à remplir.

Comme nous l'avons écrit plus haut, la saturation transforme les composantes connexes des ensembles de niveau en formes :

Définition 2.8 *Etant donnée une image u , nous appelons formes de type inférieur (resp. supérieur) les ensembles*

$$\text{sat}(\text{cc}([u < \mu], \mathbf{x})) \text{ (resp. } \text{sat}(\text{cc}([u \geq \lambda], \mathbf{x}))).$$

Des exemples d'opérateurs de saturation intéressants seront exposés plus loin, mais en voici un trivial : considérons l'opérateur qui transforme \emptyset en \emptyset ou X et tout autre ensemble en X . Cet opérateur détruit toute information des composantes connexes d'ensembles de niveau d'une image et interdit la reconstruction d'une image à partir de ses formes, ce qui est pourtant un de nos objectifs.

2.3.2 Saturation du complémentaire

Nous déduisons de la définition les propriétés essentielles d'un opérateur de saturation sur un espace topologique X .

Définition 2.9 *Nous disons que $A \subset X$ est un ensemble simple lorsque A est connexe et $\text{sat}(A) = A$.*

En d'autres termes, un ensemble simple est un ensemble connexe sans trou, c'est-à-dire un point fixe connexe de sat .

Le premier résultat est qu'un trou dans un ensemble connexe est un ensemble simple, ou alors que sa saturation est X .

Lemme 2.10 *Soit A un sous-ensemble connexe de X et H un trou de A . Alors, soit H est un ensemble simple, soit $\text{sat}(H) = X$, le dernier cas impliquant $\text{sat}(A) = X$.*

Preuve. H étant une composante connexe du complémentaire d'un ensemble connexe (A) dans un ensemble connexe, nous savons que $X \setminus H$ est connexe (voir [62, IV.3, Theorem 3.3]). Donc cet ensemble est soit un trou de H , auquel cas $\text{sat}(H) = X$, ou l'extérieur de H , auquel cas $\text{sat}(H) = H$.

Si $\text{sat}(H) = X$, alors puisque $H \subset \text{sat}(A)$, la monotonie de sat donne

$$X = \text{sat}(H) \subset \text{sat}(\text{sat}(A)) = \text{sat}(A).$$

□

Cela donne immédiatement

Corollaire 2.11 *Soit A un sous-ensemble connexe de X et H un trou de A . Alors $\text{sat}(H) \subset \text{sat}(A)$.*

Remarque 2.1. Ce résultat, essentiel pour les principaux théorèmes que nous prouverons, est une justification pour l'exigence du quatrième axiome de l'opérateur de saturation, l'idempotence. Le troisième axiome était également essentiel.

Le Lemme 2.10, joint à l'axiome 3 de la Définition 2.6, définit complètement ce qu'est la saturation d'une composante connexe du complémentaire d'un ensemble connexe A .

2.3.3 Propriétés de la saturation

Nous nous intéressons ici aux propriétés topologiques des ensembles simples, en particulier leur position relativement à leur frontière. Il s'avère que les situations pathologiques ne peuvent se produire lorsque l'espace X est localement connexe (voir la Définition 2.1). Notons que grâce à l'idempotence de l'opérateur de saturation (axiome 4 de la Définition 2.6), les ensembles simples sont l'image par l'opérateur de saturation de certains ensembles, en d'autres termes des ensembles qui sont déjà saturés. La réciproque (i.e., la saturation d'un ensemble est un ensemble simple) serait vraie à condition que cet ensemble saturé soit connexe.

La saturation conserve la connexité

D'abord, nous prouvons que la saturation préserve la connexité. Ceci sera une conséquence directe du lemme suivant :

Lemme 2.12 *Soit X un espace topologique connexe. Supposons que X est localement connexe. Si $A \subset X$, A est connexe et H est une composante connexe de $X \setminus A$, alors $A \cup H$ est connexe.*

Preuve. Supposons que $A \cup H$ n'est pas connexe. Alors A et H étant connexes, ce sont les composantes connexes de $A \cup H$. Donc A et H sont fermés dans $A \cup H$, et chacun étant le complémentaire de l'autre dans cet espace, ils sont aussi ouverts. Donc, il existe un ouvert U de X tel que $H \subset U$ et $U \cap A = \emptyset$. Nous pouvons supposer U connexe, sinon il suffit de prendre la composante connexe de U qui contient A (il en existe une puisque A est connexe), et cette composante est ouverte puisque X est localement connexe. U est donc connexe, incluse dans $X \setminus A$ et contient H . Puisque H est une composante connexe de $X \setminus A$, ceci implique $H = U$, un ouvert.

Comme H est ouvert dans $A \cup H$, $\overline{H} \cap A = \emptyset$, et \overline{H} étant connexe, $H = \overline{H}$. Comme $\emptyset \neq H \neq X$, le fait que H soit ouvert et fermé est une contradiction avec la connexité de X . \square

Remarque 2.2. Ce résultat est très proche du Théorème 3.4 dans [62, IV,3] : si H est ouvert et fermé dans $X \setminus A$ et X est connexe (X n'a pas besoin d'être localement connexe), $A \cup H$ est connexe. Le Lemme 2.12 n'est toutefois pas une conséquence de ce résultat, puisqu'une composante connexe de $X \setminus A$ n'est pas nécessairement ouverte dans $X \setminus A$ (même si X est localement connexe, car $X \setminus A$ n'est pas supposé ouvert).

Comme annoncé, ce lemme nous permet de prouver la propriété de conservation de la connexité de la saturation :

Proposition 2.13 *Soit X un espace topologique connexe et localement connexe, sat un opérateur de saturation sur X et $A \subset X$ un ensemble connexe. Alors $\text{sat}(A)$ est connexe.*

Preuve. Il suffit d'écrire

$$\text{sat}(A) = \bigcup_{H \in \mathcal{H}_A} (A \cup H),$$

une union d'ensembles connexes (grâce au Lemme 2.12) ayant une intersection non vide (A). $\text{sat}(A)$ est alors connexe. \square

Comme conséquence de la Proposition 2.13, toutes les propriétés prouvées ci-dessous s'appliquent aux formes d'une image définie sur X , puisque les formes sont des ensembles simples.

La saturation conserve la topologie

Nous démontrons maintenant que la saturation conserve la topologie :

Lemme 2.14 *Soit X un espace connexe, sat une saturation sur X et $A \subset X$. Si A est ouvert, $\text{sat}(A)$ est aussi ouvert. Si X est localement connexe et A est fermé, alors $\text{sat}(A)$ est aussi fermé.*

Preuve. Si $\text{sat}(A) = X$, les assertions deviennent triviales, donc nous supposons que ce n'est pas le cas.

$X \setminus \text{sat}(A)$ est une composante connexe de $X \setminus A$, de telle sorte que c'est un fermé de $X \setminus A$, qui est fermé pourvu que A soit ouvert. Donc $X \setminus \text{sat}(A)$ est fermé dans X , ce qui prouve que $\text{sat}(A)$ est ouvert.

Si A est fermé, alors $X \setminus A$ est ouvert, et $X \setminus \text{sat}(A)$ est une composante connexe de $X \setminus A$, donc $X \setminus \text{sat}(A)$ est ouvert (puisque X est localement connexe), prouvant que $\text{sat}(A)$ est fermé. \square

Remarque 2.3. Une conséquence directe du Lemme 2.14 est que les seules formes d'une image *semicontinue supérieurement* u qui soient de types inférieur et supérieur sont \emptyset et X . En effet, puisque les composantes connexes d'ensembles de niveau supérieurs (resp. inférieurs) sont fermées (resp. ouvertes puisque X est localement connexe), leur saturation est également fermée (resp. ouverte). Donc une forme étant à la fois de types inférieur et supérieur serait ouverte et fermée, la connexité de X impliquant que cette forme serait X ou \emptyset . Remarquons que ceci devient faux lorsque u n'est pas semicontinue supérieurement, comme le montre la Figure 2.5.

Frontière des ensembles saturés

Nous montrons maintenant que la frontière de la saturation d'un ensemble A est un sous-ensemble de la frontière de A . Pour cela, le lemme suivant sera utilisé (voir [35, §44,III,3]) :

Lemme 2.15 *Si A est un sous-ensemble d'un espace localement connexe X , et $\{A_i, i \in I\}$ sont ses composantes connexes, alors*

$$\bigcup_{i \in I} \partial A_i \subset \partial A.$$

Preuve. Soit $i \in I$. D'une part, nous avons :

$$\partial A_i \subset \overline{A_i} \subset \overline{A}.$$

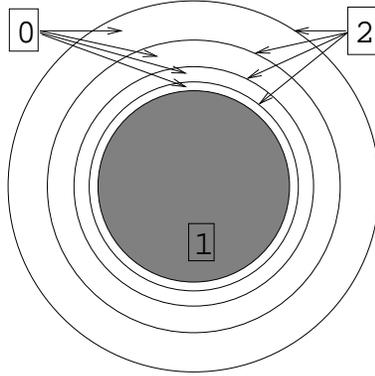


FIG. 2.5 – Pour une image qui n'est pas semicontinue supérieurement, une forme non triviale peut être de types inférieur *et* supérieur. Dans cet exemple, le disque central est approximé par une suite de cercles décroissants au niveau 2, alors que les anneaux entre les cercles sont au niveau 0. Ce disque est une composante connexe de $\mathcal{X}^{\geq 1} u$ et $\mathcal{X}^{< 2} u$, sans trous pour la saturation naturelle de \mathbb{R}^2 (voir Section 2.4).

D'autre part, $X \setminus A \subset \overline{X \setminus A}$, de telle sorte qu'en prenant le complémentaire de chaque membre nous obtenons

$$A \supset X \setminus \overline{X \setminus A}. \quad (*)$$

Alors

$$\begin{aligned} (\partial A_i) \cap (X \setminus \overline{X \setminus A}) &\subset (\partial A_i) \cap A \\ &\subset \overline{A_i} \cap A \\ &\subset A_i, \end{aligned} \quad (**)$$

la dernière inclusion venant du fait que $\overline{A_i} \cap A = A_i$, exprimant le fait que A_i est fermé dans A , puisque c'est une composante connexe de A . Comme $X \setminus \overline{X \setminus A}$ est ouvert et X est localement connexe, ses composantes connexes sont aussi ouvertes. Grâce à (*), chaque composante connexe de $X \setminus \overline{X \setminus A}$ est contenue dans une composante connexe de A . Donc, $X \setminus \overline{X \setminus A}$ étant ouvert, chacune de ses composantes connexes est contenue dans l'intérieur d'une composante connexe de A . Grâce à (**), nous obtenons

$$(\partial A_i) \cap (X \setminus \overline{X \setminus A}) \subset \overset{\circ}{A}_i$$

ce qui implique que $(\partial A_i) \cap (X \setminus \overline{X \setminus A}) = \emptyset$ car $(\partial A_i) \cap \overset{\circ}{A}_i = \emptyset$, c'est-à-dire

$$\partial A_i \subset \overline{X \setminus A}.$$

□

Remarque 2.4. Sans hypothèses supplémentaires, l'inclusion inverse est fautive (voir néanmoins [35, §44,III,1]). Prenons comme exemple $X = \mathbb{R}$ avec la topologie usuelle et $A = \mathbb{Q}$. Alors $\partial A = X$ alors que les composantes connexes de A sont composées d'un rationnel, donc pour tout i , $\overline{A_i} = A_i$ et $\bigcup_i \overline{A_i} = A$. Néanmoins, si I est fini, le fait que les A_i sont des composantes connexes de A implique $A = \bigcup_{i \in I} A_i$, ce qui est suffisant pour prouver l'inclusion inverse.

Proposition 2.16 *Si X est localement connexe et $A \subset X$,*

$$\partial \text{sat}(A) \subset \partial A.$$

Preuve. Si $\text{sat}(A) = X$, nous obtenons $\partial \text{sat}(A) = \emptyset$ et le résultat est évident. Maintenant, supposons que $X \setminus \text{sat}(A) \neq \emptyset$.

$\partial \text{sat}(A) = \partial(X \setminus \text{sat}(A))$ et $X \setminus \text{sat}(A)$ est une composante connexe de $X \setminus A$. Donc, selon le Lemme 2.15,

$$\partial(X \setminus \text{sat}(A)) \subset \partial(X \setminus A),$$

signifiant $\partial \text{sat}(A) \subset \partial A$. □

Le prochain résultat important relie la saturation d'un ensemble à la saturation de sa frontière. Le lemme simple suivant sera utilisé (voir [62, IV,3,Théorème 3.2]) :

Lemme 2.17 *Soit X un espace topologique et $A \subset X$ un ouvert connexe. Alors A est une composante connexe de $X \setminus \partial A$.*

Preuve. Puisque A est ouvert, $A \subset X \setminus \partial A$ et de plus

$$A = \overline{A} \cap (X \setminus \partial A),$$

prouvant que A est fermé dans $X \setminus \partial A$, et comme il est aussi ouvert dans cet ensemble et connexe, c'est une composante connexe de $X \setminus \partial A$. □

Le lemme ci-dessus sera utilisé au cours de la démonstration de la proposition importante :

Proposition 2.18 *Soit X un espace topologique connexe et localement connexe et $A \subset X$ tel que $\text{sat}(A) \neq X$. Alors $\text{sat}(A) \subset \text{sat}(\partial A)$, et si A est fermé, nous obtenons $\text{sat}(A) = \text{sat}(\partial A)$.*

Preuve. Montrons d'abord l'égalité lorsque A est fermé. D'une part, nous avons $\partial A \subset A$, de telle sorte que

$$\partial A \subset \text{sat}(\partial A) \subset \text{sat}(\overline{A}) = \text{sat}(A).$$

D'autre part, soit C une composante connexe de $\overset{\circ}{A}$. Comme X est localement connexe, C est ouvert. Appliquant le Lemme 2.17 à C , C est une composante connexe de $X \setminus \partial C$. Avec le Lemme 2.15, nous savons que $\partial C \subset \partial \overset{\circ}{A}$, alors $X \setminus \partial \overset{\circ}{A} \subset X \setminus \partial C$ et comme il est clair que $C \cap \partial \overset{\circ}{A} = \emptyset$, il s'ensuit que C est une composante connexe de $X \setminus \partial \overset{\circ}{A}$. Comme $C \subset A$, $\text{sat}(C) \neq X$, ainsi nous trouvons $C \subset \text{sat}(\partial \overset{\circ}{A})$ et comme $\partial \overset{\circ}{A} \subset \partial A$ nous déduisons $C \subset \text{sat}(\partial A)$. Finalement, cela donne

$$\overset{\circ}{A} \subset \text{sat}(\partial A)$$

et donc $A = \overset{\circ}{A} \cup \partial A \subset \text{sat}(\partial A)$ et du fait de la monotonie de la saturation, en prenant la saturation de chaque membre nous obtenons $\text{sat}(A) \subset \text{sat}(\partial A)$.

Maintenant, prouvons l'inégalité lorsque A est un sous-ensemble arbitraire de X tel que $\text{sat}(A) \neq X$. Nous avons $A \subset \overline{A}$, et par croissance de l'opérateur de saturation, $\text{sat}(A) \subset \text{sat}(\overline{A})$. Comme \overline{A} est fermé, si $\text{sat}(\overline{A}) \neq X$, nous utilisons l'égalité pour les ensembles fermés :

$$\text{sat}(\overline{A}) = \text{sat}(\partial \overline{A})$$

et puisque $\partial \overline{A} \subset \partial A$, cela donne $\text{sat}(A) \subset \text{sat}(\partial A)$.

Il reste à montrer la même inégalité lorsque $\text{sat}(\overline{A}) = X$. Si de plus $\text{sat}(\partial A) = X$, le résultat est évident. Sinon, $\text{Ext } \partial A$ est un sous-ensemble connexe de $X \setminus \partial A$, donc ne peut rencontrer à la fois A et $X \setminus A$.

Si $\text{Ext } \partial A \subset A$, nous obtenons

$$\text{sat}(\text{Ext } \partial A) \subset \text{sat}(A) \neq X$$

et par le troisième axiome de la Définition 2.6, $\text{sat}(\text{Ext } \partial A) = \emptyset$, ce qui est exclu.

Donc $\text{Ext } \partial A \subset X \setminus A$, et comme c'est un connexe, il est inclus dans une composante connexe de $X \setminus A$. Il ne peut être inclus dans un trou H de A pour la même

raison qu'il n'est pas dans A : sinon nous aurions

$$\text{sat}(\text{Ext } \partial A) \subset \text{sat}(H)$$

et $\text{sat}(H) = H$ suivant le Lemme 2.10 et donc $\text{sat}(\text{Ext } \partial A) \subset \text{sat}(A)$, ce qui a été prouvé impossible.

La seule possibilité restante est $\text{Ext } \partial A \subset \text{Ext } A$, ce qui revient à écrire $\text{sat}(A) \subset \text{sat}(\partial A)$. \square

2.3.4 Décomposition d'une image en formes

Les résultats ci-dessus concernant les propriétés de l'opérateur de saturation sont les outils nécessaires pour prouver que les formes ont une structure d'arbre d'inclusion. Toutefois, ceci nécessite des hypothèses supplémentaires sur l'espace X , qui, comme nous verrons, sont vérifiées avec \mathbb{R}^n .

Notre première proposition est la partie facile de notre théorème général, et n'exige pas d'hypothèses supplémentaires concernant X . Elle compare la saturation de composantes connexes du même type d'ensemble de niveau.

Proposition 2.19 *Soit X un espace connexe et localement connexe et u une image définie sur X . Soit A et B deux formes de u du même type telles que $A \cap B \neq \emptyset$. Alors soit $A \subset B$, soit $B \subset A$.*

Preuve. Supposons que tous deux puissent être écrits comme formes supérieures, $A = \text{sat}(\text{cc}([u \geq \lambda], \mathbf{x}))$ et $B = \text{sat}(\text{cc}([u \geq \lambda'], \mathbf{y}))$ (le cas où ils sont tous deux de type inférieur est symétrique).

1. $\text{cc}([u \geq \lambda'], \mathbf{y}) \cap \text{cc}([u \geq \lambda], \mathbf{x}) \neq \emptyset$. Sans perte de généralité, supposons par exemple $\lambda < \lambda'$. Alors, selon la Proposition 2.4, nous obtenons $\text{cc}([u \geq \lambda'], \mathbf{y}) \subset \text{cc}([u \geq \lambda], \mathbf{x})$, et en appliquant l'opérateur de saturation à chaque membre

$$B = \text{sat}(\text{cc}([u \geq \lambda'], \mathbf{y})) \subset \text{sat}(\text{cc}([u \geq \lambda], \mathbf{x})) = A,$$

ce qui est le résultat annoncé.

Si ce cas n'est pas vérifié, cela signifie que $\text{cc}([u \geq \lambda'], \mathbf{y})$ est dans $X \setminus \text{cc}([u \geq \lambda], \mathbf{x})$ et, étant connexe, soit dans un trou de $\text{cc}([u \geq \lambda], \mathbf{x})$ soit dans l'extérieur de $\text{cc}([u \geq \lambda], \mathbf{x})$. Examinons successivement chacun de ces cas.

2. $\text{cc}([u \geq \lambda'], \mathbf{y}) \subset H$ où H est un trou dans $\text{cc}([u \geq \lambda], \mathbf{x})$. Alors par monotonie de sat , $B \subset \text{sat}(H)$ et en appliquant le Corollaire 2.11, $\text{sat}(H) \subset \text{sat}(A) = A$, donc

$B \subset A$.

3. $\text{cc}([u \geq \lambda], \mathbf{y}) \subset X \setminus A$. Puisque $A \cap B \neq \emptyset$, alors A rencontre un trou H de $\text{cc}([u \geq \lambda], \mathbf{y})$, et puisqu'il est connexe et contenu dans $X \setminus \text{cc}([u \geq \lambda], \mathbf{y})$, A est inclus dans H , et cela ramène au cas précédent, inversant les rôles de A et B . \square

Voici la partie difficile de notre théorème. Elle traite de la comparaison des saturations des composantes connexes des ensembles de niveau de différents types. Notons que cela implique une hypothèse forte sur la frontière de la forme ouverte, ce qui explique pourquoi des hypothèses supplémentaires sur X sont demandées; de cette manière, l'hypothèse est automatiquement vérifiée pour toute forme ouverte. Remarquons que la proposition est formulée de telle façon que les deux composantes connexes ont un point (\mathbf{x}) en commun.

Proposition 2.20 *Soit u une image semicontinue supérieurement sur X , $A = \text{sat}(\text{cc}([u \geq \lambda], \mathbf{x}))$ et $B = \text{sat}(\text{cc}([u < \mu], \mathbf{x}))$ deux formes de u . Supposons également que ∂B est connexe. Alors, soit $A \subset B$, soit $B \subset A$.*

Preuve.

1. Puisque $\mathbf{x} \in \text{cc}([u \geq \lambda], \mathbf{x}) \cap \text{cc}([u < \mu], \mathbf{x})$, $\lambda \leq u(\mathbf{x}) < \mu$. Comme u est semicontinue supérieurement, $\text{cc}([u \geq \lambda], \mathbf{x})$ est fermé et $\text{cc}([u < \mu], \mathbf{x})$ est ouvert, et grâce au Lemme 2.14, A est fermé et B est ouvert.

2. Par définition, $\text{cc}([u < \mu], \mathbf{x})$ est une composante connexe de $\mathcal{X}^{<\mu} u$, donc fermé dans $\mathcal{X}^{<\mu} u$, ce qui signifie $\overline{\text{cc}([u < \mu], \mathbf{x})} \cap \mathcal{X}^{<\mu} u = \text{cc}([u < \mu], \mathbf{x})$; alors puisque $\text{cc}([u < \mu], \mathbf{x})$ est ouvert,

$$\partial \text{cc}([u < \mu], \mathbf{x}) = \overline{\text{cc}([u < \mu], \mathbf{x})} \setminus \text{cc}([u < \mu], \mathbf{x})$$

et donc

$$\partial \text{cc}([u < \mu], \mathbf{x}) \cap \mathcal{X}^{<\mu} u = \overline{\text{cc}([u < \mu], \mathbf{x})} \cap \mathcal{X}^{<\mu} u \setminus \text{cc}([u < \mu], \mathbf{x}) = \emptyset.$$

Donc $\partial \text{cc}([u < \mu], \mathbf{x}) \subset \mathcal{X}^{\geq \mu} u \subset \mathcal{X}^{\geq \lambda} u$ car $\lambda < \mu$.

3. Comme, selon la Proposition 2.16, $\partial B \subset \partial \text{cc}([u < \mu], \mathbf{x})$, nous avons également $\partial B \subset \mathcal{X}^{\geq \lambda} u$.

4. Si $\partial B \cap A = \emptyset$, comme A est connexe, nous avons soit $A \subset B$, soit $A \subset X \setminus \overline{B}$, le dernier cas étant exclu puisque $A \cap B \neq \emptyset$.

5. Si $\partial B \cap A \neq \emptyset$, ∂B étant connexe par hypothèse, appartenant à $\mathcal{X}^{\geq \lambda} u$ comme nous l'avons prouvé, cette frontière est contenue dans une composante connexe de

$\mathcal{X}^{\geq \lambda} u$, disons $\text{cc}([u \geq \lambda], \mathbf{y})$. Si $\text{cc}([u \geq \lambda], \mathbf{y}) \neq \text{cc}([u \geq \lambda], \mathbf{x})$, alors $\text{cc}([u \geq \lambda], \mathbf{y}) \cap \text{cc}([u \geq \lambda], \mathbf{x}) = \emptyset$ et ∂B , étant inclus dans $\text{cc}([u \geq \lambda], \mathbf{y})$, est dans une composante connexe de $X \setminus \text{cc}([u \geq \lambda], \mathbf{x})$, et puisqu'il rencontre A , il est inclus dans un trou de $\text{cc}([u \geq \lambda], \mathbf{x})$. Dans tous les cas, nous avons $\partial B \subset A$. En appliquant la Proposition 2.18 nous obtenons alors :

$$B = \text{sat}(B) \subset \text{sat}(\partial B) \subset \text{sat}(A) = A.$$

□

Le lemme suivant traite du dernier cas : lorsque les composantes connexes des ensembles de niveau sont disjointes.

Lemme 2.21 *Soit A et B deux ensembles connexes disjoints d'un espace topologique connexe et localement connexe. Alors $\text{sat}(A)$ et $\text{sat}(B)$ sont soit emboîtés soit disjointes.*

Preuve. Si $\text{sat}(A)$ ou $\text{sat}(B)$ est X , le résultat est évident. Supposons donc que A et B ont un extérieur. A est contenu soit dans l'extérieur de B , soit dans un trou de B , car connexe et contenu dans le complémentaire de B . S'il est inclus dans un trou H de B , alors puisque H est un ensemble simple, nous obtenons $\text{sat}(A) \subset H$, et finalement $\text{sat}(A) \subset \text{sat}(B)$. Si A est contenu dans l'extérieur de B , alors B est inclus soit dans l'extérieur de A soit dans un trou de A . Dans le dernier cas, la même preuve s'applique et nous obtenons $\text{sat}(B) \subset \text{sat}(A)$. Dans le premier cas, aucun trou de A ne rencontre B , de telle sorte que $\text{sat}(A)$ est contenu dans l'extérieur de B . Cela signifie qu'aucun trou de B ne rencontre $\text{sat}(A)$ et donc $\text{sat}(A) \cap \text{sat}(B) = \emptyset$. □

Le théorème suivant rassemble les trois résultats précédents et représente l'achèvement de cette section.

Théorème 2.22 *Soit u une image semicontinue supérieurement sur l'espace connexe et localement connexe X , A et B deux formes de u à frontière connexe. Alors A et B sont disjointes ou emboîtés.*

Preuve.

1. Si $A = \text{sat}(\text{cc}([u \geq \lambda], \mathbf{x}))$ et $B = \text{sat}(\text{cc}([u \geq \mu], \mathbf{y}))$.
 - Si $\text{cc}([u \geq \lambda], \mathbf{x}) \cap \text{cc}([u \geq \mu], \mathbf{y}) \neq \emptyset$, supposons par exemple que $\lambda \leq \mu$. Alors,

de la Proposition 2.4 nous déduisons

$$\text{cc}([u \geq \mu], \mathbf{y}) \subset \text{cc}([u \geq \lambda], \mathbf{x}) \subset A$$

et puisque A est un ensemble simple, nous obtenons $B \subset A$.

– Si $\text{cc}([u \geq \lambda], \mathbf{x}) \cap \text{cc}([u \geq \mu], \mathbf{y}) = \emptyset$, alors le Lemme 2.21 s'applique et le résultat s'ensuit.

2. Si $A = \text{sat}(\text{cc}([u < \lambda], \mathbf{x}))$ et $B = \text{sat}(\text{cc}([u < \mu], \mathbf{y}))$, la même preuve s'applique.

3. Si $A = \text{sat}(\text{cc}([u \geq \lambda], \mathbf{x}))$ et $B = \text{sat}(\text{cc}([u < \mu], \mathbf{y}))$, si $\text{cc}([u \geq \lambda], \mathbf{x}) \cap \text{sat}(\text{cc}([u < \mu], \mathbf{y})) = \emptyset$, nous pouvons utiliser le Lemme 2.21 appliqué à $\text{cc}([u \geq \lambda], \mathbf{x})$ et $\text{sat}(\text{cc}([u < \mu], \mathbf{y}))$ et la preuve est faite. Sinon, nous avons $\text{cc}([u \geq \lambda], \mathbf{x}) \cap \text{sat}(\text{cc}([u < \mu], \mathbf{y})) \neq \emptyset$ et si \mathbf{z} est dans l'intersection nous pouvons réécrire $\text{cc}([u \geq \lambda], \mathbf{x}) = \text{cc}([u \geq \lambda], \mathbf{z})$ et $\text{cc}([u < \mu], \mathbf{y}) = \text{cc}([u < \mu], \mathbf{z})$, et le résultat est une conséquence de la Proposition 2.20. \square

Ce théorème implique directement la structure d'arbre d'inclusion des formes : cela peut être démontré exactement par le même argument trivial qui a été utilisé pour prouver le Corollaire 2.5 à partir de la Proposition 2.4.

2.3.5 Espaces unicohérents

Comme nous l'avons vu, les formes d'une image ont une structure d'arbre d'inclusion sous une condition restrictive sur u : que ses formes aient une frontière connexe. En fait, cela peut être assuré par la semicontinuité supérieure de u si l'ensemble de définition X est unicohérent. Nous rappelons la définition d'un espace unicohérent (voir [35, §41,X]) :

Définition 2.23 *Un espace topologique X est dit unicohérent s'il est connexe et quels que soient les sous-ensembles connexes fermés F et F' tels que $X = F \cup F'$, nous avons : $F \cap F'$ est connexe.*

Donnons des exemples d'espaces unicohérents¹. \mathbb{R} , et tout intervalle I de \mathbb{R} , sont unicohérents. En effet, un sous-ensemble connexe de $X = \mathbb{R}$ ou I est un intervalle. Donc si X est l'union de deux intervalles fermés, ils se rencontrent et leur intersection est un intervalle fermé, donc un ensemble connexe. Il est plus difficile de prouver que \mathbb{R}^n et tout hypercube de \mathbb{R}^n sont unicohérents. En particulier, la fermeture d'un

¹Prouver qu'un espace topologique est unicohérent n'est généralement pas une tâche aisée

domaine de Jordan² dans \mathbb{R}^n est uncohérent, puisque homéomorphe à un hypercube de \mathbb{R}^n .

En ce qui concerne les sphères, S^n est uncohérent pour $n \geq 2$.

Voici également quelques exemples génériques : un arbre topologique (une dendrite), c'est-à-dire un continu localement connexe ne contenant aucune courbe simple fermée, est uncohérent, voir Kuratowsky [35, §46, VI, 1]. Tout sous-ensemble connexe par arcs d'un continu localement connexe et uncohérent est uncohérent, voir [35, §47,I,9].

Des exemples d'espaces connexes mais *non* uncohérents sont : un cylindre, un tore et S^1 . Le cercle S^1 a même la propriété remarquable d'être *discohérent* : s'il est décomposé en une réunion de deux sous-ensembles fermés propres, leur intersection n'est pas connexe.

Nous montrons que si X est uncohérent, les formes ont une frontière connexe :

Proposition 2.24 *Si X est un espace uncohérent et localement connexe, sat une saturation sur X et u une image semicontinue supérieurement définie sur X , alors les formes de u ont une frontière connexe.*

Preuve. Soit A une forme de u . Nous savons que A est soit fermé soit ouvert. Supposons A fermé. Alors $X \setminus A$ est connexe et donc $\overline{X \setminus A}$ est aussi connexe, fermé et $X = A \cup \overline{X \setminus A}$. X étant uncohérent, nous déduisons que $A \cap \overline{X \setminus A} = \partial A$ est connexe.

Si A est ouvert, $X \setminus A$ est fermé et connexe, ainsi que \overline{A} et puisque $X = \overline{A} \cup (X \setminus A)$, nous déduisons que leur intersection, ∂A , est également connexe. \square

Nous en tirons immédiatement le

Corollaire 2.25 *Dans un espace X uncohérent et localement connexe avec une saturation, deux formes d'une image semicontinue supérieurement définie sur X sont soit disjointes soit emboîtées.*

Remarque 2.5. Il est facile de construire une image semicontinue supérieurement avec deux formes non emboîtées et d'intersection non vide si X n'est pas uncohérent. Par exemple, considérons la fonction

$$f(\theta) = \theta \chi_{(0,\pi]} + (2\theta + \pi) \chi_{(-\pi, 0]}$$

²C'est-à-dire, la fermeture de la composante connexe bornée du complémentaire d'un sous-ensemble de \mathbb{R}^n homéomorphe à S^{n-1} , la sphère de \mathbb{R}^n .

pour $\theta \in (-\pi, \pi]$ définie sur le cercle S^1 , où θ est l'angle. Une composante connexe de $[f \geq \frac{\pi}{3}]$ est $[\frac{\pi}{3}, \pi]$. Une composante connexe de $[f < \frac{2\pi}{3}]$ est $(0, \frac{2\pi}{3})$. Le complémentaire de chacune est connexe, ils se rencontrent et leur union n'est pas S^1 . De ceci, il est aisé de construire un opérateur de saturation sur S^1 tel que ces ensembles soient simples relativement à cette saturation.

Dorénavant, nous supposons que X est univoqué et localement connexe. Répétons que des exemples de tels espaces comprennent en particulier \mathbb{R}^n ($n \geq 1$) et S^n (la sphère n -dimensionnelle, $n \geq 2$).

2.4 Applications

Jusqu'à présent, nous avons montré que sous certaines hypothèses sur l'espace topologique X , les formes d'une image semicontinue supérieurement définie sur X ont une structure d'arbre d'inclusion. Mais la définition des formes requiert un opérateur de saturation sur X . Le but de cette section est de mettre en évidence des opérateurs de saturation ayant un sens en analyse d'images. Nous le ferons dans les deux cas les plus importants pour les applications : lorsque $X = \mathbb{R}^n$ et lorsque X est la fermeture d'un domaine de Jordan dans \mathbb{R}^n (par exemple un hypercube), pour $n \geq 2$.

2.4.1 Image définie sur \mathbb{R}^n

Ici, nous prendrons $X = \mathbb{R}^n$ avec la topologie usuelle. Nous définissons l'opérateur de saturation naturel dans \mathbb{R}^n par

$$\text{sat} : \mathcal{P}(\mathbb{R}^n) \rightarrow \mathcal{P}(\mathbb{R}^n)$$

$$A \mapsto \begin{cases} \mathbb{R}^n & \text{si } A \text{ n'est pas borné,} \\ A \cup \bigcup_{t \in T} H_t & \text{si } A \text{ est borné,} \end{cases} \quad (2.4)$$

où les H_t sont les composantes connexes bornées de $\mathbb{R}^n \setminus A$. Que cela définisse un opérateur de saturation sera montré plus bas dans la Proposition 2.26. Notons que dans un certain sens ceci correspond à une notion de trou naturelle. Un trou est une composante connexe bornée du complémentaire. Pour les ensembles bornés, il n'y a qu'une composante connexe non bornée du complémentaire (comprise comme le fond) et il n'y a pas d'ambiguïté. Pour les ensembles non bornés, il peut y avoir plusieurs composantes connexes non bornées du complémentaire, et nous ne cherchons pas à enlever l'ambiguïté entre le fond et la forme. Nous adoptons la solution de définir toutes les composantes connexes du complémentaire comme des trous. Cette

paresse a un prix : nous ne serons pas capable de reconstruire une image à partir de ses formes. Cependant, c'est suffisant pour les images qui sont constantes dans un voisinage de l'infini.

Démontrons que nous avons bien défini un opérateur de saturation sur \mathbb{R}^n .

Proposition 2.26 *L'opérateur sat défini sur \mathbb{R}^n par la Formule (2.4) est un opérateur de saturation lorsque $n \geq 2$.*

Preuve.

1. Soit $A \subset X$. Si A n'est pas borné, $X \setminus \text{sat}(A) = \emptyset$. Sinon, soit B une boule contenant A . $\mathbb{R}^n \setminus B$ est connexe puisque $n \geq 2$, donc inclus dans une composante connexe de $\mathbb{R}^n \setminus A$, montrons que toutes les autres composantes connexes de $\mathbb{R}^n \setminus A$ sont dans B , donc incluses dans $\text{sat}(A)$. Finalement, $\mathbb{R}^n \setminus \text{sat}(A)$ est une composante connexe de $\mathbb{R}^n \setminus A$, celle qui n'est pas bornée.

2. Soit $A \subset B \subset \mathbb{R}^n$. Si B n'est pas borné, $\text{sat}(B) = \mathbb{R}^n$, donc l'inclusion est triviale. Si B est borné, A est aussi borné et $\mathbb{R}^n \setminus B \subset \mathbb{R}^n \setminus A$, de telle sorte que les composantes connexes non bornées de $X \setminus B$ sont incluses dans les composantes connexes non bornées de $X \setminus A$, et leur union, qui est $X \setminus \text{sat}(B)$, est incluse dans l'union des composantes connexes non bornées de $X \setminus A$, qui est $X \setminus \text{sat}(A)$. Cela donne $\text{sat}(A) \subset \text{sat}(B)$.

3. Soit $A \subset X$ borné. Comme nous l'avons vu, $\mathbb{R}^n \setminus \text{sat}(A)$ n'est pas borné, donc $\text{sat}(\mathbb{R}^n \setminus \text{sat}(A)) = \mathbb{R}^n$.

4. Soit $A \subset X$. Si A n'est pas borné, $\text{sat}(A) = \mathbb{R}^n$ et nous avons $\text{sat} \circ \text{sat}(A) = \text{sat}(A) = \mathbb{R}^n$. Si A est borné, $\text{Ext } A = \mathbb{R}^n \setminus \text{sat}(A)$ est connexe et non borné, donc $\text{sat} \circ \text{sat}(A) = \text{sat}(A)$. \square

Remarque 2.6. Le seul axiome non vérifié pour $n = 1$ est le second. En fait, le complémentaire d'un intervalle borné a deux composantes connexes.

En rassemblant les résultats du Théorème 2.22 et comme \mathbb{R}^n est univoqué, nous atteignons notre but :

Corollaire 2.27 *Soit u une image semicontinue supérieurement définie sur \mathbb{R}^n ($n \geq 2$). Soit A et B deux formes de u . Alors A et B sont emboîtés ou disjoints.*

2.4.2 Image définie sur un sous-ensemble borné de \mathbb{R}^n

Lorsque l'image u n'est définie que sur un sous-ensemble borné de \mathbb{R}^n , nous voudrions avoir une propriété similaire au Théorème 2.22, où les formes devraient

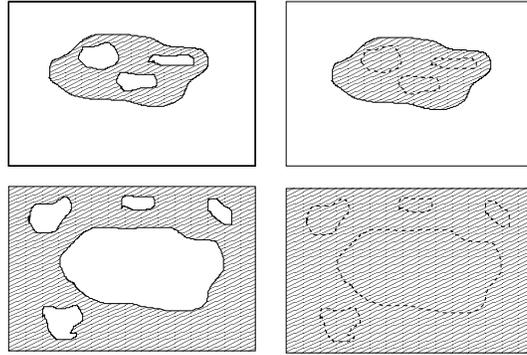


FIG. 2.6 – Saturation de quelques ensembles dans un ensemble de définition borné. Gauche : deux ensembles (hachurés) dans leur image respective. Droite : leur saturation (hachurée). L'ensemble en haut à gauche ne rencontre pas le cadre de l'image. Il est saturé comme si l'image était infinie (quel que soit le contenu de l'image en dehors de l'ensemble de définition, le résultat est l'ensemble en haut à droite). L'ensemble en bas à gauche contient le cadre de l'image. Il est aussi saturé comme si l'image était infinie (quel que soit le contenu de l'image en dehors de l'ensemble de définition, le résultat contiendrait l'ensemble de définition dans son intégralité, montré en bas à droite). La saturation d'un ensemble contenant le cadre de l'image est toujours l'ensemble de définition entier.

avoir une interprétation facile en termes d'analyse d'image. L'idée est que seule une partie d'une image définie sur \mathbb{R}^n est observée.

La première solution (mauvaise) serait de prolonger l'image u à \mathbb{R}^n par une valeur arbitraire. Le problème est précisément que cette valeur est arbitraire, et des valeurs différentes donneraient des arbres différents.

Nous voudrions que les « objets » totalement inclus dans l'ensemble de définition soient décrits de la même manière qu'ils le seraient si l'image entière sur \mathbb{R}^n était observée. De telle sorte que les composantes connexes d'ensembles de niveau ne rencontrant pas le cadre de l'ensemble de définition sont supposées ne pas être coupées. A cette condition, quelle que soit le contenu de l'image u en dehors de l'ensemble de définition, ses trous sont des composantes du complémentaire ne rencontrant pas le cadre. Pour la même raison, la saturation d'une composante connexe contenant le cadre est l'ensemble de définition lui-même (voir la Figure 2.6). Il reste à traiter les composantes connexes d'ensembles de niveau qui rencontrent le cadre sans le contenir.

La notion intuitive de trou est celle d'une composante connexe du complémentaire « plus petite » que l'extérieur. Lorsque l'ensemble de définition est \mathbb{R}^n , dans un certain sens un ensemble borné est « plus petit » qu'un ensemble non borné, de

telle sorte que nous pouvons définir les trous et l'extérieur en accord avec l'intuition. Lorsque u est définie sur un ensemble borné, nous quantifions cette notion à l'aide de la théorie de la mesure. Donc, nous devons supposer que nous avons une mesure sur l'ensemble de définition.

Nous avons également besoin que l'ensemble de définition soit uncohérent. Ceci impose des contraintes fortes. Nous supposons que X est la fermeture d'un domaine de Jordan dans \mathbb{R}^2 , ou plus généralement dans \mathbb{R}^n , i.e., la fermeture de l'intérieur d'un sous-ensemble de \mathbb{R}^n homéomorphe à S^{n-1} . Alors nous savons que X est un sous-ensemble connexe et localement connexe de \mathbb{R}^n ($n \geq 2$), et également uncohérent, pour la topologie usuelle induite par celle de \mathbb{R}^n . Nous supposons aussi donnée une mesure de Borel μ sur X . Donc, puisque X est compact, $\mu(X) < +\infty$. La frontière de X en tant que sous-ensemble de \mathbb{R}^n , notée ∂X , sera appelée le cadre de l'ensemble de définition. C'est un ensemble connexe (une hypersurface de Jordan).

A partir des remarques ci-dessus, nous définissons l'opérateur de saturation comme ceci :

Définition 2.28 *Soit A un sous-ensemble mesurable de X . Nous définissons $\text{sat}(A)$ par :*

$$\begin{cases} A \cup \{H = \text{cc}(X \setminus A) : H \cap \partial X = \emptyset\} & \text{si } A \cap \partial X = \emptyset, \\ X & \text{si } \partial X \subset A, \\ X \setminus \{H = \text{cc}(X \setminus A) : H \cap \partial X \neq \emptyset \text{ et } \mu(H) > \mu(X)/2\} & \text{si } \emptyset \neq A \cap \partial X \neq \partial X. \end{cases} \quad (2.5)$$

Comme le Lemme 2.30 le montrera, les formes associées à des ensembles dans les deux premiers ensembles sont construites suivant les mêmes règles que si l'image était définie dans \mathbb{R}^n . Le cas nouveau concerne les ensembles qui rencontrent le cadre de l'image sans le contenir. La construction de la forme associée est illustrée dans la Figure 2.7. Le fait que la moitié de l'aire de l'image joue un rôle spécifique se justifie par la propriété que cela donne un opérateur de saturation. En particulier, si nous avons pris comme extérieur d'un ensemble (rencontrant le cadre sans le contenir) la plus grande composante connexe de son complémentaire rencontrant le cadre (pourvu qu'elle soit unique), nous ne respecterions pas le troisième axiome de la Définition 2.6. En effet, l'image définie sur le rectangle $[0, 1]^2$ par $u = \chi_{x \geq 1/3} + \chi_{x \geq 2/3}$ donnerait deux formes non emboîtées et se rencontrant, à savoir $[u < 2] = [0, 2/3] \times [0, 1]$ et $[u \geq 1] = [1/3, 1] \times [0, 1]$.

Nous montrons ci-dessous que l'opérateur sat défini plus haut est bien une saturation. Il est à noter que la définition ne se conforme pas exactement à la Définition 2.6 puisque nous n'avons pas défini l'action de sat sur des ensembles non mesurables.

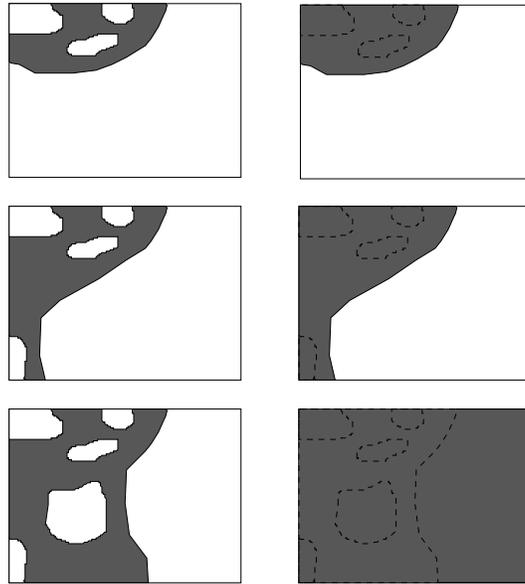


FIG. 2.7 – Construction de la forme associée à un ensemble rencontrant le cadre de l'image sans le contenir. C'est le cas qui n'avait pas été illustré dans la Figure 2.6. Gauche : ensembles. Droite : formes associées. Dans les deux premiers cas (deux premières lignes), une composante connexe du complémentaire a une mesure (de Lebesgue) supérieure à la moitié de celle de l'image, c'est l'extérieur de l'ensemble. Les autres composantes connexes sont les trous. Dans le troisième cas (troisième ligne), aucune composante connexe du complémentaire n'a une mesure suffisante, elles sont toutes considérées comme des trous et la forme associée est l'image entière.

Néanmoins, nous montrerons que nous pouvons nous restreindre à des ensembles mesurables pour deux raisons :

- les ensembles que nous traitons (en particulier les composantes connexes des ensembles de niveau) sont mesurables ;
- la saturation d'un ensemble mesurable reste mesurable, ainsi nous pouvons composer la saturation avec elle-même sur les ensembles mesurables.

Ceci explique que nous continuions de parler d'opérateur de saturation : il suffit de remplacer $\mathcal{P}(X)$ par les sous-ensembles μ -mesurables de X dans la Définition 2.6.

Notre utilisation d'une mesure de Borel donne

Lemme 2.29 *Si $A \subset X$ est mesurable, alors toute composante connexe de A et de $X \setminus A$ est mesurable.*

Preuve.

1. Si C est une composante connexe de A , c'est un fermé dans A , ainsi nous pouvons écrire $C = \overline{C} \cap A$, et puisque \overline{C} est mesurable (car fermé) et A est aussi mesurable, C est mesurable.

2. De plus, $X \setminus A$ est aussi mesurable, et en appliquant la preuve précédente à $X \setminus A$, nous déduisons que toute composante connexe de $X \setminus A$ est mesurable. \square

Le lemme suivant fait le lien avec le cas de \mathbb{R}^n , comme il a été étudié dans la section précédente.

Lemme 2.30 *Soit A un sous-ensemble mesurable de X . A est aussi un sous-ensemble de \mathbb{R}^n , de telle sorte que nous pouvons définir $\text{sat}(A)$ comme dans la Formule (2.5) et $\widetilde{\text{sat}}(A)$ comme dans la Formule (2.4).*

Si $\partial X \cap A = \emptyset$ ou $\partial X \subset A$, alors $\text{sat}(A) = \widetilde{\text{sat}}(A)$.

Preuve.

1. Si $\partial X \cap A = \emptyset$, comme ∂X est connexe, c'est une composante connexe E de $X \setminus A$, et toute autre composante connexe de $X \setminus A$ est donc incluse dans $\text{sat}(A)$. $E \cup (\mathbb{R}^n \setminus X)$ est connexe, non borné et dans $\mathbb{R}^n \setminus A$, donc c'est l'extérieur de $\widetilde{\text{sat}}(A)$. Nous avons

$$E \cup (\mathbb{R}^n \setminus X) = \mathbb{R}^n \setminus \text{sat}(A) = \mathbb{R}^n \setminus \widetilde{\text{sat}}(A)$$

et le résultat est prouvé.

2. Si $\partial X \subset A$, aucune composante connexe de $X \setminus A$ ne rencontre ∂X , de telle sorte que chacune est incluse dans $\text{sat}(A)$. Donc, $\text{sat}(A) = X$, et il est clair que $\widetilde{\text{sat}}(\partial X)$ est aussi X . \square

Nous sommes en mesure de prouver que l'opérateur sat , comme défini dans 2.28, est bien une saturation.

Proposition 2.31 *L'opérateur sat de la Formule (2.5) est un opérateur de saturation sur X .*

Preuve.

1. Soit A un sous-ensemble mesurable de X tel que $\text{sat}(A) \neq X$. Le cas où $A \cap \partial X = \emptyset$ est évident : toute composante connexe de $X \setminus \text{sat}(A)$ en est une de $X \setminus A$ rencontrant ∂X d'après la Définition 2.28, et comme ∂X est borné, il n'y en a qu'une. Sinon, nous sommes dans le cas où $\emptyset \neq A \cap \partial X \neq \partial X$. Alors $X \setminus \text{sat}(A)$ est composé de composantes connexes de $X \setminus A$ de mesure strictement plus grande que $\mu(X)/2$ et puisqu'un tel ensemble est mesurable, il est unique. S'il y en avait au moins deux, H_1 et H_2 , nous aurions

$$\mu(X) \geq \mu(X \setminus \text{sat}(A)) \geq \mu(H_1 \cup H_2) \quad (*)$$

et comme H_1 et H_2 sont disjoints et mesurables (d'après le Lemme 2.29), nous aurions

$$\mu(H_1 \cup H_2) = \mu(H_1) + \mu(H_2) > \mu(H),$$

en contradiction avec l'inégalité (*).

2. Soit $A \subset B \subset X$, A et B mesurables. Si $\partial X \subset B$, $\text{sat}(B) = X \supset \text{sat}(A)$. Si $B \cap \partial X = \emptyset$, nous avons aussi $A \cap \partial X = \emptyset$ et grâce au Lemme 2.30, le résultat $\text{sat}(A) \subset \text{sat}(B)$ provient du fait que $\widetilde{\text{sat}}$ est croissant. La possibilité restante est $\emptyset \neq \partial X \cap B \neq \partial X$. Si $\partial X \cap A = \emptyset$, $X \setminus \text{sat}(A)$ est l'unique composante connexe de $X \setminus A$ contenant ∂X et contient évidemment $X \setminus \text{sat}(B)$. Sinon, $\emptyset \neq \partial X \cap A \neq \partial X$, et par monotonie de la mesure, une composante connexe de $X \setminus \text{sat}(B)$ doit rencontrer le cadre et être de mesure strictement plus grande que $\mu(X)/2$, donc être contenue dans une composante connexe de $X \setminus \text{sat}(A)$. Ceci donne $\text{sat}(A) \subset \text{sat}(B)$.

3. Soit A mesurable tel que $\text{sat}(A) \neq X$. Si $\partial A \cap X = \emptyset$, alors $X \setminus \text{sat}(A)$ est mesurable et contient $X \setminus A$, de telle sorte que $\text{sat}(X \setminus \text{sat}(A)) = X$. Sinon, nous avons $\emptyset \neq \partial X \cap A \neq \partial X$ et $X \setminus \text{sat}(A)$ est aussi mesurable et vérifie la même propriété, mais est de mesure strictement plus grande que $\mu(X)/2$, ainsi son complémentaire est de mesure strictement plus petite que $\mu(X)/2$, donc est contenu dans $\text{sat}(X \setminus \text{sat}(A))$. Donc $\text{sat}(X \setminus \text{sat}(A)) = X$.

4. Soit A mesurable. Si $A \cap \partial X = \emptyset$ ou ∂X , nous savons que $\text{sat}(\text{sat}(A)) = \text{sat}(\widetilde{\text{sat}}(A))$ et comme $\widetilde{\text{sat}}(A)$ est aussi mesurable et que nous avons $\widetilde{\text{sat}}(A) \cap \partial X =$

\emptyset ou ∂X ,

$$\text{sat}(\text{sat}(A)) = \widetilde{\text{sat}}(\widetilde{\text{sat}}(A)) = \widetilde{\text{sat}}(A) = \text{sat}(A).$$

Si $\emptyset \neq A \cap \partial X \neq \partial X$ et $\text{sat}(A) \neq X$, nous obtenons aussi $\emptyset \neq \text{sat}(A) \cap \partial X \neq \partial X$ et $\mu(X \setminus \text{sat}(A)) > \mu(X)/2$, de telle sorte que $\text{sat}(\text{sat}(A)) \cap (X \setminus \text{sat}(A)) = \emptyset$, donnant $\text{sat}(\text{sat}(A)) = A$. \square

Comme dans le cas de \mathbb{R}^n (voir Section 2.4.1), ceci implique que les formes (suivant l'opérateur de saturation de la Définition 2.28) d'une image semicontinue supérieurement définie sur X -la fermeture d'un domaine de Jordan dans \mathbb{R}^n ($n \geq 2$)- ont une structure d'arbre.

2.5 Reconstruction

2.5.1 Cadre de travail

Dans les Sections 2.3 et 2.4 nous avons vu sous quelles conditions une image (semicontinue supérieurement) peut se décomposer en ses formes, avec une structure d'arbre reliant ces formes, et nous avons donné la définition de formes dans deux cas fondamentaux : lorsque X est \mathbb{R}^n ($n \geq 2$) et lorsque X est la fermeture d'un domaine de Jordan dans \mathbb{R}^n , donc en particulier dans le cas d'un rectangle dans le plan, ce qui est le cas le plus intéressant (le seul?) dans la pratique. Dans cette section, la question réciproque est posée : étant données les formes d'une image semicontinue supérieurement u , peut-on reconstruire u ? La réponse est positive si nous ajoutons des conditions supplémentaires, qui sont remplies automatiquement dans le cas du rectangle.

Plus précisément, soit X un espace unicohérent et localement connexe avec un opérateur de saturation. Supposons que pour chaque $\lambda \in \mathbb{R}$ et pour chaque $\mathbf{x} \in X$ nous avons la donnée $(\lambda, \text{sat}(\text{cc}([u \geq \lambda], \mathbf{x})))$ et $(\lambda, \text{sat}(\text{cc}([u < \lambda], \mathbf{x})))$, mais rien de plus. Sommes-nous capable de retrouver l'image u à partir de ces paires? La première réponse évidente est que ce n'est pas systématique.

Si nous considérons la saturation triviale qui transforme tout ensemble en X , toutes les images ont les mêmes formes, donc la reconstruction est impossible. Ceci montre que nous avons besoin d'un opérateur de saturation adéquat. Même avec un opérateur de saturation non trivial, tel que celui défini dans le cas de \mathbb{R}^2 dans la Section 2.4.1, la réponse peut être négative. Considérons l'image $u = \chi_{x \geq 0}$; u est semicontinue supérieurement alors que toutes ses formes sont \mathbb{R}^2 , comme pour une

image constante.

Pour éviter de tels défauts, nous examinerons la question de la reconstruction dans le cas où X est la fermeture d'un domaine de Jordan dans \mathbb{R}^n ou \mathbb{R}^n en entier, et u est constante en dehors d'un certain sous-ensemble borné dans ce dernier cas.

Définition 2.32 Appelons niveau-forme une paire $(\lambda, \text{sat}(\text{cc}([u \geq \lambda], \mathbf{x})))$, si $u(\mathbf{x}) \geq \lambda$, ou une paire $(\lambda, \text{sat}(\text{cc}([u < \lambda], \mathbf{x})))$, si $u(\mathbf{x}) < \lambda$.

Remarque 2.7. De façon logique, nous dirons que c'est un niveau-forme supérieur dans le premier cas et un niveau-forme inférieur dans le second cas. Notons que les formes impliquées dans les niveaux-formes sont toujours supposées non vides.

Notation : La famille des niveaux-formes contenant un point \mathbf{x} est notée $\text{LS}_{\mathbf{x}}$.

Un théorème classique de topologie, utilisé plusieurs fois dans la suite, est dû à Zoretti, voir [35, §42,II,5] ou [62, IV.5, Theorem 5.3] :

Théorème 2.33 (Zoretti) *L'intersection d'une suite décroissante de continus est un continu.*

Remarque 2.8. En fait, c'est une conséquence d'un autre résultat dû également à Zoretti : si une suite de continus a une limite inférieure non vide, sa limite supérieure est un continu. La limite inférieure d'une suite d'ensembles C_n est l'ensemble des points limites de suites de points $\mathbf{x}_n \in C_n$, tandis que la limite supérieure est l'ensemble des points limites de suites extraites de telles suites (c'est-à-dire les points d'adhérence de ces suites).

Un autre résultat important, dû à Lindelöf est le suivant, voir [35, §17,I] :

Théorème 2.34 (Lindelöf) *Si X est un espace métrique séparable, et $(G_i)_{i \in I}$ (resp. $(F_i)_{i \in I}$) est une famille infinie d'ouverts (resp. de fermés), il existe une suite G_{i_1}, G_{i_2}, \dots (resp. F_{i_1}, F_{i_2}, \dots) telle que*

$$\bigcup_{n=1}^{\infty} G_{i_n} = \bigcup_{i \in I} G_i \quad (\text{resp.} \quad \bigcap_{n=1}^{\infty} F_{i_n} = \bigcap_{i \in I} F_i)$$

De ces deux théorèmes, nous dérivons facilement le corollaire suivant (nous rappelons qu'une famille monotone d'ensembles vérifie par définition la propriété que deux quelconques de ses éléments sont emboîtés) :

Corollaire 2.35 *Si X est un espace métrique séparable, l'intersection d'une famille monotone de continus est un continu.*

Preuve. Si les $(C_i)_{i \in I}$ sont des continus de partie commune C , nous pouvons extraire une suite C_1, C_2, \dots (grâce au théorème de Lindelöf) telle que $C = \bigcap_{i=1}^{\infty} C_i$, et en remplaçant C_i par $C'_i = \bigcap_{j=1}^i C_j$, les C'_i sont clairement des continus décroissants,

et leur intersection est C . Selon le théorème de Zoretta, C est donc un continu. \square

Le lemme suivant joue un rôle fondamental dans les preuves des théorèmes de reconstruction que nous exposerons.

Lemme 2.36 *Soit $\mathbf{x}, \mathbf{y} \in X$ et $\lambda \in \mathbb{R}$ tels que*

$$\mathbf{x} \in \text{sat}(\text{cc}([u \geq \lambda], \mathbf{y})) \text{ mais } \mathbf{x} \notin \text{cc}([u \geq \lambda], \mathbf{y}).$$

Alors

$$\exists \mathbf{z} \in \text{sat}(\text{cc}([u \geq \lambda], \mathbf{y})) \text{ t.q. } \mathbf{x} \in \text{sat}(\text{cc}([u < \lambda], \mathbf{z})) \subset \text{sat}(\text{cc}([u \geq \lambda], \mathbf{y})).$$

De même, si

$$\mathbf{x} \in \text{sat}(\text{cc}([u < \lambda], \mathbf{y})) \text{ mais } \mathbf{x} \notin \text{cc}([u < \lambda], \mathbf{y}),$$

alors

$$\exists \mathbf{z} \in \text{sat}(\text{cc}([u < \lambda], \mathbf{y})) \text{ t.q. } \mathbf{x} \in \text{sat}(\text{cc}([u \geq \lambda], \mathbf{z})) \subset \text{sat}(\text{cc}([u < \lambda], \mathbf{y})).$$

Preuve.

1. \mathbf{x} est dans un trou H de $\text{cc}([u \geq \lambda], \mathbf{y})$. H est ouvert et sa frontière est connexe car X est univoqué.

2. Supposons que

$$\forall \mathbf{z} \in H, \mathbf{x} \notin \text{sat}(\text{cc}([u < \lambda], \mathbf{z})). \quad (*)$$

Notons que cela implique $u(\mathbf{x}) \geq \lambda$.

3. Considérons la famille des ensembles $\text{cc}([u < \lambda], \mathbf{z})$ où $\mathbf{z} \in H$ et $u(\mathbf{z}) < \lambda$. Ces ensembles sont ouverts et disjoints, donc ils sont au plus un nombre dénombrable. La même propriété est vraie pour leurs saturations. Enumérons-les C_1, C_2, \dots . Montrons que pour tout n l'ensemble $D_n = \overline{H} \setminus \bigcup_{i=1}^n C_i$ est connexe. Il est clair que D_n est fermé. Soit $G \neq \emptyset$ un sous-ensemble ouvert et fermé de D_n . Nous avons

$$\partial D_n = \partial H \cup \bigcup_{i=1}^n \partial C_i.$$

et chaque terme de cette réunion est connexe. Alors si G rencontre l'un des ∂C_i , il le contient. Considérons G' la réunion de G et des C_i dont la frontière est dans G . Visiblement, G' est ouvert dans \overline{H} . De plus pour un tel i , $G \cup C_i = G \cup \overline{C_i}$ et donc G' est aussi fermé dans \overline{H} . Par connexité de \overline{H} , nous déduisons $G' = \overline{H}$, donnant

Version light : certaines figures sont absentes

$G = D_n$. Ceci prouve que D_n est connexe.

4. Si le nombre de C_i est fini, cela prouve que $\overline{H} \cap [u \geq \lambda]$ est connexe. Si leur nombre est infini, les D_n sont des continus décroissants. Selon le théorème de Zoratti, nous concluons que leur intersection, i.e., $\overline{H} \cap [u \geq \lambda]$, est aussi connexe.

5. Dans tous les cas, comme ∂H est dans $\text{cc}([u \geq \lambda], \mathbf{y})$, nous déduisons que $\overline{H} \cap [u \geq \lambda]$ est inclus dans $\text{cc}([u \geq \lambda], \mathbf{y})$, ce qui est impossible puisque \mathbf{x} est dans le premier mais pas dans le second. Donc l'hypothèse (*) ne tient pas.

6. La preuve dans le second cas est plus facile : si $\mathbf{x} \in \text{sat}(\text{cc}([u < \lambda], \mathbf{y}))$ mais pas dans $\text{cc}([u < \lambda], \mathbf{y})$, il est dans un trou H de $\text{cc}([u < \lambda], \mathbf{y})$, et ∂H est dans $[u \geq \lambda]$ et connexe, avec $\text{sat}(\partial H) = H$. Il suffit de prendre pour \mathbf{z} un point quelconque de ∂H . \square

Suivant l'intuition, nous dirions que le niveau de gris en un point \mathbf{x} est le niveau de la plus petite forme le contenant. Malheureusement, cette plus petite forme n'existe pas toujours, donc nous aurons besoin d'utiliser une limite. Pour cela, nous ordonnons d'abord les niveaux-formes.

Les formes contenant un point donné $\mathbf{x} \in X$ sont totalement ordonnées par la relation d'inclusion. Néanmoins, la même forme peut être extraite de niveaux de gris différents, donc nous devons préciser les choses.

Pour un $\mathbf{x} \in X$, nous définissons la relation \preceq dans la famille des niveaux-formes $(\lambda, S) \in \text{LS}_{\mathbf{x}}$ tels que $S \subsetneq X$ par :

$$\begin{aligned}
 & A \subsetneq B \text{ ou} \\
 (\lambda, A) \preceq (\mu, B) & \Leftrightarrow A = B \text{ et } \begin{cases} \lambda \leq \mu & \text{si } A \text{ et } B \text{ sont des formes inférieures} \\ \lambda \geq \mu & \text{si } A \text{ et } B \text{ sont des formes supérieures} \end{cases}
 \end{aligned} \tag{2.6}$$

Remarque 2.9. La définition est bien posée : le cas $A = B$ implique que les formes sont du même type (toutes deux inférieures ou toutes deux supérieures), sinon elles seraient ouvertes et fermées et par hypothèse elles ne sont ni \emptyset ni X .

Lemme 2.37 \preceq est une relation d'ordre total dans la famille des niveaux-formes contenant un point donné \mathbf{x} , dont la forme n'est pas X .

Preuve. La définition est très proche d'un ordre lexicographique sur les paires, mais pas exactement, donc nous détaillons la preuve.

1. La réflexivité est contenue dans la définition.
2. Si $(\lambda, A) \preceq (\mu, B)$ et $(\mu, B) \preceq (\lambda, A)$, alors trivialement $A = B$ et donc $\lambda = \mu$.

3. En ce qui concerne la transitivité, supposons $(\lambda, A) \preceq (\mu, B)$ et $(\mu, B) \preceq (\nu, C)$. Si $A \subsetneq B$ ou $B \subsetneq C$, nous avons $A \subsetneq C$ et donc $(\lambda, A) \preceq (\nu, C)$. Sinon, $A = B = C$ et si ce sont des formes inférieures (resp. supérieures), ça signifie $\lambda \leq \mu \leq \nu$ (resp. $\lambda \geq \mu \geq \nu$). Dans tous les cas, $(\lambda, A) \preceq (\nu, C)$.

4. Donc, nous avons à faire à une relation d'ordre. Pour tout (λ, A) et (μ, B) , comme A et B contiennent \mathbf{x} , ils sont emboîtés. S'ils ne sont pas égaux, (λ, A) et (μ, B) sont comparables, puisque l'ordre est déterminé par l'ordre de A et B . Si $A = B$ et si c'est une forme inférieure (resp. supérieure), ce n'est pas une forme supérieure (resp. inférieure), sinon ce serait un ouvert et un fermé, contredisant la connexité de X . Donc (λ, A) et (μ, B) sont comparables. \square

Dans le reste du chapitre, nous donnerons deux méthodes de reconstruction d'une image à partir de ses niveaux-formes. La première, que nous appelons reconstruction directe, est une formule explicite, mais n'est pas très facile à utiliser, car cette formule implique une limite. La seconde est appelée indirecte car elle reconstruit en fait les ensembles de niveau de l'image (qui à leur tour permettent de reconstruire l'image). Cette deuxième solution est plus facile à utiliser, car seules des manipulations algébriques d'ensembles sont nécessaires.

2.5.2 Reconstruction directe

Voici notre formule de reconstruction directe :

Théorème 2.38 *Soit u une image semicontinue supérieurement. Pour $\mathbf{x} \in X$, nous avons*

$$u(\mathbf{x}) = \lim_{\text{LS}_{\mathbf{x}} \ni (\lambda, S) \searrow} \lambda \quad (2.7)$$

s'il existe une forme S contenant \mathbf{x} et incluse strictement dans X .

$$u(\mathbf{x}) = \inf_{(\lambda, X) \in \text{LS}_{\mathbf{x}}^<} \lambda = \max_{(\lambda, X) \in \text{LS}_{\mathbf{x}}^>} \lambda \quad (2.8)$$

si la seule forme contenant \mathbf{x} est X et pourvu que la condition suivante soit vérifiée :

$$\begin{aligned} C_1, C_2 \text{ composantes connexes d'ensembles de niveau de } u, \\ \text{sat}(C_1) = \text{sat}(C_2) = X \Rightarrow C_1 \cap C_2 \neq \emptyset. \end{aligned} \quad (2.9)$$

Remarque 2.10. La limite présente dans l'Equation 2.7 nécessite quelques explications. Elle doit être lue comme « limite du niveau lorsque les niveaux-formes

contenant \mathbf{x} décroissent ». Sa définition est

$$\begin{aligned} \lim_{\text{LS}_{\mathbf{x}} \ni (\lambda, S) \searrow} \lambda = l \in \mathbb{R} &\Leftrightarrow \\ \forall \varepsilon > 0, \exists (\lambda, S) \in \text{LS}_{\mathbf{x}}, \forall (\mu, T) \in \text{LS}_{\mathbf{x}}, & (\mu, T) \preccurlyeq (\lambda, S) \Rightarrow |l - \mu| \leq \varepsilon. \end{aligned} \quad (2.10)$$

De même, nous définissons les limites inférieures et supérieures :

$$\begin{aligned} \liminf_{\text{LS}_{\mathbf{x}} \ni (\lambda, S) \searrow} \lambda = l \in \mathbb{R} &\Leftrightarrow \\ \forall \varepsilon > 0, \exists (\lambda, S) \in \text{LS}_{\mathbf{x}}, & |l - \inf \{ \mu : (\mu, T) \in \text{LS}_{\mathbf{x}}, (\mu, T) \preccurlyeq (\lambda, S) \}| \leq \varepsilon. \end{aligned} \quad (2.10'')$$

$$\begin{aligned} \limsup_{\text{LS}_{\mathbf{x}} \ni (\lambda, S) \searrow} \lambda = l \in \mathbb{R} &\Leftrightarrow \\ \forall \varepsilon > 0, \exists (\lambda, S) \in \text{LS}_{\mathbf{x}}, & |l - \sup \{ \mu : (\mu, T) \in \text{LS}_{\mathbf{x}}, (\mu, T) \preccurlyeq (\lambda, S) \}| \leq \varepsilon. \end{aligned} \quad (2.10''')$$

et nous avons facilement l'équivalence classique :

$$\lim_{\text{LS}_{\mathbf{x}} \ni (\lambda, S) \searrow} \lambda = l \Leftrightarrow \liminf_{\text{LS}_{\mathbf{x}} \ni (\lambda, S) \searrow} \lambda = \limsup_{\text{LS}_{\mathbf{x}} \ni (\lambda, S) \searrow} \lambda = l$$

Notons toutefois que contrairement aux définitions classiques pour des suites de nombres réels, il n'est pas clair que les limites inférieures et supérieures existent toujours.

Nous prouvons maintenant le Théoreme 2.38.

Preuve.

1. Soit \mathbf{x} un point tel qu'il existe une forme le contenant $S^* \subsetneq X$, de niveau associé λ^* .

2. Considérons la forme $S_0 = \text{sat}(\text{cc}([u \geq u(\mathbf{x})], \mathbf{x}))$. Il est clair que $(u(\mathbf{x}), S_0) \in \text{LS}_{\mathbf{x}}$.

3. Soit $(\lambda, S) \in \text{LS}_{\mathbf{x}}$ tel que $(\lambda, S) \preccurlyeq (u(\mathbf{x}), S_0)$ si $S_0 \subsetneq X$, ou $S \subsetneq X$ si $S_0 = X$. Nous prétendons que $\lambda \geq u(\mathbf{x})$, cette inégalité étant stricte si $S_0 = X$.

4. Si S est une forme inférieure, nous avons $S \subsetneq S_0$. Comme $S = \text{sat}(\text{cc}([u < \lambda], \mathbf{y}))$, si nous avons $\lambda \leq u(\mathbf{x})$, nous déduisons que $\text{cc}([u < \lambda], \mathbf{y})$ ne contient pas \mathbf{x} , et donc que \mathbf{x} est dans un de ses trous, tout comme $\text{cc}([u \geq u(\mathbf{x})], \mathbf{x})$ et donc aussi sa saturation, ce qui est impossible. Donc $\lambda > u(\mathbf{x})$.

5. Si S est une forme supérieure, nous écrivons $S = \text{sat}(\text{cc}([u \geq \lambda], \mathbf{y}))$. Alors si $\mathbf{x} \in \text{cc}([u \geq \lambda], \mathbf{y})$, nous avons $u(\mathbf{x}) \geq \lambda$, auquel cas $\text{cc}([u \geq \lambda], \mathbf{y}) \supset \text{cc}([u \geq$

$u(\mathbf{x}), \mathbf{x}$), impliquant en fait que $S_0 \neq X$ et $\text{cc}([u \geq \lambda], \mathbf{y}) = \text{cc}([u \geq u(\mathbf{x})], \mathbf{x})$. La relation d'ordre donne alors $\lambda \geq u(\mathbf{x})$, et donc égalité. Sinon \mathbf{x} est dans un trou de $\text{cc}([u \geq \lambda], \mathbf{y})$ et puisque $\text{cc}([u \geq u(\mathbf{x})], \mathbf{x})$ n'est pas inclus dans ce trou, il rencontre $\text{cc}([u \geq \lambda], \mathbf{y})$ donc le contient (strictement), d'où $\lambda > u(\mathbf{x})$.

6. Pour $\varepsilon > 0$, considérons la forme $S_\varepsilon = \text{sat}(\text{cc}([u < u(\mathbf{x}) + \varepsilon], \mathbf{x}))$ et supposons $S_\varepsilon \neq X$. Soit $(\lambda, S) \in \text{LS}_{\mathbf{x}}$ tel que $(\lambda, S) \preccurlyeq (u(\mathbf{x}) + \varepsilon, S_\varepsilon)$.

7. Supposons que S est une forme supérieure, $S = \text{cc}([u \geq \lambda], \mathbf{y}) \neq X$. Alors si $\lambda \geq u(\mathbf{x}) + \varepsilon$, $\mathbf{x} \notin \text{cc}([u \geq \lambda], \mathbf{y})$ et \mathbf{x} est dans un trou H de cet ensemble, entraînant que $\text{cc}([u < u(\mathbf{x}) + \varepsilon], \mathbf{x})$ est aussi dans H , et comme $S \neq X$ par hypothèse, nous avons $\text{sat}(H) = H$ (voir le Lemme 2.10), et $S_\varepsilon \subset H \subsetneq S$, ce qui est contraire à l'hypothèse que $S \subset S_\varepsilon$. Donc $\lambda < u(\mathbf{x}) + \varepsilon$.

8. Si $S = \text{sat}(\text{cc}([u < \lambda], \mathbf{y}))$ est une forme inférieure, dans le cas où $S = S_\varepsilon$ nous devons avoir $\lambda \leq u(\mathbf{x}) + \varepsilon$. Dans le cas où $S \subsetneq S_\varepsilon$, nous avons soit $\mathbf{x} \in \text{cc}([u < \lambda], \mathbf{y})$, et alors $\lambda < u(\mathbf{x}) + \varepsilon$, soit \mathbf{x} est dans un trou H de $\text{cc}([u < \lambda], \mathbf{y})$, et puisque $\text{cc}([u < u(\mathbf{x}) + \varepsilon], \mathbf{x}) \not\subset H$, nous obtenons $\text{cc}([u < u(\mathbf{x}) + \varepsilon], \mathbf{x}) \cap \text{cc}([u < \lambda], \mathbf{y}) \neq \emptyset$, donc $\text{cc}([u < \lambda], \mathbf{y}) \subset \text{cc}([u < u(\mathbf{x}) + \varepsilon], \mathbf{x})$ et enfin $\lambda \leq u(\mathbf{x}) + \varepsilon$.

9. Ces résultats permettent de conclure dans les deux configurations possibles : $S_0 = X$ ou $S_0 \subsetneq X$.

10. Si $S_0 = X$, nous avons $S^* \subsetneq X$, impliquant (voir ci-dessus) $\lambda^* > u(\mathbf{x})$ et nous déduisons facilement

$$(\lambda^*, S_{\lambda^* - u(\mathbf{x})}) \preccurlyeq (\lambda^*, S^*).$$

Alors pour tout $\varepsilon > 0$, nous prenons $\varepsilon' = \min(\varepsilon, \lambda^* - u(\mathbf{x}))$ et si $(\lambda, S) \preccurlyeq (u(\mathbf{x}) + \varepsilon', S_{\varepsilon'})$, puisque $S_{\varepsilon'} \subsetneq X$, les résultats ci-dessus prouvent que $\lambda \leq u(\mathbf{x}) + \varepsilon'$, et puisque $S \subsetneq X$, $\lambda > u(\mathbf{x})$. Donc $|\lambda - u(\mathbf{x})| \leq \varepsilon' \leq \varepsilon$, prouvant l'Equation (2.7).

11. Si $S_0 \subsetneq X$, si de plus pour un certain $\varepsilon > 0$, nous avons $S_\varepsilon \subsetneq X$, nous concluons de la même manière que dans le cas précédent. Sinon, soit $(\lambda, S) \preccurlyeq (u(\mathbf{x}), S_0)$. Ceci implique $\lambda \geq u(\mathbf{x})$. Si $S = \text{sat}(\text{cc}([u < \lambda], \mathbf{y}))$ est de type inférieur, nécessairement $\lambda = u(\mathbf{x})$ et \mathbf{x} est dans un trou H de $\text{cc}([u < \lambda], \mathbf{y})$ et donc $\text{cc}([u \geq u(\mathbf{x})], \mathbf{x}) \subset H \subsetneq S$, ce qui contredit $(\lambda, S) \preccurlyeq (u(\mathbf{x}), S_0)$. Alors $S = \text{sat}(C)$ est de type supérieur, et si $\lambda > u(\mathbf{x})$, \mathbf{x} est dans un trou H de C , et d'après le Lemme 2.36, il existe une forme inférieure S' au niveau λ contenue dans H et contenant \mathbf{x} . Ceci entraîne facilement que $S_{\lambda - u(\mathbf{x})} \subset S' \subsetneq X$, contredisant l'hypothèse. En conclusion, $(\lambda, S) \preccurlyeq (u(\mathbf{x}), S_0)$ implique $\lambda = u(\mathbf{x})$ (et $S = S_0$), prouvant l'Equation (2.7), la borne inférieure étant en fait atteinte en $(u(\mathbf{x}), S_0)$.

12. Maintenant, soit \mathbf{x} un point dont l'unique forme le contenant est X , et nous supposons que l'hypothèse (2.9) est vérifiée. Soit $\lambda \leq u(\mathbf{x})$. Alors $C_1 = \text{cc}([u \geq$

$\lambda], \mathbf{x}) \neq \emptyset$ et donc $\text{sat}(C_1) = X$. D'après la condition (2.9), ceci implique que si C_2 est une composante connexe de $[u < \lambda]$, $(\lambda, \text{sat}(C_2)) \notin \text{LS}_{\mathbf{x}}^<$. Donc

$$u(\mathbf{x}) \leq \inf_{(\lambda, X) \in \text{LS}_{\mathbf{x}}^<} \lambda.$$

Réciproquement, pour $\lambda_n = u(\mathbf{x}) + \frac{1}{n}$, $\text{cc}([u < \lambda_n], \mathbf{x}) \neq \emptyset$, d'où $\text{sat}(\text{cc}([u < \lambda_n], \mathbf{x})) = X$ et donc $(\lambda_n, X) \in \text{LS}_{\mathbf{x}}^<$. Passer à la limite lorsque n tend vers l'infini donne la première égalité de (2.8). La seconde s'ensuit des mêmes arguments, sauf que la borne supérieure est atteinte puisque $(u(\mathbf{x}), X) \in \text{LS}_{\mathbf{x}}^{\geq}$. \square

Remarque 2.11. La condition supplémentaire pour reconstruire le fond n'est pas toujours vraie. Pour l'opérateur de saturation que nous avons défini dans \mathbb{R}^n , elle est vraie, car une composante connexe d'ensemble de niveau dont la saturation est \mathbb{R}^n contient un voisinage de l'infini. Toutefois, pour l'opérateur de saturation défini sur un sous-ensemble borné de \mathbb{R}^n , des exceptions peuvent arriver. C'est le cas si une ligne de niveau rencontre le cadre et coupe l'image exactement en moitiés, par exemple pour l'image définie sur $[0, 1]^2$ par $\chi_{[0, 1/2] \times [0, 1]}$. Néanmoins, mis à part ces cas particuliers, la condition est satisfaite. Cette condition peut être rapprochée de l'axiome de la saturation disant que si $C \subset X$, soit C soit $X \setminus C$ a pour saturation X . Ceci pousse à prendre X comme saturation de certains ensembles, alors que la condition ci-dessus pousse à rendre ce cas étroitement contrôlé. Dans la construction d'un opérateur de saturation, cet équilibre est délicat, par exemple pas totalement satisfaisant pour la saturation que nous avons définie sur un sous-ensemble borné de \mathbb{R}^n . Ceci demanderait des raffinements pour les composantes connexes d'ensembles de niveau rencontrant le cadre, et dont une composante connexe du complémentaire a une aire moitié de celle de l'image. Nous préférons ne pas entrer dans ce niveau de détails, parce que cela ne concerne qu'une famille très particulière d'images.

Donnons quelques exemples concernant la reconstruction. Il est clair que si \mathbf{x} appartient à un maximum régional de u , c'est-à-dire que tous les points de la composante connexe de l'ensemble isoniveau contenant \mathbf{x} sont des maxima locaux, la limite est en fait atteinte. Le niveau-forme $(u(\mathbf{x}), \text{sat}(\text{cc}([u \geq u(\mathbf{x})], \mathbf{x})))$ est plus petit ou égal que tout autre niveau-forme de $\text{LS}_{\mathbf{x}}$. Au contraire, à cause de l'inégalité stricte dans la définition des ensembles de niveau inférieurs, s'il y a un niveau-forme inférieur (λ, S) plus petit ou égal que tout niveau-forme supérieur contenant \mathbf{x} , la limite n'est pas atteinte. Effectivement, cela entraîne $u(\mathbf{x}) < \lambda$. C'est en particulier le cas lorsque \mathbf{x} est un maximum local.

Ces deux cas sont faciles à traiter, car dans la « minimisation » des niveaux-formes contenant \mathbf{x} , des niveaux monotones interviennent : il existe un niveau-forme (λ_0, S_0) tel que

$$(\mu, T) \preceq (\lambda, S) \preceq (\lambda_0, S_0) \quad \Rightarrow \quad \mu \leq \lambda \text{ (resp. } \mu \geq \lambda \text{)}.$$

Un cas plus complexe, où les deux types de niveaux-formes interviennent, est illustré dans la Figure 2.8. Cette figure montre le graphe de la fonction numérique définie par :

$$f(x) = \begin{cases} 0 & \text{pour } |x| \leq 1 \text{ ou } |x| > 4; \\ |x| - 1 & \text{pour } 1 \leq |x| \leq 4 \text{ et } |x| \notin 1 + \{1, 1/2 \cdots 1/n \cdots\}; \\ 2(|x| - 1) & \text{pour } |x| \in 1 + \{1, 1/2 \cdots 1/n \cdots\}. \end{cases} \quad (2.11)$$

La fonction radiale $u(\mathbf{x}) = f(\|\mathbf{x}\|)$ est semicontinue supérieurement (puisque f est elle-même semicontinue supérieurement). Une suite minimisante de niveaux-formes contenant l'origine \mathbf{O} est donnée par $(LS_n)_{n \in \mathbb{N}}$ avec

$$\begin{aligned} LS_{2n} &= (2/n, \text{sat}(\text{cc}([u \geq 2/n], (1 + 1/n, 0)))) \\ LS_{2n+1} &= (2/(n+1), \text{sat}(\text{cc}([u < 2/(n+1)], (1 + 1/n + 1/(2n(n+1)), 0)))) . \end{aligned}$$

C'est une suite décroissante de niveaux-formes, de type alterné, et il est clair que quel que soit le niveau-forme associé à l'origine, il existe un n tel que LS_n soit strictement plus petit que ce niveau-forme et de type différent.

Remarque 2.12. Même pour les fonctions continues u , ce cas complexe peut apparaître. Par exemple, nous pouvons modifier légèrement f dans l'exemple ci-dessus pour être linéaire dans les intervalles $[1 + \frac{1}{n} - \varepsilon_n, 1 + \frac{1}{n}]$ et $[1 + \frac{1}{n}, 1 + \frac{1}{n} + \varepsilon_n]$, pourvu que (ε_n) soit une suite vérifiant

$$1 + \frac{1}{n+1} + \varepsilon_{n+1} < 1 + \frac{1}{n} - \varepsilon_n.$$

2.5.3 Reconstruction indirecte

Le problème de la Formule (2.7) est qu'elle implique une quantité infinie de niveaux-formes, qui peut n'être pas même dénombrable. Cette difficulté vient principalement du fait que tous les niveaux-formes (contenant un point donné \mathbf{x}) sont pris en compte. Nous pouvons l'éviter si nous considérons seulement les formes ex-

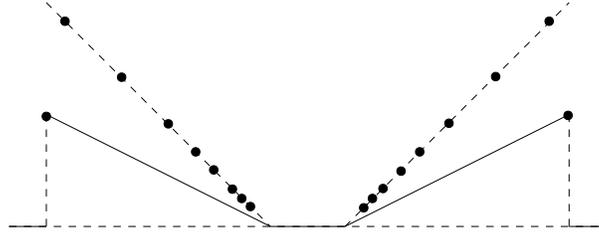


FIG. 2.8 – La « minimisation » des niveaux-formes contenant un point donné peut impliquer les deux types de niveaux-formes. Pour l'image radiale $u(\mathbf{y}) = f(\|\mathbf{x}\|)$ dont le graphe de la fonction réelle f apparaît ici, quel que soit le niveau-forme contenant l'origine \mathbf{O} , il existe un niveau-forme de type différent strictement plus petit pour la relation d'ordre \preceq . f est définie par la Formule (2.11). Il est facile de vérifier que f est semicontinue supérieurement (et donc également u).

traites d'un ensemble de niveau fixé : dans ce cas nous montrons que la donnée de l'ensemble de niveau ou de ses formes est équivalent. Cela donne une reconstruction en deux étapes de l'image : reconstruction des ensembles de niveau à partir des niveaux-formes associés, puis application de la formule (1.5). Cette section explique comment reconstruire un ensemble de niveau à partir de ses niveaux-formes.

Pour un niveau $\lambda \in \mathbb{R}$ et un point $\mathbf{x} \in X$, nous notons :

$$F_{\lambda, \mathbf{x}} = \{F \text{ fermé } \subset X : (\lambda, F) \in \text{LS}_{\mathbf{x}}\}$$

et

$$G_{\lambda, \mathbf{x}} = \{G \text{ ouvert } \subset X : (\lambda, G) \in \text{LS}_{\mathbf{x}}\}.$$

La réunion des deux ensembles est la famille des formes au niveau λ contenant \mathbf{x} , noté $\text{LS}_{\lambda, \mathbf{x}}$.

Le résultat permettant une reconstruction facile de l'ensemble de niveau λ de u à partir de $\text{LS}_{\lambda, \mathbf{x}}$ est le suivant :

Théorème 2.39 *La famille d'ensembles $\text{LS}_{\lambda, \mathbf{x}}$ est fermée par intersection (finie ou non), c'est-à-dire que si $I \subset \text{LS}_{\lambda, \mathbf{x}}$, alors $\bigcap_{A \in I} A \in \text{LS}_{\lambda, \mathbf{x}}$.*

Preuve.

1. Puisque les formes de $\text{LS}_{\lambda, \mathbf{x}}$ se rencontrent (au moins en \mathbf{x}), elles sont emboîtées.

2. Nous pouvons remplacer I par

$$I \cup \{A \in \text{LS}_{\lambda, \mathbf{x}} : \exists B \in I, B \subset A\},$$

tout en ne changeant pas l'ensemble $\bigcap_{A \in I} A$.

3. Supposons $\exists G \in I \cap G_{\lambda, \mathbf{x}}$ tel que $\forall A \in I, G \subset A$. Alors clairement $G = \bigcap_{A \in I} A$, et la preuve est faite. Si cette hypothèse est fausse, nous pouvons écrire

$$\forall G \in I \cap G_{\lambda, \mathbf{x}}, \exists F \in I, F \subset G \quad (*)$$

et supposer que $F \in F_{\lambda, \mathbf{x}}$, car si $F \in G_{\lambda, \mathbf{x}}$, il existe un $F' \in F_{\lambda, \mathbf{x}}$ tel que $F \subset F' \subset G$ (grâce au Lemme 2.36), et avec le remplacement de I effectué ci-dessus, nous avons $F' \subset I$, de telle sorte que nous pouvons remplacer F par F' .

4. L'assertion (*) donne facilement, avec l'hypothèse que $F \in F_{\lambda, \mathbf{x}}$:

$$\bigcap_{A \in I} A = \bigcap_{F \in I \cap F_{\lambda, \mathbf{x}}} F.$$

Cela montre que nous pouvons restreindre le reste de la preuve au cas où $I \subset F_{\lambda, \mathbf{x}}$.

5. Puisque les éléments de I sont des continus, leur intersection K est un continu, d'après le Corollaire 2.35. $X \setminus K$ peut s'écrire :

$$X \setminus K = \bigcup_{F \in I} (X \setminus F)$$

et pour $F \in I$, nous avons $X \setminus F$ connexe, ainsi $X \setminus K$ est une réunion d'ensembles connexes rencontrant un ensemble connexe $X \setminus F_0$ (F_0 est un élément arbitraire de I), donc connexe. Puisque X est univoqué, ceci entraîne que ∂K est connexe.

6. Il reste à montrer que $\partial K \subset [u \geq \lambda]$, ce qui impliquerait que ∂K est dans une composante connexe de $[u \geq \lambda]$ et puisque $K = \text{sat}(K) = \text{sat}(\partial K)$, nous pourrions écrire

$$K = \text{sat}(\text{cc}([u \geq \lambda], \mathbf{y}))$$

pour un $\mathbf{y} \in \partial K$, prouvant que $K \in F_{\lambda, \mathbf{x}}$.

7. En effet, soit $\mathbf{y} \in \partial K$ et U un voisinage connexe de \mathbf{y} (nous utilisons la connexité locale de X). Par définition de la frontière de K , il existe un point $\mathbf{z} \in U \setminus K$, et donc il y a un F dans I tel que $\mathbf{z} \notin F$. De plus, $U \cap F \neq \emptyset$, et comme U est connexe, $U \cap \partial F \neq \emptyset$. Donc $U \cap [u \geq \lambda] \neq \emptyset$, et U étant un voisinage arbitraire (connexe) de \mathbf{y} ,

$$\mathbf{y} \in \overline{[u \geq \lambda]} = [u \geq \lambda].$$

□

Remarque 2.13. Au cours de la démonstration, l’assertion plus précise suivante a été prouvée : soit il existe un $G \in I \cap G_{\lambda, \mathbf{x}}$ tel que $G = \bigcap_{A \in I} A$, soit cette intersection est dans $F_{\lambda, \mathbf{x}}$ (mais pas nécessairement dans I).

Remarque 2.14. Seule une version faible du Théorème 2.39 est utilisée pour prouver la formule de reconstruction indirecte : $\bigcap_{A \in \text{LS}_{\lambda, \mathbf{x}}} A \in \text{LS}_{\lambda, \mathbf{x}}$. Toutefois, le résultat plus général exposé ici sera utilisé dans le Chapitre 4.

La formule de reconstruction indirecte est la suivante :

Théorème 2.40 *Pour $\mathbf{x} \in X$ et $\lambda \in \mathbb{R}$,*

$$u(\mathbf{x}) \geq \lambda \Leftrightarrow \bigcap_{A \in \text{LS}_{\lambda, \mathbf{x}}} A \in F_{\lambda, \mathbf{x}}; \quad (2.12)$$

et inversement

$$u(\mathbf{x}) < \lambda \Leftrightarrow \bigcap_{A \in \text{LS}_{\lambda, \mathbf{x}}} A \in G_{\lambda, \mathbf{x}}. \quad (2.13)$$

Preuve.

1. Supposons que $u(\mathbf{x}) \geq \lambda$ et considérons la forme

$$F = \text{sat}(\text{cc}([u \geq \lambda], \mathbf{x})) \in F_{\lambda, \mathbf{x}}.$$

Nous montrons que $F = \bigcap_{A \in \text{LS}_{\lambda, \mathbf{x}}} A$. En effet, si

$$G = \text{sat}(\text{cc}([u < \lambda], \mathbf{y})) \in G_{\lambda, \mathbf{x}},$$

\mathbf{x} doit être dans un trou de $\text{cc}([u < \lambda], \mathbf{y})$, donc également $\text{cc}([u \geq \lambda], \mathbf{x})$. Cela donne $F \subset G$. De même, si $F' \subsetneq F$ et $F' \in F_{\lambda, \mathbf{x}}$, nous avons un G de $G_{\lambda, \mathbf{x}}$ (d’après le Théorème 2.39) entre F' et F , ce qui a été prouvé impossible.

2. Inversement, si $F = \bigcap_{A \in \text{LS}_{\lambda, \mathbf{x}}} A \in F_{\lambda, \mathbf{x}}$, nous pouvons écrire $F = \text{sat}(\text{cc}([u \geq \lambda], \mathbf{y}))$, et si \mathbf{x} était dans un trou de $\text{cc}([u \geq \lambda], \mathbf{y})$, nous aurions un élément de $G_{\lambda, \mathbf{x}}$ contenu dans F , grâce au Théorème 2.39, contredisant le fait que F est minimal.

3. L’Equation (2.13) est une conséquence directe de l’Equation (2.12) et du Théorème 2.39. □

La conséquence de ce théorème est que tout point a une plus petite forme qui le contient à un niveau donné. Si cette forme est du type inférieur, le point n'est pas dans l'ensemble de niveau, et si cette forme est du type supérieur, il appartient à l'ensemble de niveau. C'est ce qu'exprime le corollaire suivant :

Corollaire 2.41 *Pour $\lambda \in \mathbb{R}$, nous pouvons reconstruire l'ensemble de niveau au niveau λ d'une image semicontinue supérieurement par les formules :*

$$[u \geq \lambda] = \bigcup_{F \in F_{\lambda, \mathbf{x}}} (F \setminus \bigcup_{G \in G_{\lambda, \mathbf{x}}, G \subset F} G) \quad (2.14)$$

$$[u < \lambda] = \bigcup_{G \in G_{\lambda, \mathbf{x}}} (G \setminus \bigcup_{F \in F_{\lambda, \mathbf{x}}, F \subset G} F) \quad (2.15)$$

Preuve. Ces égalités sont des conséquences directes du Théorème 2.40. □

Chapitre 3

Fast Level Set Transform

3.1 Intérêt de l’algorithme

La structure d’arbre d’inclusion de formes, étudiée dans le chapitre précédent, a son équivalent pour les images numériques. Cette structure, interprétée dans le cadre discret, est d’un grand intérêt. C’est une représentation invariante par changement de contraste de l’image, codant la structure topologique de l’image. Ceci sera utilisé pour appliquer divers filtres dans le chapitre suivant. Nous avons implémenté un algorithme rapide pour décomposer une image numérique (permettant également une reconstruction triviale), qui repose fortement sur la structure d’arbre des formes. Une version préliminaire de l’algorithme, moins efficace, a été présentée dans [59].

3.2 Domaine continu vs. domaine discret

3.2.1 Généralités

D’abord nous avons besoin de trouver une définition appropriée de la saturation, des trous, des formes pour des images numériques. C’est un problème classique en traitement d’image : les images définies sur un domaine continu sont plus faciles à étudier dans un cadre théorique, car la plupart des outils mathématiques leur sont adaptés, mais du côté de l’ordinateur, nous devons traiter des images discrètes¹. La traduction des images définies sur un domaine continu aux images discrètes n’est que rarement directe, et implique souvent pour le moins des problèmes numériques,

¹Dans cette section, le terme «image définie sur un domaine continu» signifie une image définie sur un continu, par opposition aux images discrètes (ou numériques), ce qui signifie un tableau de valeurs.

comme par exemple quand nous devons calculer des dérivées. Ceci serait moins crucial si les images discrètes étaient échantillonnées en accord avec le taux de Nyquist, mais ce n'est quasiment jamais le cas, ainsi nous n'avons aucun moyen de retrouver l'image définie sur un domaine continu à partir de l'image discrète.

Pour ce problème de traduction, deux stratégies peuvent être adoptées : interpréter l'image discrète comme une image définie sur un domaine continu, ou redéfinir les notions utilisées dans le cas du domaine continu au cas discret. La deuxième solution demande plus de travail, puisque les contreparties discrètes des notions du domaine continu doivent être définies, mais de plus les propriétés prouvées dans le domaine continu ne sont pas forcément vraies avec les notions discrètes. Par exemple, des inconsistances topologiques sont bien connues pour la connexité : dans le cas discret, deux notions de connexité existent, la 4- et la 8-connexité, mais aucune n'est satisfaisante seule, car toutes deux peuvent produire des situations qui ne peuvent arriver dans le cas du domaine continu. Un énorme travail existe dans l'étude de la topologie discrète, voir par exemple Rosenfeld [67, 68] et la revue de Kong et Rosenfeld dans [31]. La première solution, interpréter l'image discrète comme une image définie sur un domaine continu, implique une étape d'interpolation, qui peut être plus ou moins chère en tant de calcul. Notons que cette solution ne prétend pas travailler sur l'image définie sur un domaine continu dont la version échantillonnée est observée, puisque cette image définie sur un domaine continu ne peut pas être retrouvée. A la place, une image définie sur un domaine continu est définie en se basant sur les valeurs des pixels.

3.2.2 Notre modèle d'interpolation

La solution adoptée est d'interpréter l'image discrète comme une image définie sur un domaine continu, donc en prenant avantage de tous les résultats prouvés dans le chapitre précédent. Nous supposons ainsi que l'image numérique u_d fournit des valeurs sur la grille régulière à coordonnées demi entières :

$$u_d : \Omega_d = \{1/2, 3/2 \dots W - 1/2\} \times \{1/2, 3/2 \dots H - 1/2\} \rightarrow \mathbb{R},$$

où W et H sont la largeur et la hauteur de l'image. L'image u_d est la donnée de $W \times H$ valeurs. Nous interpolons ces valeurs avec une interpolation très primitive, assurant néanmoins que l'image définie sur un domaine continu en résultant est semi-continue supérieurement et déduite de u_d d'une manière invariante par changement

de contraste. D'abord, nous définissons une interpolation préliminaire \tilde{u}_c par

$$\tilde{u}_c : \quad \bar{\Omega} = [0, W] \times [0, H] \rightarrow \mathbb{R}$$

$$\mathbf{x} \mapsto \begin{cases} u_d(\mathbf{x}) & \text{if } \mathbf{x} \in \Omega_d \\ -\infty & \text{if } \mathbf{x} \in \bar{\Omega} \setminus \Omega_d \end{cases}$$

Puis nous appliquons à \tilde{u}_c une dilatation par l'élément structurant $B = [-1/2, 1/2]^2$ (carré fermé) :

$$u_c(\mathbf{x}) = D_B \tilde{u}_c(\mathbf{x}) = \sup_{\mathbf{y} \in \mathbf{x} + B} \tilde{u}_c(\mathbf{y}).$$

Comme \tilde{u}_c , considéré comme une fonction de $\bar{\Omega}$ dans \mathbb{R} , est semicontinue supérieurement, et comme u_c est le dilaté de \tilde{u}_c par un ensemble *fermé*, u_c est aussi semicontinue supérieurement. De la même manière, puisque \tilde{u}_c se déduit de u_d par un procédé invariant par changement de contraste, et u_c de \tilde{u}_c par une dilatation (qui est un filtre morphologique), nous concluons que u_c est une interpolée de u_d par une interpolation invariante par changement de contraste².

Le résultat de cette interpolation est aisé à établir : un point \mathbf{x} dont les deux coordonnées ne sont pas des entiers reçoit la valeur de u_d au point de Ω_d le plus proche (il est unique), et si \mathbf{x} a au moins une coordonnée entière, il peut y avoir plusieurs points les plus proches dans Ω_d , et sa valeur par u_c est le maximum des valeurs de u_d à ces points les plus proches (voir la Figure 3.1). En d'autres termes, le pixel ouvert (image element i, j) $(i, i + 1) \times (j, j + 1)$ reçoit la même valeur $u_d(i + 1/2, j + 1/2)$, l'edgel (edge element i, j) vertical ouvert $\{i\} \times (j, j + 1)$ la valeur $\max(u_d(i - 1/2, j + 1/2), u_d(i + 1/2, j + 1/2))$ pourvu que $1 \leq i \leq W - 1$ ($u_d(1/2, j + 1/2)$ si $i = 0$ et $u_d(W - 1/2, j + 1/2)$ si $i = W$), l'edgel horizontal ouvert $(i, i + 1) \times \{j\}$ la valeur $\max(u_d(i + 1/2, j - 1/2), u_d(i + 1/2, j + 1/2))$ pourvu que $1 \leq j \leq H - 1$ ($u_d(i + 1/2, 1/2)$ si $j = 0$ et $u_d(i + 1/2, H - 1/2)$ si $j = H$) et le pointel (point element i, j) $\{i\} \times \{j\}$ la valeur

$$\max(u_d(i - 1/2, j - 1/2), u_d(i + 1/2, j - 1/2), u_d(i - 1/2, j + 1/2), u_d(i + 1/2, j + 1/2))$$

(en remplaçant dans la max les valeurs à des points hors de Ω_d par $-\infty$, s'il en existe).

²Notons qu'une interpolation invariante par translation et linéaire, c'est-à-dire une convolution, ne serait pas une interpolation invariante par changement de contraste.

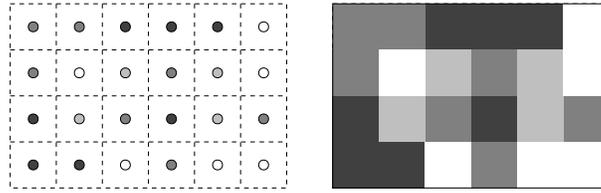


FIG. 3.1 – Interpolation (simple) invariante par changement de contraste que nous utilisons pour considérer l’image numérique u_d (figure de gauche) comme une image définie sur un domaine continu u_c (image de droite). Aucun niveau de gris nouveau n’est introduit et les valeurs aux edgels et pointels sont telles que u_c est semicontinue supérieurement.

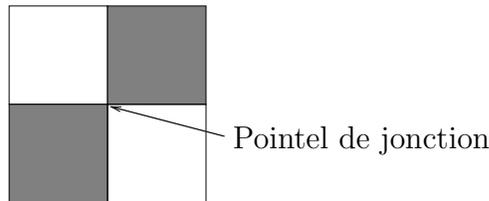


FIG. 3.2 – La connexité des pixels clairs ou foncés dépend de la valeur attribuée par l’image au pointel de jonction.

3.2.3 Conséquence sur la connexité

Dans ce cadre, le problème classique du choix de la connexité discrète est implicitement résolu. Dans la situation ambiguë illustrée dans la Figure 3.2, où $u_d(i - 1/2, j - 1/2) = u_d(i + 1/2, j + 1/2) = 1$ et $u_d(i - 1/2, j + 1/2) = u_d(i + 1/2, j - 1/2) = 0$, la connexité des pixels opposés en diagonale dépend de la valeur au pointel (i, j) . Si cette valeur est 1 (resp. 0), les pixels haut-gauche et bas-droite sont connectés (resp. déconnectés) tandis que les pixels haut-droite et bas-gauche sont déconnectés (resp. connectés). Avec notre choix, ce pointel reçoit la valeur 1. En d’autres termes, cela signifie que nous utilisons la 8-connexité pour les ensembles de niveau supérieurs et la 4-connexité pour les ensembles de niveau inférieurs.

3.2.4 Formes dans les images numériques

Formes numériques

Nous appellerons *pixel* de u_d un point de Ω_d . $u_d(P) = u_c(P)$ est appelé la valeur au pixel P . Le *pixel ouvert* associé à $P = (i + 1/2, j + 1/2)$, noté $|P|$, est le carré ouvert $(i, i + 1) \times (j, j + 1)$. Comme u_c est constante sur un pixel ouvert, nous pouvons

parler de la valeur de u en un pixel ouvert.

Si S est un ensemble de pixels, nous notons $|S|$ l'union des pixels ouverts associés aux pixels de S :

$$|S| = \bigcup_{P \in S} |P|.$$

Si S_c est une forme de u_c , nous appelons forme numérique S_d associée à S_c l'ensemble des pixels appartenant à S_c . Donc, c'est un sous-ensemble de Ω_d . Des définitions similaires s'appliquent aux ensembles de niveau supérieurs numériques et ensembles de niveau inférieurs numériques. Nous parlerons aussi de composantes connexes numériques de ces ensembles.

Le lemme suivant se révélera utile :

Lemme 3.1 *Soit u_c une image définie sur un domaine continu déduite d'une image numérique u_d . Soit $\mathbf{x} \in \bar{\Omega}$.*

1. *Il existe un voisinage U de \mathbf{x} tel que pour tout pixel ouvert $|P|$,*

$$|P| \cap U \neq \emptyset \quad \Rightarrow \quad \mathbf{x} \in \overline{|P|}$$

2. *Si nous notons $P_i, i = 1 \dots k$, les pixels ouverts P tels que $\mathbf{x} \in \overline{|P|}$, alors*

$$\mathbf{x} \in \overline{\bigcup_{i=1}^k |P_i|}$$

Preuve.

1. Considérons l'union V des pixels ouverts $|P|$ tels que $\mathbf{x} \in \overline{|P|}$. Alors $U = \overset{\circ}{V}$ répond à la question. En effet, $|P|$ étant un pixel ouvert rencontrant U , nous avons $\bar{V} \cap |P| \neq \emptyset$, ce qui implique que $|P|$ appartient à V , car nous pouvons écrire

$$|P| \cap \bar{V} = |P| \cap \bigcup_{|Q| \in V} \overline{|Q|} = \bigcup_{|Q| \in V} |P| \cap \overline{|Q|}$$

et si $|P|$ et $|Q|$ sont différents, alors $|P| \cap \overline{|Q|} = \emptyset$.

2. Puisque pour tout $i, \mathbf{x} \in \overline{|P_i|}$, nous avons

$$\mathbf{x} \in \overline{\bigcup_{i=1}^k |P_i|}.$$

Soit U un voisinage de \mathbf{x} comme dans le point précédent et $\mathbf{y} \in U$. Soit $|P|$ tel que

$\mathbf{y} \in \overline{|P|}$. Comme U est un voisinage de \mathbf{y} , nous avons $U \cap \overline{|P|} \neq \emptyset$. Par propriété de U , ceci implique que $\mathbf{x} \in \overline{|P|}$, d'où P est un des P_i . Donc $\mathbf{y} \in \bigcup_{i=1}^k |P_i|$, ce qui montre que U est contenu dans $\bigcup_{i=1}^k |P_i|$ et puisque c'est un voisinage de \mathbf{x} ,

$$\mathbf{x} \in \bigcup_{i=1}^k |P_i|.$$

□

Montrons quelques propriétés élémentaires liant les formes aux formes numériques.

Proposition 3.2 *Soit u_c une image définie sur un domaine continue déduite d'une image numérique u_d , S_c une forme (resp. une composante connexe d'ensemble de niveau) de u_c et S_d sa forme numérique associée (resp. composante connexe numérique).*

1. Si S_c est de type supérieur, alors $S_c = \overline{|S_d|}$;
2. Si S_c est de type inférieur, alors $S_c = \overline{\overset{\circ}{|S_d|}}$.

Preuve.

1. Quel que soit le type de S_c , par définition de S_d , nous avons $S_d \subset S_c$. Si $P \in S_d$, u_c est constante sur $|P|$, donc une composante connexe d'ensemble de niveau rencontrant $|P|$ contient $|P|$. Si S_c est une composante connexe d'ensemble de niveau, comme $P \in S_c \cap |P|$, alors $|P| \subset S_c$. Si S_c est une forme, si P appartient à la composante connexe d'ensemble de niveau C sur laquelle elle repose, le résultat est le même; sinon, P appartient à un trou H de C , et C ne peut pas rencontrer $|P|$, et comme $|P|$ est connexe, $|P| \subset H \subset S_c$. Cela donne $|S_d| \subset S_c$.

2. Si S_c est de type supérieur, c'est un fermé, de telle sorte que nous avons immédiatement $\overline{|S_d|} \subset S_c$. Réciproquement, soit $\mathbf{x} \in S_c$. Si \mathbf{x} a des coordonnées non entières, \mathbf{x} est dans un pixel ouvert $|P|$, and donc $\mathbf{x} \in |P| \subset |S_d|$. Si \mathbf{x} a au moins une coordonnée entière, il y a un pixel P tel que $\mathbf{x} \in \overline{|P|}$ et $u_c(P) = u_c(\mathbf{x})$. Donc $|P| \cup \{\mathbf{x}\}$ est connexe et contenu dans un ensemble isoniveau de u_c . Donc $P \in S_c$, montrant que $\mathbf{x} \in \overline{|S_d|}$.

3. Si S_c est de type inférieur, c'est un ouvert. Soit $\mathbf{x} \in S_c$. Si \mathbf{x} a des coordonnées non entières, \mathbf{x} appartient à un pixel ouvert $|P|$ qui donc est inclus dans S_c . Sinon, il existe une boule ouverte centrée en \mathbf{x} contenue dans S_c et de rayon suffisamment petit telle que tous les pixels ouverts la rencontrant contiennent \mathbf{x} dans leur frontière.

Alors il existe dans cette boule un point \mathbf{y} avec des coordonnées non entières, appartenant à un pixel ouvert $|P|$ et donc $\mathbf{x} \in \overline{|P|}$. Comme $|P| \cap S_c \neq \emptyset$, nous déduisons que $P \in S_d$ et donc $S_c \subset \overline{|S_d|}$ et puisque S_c est ouvert, $S_c \subset \overline{\overset{\circ}{|S_d|}}$. Réciproquement, soit $\mathbf{x} \in \overline{\overset{\circ}{|S_d|}}$. Si \mathbf{x} a des coordonnées non entières, \mathbf{x} appartient à un pixel ouvert $|P|$ et comme $|S_d| \subset S_c$, il appartient à S_c . Si \mathbf{x} a au moins une coordonnée entière, soit B une boule de centre \mathbf{x} contenue dans $\overline{|S_d|}$ et de rayon tel que tout pixel ouvert rencontrant B contient \mathbf{x} dans sa frontière. Clairement, ces pixels ouverts sont dans $\overline{|S_d|}$, soit $|P|$ un tel pixel ouvert et $\mathbf{y} \in |P|$. Comme $|P|$ est un voisinage de \mathbf{y} , il existe un point de $|S_d|$ dans $|P|$ et donc $P \in S_d$. Cela montre que tous les pixels ouverts rencontrant B sont dans $|S_d|$ et donc que $\mathbf{x} \in |S_d|$, qui à son tour est dans S_c . \square

Connexité numérique

Puisque les deux notions de connexité discrète sont utilisées, nous ne pouvons pas parler de la connexité de n'importe quel ensemble de pixels en général sans référence à l'image. La connexité dépend des valeurs aux pixels !

Plus précisément, à un pixel $P \in \Omega_d$, nous associons l'ensemble P^b de $\bar{\Omega}$ suivant :

$$P^b = \overline{|P|} \cap \{\mathbf{x} \in \bar{\Omega} : u_c(\mathbf{x}) = u_d(P)\}.$$

C'est-à-dire $|P|$ et les edgels au même niveau ainsi que les pointels au même niveau. A un ensemble discret $D \in \Omega_d$, nous associons D^b défini par :

$$D^b = \bigcup_{P \in D} P^b,$$

et nous disons que D est connexe si et seulement si D^b l'est.

Donc, une composante connexe d'ensemble de niveau inférieur de u_d est une composante 4-connexe, une composante connexe d'un ensemble de niveau supérieur de u_d est une composante 8-connexe et pour un ensemble isoniveau, cela dépend : certains pixels peuvent être considérés en 4-connexité alors que d'autres le sont en 8-connexité (voir Figure 3.3).

Un ensemble 4-connexe de pixels reste connexe indépendamment de u . La connexité relativement à u d'un ensemble 8-connexe de pixels sans être 4-connexe dépend des valeurs de u en ses 4-voisins.

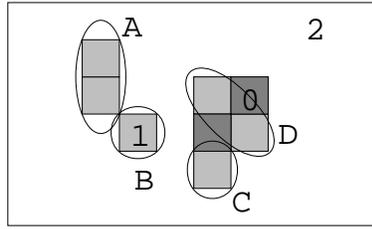


FIG. 3.3 – L'ensemble numérique isoniveau $u_c = 1$ a 4 composantes connexes, A , B , C et D . Cela montre que certains pixels voisins sont considérés en 4-connexité et d'autres en 8-connexité, en fonction des valeurs aux autres pixels voisins.

Quand nous parlons d'une composante connexe C d'ensemble de niveau de u_d , nous appelons *voisins* de C les pixels 4-adjacents à un pixel de C et hors de C si C provient d'un ensemble de niveau inférieur, ou bien les pixels 8-adjacents si C provient d'un ensemble de niveau supérieur. Pour un ensemble qui n'est pas une composante connexe d'ensemble de niveau, nous devons toujours préciser si nous entendons 4-voisins ou 8-voisins.

Niveaux associés aux formes

Dans un but de reconstruction, nous avons besoin de stocker des niveaux-formes, pas seulement des formes. C'est-à-dire qu'il faut enregistrer le niveau auquel une composante de l'ensemble de niveau donne la forme. Nous nous intéressons dans cette section au niveau que nous devons associer à la forme.

Avec notre modèle d'interpolation, les valeurs de l'image numérique u_d et de l'image définie sur un domaine continu u_c sont les mêmes : $u_d(\Omega_d) = u_c(\bar{\Omega})$, donc un ensemble fini de valeurs. C'est pourquoi nous voudrions décrire la famille des formes de u_d par une structure finie. La définition des formes de u_d est directe : si S_c est une forme de u_c , nous définissons la forme numérique S_d comme les points de Ω_d appartenant à S_c . Si nous énumérons l'ensemble des valeurs en ordre croissant :

$$u_d(\Omega_d) = u_c(\bar{\Omega}) = \{u_1, u_2 \cdots u_n\},$$

toute composante connexe d'ensemble de niveau de u_c est associée à plusieurs (une infinité de) niveaux. Si la composante connexe d'ensemble de niveau supérieur U_c est écrite $U_c = \text{cc}([u \geq \lambda], \mathbf{x})$ avec $u_i < \lambda \leq u_{i+1}$ alors nous avons

$$\forall \mu \in (u_i, u_{i+1}], \quad U_c = \text{cc}([u \geq \mu], \mathbf{x}).$$

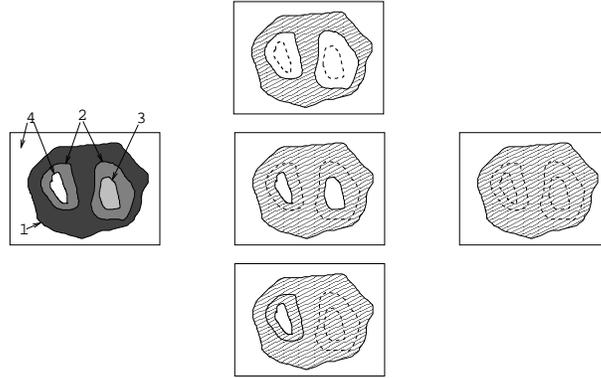


FIG. 3.4 – Deux composantes connexes différentes d’ensembles de niveau peuvent donner la même forme. Gauche : un image u avec les niveaux de gris indiqués. Milieu : de haut en bas, les composantes connexes d’ensembles de niveau inférieurs $cc([u \leq 1])$, $cc([u \leq 2])$ et $cc([u \leq 3])$. Droite : leur forme associée commune.

De même, si la composante connexe d’ensemble de niveau inférieur L_c est écrite $L_c = cc([u < \lambda], \mathbf{x})$ avec $u_i < \lambda \leq u_{i+1}$ alors

$$\forall \mu \in (u_i, u_{i+1}], \quad L_c = cc([u < \mu], \mathbf{x}).$$

Notons que dans ce dernier cas, nous pouvons aussi écrire :

$$L_c = cc([u \leq u_i], \mathbf{x}).$$

Un autre fait important à noter est que différentes composantes connexes d’ensembles de niveau peuvent donner la même forme, même si cette forme n’est pas le support complet de l’image $\bar{\Omega}$: il suffit que la composante grossisse par les trous seulement, lorsque l’on change le seuil. Ceci est illustré dans la Figure 3.4. Bien sûr, lorsque cette forme n’est pas $\bar{\Omega}$, tous ces ensembles de niveau sont du même type. En vue de la reconstruction, nous ne stockerons dans de telles situations que le niveau le plus strict, c’est-à-dire le niveau correspondant au plus petit niveau-forme pour la relation \preceq . Cela signifie que nous enregistrons le plus grand niveau de gris pour une forme supérieure et le plus petit pour une forme inférieure. Comme nous l’avons remarqué, ces niveaux les plus stricts sont dans $u_d(\Omega_d)$.

En ce qui concerne la forme numérique Ω_d , la racine de l’arbre, correspondant à la forme du domaine continu $\bar{\Omega}$, tous les points de $\bar{\Omega}$ non inclus dans une autre forme ont le même niveau de gris (sinon, nous ne pourrions pas reconstruire l’image). Nous enregistrons ce niveau en association à Ω_d .

Le nombre de formes

Nous montrons que le nombre de formes numériques ne dépasse pas le nombre de pixels. C'est une propriété cruciale dans la pratique, puisque, pourvu que la taille nécessaire pour stocker une forme est constante, cela assure que la mémoire maximale nécessaire pour l'arbre des formes est bornée par la taille de l'image, indépendamment de son contenu.

Lemme 3.3 *Si une forme de u_c rencontre une composante connexe d'ensemble isoniveau, cette composante connexe d'ensemble isoniveau est incluse dans la forme.*

Preuve. En effet, nous avons deux possibilités : soit la composante connexe d'ensemble isoniveau rencontre la composante connexe d'ensemble de niveau sur laquelle la forme repose, auquel cas la composante d'ensemble isoniveau est contenue dedans, soit elle rencontre un trou de la composante connexe d'ensemble de niveau, auquel cas la composante d'ensemble isoniveau est dans le trou. \square

Proposition 3.4 *Le nombre de formes numériques dans une image numérique est fini.*

Preuve. Comme nous l'avons prouvé, toute composante connexe d'ensemble de niveau de u_c peut s'écrire comme :

$$U_c = \text{cc}([u \geq u_i], \mathbf{x}) \quad \text{or} \quad L_c = \text{cc}([u \leq u_i], \mathbf{x}),$$

pour un certain i .

Soit $i \in \{1 \cdots n\}$ et $\mathbf{x} \in \bar{\Omega}$. Si \mathbf{x} est dans un pixel ouvert (i.e., aucune de ses coordonnées n'est entière), la composante connexe d'ensemble isoniveau contenant \mathbf{x} contient le pixel ouvert contenant \mathbf{x} . Si \mathbf{x} est dans un edgel ou est un pointel, \mathbf{x} est au même niveau qu'un des pixels ouverts adjacents. Donc ce pixel ouvert est contenu dans la composante connexe d'ensemble isoniveau contenant \mathbf{x} . Cela prouve que chaque composante connexe d'ensemble de niveau de u_c au niveau u_i contient au moins un pixel ouvert. Puisque ces composantes connexes sont disjointes, et que le nombre de pixels ouverts est fini, nous déduisons que le nombre de composantes connexes d'ensembles de niveau à un niveau donné est fini, et donc également le nombre de formes. Puisqu'il n'y a qu'un nombre fini de i , le nombre de formes est fini. \square

Pour le prochain résultat, nous utilisons à plusieurs reprises le lemme suivant :

Lemme 3.5 *La réunion d'un nombre fini (≥ 2) de fermés connexes disjoints n'est pas connexe.*

Preuve. Si les C_i ($i = 1 \dots n$) sont disjoints, fermés et connexes, C_1 est fermé relativement à $\bigcup C_i$, et également ouvert dans $\bigcup C_i$ puisque son complémentaire dans $\bigcup C_i$ est l'union des C_i pour $i \geq 2$, chacun étant fermé. \square

Proposition 3.6 *Toute forme S d'une image numérique contient au moins un pixel ouvert n'appartenant à aucune forme strictement incluse dans S .*

Preuve.

1. Si S ne contient pas strictement une autre forme, il suffit de montrer que S contient au moins un pixel ouvert, ce qui est une partie de la démonstration de la Proposition 3.4. Pour la suite, nous supposons que S contient strictement au moins une autre forme.

2. Considérons la famille de formes strictement contenues dans S , partiellement ordonnées par inclusion. Puisque leur nombre est fini, nous pouvons trouver les éléments maximaux, $\text{sat}(U_1) \dots \text{sat}(U_k)$ et $\text{sat}(L_1) \dots \text{sat}(L_l)$ (k ou l peut être 0, mais pas les deux). Il suffit de montrer que S contient *strictement* leur union pour conclure, à l'aide du même argument que dans la preuve de la Proposition 3.4.

3. Supposons que S n'est pas $\bar{\Omega}$. Si S est fermé, ∂S est connexe, donc ce n'est pas $\partial \text{sat}(U_1) \cup \dots \cup \partial \text{sat}(U_k)$ puisque ces frontières sont deux à deux disjointes. Soit

$$\mathbf{x} \in \partial S \setminus \bigcup_{m=1}^k \partial \text{sat}(U_m).$$

Puisque les autres formes sont ouvertes, \mathbf{x} n'appartient à aucune d'elles. Donc \mathbf{x} est dans S et pas dans une forme strictement contenue dans S .

4. Supposons maintenant que S est ouvert. Si $l = 0$, alors $\text{sat}(U_1) \cup \dots \cup \text{sat}(U_k)$ est fermé et non égal à S car S est ouvert (et différent de $\bar{\Omega}$) et la preuve est faite. Sinon, considérons $\partial \text{sat}(L_1)$. C'est un ensemble connexe. Supposons que

$$\partial \text{sat}(L_1) \subset \partial \text{sat}(U_1) \cup \dots \cup \partial \text{sat}(U_k). \quad (*)$$

Alors sélectionnons les $\partial \text{sat}(U_j)$ qui rencontrent $\partial \text{sat}(L_1)$. Il n'en existe qu'une, puisque sinon nous aurions un ensemble connexe ($\partial \text{sat}(L_1)$) qui rencontre chacune des composantes connexes $\partial \text{sat}(U_j)$, donc leur réunion, c'est-à-dire $\partial \text{sat}(U_1) \cup \dots \cup$

$\partial\text{sat}(U_k)$, serait connexe, ce qui n'est pas le cas (voir le Lemme 3.5). Donc il y a un j tel que $\partial\text{sat}(L_1) \subset \partial\text{sat}(U_j)$. Comme $\text{sat}(U_j)$ est fermé, nous avons

$$\text{sat}(\partial\text{sat}(U_j)) = \text{sat}(\text{sat}(U_j)) = \text{sat}(U_j)$$

et donc $\text{sat}(\partial\text{sat}(L_1)) \subset \text{sat}(U_j)$ and d'après la Proposition 2.18, nous déduisons

$$\text{sat}(L_1) = \text{sat}(\text{sat}(L_1)) \subset \text{sat}(\partial\text{sat}(L_1)) \subset \text{sat}(U_j)$$

ce qui contredit la maximalité de $\text{sat}(L_1)$ parmi les formes strictement contenues dans S . Donc l'Equation (*) est fausse.

5. De plus, nous savons que $\partial\text{sat}(L_1) \subset \bar{S}$. Supposons que

$$\partial\text{sat}(L_1) \subset \partial S \cup \bigcup_{m=1}^k \partial\text{sat}(U_m). \quad (**)$$

Comme (*) est faux, nous avons $\partial\text{sat}(L_1) \cap \partial S \neq \emptyset$, et puisque $L_1 \neq S$ et que S est connexe, $\partial\text{sat}(L_1) \not\subset \partial S$. Donc (**) donne :

$$\partial\text{sat}(L_1) \cap \bigcup_{m=1}^k \partial\text{sat}(U_m) \neq \emptyset$$

et du fait que $\partial\text{sat}(L_1)$ est connexe,

$$\partial S \cup \bigcup_{m=1}^k \partial\text{sat}(U_m)$$

est connexe. Mais pour tout m ,

$$\partial S \cap \partial\text{sat}(U_m) \subset \partial S \cap \partial U_m \subset \partial S \cap U_m \subset (\bar{\Omega} \setminus S) \cap U_m = \emptyset.$$

ce qui montre que

$$\partial S \cup \bigcup_{m=1}^k \partial\text{sat}(U_m)$$

ne peut être connexe. Donc (**) ne peut pas être vérifié et nous concluons que

$$\partial\text{sat}(L_1) \cap \left(S \setminus \bigcup_{m=1}^k \partial\text{sat}(U_m) \right) \neq \emptyset$$

et comme il est clair que $\partial\text{sat}(L_1) \cap \text{sat}(L_j) = \emptyset$ si $2 \leq j \leq l$, alors nous avons un point de $\partial\text{sat}(L_1)$ appartenant à S mais à aucune forme strictement incluse dans S .

6. Le dernier cas à examiner est $S = \bar{\Omega}$. Comme S ne peut être partitionné en un nombre fini d'ensembles fermés disjoints $\text{sat}(U_m)$ (cela contredirait sa connexité), et si $l = 0$, il suffit de prendre un point de S hors de tout $\text{sat}(U_m)$. Donc supposons maintenant que $l \geq 1$. Comme dans le cas précédent, nous pouvons prouver que (*) n'est pas possible. De là, la conclusion est plus facile que dans le cas précédent puisque $\partial S = \emptyset$, montrant que tout point de

$$\partial\text{sat}(L_1) \setminus \bigcup_{m=1}^k \partial\text{sat}(U_m)$$

répond à la question. □

L'interprétation de la Proposition 3.6 est que toute forme S contient un point de Ω_d qui n'est pas dans une forme strictement incluse, c'est-à-dire un point dont la plus petite forme le contenant est S . Cela donne une injection entre les formes de u_c et les points de Ω_d , prouvant que le nombre de formes ne peut dépasser le nombre de pixels.

Remarque 3.1. Dans le cas discret, une petite liberté peut être prise par rapport aux axiomes de l'opérateur de saturation. Nous pouvons tolérer que la saturation du complémentaire d'un ensemble saturé n'est pas Ω_d mais lui-même, et pourtant avoir une extraction et une reconstruction, dans la cas où cet ensemble saturé rencontre le cadre de l'image et soit d'aire exactement moitié celle de l'image. Dans ce cas, nous pouvons tolérer que l'image soit coupée en deux formes (ceci concerne des images très particulières, et est très improbable dans les images naturelles). Dans ce cas, la racine de l'arbre ne contient pas de pixel propre, et le nombre de formes peut atteindre le nombre de pixels plus un, mais pas plus.

Plus petite forme

Pour le point $P = (i + 1/2, j + 1/2)$, nous considérons les deux formes :

$$S_P^> = \text{sat}(\text{cc}([u \geq u_d(P)], P)) \quad \text{et} \quad S_P^< = \text{sat}(\text{cc}([u \leq u_d(P)], P)).$$

Nous affirmons que l'une d'elles est la plus petite forme contenant P . En effet, la famille des formes contenant P est finie, et elles sont par définition non disjointes,

donc elles sont emboîtées, montrant qu'il existe une plus petite forme contenant P . Soit C une composante connexe d'ensemble de niveau (notons son niveau λ) dont la saturation est cette plus petite forme contenant P . Alors P appartient à C , sinon P serait contenu dans un trou de C , qui, d'après ce qu'on a montré plus haut, contredirait la minimalité de la forme. Si C est extrait d'un ensemble de niveau supérieur (resp. inférieur), alors $u_d(P) \geq \lambda$ (resp. $u_d(P) \leq \lambda$) et C contient la composante connexe d'ensemble de niveau du même type au niveau $u_d(P)$ et par argument de minimalité, elles devraient être égales.

Proposition 3.7 *Un trou d'une composante connexe d'ensemble de niveau dans une image numérique est une forme.*

Preuve.

1. Soit C une composante connexe d'ensemble de niveau, C_d sa contrepartie numérique et H un trou de C . Pour tout $\mathbf{x} \in \partial H$, montrons qu'il existe un pixel ouvert dans H dont la frontière contient \mathbf{x} . Sinon, tous les pixels ouverts dont la frontière contient \mathbf{x} appartiendraient à C ou une autre composante connexe du complémentaire de C . Ce dernier cas est impossible puisqu'une telle composante D serait telle que $\overline{D} \cap H \neq \emptyset$ et donc $H \cup D$ serait connexe. Le premier cas l'est également car il impliquerait que tous les pixels ouverts dont la frontière contient \mathbf{x} seraient dans C , et si nous notons P_i les pixels associés, nous aurions grâce au point 2 du Lemme 3.1 :

$$\mathbf{x} \in \bigcup_i \overset{\circ}{P}_i \subset \overset{\circ}{|C_d|}.$$

Selon la Proposition 3.2, nous aurions

$$\overset{\circ}{|C_d|} = \overset{\circ}{C}$$

et donc $\mathbf{x} \in \overset{\circ}{C}$, contrairement à l'hypothèse $\mathbf{x} \in \partial H$. Soit F la réunion des pixels ouverts de H dont la frontière rencontre ∂H , qui est donc non vide.

2. Supposons que C est de type supérieur, $C = \text{cc}([u \geq \lambda], \mathbf{x})$. Soit μ le maximum de u_c sur F (ce maximum existe, puisque u_c prend seulement un nombre fini de valeurs). Alors si $\mathbf{y} \in F$

$$H = \text{sat}(\text{cc}([u \leq \mu], \mathbf{y})).$$

En effet, si nous notons H' la forme du membre de droite, nous avons $\mu < \lambda$, et donc $\text{cc}([u \leq \mu], \mathbf{y}) \subset H$, ce qui donne $H' \subset H$. Il est clair que $\partial H \cap \partial H' \neq \emptyset$. Comme

$$\partial H' \subset \partial \text{cc}([u \leq \mu], \mathbf{y})$$

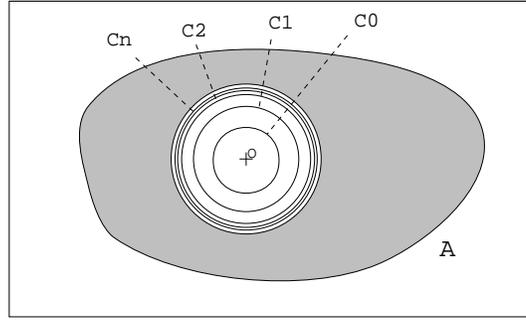


FIG. 3.5 – Pour une image semicontinue générale, il peut arriver qu'un trou dans une composante connexe d'ensemble de niveau ne soit pas une forme. La fonction caractéristique de l'ensemble $F = A \cup C_1 \cup \dots \cup C_n \cup \dots$ est semicontinue supérieurement. A est une composante connexe d'ensemble de niveau, ayant un disque comme trou, qui est approché par les C_n , cercles de rayon croissant. Pour tout point \mathbf{x} dans le trou et non dans F , la forme $\text{sat}(\text{cc}([u < 1], \mathbf{x}))$ est strictement incluse dans le trou. Néanmoins, le trou est l'union de toutes ces formes.

et $\partial \text{cc}([u \leq \mu], \mathbf{y}) \subset [u > \mu]$, si nous notons ν la valeur prise par u immédiatement supérieure, nous obtenons $\partial H' \subset [u \geq \nu]$. Soit D la composante connexe de $[u \geq \nu]$ contenant $\partial H'$. D et F sont disjoints, supposons que D contient un pixel ouvert de H . D est composé de la fermeture des pixels ouverts qu'il contient (Proposition 3.2), donc sa frontière ne peut rencontrer ∂H , car tous les pixels ouverts de H dont la fermeture rencontre ∂H sont dans F , donc ont une valeur strictement plus petite que ν . Or nous avons évidemment $\partial H' \subset \partial D$. Cela contredit le fait que $\partial H'$ rencontre ∂D . Donc D ne peut rencontrer H , et par conséquent $\partial H' = \partial H$, ainsi H' est un sous-ensemble de H sans frontière dans H . Par connexité de H , nous obtenons $H = H'$.

3. Si C est de type inférieur, H est fermé et donc $\overline{F} \subset H$. Comme $\partial H \subset \overline{F}$ et ∂H est connexe, nous déduisons que \overline{F} est connexe. Alors si μ est le minimum de u_c sur F , par semicontinuité supérieure de u_c , nous déduisons que \overline{F} est dans une composante connexe de $[u \geq \mu]$, qui ne rencontre pas C . Alors

$$H = \text{sat}(H) = \text{sat}(\partial H) \subset \text{sat}(\overline{F}) \subset \text{sat}(\text{cc}([u \geq \mu], \mathbf{y})).$$

Cela donne facilement $H = \text{sat}(\text{cc}([u \geq \mu], \mathbf{y}))$. □

Remarquons que cette propriété est spécifique aux image numériques, elle n'est pas valide pour une image semicontinue générale, voir la Figure 3.5.

3.3 La FLST

La FLST, pour `Fast Level Set Transform`, est notre algorithme pour extraire les formes d'une image numérique et les représenter dans une structure d'arbre.

3.3.1 Entrée et sortie de la FLST

En entrée, la FLST prend simplement une image numérique. En sortie, elle donne l'arbre des formes. Détaillons la structure de données utilisée.

Les valeurs de l'image sont données dans un tableau de taille $W \times H$. Il est commode de les énumérer ligne par ligne, chaque ligne de gauche à droite (comme dans la lecture d'un texte), nous notons les valeurs successives $u_1, u_2 \cdots u_N$ avec $N = W.H$.

Une structure de donnée pour une forme numérique facile à manipuler est :

```

{Structure d'une forme numérique (nœud de l'arbre d'in-
clusion)}
Parent, Enfant, FrèreSuivant {Pointeurs sur d'autres nœuds}
Type {Forme inférieure ou supérieure}
Gris {Le niveau de gris associé}
Données {Caractéristique de la forme}

```

La structure locale de l'arbre est codée dans la forme par les champs `Parent`, `Enfant` and `FrèreSuivant`. la racine de l'arbre, i.e., la forme Ω_d a un parent nul, les feuilles de l'arbre, i.e., les maxima régionaux de l'image sans trous, ont un enfant nul. Les frères (c'est-à-dire des formes différentes ayant le même `Parent`) sont codées dans une liste, le lien étant donné par `FrèreSuivant`. De cette manière, la procédure pour énumérer la liste des enfants d'une forme est :

```

Require:  $P$  {La forme parent}
 $C$  {Devient successivement chaque enfant
de  $P$ }
 $C \leftarrow \text{Enfant}(P)$ 
while  $C \neq \emptyset$  do
  Action sur  $C$  { $C$  est un enfant de  $P$ }
   $C \leftarrow \text{ProchainFrère}(C)$ 
end while

```

Le champ `Gris` de la forme est le niveau de gris associé à la forme, comme

expliqué dans la Section 3.2.4. Notons que le champ `Type` n'a pas de signification pour la racine.

L'arbre d'inclusion des formes a la structure de donnée suivante :

```

{Structure de l'arbre d'inclusion}
  W, H {Dimensions de l'image}
  Racine {La racine de l'arbre}
  PlusPetitesFormes = S1...SN {Tableau des plus
    petites formes}

```

Le tableau `PlusPetitesFormes` donne la plus petite forme contenant chaque pixel. Pour le pixel d'index i , S_i pointe sur la plus petite forme le contenant.

3.3.2 Description de l'algorithme

Reconstruction

L'algorithme de reconstruction est direct et de complexité linéaire, voir Algorithme 1. Chaque pixel prend le niveau de gris de la plus petite forme associée. Notons que le `Type` de la forme n'est pas utilisé.

Algorithm 1 Reconstruction de l'image à partir de son arbre d'inclusion

Require: T {L'arbre d'inclusion}

$u \leftarrow$ Tableau de taille $N = W.H$ {Création du tableau}

for $i = 1$ à N **do**

$u_i \leftarrow \text{Gris}(S_i)$

end for

Bases de l'extraction

L'algorithme d'extraction est bien plus complexe. Durant l'exécution de l'algorithme, l'image est modifiée. A la fin de l'algorithme, l'image est uniforme au niveau de gris de la racine.

Nous utilisons aussi une image de marqueurs, `Marque(P)` pour un pixel P . A l'origine, `Marque(P)` est faux pour tout pixel P ; puis, une fois qu'un pixel est examiné, il est marqué vrai.

Nous utilisons une structure de queue, stockant un tableau de pixels, tel que nous pouvons en *temps constant* :

- ajouter une nouveau pixel ;
- retourner et enlever tous les pixels à un niveau de gris donné ;
- Retourner les valeurs minimum et maximum de ses pixels.

L'idée est d'avoir pour chaque niveau de gris possible g un tableau contenant les pixels au niveau de gris g . Cela est raisonnable pourvu que le nombre de niveaux de gris soit petit. C'est le cas pour une image 8-bits, le nombre de niveaux de gris possibles étant 256. Toutefois, pour une image où chaque pixel est une valeur réelle, il peut y avoir autant de niveaux de gris que de pixels, donc une telle structure de donnée ne serait pas adaptée. Nous restreindrons notre discussion aux images 8-bits³.

Les pixels que nous stockerons dans cette queue sont les voisins de la région actuelle, donc elle ne contiendra pas plus de pixels que l'image. Au lieu d'allouer 256 tableaux de N pixels, ce qui serait consommateur de mémoire, nous pouvons faire la même chose avec seulement N pixels et 256 indexs. Cette queue sera notée \mathcal{N} et sa structure est :

```

{Structure de la queue  $\mathcal{N}$ }
  Min, Max {Niveaux de gris minimum et maximum des
  pixels}
  Taille {Le nombre de pixels dans la queue}
  Pixels[ $N$ ] {Tableau de Taille pixels, ne dépassant
  pas  $N$ }
  Suivant[ $N$ ] {Tableau de Taille pointeurs sur des
  éléments de Pixels}
  Premier[256] {Tableau de pointeurs sur des éléments
  dans Pixels}

```

Pour chaque élément dans le tableau `Pixels`, l'élément au même index dans `Suivant` pointe sur le pixel suivant au même niveau de gris. Le premier est donné

³En fait, l'algorithme peut s'adapter aux images avec des niveaux de gris codés sur n'importe quel nombre de bits. Pour cela, nous utiliserions une queue de priorité ; les voisins de la région courante sont triés dans un arbre binaire équilibré avec la propriété suivante : tout nœud a un niveau de gris au moins égal à ses deux enfants. Donc le voisin de plus grand niveau de gris est toujours à la racine de l'arbre. Lorsque nous cherchons une composante connexe de type supérieur, c'est ce dont nous avons besoin, pour le type inférieur, nous appliquons les changements appropriés. La taille maximum de cet arbre en mémoire est la taille de l'image, l'insertion d'un nouveau voisin prend $O(d)$ où d est la profondeur de l'arbre, de même que la suppression d'un voisin. C'est la structure utilisée dans l'algorithme `jheapsort` [1]. En pratique, la performance baisse d'environ 30% pour des réels à virgule flottante, mais il apparaît que c'est principalement dû au fait que des types de valeurs différents sont comparés : des entiers sur un octet dans le premier cas, et des réels à virgule flottante dans le second cas.

par le tableau `First`. De cette manière, l'ensemble des pixels ayant un niveau de gris donné s'extrait en prenant `First` et en suivant les liens. Ajouter un pixel au niveau g dans \mathcal{N} est facile. Nous notons cette opération $\mathcal{N} \leftarrow \mathcal{N} \cup (P, g)$:

{Ajout d'un pixel dans \mathcal{N} }

Require: Pixel P de niveau g

`Taille` ← `Taille`+1

`Pixels`_{`Taille`} ← P {Ajout du pixel P à un emplacement libre}

`Suivant`_{`Taille`} ← `Premier` _{g}

`Premier` _{g} ← `Pixels`_{`Taille`}

`Min` ← $\min(g, \text{Min})$

`Max` ← $\max(g, \text{Max})$

Enlever les pixels de \mathcal{N} au niveau g consiste en leur extraction, comme expliqué plus haut, puis en la réinitialisation de `Premier` _{g} à \emptyset et :

- Si $g = \text{Min}$, incrémenter `Min` jusqu'à ce que `Premier`_{`Min`} $\neq \emptyset$.
- Si $g = \text{Max}$, décrémenter `Max` jusqu'à ce que `Premier`_{`Max`} $\neq \emptyset$.

Nous notons cette procédure $\mathcal{N} \leftarrow \mathcal{N} \setminus [\mathcal{N} = g]$, où $[\mathcal{N} = g]$ représente les pixels de \mathcal{N} de niveau de gris g .

Comme les pixels du même niveau de gris n'ont pas de raison particulière d'être à la fin du tableau `Pixels`, les extraire laisse des trous dans les tableaux. Au lieu de déplacer les éléments de `Pixels` et de mettre à jour les pointeurs `Suivant` correspondants, une opération fastidieuse, nous pouvons garder en mémoire un tableau d'emplacements libres, et quand un nouveau pixel est ajouté, il occupe la première place libre s'il y en a une, sinon il est ajouté à la fin du tableau `Pixels`.

Boucle principale

D'abord, des initialisations de variables ont lieu : `Marque`(u_i) est initialisée à faux pour tout i , \mathcal{N} est vide, et l'arbre T contient seulement la racine.

La boucle principale de l'algorithme est simple : c'est juste un parcours de l'image, appelant une procédure d'extraction lorsqu'un extremum local non marqué de l'image est trouvé. La procédure `ExtractionBranche` est expliquée plus loin.

La définition d'extremum local demande quelque explication. La localité est représentée par les voisins. Le pixel P est dit un extremum local de u si et seulement si :

1. $\forall Q$ 4-voisin de P , $u(P) \leq u(Q)$ (minimum local) ou

2. $\forall Q$ 8-voisin de P , $u(P) \geq u(Q)$ (maximum local) ;
3. $\exists Q$ k -voisin de P tel que $u(P) \neq u(Q)$.

Le k dans le troisième point est 4 dans le cas 1 et 8 dans le cas 2. Bien sûr, ces deux cas s'excluent mutuellement.

Algorithm 2 La boucle principale de la FLST.

```

for  $i = 1$  à  $N$  do
  if (non Marque( $u_i$ )) et ( $u_i$  est un extremum local) then
    Appeler ExtractionBranche( $u_i$ )
  end if
end for

```

Extraction de branche

Nous décrivons la procédure `ExtractionBranche`, qui extrait la branche contenant un pixel donné.

Définition 3.8 Pour un pixel P , nous appelons *branche en P* l'ensemble des formes S contenant P et telles que pour toute forme S' , $S' \subset S \Rightarrow P \in S'$.

Notons que la branche contenant P peut être vide.

Cette procédure extrait la branche en P seulement si P appartient à un extremum régional (ce n'est pas une limitation, comme la Section 3.3.3 le montrera).

Définition 3.9 Un sous-ensemble R de Ω_d est dit un *minimum* (resp. *maximum*) régional si c'est un sous-ensemble maximal pour la relation \mathcal{J} être connexe et contenir seulement des minima locaux larges \mathcal{J} (resp. maxima locaux larges).

Remarque 3.2. Les notions de maximum et minimum sont prises ici au sens large : si tous les pixels voisins sont au même niveau, nous disons aussi que c'est un extremum.

Remarque 3.3. Bien sûr, cette définition donnée pour les pixels -où le terme \mathcal{J} connexe \mathcal{J} signifie, comme d'habitude dans cette thèse, 4-connexe pour un minimum et 8-connexe pour un maximum- reste identique pour les images définies sur un domaine continu, avec la définition topologique de la connexité.

Comme nous pouvions le prévoir, les extrema régionaux sont disjoints.

Lemme 3.10 Les extrema régionaux d'une image numérique sont disjoints.

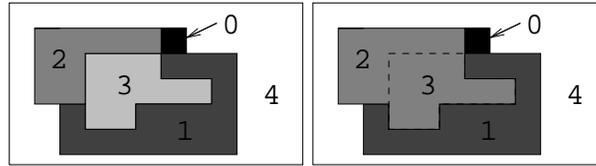


FIG. 3.6 – Suppression d’une forme. Gauche : image initiale. Droite : image lorsque la forme S au niveau 3 est supprimée. S est une composante connexe d’ensemble de niveau supérieur dans l’image initiale, ses pixels prennent le niveau de gris maximum en ses voisins, 2.

Preuve.

1. Les sous-ensembles \mathbb{I} connexes et ne contenant que des minima locaux i_i (resp. maxima locaux) sont fermés par union quand ils ne sont pas disjoints. En effet, si deux sous-ensembles contiennent seulement des minima (resp. maxima) locaux et se rencontrent, leur réunion reste connexe, et ne contient également que des minima (resp. maxima) locaux.

2. Si un pixel P est commun à un maximum régional M_1 et un minimum régional M_2 , alors par maximalité tous les pixels hors de M_1 et voisins de M_1 sont à un niveau de gris strictement plus petit, donc les pixels de la frontière numérique de M_1 (i.e., les pixels de M_1 ayant un voisin hors de M_1) ne peuvent être des minima locaux, et M_2 étant connexe, cela signifie que $M_2 \subset M_1$. Pour la même raison, $M_1 \subset M_2$ et cela donne $M_1 = M_2$. Finalement, cela entraîne que M_1 n’a pas de voisin, donc que l’image est constante. \square

Remarque 3.4. Un minimum (resp. maximum) régional est donc une composante connexe d’ensemble isoniveau pour laquelle tous les voisins ont un niveau de gris plus grand (resp. plus petit).

L’idée de `ExtractionBranche` est de supprimer une forme dès qu’elle est détectée et stockée dans l’arbre. Ce que nous voulons dire par supprimer une forme est de mettre tous ses pixels au niveau de gris le plus petit des voisins, dans le cas où c’est une forme inférieure, et au niveau de gris le plus grand des voisins, si c’est une forme supérieure (voir la Figure 3.6).

Comme une forme contenant P contient sa composante connexe d’ensemble isoniveau, nous avons d’abord besoins d’une procédure pour trouver cet ensemble isoniveau. La stratégie est la croissance de région, en utilisant la queue \mathcal{N} . Cette procédure, `ExtractionExtremumRégional`, est décrite dans l’Algorithme 3. Notons que cet algorithme extrait aussi bien un minimum régional qu’un maximum régional.

Puisque nous ne pouvons pas savoir à l'avance si P appartient à un minimum régional ou à un maximum régional (ou aucun des deux!), nous avons le problème du choix de la connexité. La solution est de choisir la notion la plus restrictive, à savoir la 4-connexité, et de passer en 8-connexité lorsque nous rencontrons un voisin de niveau de gris strictement plus petit, puisque dans ce cas nous nous attendons à un maximum régional. Ceci ne perd pas des voisins au cas où P appartient en fait à un maximum régional, car les voisins diagonaux d'un pixel Q sont des 4-voisins des 4-voisins de Q .

Remarque 3.5. Lorsque nous ajoutons un voisin à \mathcal{N} , nous devons vérifier s'il n'a pas été déjà ajouté (car il peut être le voisin de plusieurs pixels dans R). Un moyen facile de le vérifier est d'avoir une image de marqueurs, `ImageVoisins`, où chaque marqueur indique si le pixel a été ajouté comme voisin de la région actuelle ou non⁴. `ImageVoisins` est aussi utilisé pour compter le nombre de trous de l'ensemble actuel, comme nous l'expliquerons plus loin.

La procédure `ExtractionBranche` (voir Algorithme 4) extrait d'abord un extremum régional, l'enregistre en tant que forme s'il n'a pas de trou, le supprimer, et à nouveau extrait l'extremum régional, etc. Il y a trois conditions de rupture :

1. P n'appartient pas à un extremum régional ;
2. P appartient à un extremum régional avec trou ;
3. L'extremum régional contenant P rencontre le cadre de l'image et son aire est au moins la moitié de celle de l'image.

Dans les deux premiers cas, il y a une autre forme, ne contenant pas P , mais incluse dans la plus petite forme contenant P (un extremum régional sans trou dans le premier cas, le trou ou un sous-ensemble de ce trou dans le second cas). Dans le troisième cas, la plus petite forme contenant P est la racine de l'arbre, de telle sorte que le niveau associé à la racine est $u(P)$ (voir la Figure 3.7).

Notons qu'au lieu de réellement supprimer l'extremum régional actuel lorsqu'il est détecté et vu comme une forme, c'est-à-dire de mettre tous ses pixels au niveau de gris des voisins le plus proche, puisque par la suite nous continuons la croissance de région, il suffit de faire *comme si* la forme avait été supprimée et considérer que les pixels de la forme sont au niveau de gris actuel. De la même manière, dans l'image `ImageVoisins`, les marqueurs des pixels de la forme trouvée sont déjà à jour, donc il

⁴En fait, pour éviter de réinitialiser cette image à chaque appel à `ExtractionExtremumRégional`, nous utilisons plutôt un index `Index` incrémenté à chaque appel, et pour `ImageVoisins`, une paire (`Indexi`, `Marquei`). De cette manière, le test pour savoir si un pixel a déjà été considéré comme voisin est `!!Indexi = Index` et `Marquei !!` et l'instruction pour enregistrer le fait qu'un nouveau voisin est ajouté est `!!Indexi ← Index` et `Marquei ← TRUE !!`.

Algorithm 3 *ExtractionExtremumRégional* : extraction de la composante connexe d'ensemble isoniveau contenant P , si cet ensemble est un extremum régional.

Require: P, g {Pixel P au niveau g }

$R \leftarrow \emptyset$ {La composante connexe d'ensemble isoniveau}

$\mathcal{N} \leftarrow \{P\}$ {La queue des voisins}

while $[\mathcal{N} = g] \neq \emptyset$ **do**

$\mathcal{A} \leftarrow [\mathcal{N} = g]$ {Tableaux des pixels actuels à ajouter}

$\mathcal{N} \leftarrow \mathcal{N} \setminus [\mathcal{N} = g]$ {Les enlever de \mathcal{N} }

for $Q \in \mathcal{A}$ **do** {tous les voisins au niveau g }

 Marque(Q) \leftarrow VRAI {Marque ces pixels}

$R \rightarrow R \cup \{Q\}$ {Ajout du pixel Q dans R }

for all Q' 4-voisin de Q **do**

$\mathcal{N} \leftarrow \mathcal{N} \cup \{Q'\}$

end for

if $\text{Min} < g$ **then** {Nous nous attendons à un maximum régional}

for all Q' voisin diagonal de Q **do**

$\mathcal{N} \leftarrow \mathcal{N} \cup \{Q'\}$

end for

end if

if $\text{Min} < g < \text{Max}$ **then**

 Sortir et retourner \emptyset { P n'appartient pas à un extremum régional}

end if

end for

 Retourner R

end while

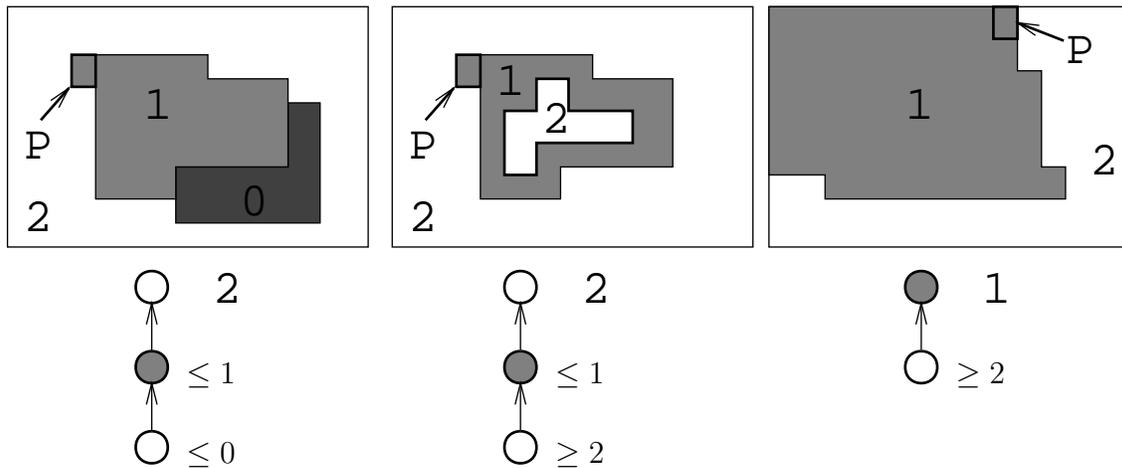


FIG. 3.7 – Conditions de rupture dans la procédure `ExtractionBranche`. Chaque colonne montre une configuration et son arbre associé, avec la plus petite forme contenant le pixel P en grisé. Dans chaque configuration, P est détecté comme minimum local. Dans le premier cas (gauche), P n'appartient pas à un minimum régional. Dans le second cas (milieu), la plus petite forme contenant P est un minimum régional, mais elle a un trou. Dans le troisième cas (droite), la plus petite forme contenant P est un minimum régional, mais rencontre le cadre et a une aire au moins moitié de celle de l'image, donc c'est la racine.

n'y a rien de spécial à faire dans cette image pour continuer la croissance de région⁵. Le niveau de gris $u(R)$ indiqué à la fin de l'Algorithme 4 signifie en fait le niveau de gris courant de la croissance de région.

Algorithm 4 La procédure `ExtractionBranche`, extrayant la branche contenant un pixel P

Require: P {Le pixel}

$\text{Fin} \leftarrow \text{FAUX}$ {Témoin de fin de la procédure}

while non Fin **do**

$R \leftarrow \text{ExtractionExtremumRégional}(P)$

if $R = \emptyset$ ou R a un trou **then**

$\text{Fin} \leftarrow \text{VRAI}$

else if $R \cap \text{cadre} \neq \emptyset$ et $\#R \geq N/2$ **then**

$\text{Gris}(\text{Racine}) \leftarrow u(R)$

else

 Stocke R comme nouvelle forme dans l'arbre

end if

end while

for all $Q \in R$ **do** {Supprime la branche dans u }

$u(Q) \leftarrow u(R)$ { $u(R)$ est le niveau de gris courant}

end for

Décompte des trous

Nous n'avons pas encore expliqué comment compter le nombre de trous dans un extremum régional. Le fait que nous utilisons la 8-connexité (resp. la 4-connexité) pour cet ensemble et la 4-connexité (resp. la 8-connexité) pour son complémentaire, permet de calculer la caractéristique d'Euler à partir de configurations *locales* de l'ensemble. Cette propriété est prouvée par Rosenfeld et Kak dans [70]. Remarquons que cette propriété se perd quand la même notion de connexité est utilisée pour l'ensemble et son complémentaire (voir Kong et Rosenfeld [32]), une autre sorte de

⁵Cela signifie que l'`Index` utilisé pour éviter la réinitialisation de `ImageVoisins` doit être incrémenté à chaque appel de la procédure `ExtractionBranche`, pas à l'appel de `ExtractionExtremumRégional`.

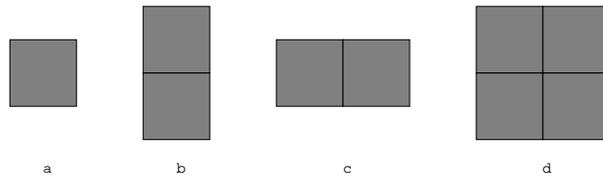


FIG. 3.8 – La caractéristique d’Euler d’un ensemble S de pixels en 4-connexité et le complémentaire en 8-connexité peut se calculer localement en comptant le nombre de configurations de blocs de taille au plus 2×2 . Elle est égale au nombre de positions possibles de dominos de forme a et d contenus dans S moins le nombre de positions possibles de dominos de forme b et c . Le nombre de positions de dominos de type a est $\#S$, le cardinal de S .

manque de cohérence de la topologie discrète par rapport à la topologie du domaine continu, lorsque seule une notion de connexité discrète est utilisée.

Rappelons que la caractéristique d’Euler d’un ensemble S est $n - m + 1$ où n est le nombre de composantes connexes de S et m le nombre de composantes connexes de son complémentaire. Puisque nous traitons d’ensemble connexes, $n = 1$, de telle sorte que la caractéristique d’Euler est $1 - h$ où $h = m - 1$ est le nombre de trous de S . Les configurations locales en question sont des blocs d’au plus 2×2 pixels (voir la Figure 3.8). Puisque notre algorithme repose sur la croissance de région, il est préférable de compter la variation de la caractéristique d’Euler d’un ensemble lorsque nous lui ajoutons un pixel. Cela veut dire que nous devons examiner le voisinage 3×3 centré au nouveau pixel pour calculer cette variation, puisque les seuls blocs 2×2 qui changent sont inclus dans ce voisinage 3×3 (voir les Figures 3.9 et 3.10).

Le dernier problème est lié au cas où nous ne savons pas en avance la connexité à choisir, car tous les voisins de l’ensemble courant sont au même niveau de gris. Comme nous l’avons expliqué plus haut, nous utilisons dans ce cas la 4-connexité pour l’ensemble (et donc la 8-connexité pour le complémentaire). Mais si nous trouvons un voisin de niveau de gris strictement plus petit, nous passons en 8-connexité. Ceci peut induire des erreurs dans le calcul de la caractéristique d’Euler, voir la Figure 3.11. La correction en est aisée. Lorsque nous passons en 8-connexité quand nous considérons un ensemble 4-connexes, nous n’ajoutons pas de trous, de telle sorte que l’ensemble, qui n’avait pas de trou avant (sinon, nous aurions rencontré un voisin de niveau de gris différent avant), reste sans trou. Notons que ce cas ambigu, un détail délicat de l’algorithme, bien que facile à régler, ne peut arriver pour la croissance de région initiale dans la procédure `ExtractionBranche`, puisque le pixel P doit avoir au moins un voisin de niveau de gris différent.

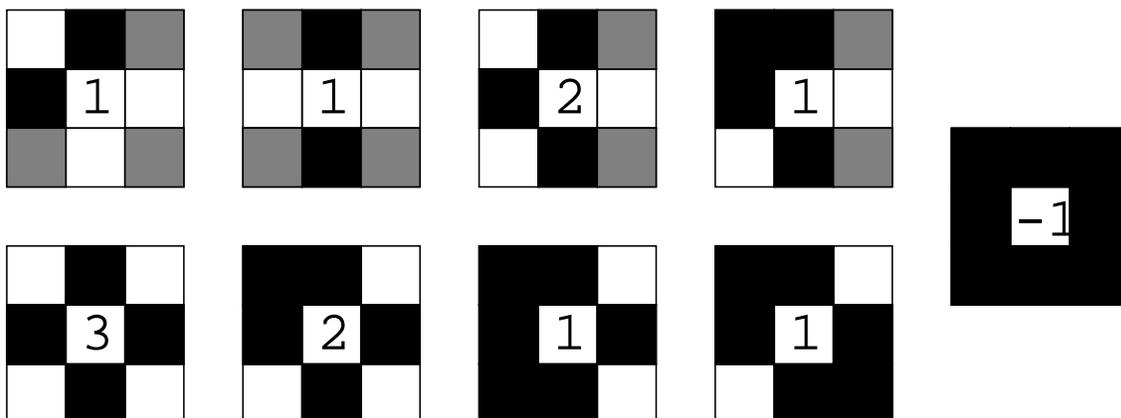


FIG. 3.9 – Variation du nombre de trous d'une forme quand nous ajoutons un 4-voisin P à un ensemble 4-connexe S , suivant les configurations locales, c'est-à-dire les voisinages 3×3 centrés en P . Les pixels de S sont représentés en noir, les pixels hors de S laissés en blanc et les pixels étant indifféremment dans S ou en dehors en gris. A ces configurations, nous devons ajouter toutes les configurations obtenues par des rotations d'un quart de tour ou d'un demi-tour, ainsi que les configurations symétriques horizontalement ou verticalement. Les configurations ne pouvant être obtenues par de telles opérations soit ne modifient pas le nombre de trous, soit ne sont pas possibles (P ne serait pas un 4-voisin de S).

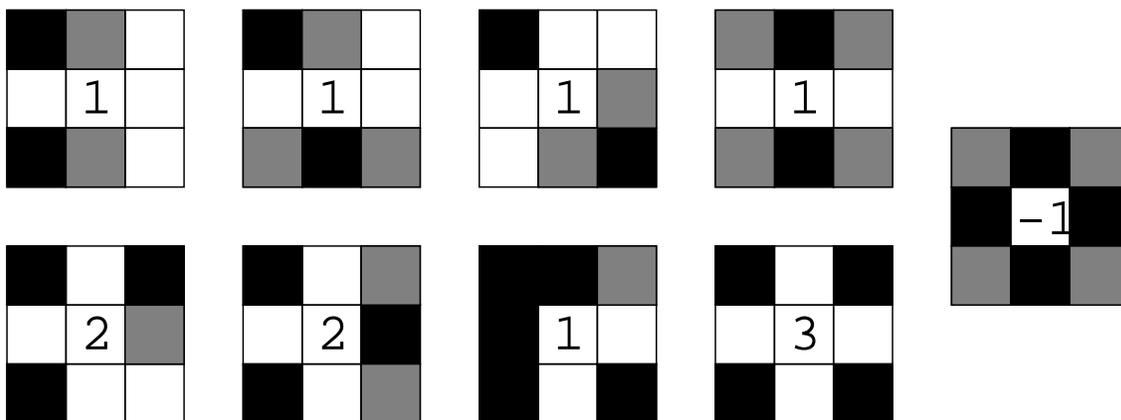


FIG. 3.10 – Variation du nombre de trous d'une forme quand nous ajoutons un 8-voisin P à un ensemble 8-connexe S , suivant les configurations locales autour de P . Les versions obtenues par rotations et symétries de ces configurations ajoutent ou soustraient le même nombre de trous. Voir la Figure 3.9 pour des explications sur les configurations.

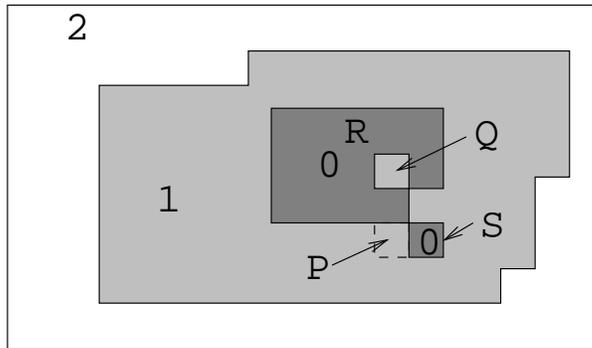


FIG. 3.11 – Erreur possible dans la caractéristique d’Euler lorsque tous les voisins de l’ensemble courant sont au même niveau de gris. La région R est une composante connexe d’ensemble de niveau inférieur, au niveau 0, donc considérée en 4-connexité, sans trou. Lorsque la forme est supprimée, tous les voisins sont au même niveau de gris, 1. Si le voisin P est ajouté en premier à la région courante, il a comme voisin S , à un niveau de gris inférieur, 0. Donc nous passons en 8-connexité. Mais cela modifie la caractéristique d’Euler de la région, qui devient 2, car le pixel Q est maintenant un trou dans la région courante. Lorsque Q est ajouté à la région, ceci décrémente la caractéristique d’Euler, qui doit être corrigée à 1 après que Q a été ajouté.

Enregistrement des formes

Le dernier point à expliquer est comment nous enregistrons les formes quand elles sont détectées. C’est-à-dire que nous avons besoin de trouver leur emplacement dans l’arbre T . C’est ce que nous faisons à l’aide du tableau `PlusPetitesFormes`; nous expliquons aussi comment remplir cet arbre. En vue de ne pas pénaliser l’algorithme, nous utilisons également un tableau `PlusGrandesFormes`, qui devient inutile à la fin de la FLST. Ce tableau, dont nous notons les éléments L_1, \dots, L_N , regroupe des pointeurs sur la plus grande forme extraite (sauf Ω_d) contenant chaque pixel dans l’arbre. Bien sûr, ce tableau se déduit facilement de `PlusPetitesFormes` et de l’arbre : il suffit de partir de S_i et de remonter dans l’arbre jusqu’à atteindre un nœud dont le parent est la racine ; ce nœud est L_i . Toutefois, garder ce tableau en mémoire sauve du temps de calcul.

Initialement, nous mettons chaque S_i à \emptyset et chaque L_i à **Racine**. Lorsqu’une nouvelle forme S est détectée, nous avons son ensemble de pixels R . Pour chaque pixel u_i de R dont la forme S_i associée est \emptyset , nous positionnons S_i à S . Pour chaque u_i de R dont la forme L_i associée n’est pas **Racine**, nous rendons la forme pointée par L_i enfant de S , le sous-arbre de racine L_i restant inchangé. Finalement, pour chaque pixel $u_i \in R$, nous positionnons la forme L_i correspondante à S .

3.3.3 Analyse et preuve de la FLST

Dans cette section, nous justifions l'algorithme et expliquons pourquoi il est correct. Cela ressort de quelques résultats que nous allons prouver. Certains d'entre eux sont valables pour une image définie sur un domaine continu, mais nous les prouverons dans le cas des images numériques, car nous ne les utilisons que pour justifier l'algorithme.

Rôle des extrema locaux

Lemme 3.11 *Une composante connexe d'ensemble de niveau inférieur (resp. supérieur) contient un minimum (resp. maximum) régional.*

Preuve. Considérons la valeur minimale prise par u sur la composante C et un point P où ce minimum est atteint. La composante connexe I d'ensemble isoniveau contenant P est alors un minimum régional ; si Q est un voisin de cette composante d'ensemble isoniveau, il y a deux possibilités :

1. $Q \in C$, dans ce cas $u(P) \leq u(Q)$ par définition de P et nous ne pouvons avoir égalité, sinon $I \cup \{Q\}$ serait connexe et dans l'ensemble isoniveau $u(P)$.

2. Q voisin d'un pixel P' de $C \cap I$, auquel cas $u(P') < u(Q)$, sinon $C \cup \{Q\}$ serait connexe et dans le même ensemble de niveau que C . Comme $u(P) = u(P')$, la preuve est faite. \square

Proposition 3.12 *Une forme contient un extremum régional sans trous, qui est une forme.*

Preuve.

1. Si la forme contient seulement un pixel, c'est une composante connexe d'ensemble de niveau sans trou, donc un extremum régional. Supposons que le résultat est prouvé pour toutes les formes d'aire $\leq n$ et prenons une forme S d'aire $n + 1$ basée sur une composante connexe d'ensemble de niveau C . Si C a au moins un trou H , H est une forme d'aire $\leq n$ et par hypothèse, cela implique le résultat. Si C n'a pas de trou, d'après le Lemme 3.11, C contient un extremum régional R . Alors $\text{sat}(R) \subset S$. Si R n'a pas de trou, R répond à la condition voulue, si R a au moins un trou, celui-ci est d'aire $\leq n$, et par hypothèse $\text{sat}(R)$, et donc aussi S , contient un extremum régional sans trou.

2. Tout extremum régional est une composante connexe d'ensemble de niveau, et s'il n'a pas de trou, c'est une forme. \square

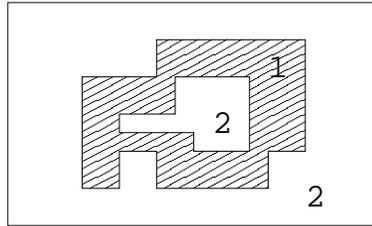


FIG. 3.12 – La forme inférieure associée à la composante connexe d'ensemble de niveau inférieur 1 contient un maximum régional sans trou, mais pas de minimum régional sans trou.

Remarque 3.6. La Proposition 3.12 ne précise pas le type d'extremum. En effet, il n'est pas vrai qu'une forme inférieure contienne un minimum régional sans trou, voir la Figure 3.12.

Remarque 3.7. Il est clair qu'un extremum régional sans trou est une forme terminale, c'est-à-dire une feuille de l'arbre d'inclusion. Inversement, si S est une forme terminale, la Proposition 3.12 montre qu'elle contient comme forme incluse un extremum régional sans trou, et puisque c'est une forme terminale, ces formes sont égales. Ceci caractérise les feuilles de l'arbre d'inclusion ; ce sont les extrema régionaux sans trou de l'image.

Suppression de branches

Nous montrons d'abord que supprimer des branches d'une image ne crée pas de nouvelles formes, bien que cela puisse créer de nouvelles branches. Nous disons qu'une branche est supprimée de l'image lorsque sa plus grande forme est supprimée. Nous rappelons qu'une forme inférieure (resp. supérieure) est supprimée si tous ces pixels ont été mis au niveau de gris le plus petit (resp. grand) de ses voisins.

La Proposition 3.13 explique l'effet de la suppression d'une forme terminale à l'arbre d'inclusion.

Proposition 3.13 *Soit u une image et S une forme terminale de u , c'est-à-dire une forme ne contenant pas d'autre forme. Soit \tilde{u} l'image u une fois la forme S supprimée. Alors l'arbre d'inclusion de \tilde{u} est l'arbre d'inclusion de u dans lequel le nœud correspondant à S est enlevé, tout comme l'arête entre S et son parent.*

Preuve.

1. Comme nous l'avons remarqué, comme S est une forme terminale de u , c'est

un extremum régional de u , donc une composante connexe d'ensemble isoniveau dans u . Soit g son niveau.

2. S est strictement incluse dans une composante connexe d'ensemble isoniveau de \tilde{u} . Le fait que S soit dans un ensemble isoniveau de \tilde{u} vient de la définition de la suppression de S . Si S est 4-connexe dans u , S est connexe dans u et \tilde{u} , et il existe un 4-voisin Q de S au même niveau dans \tilde{u} , de telle sorte que $S \cup \{Q\}$ est connexe et dans un ensemble isoniveau. Si S n'est pas 4-connexe, S est une composante connexe d'ensemble de niveau supérieur dans u , soit Q un 8-voisin de S ayant la même valeur dans \tilde{u} . Les 4-voisins de S ont la même valeur dans u et \tilde{u} , donc ils ont un niveau plus petit ou égal que Q et S . Donc les pointels de jonction de S et entre Q et S sont au même niveau, de telle sorte que $S \cup \{Q\}$ est connexe.

3. Nous affirmons que les formes de \tilde{u} sont des formes de u . Effectivement, soit C une composante connexe d'ensemble de niveau de \tilde{u} . Comme S est une partie connexe d'un ensemble isoniveau dans \tilde{u} , soit C contient S , soit $C \cap S = \emptyset$. Examinons les deux cas.

4. Si $C \cap S = \emptyset$, les pixels de C ont la même valeur dans u et \tilde{u} . Si Q' est un voisin de C au point Q , l'ordre des niveaux de gris en ces points est le même dans u et \tilde{u} ; si $Q' \notin S$, c'est évident, si $Q' \in S$, deux possibilités : si Q est un voisin de S dans u , c'est également clair, sinon Q et Q' sont des voisins diagonaux, S est de type inférieur dans u (donc $u \leq \tilde{u}$), alors que C est de type supérieur dans \tilde{u} , et si R est un 4-voisin commun à Q et Q' , nous avons

$$\tilde{u}(Q) = u(Q) \geq u(R) \geq \tilde{u}(Q') > u(Q')$$

de telle sorte que l'ordre est le même. Cela montre que C est aussi une composante connexe d'ensemble de niveau dans u ⁶.

5. Supposons maintenant que $S \subset C$. Si S est du même type dans u que C dans \tilde{u} , par exemple de type supérieur, nous écrivons $C \in [\tilde{u} \geq \lambda]$, C est 8-connexe, et comme $u|_S > \tilde{u}|_S$ et $u|_{\Omega_d \setminus S} = \tilde{u}|_{\Omega_d \setminus S}$, nous avons $C \in [u \geq \lambda]$, et C étant 8-connexe, C est une composante connexe d'ensemble de niveau supérieur dans u . Pour des raisons analogues (avec la 4-connexité cette fois), si S dans u et C dans \tilde{u} sont des composantes connexes d'ensembles de niveau inférieurs, nous trouvons à nouveau que C est une composante connexe d'ensemble de niveau inférieur dans u .

6. Si S est de type supérieur dans u et C de type inférieur dans \tilde{u} , $C \in [\tilde{u} \leq \mu]$. Comme $u|_{\Omega_d \setminus S} = \tilde{u}|_{\Omega_d \setminus S}$, nous avons $C \setminus S \in [u \leq \mu]$. Soit $Q' \in \Omega_d \setminus C$ un 4-

⁶Ceci n'est pas suffisant pour conclure directement que les formes associées à C sont les mêmes dans u et \tilde{u} , car les composantes connexes du complémentaire de C dans u et \tilde{u} doivent également être comparées.

voisin de C en Q . Nous avons $u(Q') = \tilde{u}(Q') > \mu \geq \tilde{u}(Q)$ et $Q \notin S$, sinon nous aurions $u(Q) > u(Q')$ et par définition de la suppression de S , $\tilde{u}(Q) \geq u(Q')$. Donc, $\tilde{u}(Q) = u(Q)$ et nous déduisons $u(Q') > \mu \geq u(Q)$. Si $S \in [u \leq \mu]$, nous déduisons que $C \in [u \leq \mu]$, et puisque C est 4-connexe et que tout 4-voisin de C est dans $[u > \mu]$, il s'ensuit que C est une composante connexe de $[u \leq \mu]$. Le cas restant est $S \in [u > \mu]$. Alors, S est une composante connexe de $[u > \mu]$, sinon il y aurait un 8-voisin Q de S avec $u(Q) > \mu$ et il viendrait $S \in [\tilde{u} > \mu]$. Considérons $Q \in C \setminus S$, 4-voisin de S . La composante connexe D de $(C \setminus S)^b$ (le symbole b étant compris lorsque l'on considère $C \setminus S$ comme sous-ensemble de u) contenant Q a une frontière rencontrant ∂S^b . Comme D est ouvert, nous avons $\partial D \in [u > \mu]$. Or ∂S^b connexe et dans $[u > \mu]$, donc nous avons soit $\partial S^b \in \text{sat}(D)$ soit $\partial S^b \in \bar{\Omega} \setminus \text{sat}(D)$. Le deuxième cas donnerait $(\partial \text{sat}(D)) \cap (\partial S^b) \neq \emptyset$ et comme $\partial \text{sat}(D)$ est connexe, dans $[u > \mu]$ et rencontre S^b , qui est une composante connexe de $[u > \mu]$, nous aurions $\partial \text{sat}(D) \subset S^b$, et puisque $\bar{\Omega} \setminus S^b$ est connexe, il viendrait $D = \bar{\Omega} \setminus S^b$, et donc $\text{sat}(D) = \bar{\Omega}$, ce qui est exclu par hypothèse. Donc $\partial S^b \subset \text{sat}(D)$ and par conséquent $\text{sat}(\partial S^b) = S^b \subset \text{sat}(D)$. Ceci est valable pour toute composante connexe de $(C \setminus S)^b$ dont la frontière rencontre ∂S^b . S'il y en a plusieurs, prenons deux d'entre elles, D et D' . Leurs saturations se rencontrent, donc elles sont emboîtées, par exemple $\text{sat}(D) \subsetneq \text{sat}(D')$. Donc, $S^b \cap D' = \emptyset$, puisque $\text{sat}(D) \cap D' = \emptyset$. Donc D' est ouvert et fermé dans C^b , ce qui contredit la connexité de C^b . Nous en concluons que $(C \setminus S)^b$ est connexe, et que S^b en est un trou. En d'autres termes, $C \setminus S$ est une composante connexe d'ensemble de niveau inférieur et sa forme associée contient S

7. Si S est de type inférieur dans u et C de type supérieur dans \tilde{u} , une preuve similaire montre que soit C est une composante connexe d'ensemble de niveau supérieur dans u soit $C \setminus S$ l'est, et dans ce cas S est un trou de $C \setminus S$.

8. Soit S' une forme de \tilde{u} , saturation de la composante connexe d'ensemble de niveau C . Si $\#S' = 1$, alors $S \notin S'$ et $C = S'$ est une composante connexe d'ensemble de niveau de u selon ce qui précède. Donc S' est une forme de u . Supposons que nous ayons montré que toutes les formes de \tilde{u} d'aire au plus n sont des formes de u . Soit S' d'aire $n + 1$. Si $S \in C$, alors $C \setminus S$ est une composante connexe d'ensemble de niveau de u et S est un trou de $C \setminus S$, donc $S \subset \text{sat}(C \setminus S)$ (saturation dans u). Si $S \notin C$, C est une composante connexe d'ensemble de niveau de u . Si H est un trou de C , c'est une forme d'aire au plus n de \tilde{u} , donc par hypothèse c'est une forme de u . Cela montre en tout cas que C est une forme de u .

9. Réciproquement, nous montrons que toute forme S' de u autre que S est aussi une forme de \tilde{u} . Cela suit les mêmes lignes que ci-dessus.

10. Cela montre que les formes de u et \tilde{u} sont les mêmes, sauf que S a disparu de

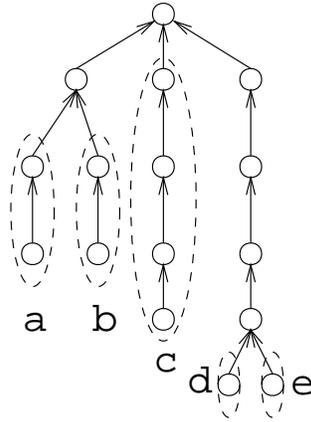


FIG. 3.13 – Un arbre et ses 5 branches non vides, de a à e .

l'ensemble des formes de \tilde{u} . Donc, les nœuds dans les arbres sont les mêmes, sauf les nœuds correspondant à S , et les arêtes de l'arbre représentant l'inclusion, celles-ci restent identiques, sauf que l'arête entre S et son parent disparaît dans l'arbre de \tilde{u} . \square

Cela donne une interprétation des branches en terme de l'arbre (voir la Figure 3.13).

Corollaire 3.14 *Soit u une image, P un pixel et S sa plus petite forme associée.*

- *Si S n'est pas terminale, la branche associée à P est \emptyset .*
- *Si S est terminale, la branche associée à P est l'union des ancêtres P de S tels que si P' est une forme, $P \supset P' \supset S \Rightarrow P'$ n'a qu'un enfant.*

Preuve. Si S n'est pas terminale, elle contient strictement un extremum régional sans trou, qui est une forme et ne contient pas P puisque S est la plus petite forme contenant P . Donc, la branche associée à P est vide.

Si S est terminale, soit \tilde{u} l'image u après suppression de S , et S' la plus petite forme associée à P dans \tilde{u} . S' , qui est aussi une forme de u , appartient à la branche associée à P si, et seulement si, elle est terminale, c'est-à-dire que son seul descendant dans l'arbre de u est S . De cette manière, nous pouvons monter dans l'arbre tant que les formes sont terminales. \square

Remarque 3.8. Cela donne une interprétation de la suppression d'une branche en termes de suppression de formes terminales : c'est simplement le résultat des

enlèvements successifs des formes de la branche en ordre croissant, chacune étant terminale quand elle est supprimée.

La proposition suivante montre que les pixels d'une branche supprimée n'ont pas de branche associée.

Proposition 3.15 *Soit \tilde{u} une image déduite de u par suppression de branches. Soit B une telle branche supprimée, de plus grande forme L . Alors si $P \in L$, la branche associée à P dans \tilde{u} est \emptyset .*

Preuve. \tilde{u} se déduit de u par des suppressions successives de branches B_0, \dots, B_k , résultant en les images $\tilde{u}_1, \dots, \tilde{u}_{k+1}$ où chaque B_i est une branche de \tilde{u}_i , et avec $\tilde{u}_0 = u$ et $\tilde{u}_{k+1} = \tilde{u}$. Supposons qu'il existe une forme terminale S dans \tilde{u} contenant P . Soit $0 \leq l \leq k$ l'index de la branche la plus récemment supprimée dont une forme contient P . Alors S est aussi une forme terminale de \tilde{u}_{l+1} , sinon tous les descendants de S auraient été supprimés à des étapes ultérieures $l' \geq l + 1$, et pour la dernière de celles-ci l' , S serait une forme dont le seul enfant est la plus grande forme de $B_{l'}$, impliquant que S serait incluse dans $B_{l'}$.

Pour la même raison, il y a une contradiction si l'on considère le plus grand $l'' \leq l$ pour lequel $B_{l''}$ contient un descendant de S . \square

Remarque 3.9. Cette proposition justifie le fait que les pixels qui sont marqués n'ont pas besoin d'être examinés à nouveau, même s'ils sont des extrema locaux. En effet, les pixels marqués correspondent à des pixels appartenant à une branche supprimée, ou bien des pixels connectés à une plus grande forme d'une branche supprimée dans un ensemble isoniveau, mais ayant un trou (donc n'étant pas associé à une forme terminale). Ceci explique que `ExtractionBranche` soit appelé de la boucle principale avec comme paramètre un extremum local *non marqué*.

Justification de l'algorithme

Les résultats précédents montrent que :

- chaque branche peut être supprimée en supprimant successivement des formes terminales ;
- il n'est pas nécessaire de parcourir plus d'une fois l'image, puisque durant un parcours, les branches associées à tous les pixels parcourus ont été extraites ;
- après un parcours complet de l'image, il ne reste plus de branche, donc toutes les formes de l'image ont été extraites.

Exemple

La Figure 3.14 présente un exemple d'exécution de l'algorithme. Chaque étape représente l'image, l'arbre d'inclusion (inconnu), et la partie de l'arbre calculée à ce stade. Les étapes sont les suivantes :

(a) L'image originale. L'arbre calculé est initialisé avec seulement la racine, dont le niveau de gris est inconnu.

(b) Le premier minimum local rencontré pendant le parcours est le pixel P . La procédure **ExtractionBranche** est appelée, trouve l'ensemble isoniveau 2, qui n'est pas un extremum régional, donc sortie. Le pixel Q est l'extremum local suivant, cela extrait l'ensemble isoniveau 5, qui est bien un extremum régional mais avec trou, donc le parcours continue. La même situation se passe pour R . S est l'extremum local suivant. La procédure **ExtractionBranche** trouve d'abord l'extremum régional au niveau 5 contenant S , n'ayant pas de trou, donc l'ajoute dans l'arbre calculé et le supprime de l'image.

(c) **ExtractionBranche**, appelée depuis le pixel S , poursuit la croissance de région et extrait l'extremum régional au niveau 3, sans trou, donc l'ajoute dans l'arbre calculé. Lorsqu'elle reprend la croissance de région, elle trouve l'extremum régional au niveau 2, avec un trou, donc sort.

(d) L'extremum local suivant rencontré pendant le parcours est T . **ExtractionBranche** trouve l'extremum régional au niveau 1, l'ajoute dans l'arbre, puis trouve le maximum régional au niveau 5, avec trou cependant, donc sort.

(e) U est l'extremum local suivant, mais sa composante d'ensemble isoniveau n'est pas une extremum régional. Puis V est rencontré, mais son extremum régional a un trou. Au contraire, en W un minimum régional sans trou au niveau 0 est extrait.

(f) La croissance de région commencée en W continue et **ExtractionBranche** trouve le maximum régional au niveau 4.

(g) **ExtractionBranche** continue la croissance de région et trouve un extremum régional rencontrant le cadre mais d'aire plus petite que la moitié de celle de l'image. Donc c'est une forme. L'isoniveau extrait ensuite a un trou, donc **ExtractionBranche** sort.

(h) L'extremum local non marqué suivant est X , **ExtractionBranche** trouve successivement les formes inférieures au niveau 1 et 3.

(i) Quand la croissance de région se poursuit, **ExtractionBranche** trouve l'image entière. Cela donne le niveau de gris de la racine, 5.

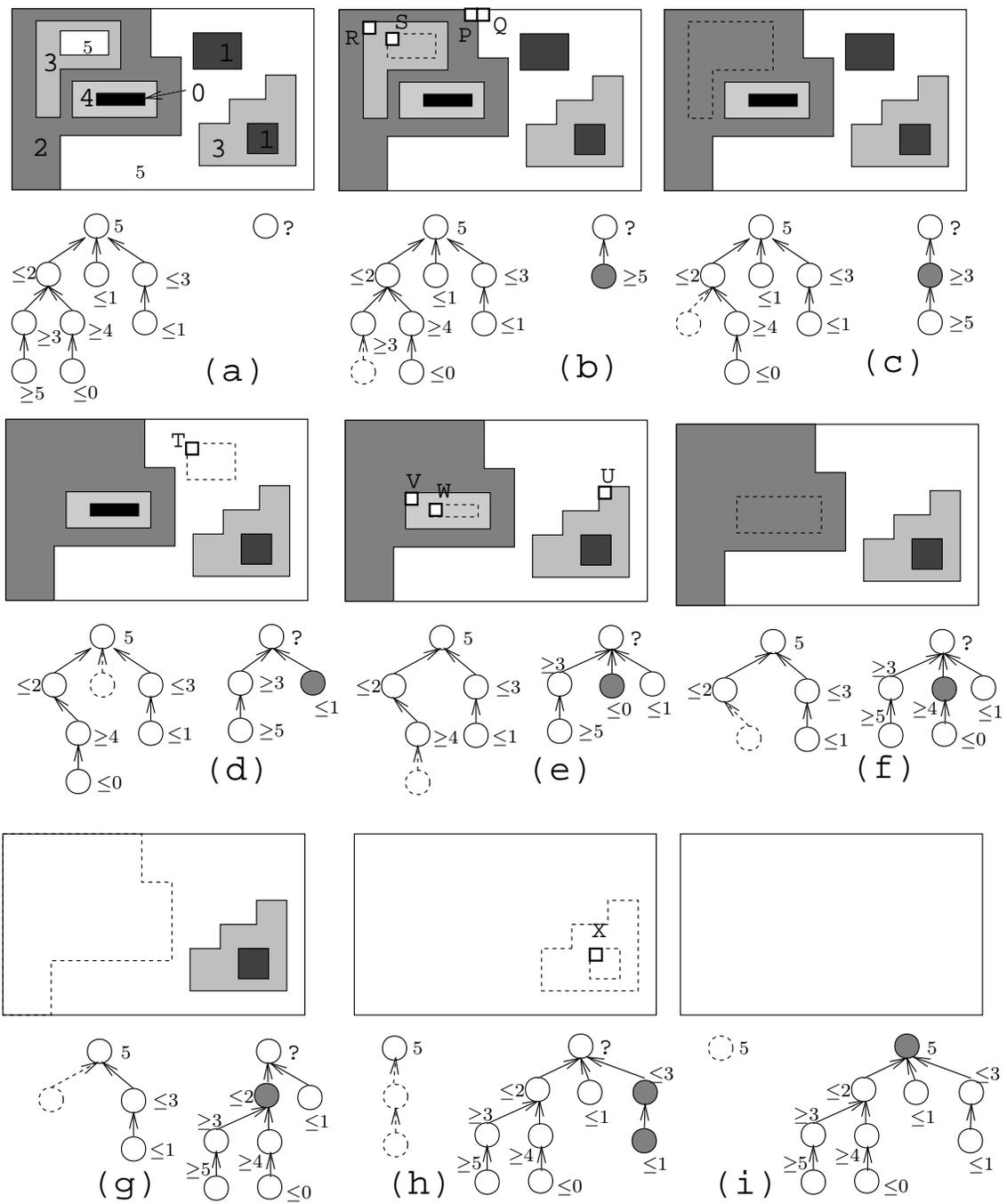


FIG. 3.14 – Un exemple d'exécution de la FLST. Voir le texte pour les détails.

Version *light* : certaines figures sont absentes

3.3.4 Complexité

Comparaisons

A la base, trouver les formes repose sur la comparaison de valeurs de pixels. C'est pourquoi l'opération de base de l'algorithme est la comparaison de deux pixels, donc notre mesure de complexité est le nombre de comparaisons de valeurs de pixels.

Il est difficile d'établir précisément la complexité de l'algorithme, comme nous ne pouvons pas savoir à l'avance combien de fois un pixel sera comparé à ses voisins⁷. La raison en est qu'il peut être le voisin de nombreuses formes.

Pour montrer comment la complexité dépend du contenu de l'image, et pas seulement de sa taille, nous exposons les résultats de l'expérience suivante : nous sommes parti d'une image de taille (6000×500) , apparaissant dans la Figure 3.15, et avons lancé la FLST sur des parties de cette image de largeur croissante, avec un pas de 100 pixels, en partant toujours du coin haut-gauche. La Figure 3.16 montre le nombre de comparaisons en fonction de la largeur de l'image, et ce nombre normalisé par le nombre de pixels. Notons la variation importante de ce nombre normalisé. Dans les 1000 premiers pixels de largeur, l'image est assez simple, avec de grandes plages de valeur uniforme, alors qu'elle devient bien plus complexe, avec des textures, par la suite.

A titre de comparaison, la FLST de l'image Lenna (taille 256×256) exige 770 888 comparaisons de pixels, ce qui revient à 11,7 par pixel. En ce qui concerne l'image hautement texturée de tapis montrée dans la Figure 3.17, de taille 1024×715 , la FLST demande 11 460 396 comparaisons de pixels, c'est-à-dire 15,6 comparaisons par pixel.

Un bon point de comparaison est de considérer l'extraction de base des composantes connexes. Pour simplifier, oublions l'extraction des trous. L'algorithme de base consiste à appliquer successivement chaque seuil possible à l'image et à extraire à chaque fois par un algorithme de temps linéaire les composantes connexes des pixels noirs et blancs dans l'image binaire. Chaque pixel doit se comparer à ses voisins nord et ouest (et le voisin nord-ouest en cas de 8-connexité), ce qui revient à au moins deux comparaisons par pixel. La complexité est donc $2g.N$ où g est le nombre de niveaux de gris dans l'image et N le nombre de pixels. C'est en apparence linéaire $O(N)$, mais avec une constante au minimum $2g$, qui peut être typiquement 200, et au plus (si tous les niveaux de gris sont utilisés) $2 \times 256 = 512$. C'est beaucoup plus élevé que nos résultats expérimentaux avec la FLST. De plus, si les niveaux de gris sont représentés avec des valeurs réelles, au pire chaque pixel a

⁷Il y a probablement un lien avec la norme BV de l'image. Ce rapport nécessiterait d'être étudié.

FIG. 3.15 – Image satellite Spot (taille 6000×500) utilisée pour effectuer les expériences sur la dépendance du nombre de comparaisons dans la FLST en fonction du contenu de l'image. L'image est affichée en 6 parties de largeur 1000 de gauche à droite et de haut en bas.

un unique niveau de gris et $g = N$, donc l'algorithme de base devient de complexité $O(N^2)$. Cette complexité le rend trop lent en pratique, alors que la FLST ne perd pas en efficacité.

Accès mémoire

Cette mesure de complexité n'est pas nécessairement la meilleure, car l'algorithme ne se réduit pas à des comparaisons : les accès à la mémoire sont spécialement importants. En effet, chaque fois qu'une nouvelle forme est trouvée, ses pixels sont énumérés pour trouver la plus grande forme contenant chacun et la mettre comme enfant de la nouvelle forme ; cela implique d'accéder aux formes par des pointeurs, et ces pointeurs ne sont pas forcément proches en mémoire, ce qui induit ce qu'on

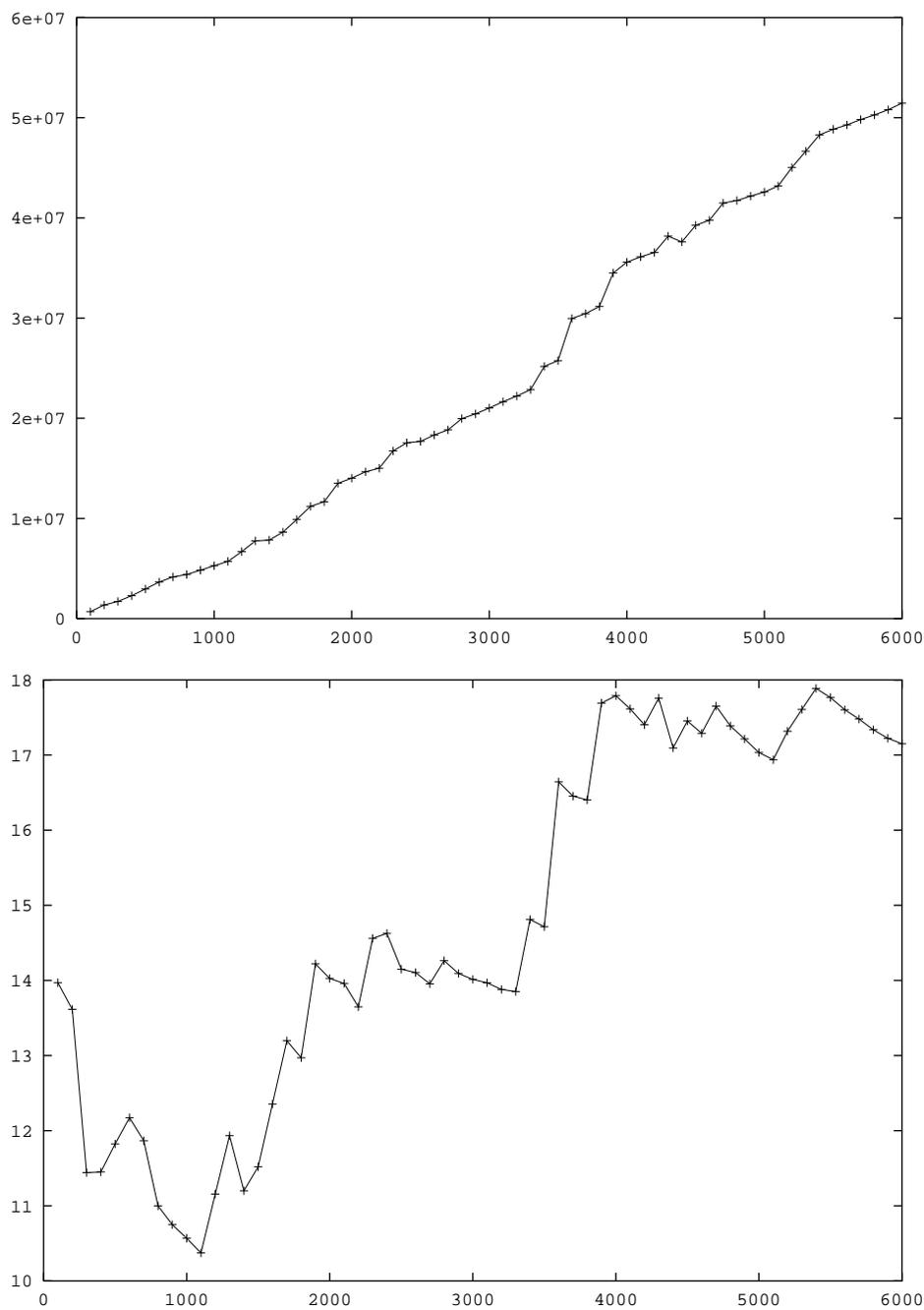


FIG. 3.16 – Dépendance du nombre de comparaisons de valeurs de pixels dans la FLST en fonction du contenu de l'image. Expérience effectuée sur des extraits de l'image satellite de la Figure 3.15. Haut : le nombre de comparaisons de valeurs de pixels en fonction de la largeur de l'image extraite. Bas : idem après division par le nombre de pixels de l'image extraite.

FIG. 3.17 – L'image de tapis, de taille 1024×715 .

appelle le *memory paging*, qui peut être très coûteux. Puisque les pixels appartiennent à plusieurs formes, le nombre de parcours pour chaque pixel peut être important, dépendant de la profondeur de l'arbre en la plus petite forme contenant ce pixel. Le nombre de tels accès mémoire est facile à compter : c'est la somme des aires de toutes les formes, moins N (l'aire de la racine).

Au mieux, l'image est uniforme, il n'y a qu'une forme, et cette étape a une complexité nulle. Au pire, nous pouvons imaginer une image où toutes les formes n'ont qu'un pixel propre, donc avec N formes, et que ces formes soient emboîtées, donnant un arbre sans ramifications. Donc le nombre d'accès mémoire serait :

$$1 + \dots + (N - 1) = \frac{N(N - 1)}{2} = O(N^2).$$

C'est une très mauvaise performance pour l'algorithme, comme les accès mémoire peuvent être très chers, bien plus que la comparaison de valeurs de pixels, en particulier si celles-ci sont codées sur un octet.

Comme nous pouvons le voir, le nombre d'accès mémoire dépend du contenu de l'image, pas seulement de sa taille, bien qu'il soit majoré par $O(N^2)$. Heureusement, les images usuelles ne présentent pas un aussi mauvais comportement, et le nombre d'accès mémoire est bien moins élevé.

La Figure 3.18 montre le nombre d'accès mémoire pour l'image satellite Spot de la Figure 3.15, relativement à la largeur de la sous-image. Pour l'image Lenna (256×256), ce nombre est 4084 219, ou 62 accès mémoire par pixel. Pour l'image du tapis (1024×715), 144 302 787, ou 197 par pixel.

Performances

La meilleure mesure pragmatique de la complexité de la FLST est donnée par le temps CPU utilisé par l'algorithme. Les tests sont effectués sur un processeur Pentium II à 300 MHz, avec 192 Mo de RAM, sous Linux.

L'image Lenna (256×256) prend 1,5 s, c'est-à-dire $23 \mu\text{s}$ par pixel. Pour l'image de tapis de la Figure 3.17 (1024×715), 50,8 s, ou encore $69 \mu\text{s}$ par pixel. La complexité de cette image, donnant de nombreuses formes incluses et un arbre très profond, en fait un véritable défi.

En ce qui concerne l'image Spot de la Figure 3.15, le temps CPU et le temps CPU par pixel, par rapport à la largeur de la sous-image que nous prenons, sont montrés dans la Figure 3.19.

Comme ces expériences le suggèrent, pour des images de taille moyenne et de complexité raisonnable, les temps de calcul font de la FLST un algorithme utilisable

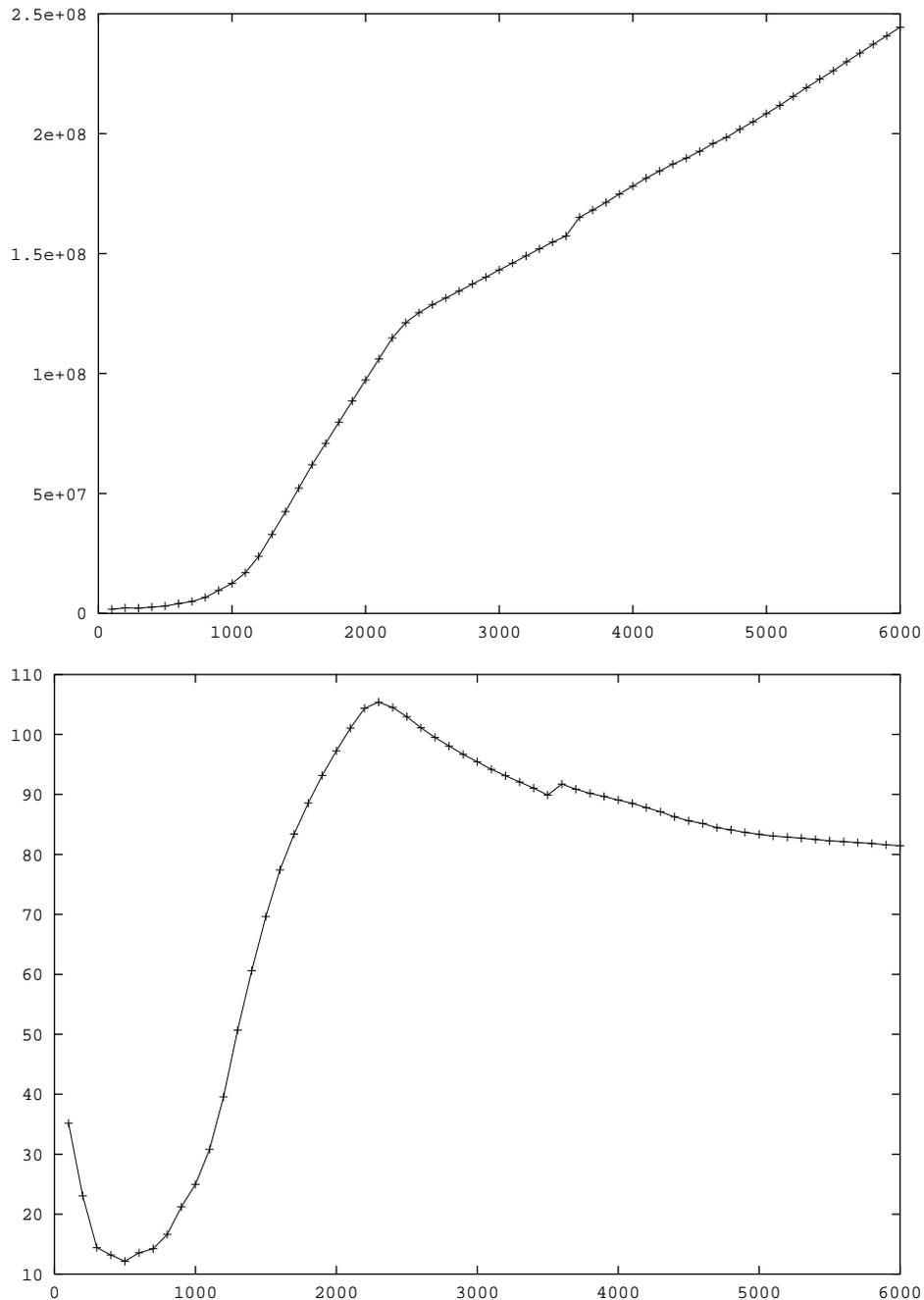


FIG. 3.18 – Dépendance du nombre d'accès mémoire dans la FLST par rapport au contenu de l'image. Expérience effectuée sur des extraits de l'image satellite de la Figure 3.15. Haut : nombre d'accès mémoire par rapport à la largeur de l'image extraite. Bas : idem après division par le nombre de pixels de l'image extraite.

dans la plupart des tâches de l'analyse d'image. Le nom de *Fast Level Set Transform* est dû à ces bonnes performances expérimentales, plutôt qu'à une complexité, qui est inconnue.

3.4 Prendre avantage de la structure d'arbre

Le fait d'avoir un arbre d'inclusion permet de calculer avec peu de mémoire ou de temps de calcul quelques caractéristiques des formes. Nous montrons deux exemples de telles possibilités. La première illustre l'économie de mémoire, la seconde l'économie de temps de calcul.

3.4.1 Stockage des pixels des formes numériques

Nous montrons ici comment nous pouvons stocker les pixels de toutes les formes avec une grande économie de mémoire. La FLST donne pour chaque forme, une fois détectée, sa liste de pixels. Mais stocker pour chacune cette liste serait gourmand en mémoire, car les formes peuvent être emboîtées, de telle sorte que chaque pixel apparaît dans différentes (éventuellement nombreuses) listes. Cela peut exiger une importante mémoire disponible, bien plus grande que la taille de l'image, N . Nous proposons à la place une méthode où chaque pixel n'est listé qu'une fois.

Pour cela, les formes doivent être énumérées en préordre, c'est-à-dire que chaque forme est énumérée avant ses enfants et que les frères sont énumérés de façon adjacente (voir Sedgewick, [82]). Ce n'est pas l'ordre dans lequel la FLST extrait les formes. Néanmoins, quel que soit l'ordre initial des formes, il est aisé et rapide de les réordonner. Cela peut se faire par une procédure récursive facile, mais la version non récursive n'est guère plus complexe à implémenter, voir Aho *et al.*, [1]. Si cette énumération est nécessaire, sa complexité est de l'ordre du nombre de formes, donc au plus $O(N)$. Nous supposons donc que les formes sont énumérées dans un tel ordre, nommons-les $\{T_1, \dots, T_k\}$, où k est le nombre de formes, de telle sorte que

1. $\forall i, j, \quad T_i \subset T_j \Rightarrow i \geq j$;
2. $\forall i, j, l, \quad T_i \subset T_l \text{ et } T_j \subset T_l \Rightarrow \forall m \text{ tel que } i \leq m \leq j, T_m \subset T_l$.

Une fois cela fait, nous stockons pour chaque forme son nombre de pixels propres. Nous appelons les pixels propres d'une forme S les pixels u_i dont la plus petite forme contenant chacun S_i est S . Il a été prouvé que chaque forme contient au moins un pixel propre. Ce nombre est l'aire de S moins la somme des aires de ses enfants, donc calculable en $O(k)$, qui est au plus $O(N)$. Nous notons ces nombres p_i . L'aire

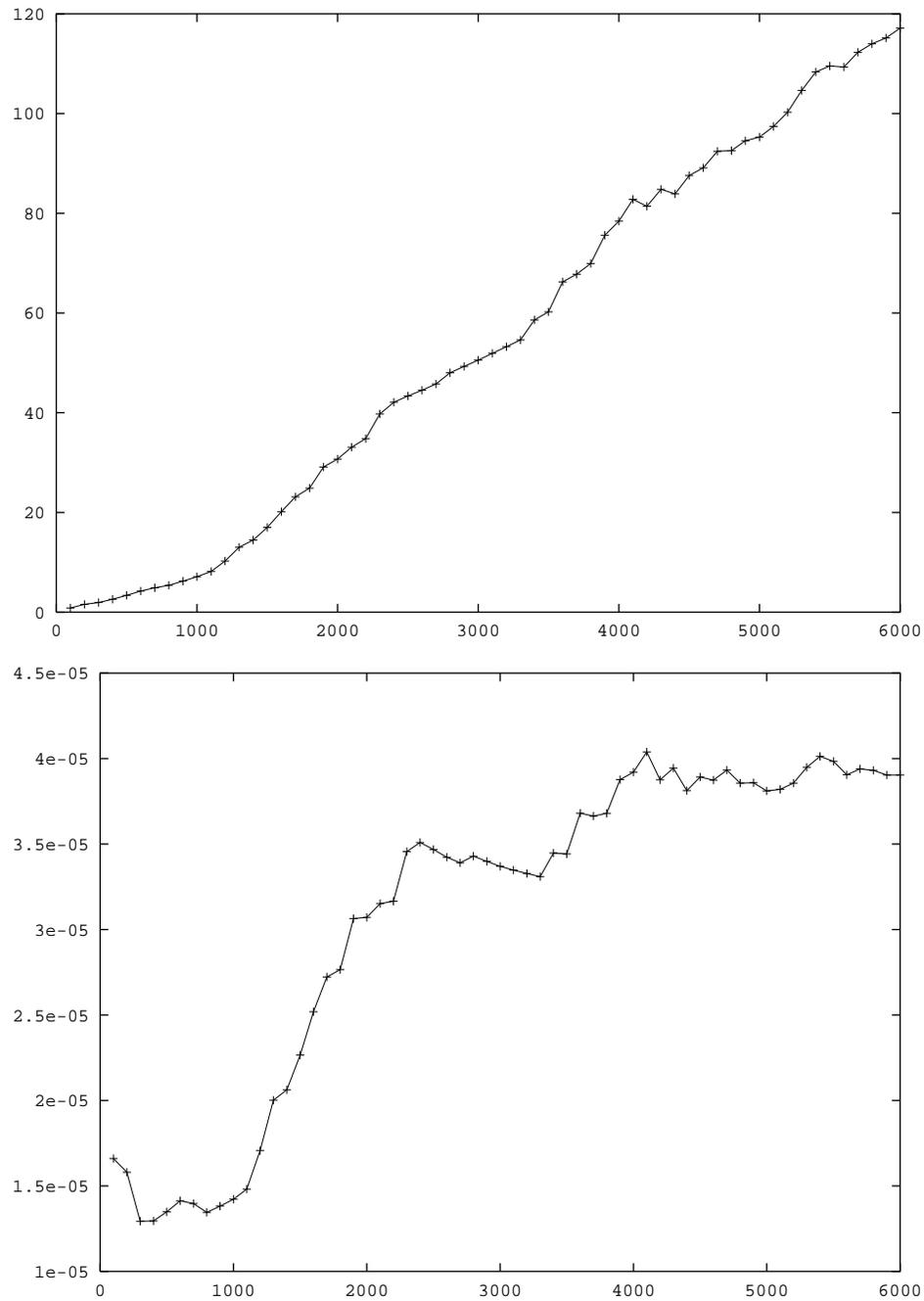


FIG. 3.19 – Haut : temps CPU de la FLST appliquée à une sous-image de l'image satellite Spot de la Figure 3.15, en fonction de la largeur de la sous-image. L'unité de l'axe des y est la seconde. Bas : temps CPU par pixel, en μs .

des formes est donnée par la FLST durant l'extraction et peut être enregistrée, ou bien peut se retrouver après l'extraction de l'arbre (voir la section suivante).

L'idée est d'énumérer les pixels dans un tableau A avec l'ordre induit par le préordre des formes, tel que les pixels appartenant à la même forme sont dans un intervalle de A . Alors chaque forme T_i n'a besoin que de stocker l'index I_i du début de l'intervalle dans A (et la longueur de l'intervalle, qui est l'aire de la forme). Ceci est expliqué dans l'Algorithme 5. Chacun de ces intervalles contient les pixels propres de la forme, suivi des pixels de ses enfants. Par exemple, les p_1 premiers pixels sont les pixels propres de la racine, les p_2 pixels suivants les pixels propres du premier enfant de la racine, etc. La complexité de cet algorithme est visiblement $O(k) + O(N) = O(N)$ et la quantité de mémoire nécessaire $O(N)$.

Algorithm 5 Stockage efficace des pixels associés à chaque forme

Require: T_i et p_i , pour $i = 1 \dots k$ {Formes en préordre et leur nombre de pixels propres}

Require: S_j , pour $j = 1 \dots N$ {Pour chaque pixel, la plus petite forme le contenant}

$l \leftarrow 1$ {Initialisation de l'index courant}

for $i = 1$ à k **do** {Boucle de calcul des indices I_i }

$I_i \leftarrow l$

$l \leftarrow l + p_i$

end for

Copier I_i dans le tableau temporaire I'_i { I'_i est la première place réservée pour un pixel propre de T_i }

for $j = 1$ à N **do** {Boucle d'énumération des pixels}

$i \leftarrow$ index de S_j , tel que $S_j = T_i$ {Index de la forme pointée par S_j }

$A_{I'_i} \leftarrow j$ {Ajouter le pixel j dans le tableau A }

$I'_i \leftarrow I'_i + 1$ {Incrémenter I'_i }

end for

3.4.2 Calcul de caractéristiques additives de forme

Nous pouvons aussi prendre avantage de la structure d'arbre pour calculer des caractéristiques additives associées aux formes. Nous appelons caractéristique additive

d'un ensemble un réel c tel que pour toute paire d'ensemble S_1 et S_2 :

$$S_1 \cap S_2 = \emptyset \quad \Rightarrow \quad c(S_1 \cup S_2) = c(S_1) + c(S_2).$$

Des exemples de telles caractéristiques sont nombreux, en particulier l'aire, plus généralement les moments, ou toute quantité intégrale. Au contraire, d'autres caractéristiques intéressantes ne sont pas additives, comme le périmètre.

Nous pouvons calculer des caractéristiques additives de toutes les formes avec une complexité linéaire $O(N)$. Ce n'est pas direct, puisque les formes peuvent être emboîtées.

L'idée est à nouveau d'utiliser la structure d'arbre d'inclusion. Supposons comme dans la section précédente que les formes sont énumérées en préordre dans un tableau $\{T_1, \dots, T_k\}$. Alors l'algorithme est en deux étapes :

1. Calculer les caractéristiques de l'ensemble des pixels propres de chaque forme.
2. Ajouter les caractéristiques de chaque forme à celles de son parent.

En effet, comme une forme est composée de ses pixels propres et de la famille des pixels propres de toutes ses formes contenues, nous sommes certain de décrire toute la forme. La méthode est décrite dans l'Algorithme 6. La complexité est évidemment de l'ordre de $O(N)$ additions.

Algorithm 6 Calcul d'une caractéristique additive c des formes

Require: $\{T_1, \dots, T_k\}$ {Les formes en préordre}

Require: $\{S_1, \dots, S_N\}$ {Pour chaque pixel, sa forme contenue la plus petite}

$c_i \leftarrow 0$ pour $i = 1 \dots k$ {Initialisation}

for $j = 1$ à N **do** {Boucle concernant les pixels propres}

$i \leftarrow$ index de S_j , tel que $S_j = T_i$ {Index de la forme pointée par S_j }

$c_i \leftarrow c_i + c(j)$ { $c(j)$ est la caractéristique de l'ensemble constitué du pixel numéro j }

end for

for $i = k$ décroissant jusqu'à 2 **do** {L'ordre est important; il est tel que les caractéristiques des enfants sont calculées avant celles de leurs parents}

$i' \leftarrow$ index du parent de T_i

$c_{i'} \leftarrow c_{i'} + c_i$ {Ajout de la caractéristique de l'enfant à son parent}

end for

3.5 Extensions

3.5.1 Changement de connexité

L'algorithme a été présenté pour une image discrète considérée comme un échantillonnage d'une image définie sur un domaine continu. Ceci permet d'utiliser des résultats topologiques dans le plan et de les traduire dans un cadre discret. En particulier, la question de la connexité est centrale dans l'algorithme. Il est apparu que l'utilisation de la 4-connexité pour un type d'ensemble de niveau de la 8-connexité pour l'autre type correspond implicitement à choisir l'image définie sur un domaine continu associée comme semicontinue supérieurement ou inférieurement. Toutefois, il est légitime de se demander si l'utilisation de seulement *une* notion de connexité conduit également à un arbre d'inclusion des formes.

Si nous travaillons avec la k -connexité ($k = 4$ ou 8), deux formes se rencontrant sont-elles emboîtées ? Les formes dans ce cas sont les composantes k -connexes d'ensembles de niveau, union leurs trous, où les trous sont les composantes k -connexes du complémentaire.

Quand $k = 4$, la réponse est négative. La Figure 3.20 exhibe une configuration où deux formes se rencontrent, alors qu'aucune d'entre elles ne contient l'autre. Il y a une explication facile à ce fait : les deux pointels nécessaires à la connexion des deux composantes 4-connexes de l'ensemble isoniveau $[u = 0]$ ont une valeur inférieure à 1, de telle sorte que la composante connexe de l'ensemble de niveau supérieur $[u \geq 1]$ ne se connecte pas au fond au niveau 2. De même, ils doivent avoir une valeur plus grande que 1, pour empêcher la connexion avec l'autre composante connexe de $[u = 0]$. Les conditions sur les valeurs de ces pointels sont donc contradictoires. La solution est d'enlever ces pointels de l'ensemble de définition de l'image. C'est-à-dire que considérer la 4-connexité pour les ensembles et leur complémentaire revient à imaginer l'image définie sur le domaine continu $\bar{\Omega}$ moins les pointels de jonction. Le problème avec cet ensemble de définition est qu'il n'est pas univoque, empêchant la représentation de l'image comme un arbre d'inclusion des formes.

Au contraire, quand $k = 8$, la réponse est, de façon surprenante, positive. Notons que cela donne une notion contre-intuitive de trou, voir la Figure 3.21, montrant un rectangle n'ayant pas d'intérieur et d'extérieur, c'est-à-dire dont le complémentaire est connexe. Nous donnerons un schéma de démonstration.

La croissance de l'opérateur de saturation reste vraie. Pour deux composantes connexes d'ensembles de niveau du même type, si elles se rencontrent, l'une contient l'autre, et donc leurs saturations sont emboîtées dans le même ordre. Si elles ne se rencontrent pas, elles sont disjointes, et chacune est incluse soit dans un trou soit

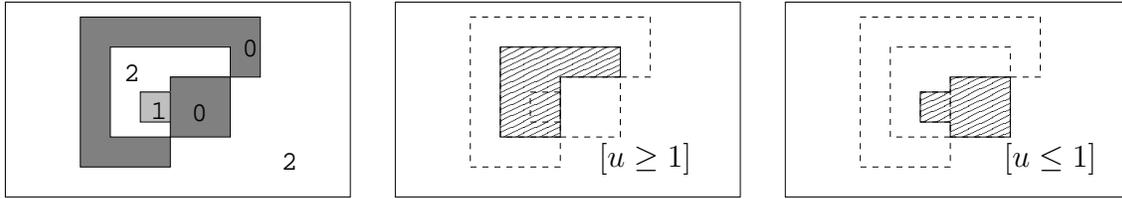


FIG. 3.20 – Une configuration où le choix de la 4-connexité pour les ensembles et aussi pour leur complémentaire ne donne pas un arbre d’inclusion des formes. Gauche : l’image u . Milieu : la composante connexe de l’ensemble de niveau supérieur $[u \geq 1]$ n’a pas de trou, c’est une forme. Droite : la composante connexe de l’ensemble de niveau inférieur $[u \leq 1]$ n’a pas de trou non plus, c’est aussi une forme. Pourtant, ces deux formes se rencontrent sans être emboîtées.

dans l’extérieur de l’autre. Le résultat s’ensuit alors facilement. De même, si nous avons une composante connexe d’ensemble de niveau supérieur $C = cc([u \geq \lambda])$ et une autre d’ensemble de niveau inférieur $C' = cc([u \leq \mu])$, si de plus $C \cap C' = \emptyset$, la même preuve s’applique. Notons que jusqu’à présent, ceci reste vrai lorsque toutes les connexions sont considérées en 4-connexité. Le seul cas restant est $C \cap C' \neq \emptyset$. Ceci implique $\mu \leq \lambda$. Si $\text{sat}(C) = \Omega_d$, rien de plus n’est à montrer. Sinon, l’ensemble N des 8-voisins de $\text{sat}(C)$ est connexe et dans $[u < \lambda] \subset [u \leq \mu]$. Seuls deux cas sont possibles :

1. $N \cap C' = \emptyset$. Comme C' est 8-connexe et rencontre $\text{sat}(C)$, ceci implique $C' \subset \text{sat}(C)$, et donc $\text{sat}(C') \subset \text{sat}(C)$.
2. $N \subset C'$. Nous avons alors $\text{sat}(N) \subset \text{sat}(C')$. Or $\text{sat}(C)$ est une composante connexe de $\Omega_d \setminus N$, donc nous avons soit $\text{sat}(C) \subset \text{sat}(N)$, auquel cas $\text{sat}(C) \subset \text{sat}(C')$, soit $\text{sat}(N) \supset \Omega_d \setminus \text{sat}(C)$, mais comme $\text{sat}(C) \neq \Omega_d$, nous avons $\text{sat}(\Omega_d \setminus \text{sat}(C)) = \Omega_d$, donc $\text{sat}(N) = \Omega_d$, et finalement $\text{sat}(C') = \Omega_d \supset \text{sat}(C)$.

Pour autoriser la connexion de voisins diagonaux aux pointels de jonction, ces pointels de jonctions doivent prendre plusieurs valeurs. Cela montre qu’il n’y a pas d’image à domaine continu associée. A la place l’image discrète doit être considérée comme l’échantillonnage d’une fonction multivaluée. Mais la théorie sur l’arbre d’inclusion pour les images multivaluées n’est pas faite.

L’avantage d’utiliser seulement cette notion de connexité est que les ensembles de niveaux supérieurs et inférieurs sont considérés de la même manière. L’arbre devient symétrique. Remarquons toutefois que considérer seulement la 8-connexité empêche le calcul local de la caractéristique d’Euler, voir Kong et Rosenfeld [32]. La conséquence en est que l’algorithme doit être modifié de la façon suivante : lorsqu’une composante connexe d’ensemble de niveau est extraite, pour trouver si elle

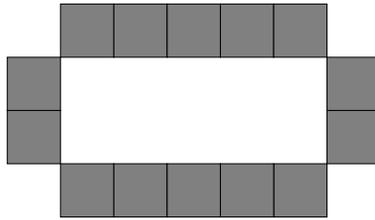


FIG. 3.21 – Lorsque nous considérons la 8-connectivité pour un ensemble et son complémentaire, les pixels gris composent un rectangle, mais le complémentaire est connexe, donc il n’y a pas de trou. Cela donne une notion de trou contraire à l’intuition.

a un trou, nous devons utiliser une extraction standard des composantes connexes du complémentaire, voir Rosenfeld and Pfaltz [71], Lumia *et al.* [41]. C’est un algorithme de complexité linéaire, néanmoins il doit être utilisé pour chaque composante connexe d’ensemble de niveau trouvée, donc cette étape devient le goulot d’étranglement de l’ensemble.

3.5.2 Dimension supérieure

La FLST peut se généraliser presque directement à la dimension supérieure. Le seul point délicat concerne la détermination de la présence ou non d’un trou dans une composante connexe d’ensemble de niveau. C’est à nouveau calculable localement, mais les différentes configurations du voisinage, au nombre de 256 pour les images 2-D, devient 2^{3^n-1} (n étant la dimension), donc ne peuvent être codées en dur dès que $n \geq 3$. Toutefois, l’identification de la configuration n’est pas vraiment plus difficile.

En 3-D en particulier, nous considérons la 6-connectivité pour les ensembles de niveau inférieurs, et la 26-connectivité pour les ensembles de niveau supérieurs (ou l’inverse). Une alternative est de considérer la 26-connectivité pour les deux (mais pas la 6-connectivité) et d’avoir aussi un arbre d’inclusion des formes (la preuve concernant la 8-connectivité dans la section précédente est valable aussi en dimension supérieure pour la $(3^n - 1)$ -connectivité). Mais de nouveau, la notion de trou ne devient pas vraiment intuitive, et la caractéristique d’Euler n’est pas localement calculable. Elle peut être calculée comme expliqué par Lee, Poston et Rosenfeld [36], Lee et Rosenfeld [37], Lumia [40].

Chapitre 4

Applications à quelques filtres morphologiques

Dans ce chapitre, nous expliquons de quelle façon l'arbre d'inclusion des formes joue un rôle dans un filtre morphologique intéressant, le filtre de grain, et introduit une quantification adaptative de l'image, qui est un filtre de choc particulier (voir Osher et Rudin [64]), reposant sur l'arbre d'inclusion. Quelques-unes de ces applications et expériences ont été présentées dans [59] et [58].

4.1 Filtres morphologiques

Comme l'information de contraste est en grande partie non pertinente, les filtres appliqués aux images doivent être insensibles à un changement de contraste, d'où la nécessité d'utiliser des filtres morphologiques. Par exemple, l'un des filtres morphologiques les plus anciens est le fameux filtre médian, voir Huang *et al.* [28]. Matheron [50] a un théorème de représentation pour les opérateurs invariants par translation, et après lui Maragos et Shafer [44, 45] font le lien entre les opérateurs invariants par changement de contraste et monotones et les opérateurs ensemblistes monotones. Ils montrent l'équivalence des deux. A tout filtre morphologique est associé un opérateur ensembliste monotone, dans le sens que les ensembles de niveau de l'image filtrée sont les ensembles de niveau de l'image originale après transformation par l'opérateur ensembliste. Inversement, d'un opérateur ensembliste nous pouvons construire un filtre morphologique tel que la même propriété s'applique. De plus, cela permet de donner une forme générale des filtres morphologiques :

$$Tu(\mathbf{x}) = \sup_{B \in \mathcal{B}_x} \inf_{\mathbf{y} \in B} u(\mathbf{y})$$

où $\mathcal{B}_{\mathbf{x}}$ est une famille d'ensembles dépendant de \mathbf{x} , et si de plus le filtre est invariant par translation, nous pouvons écrire

$$\mathcal{B}_{\mathbf{x}} = \mathcal{B} + \mathbf{x},$$

où \mathcal{B} est une famille d'ensemble indépendante de \mathbf{x} .

Alvarez *et al.* [3] classifient totalement les filtres morphologiques réguliers. Ils montrent qu'à tout filtre morphologique régulier est associé une évolution par une équation aux dérivées partielles (EDP). Par exemple, le filtre médian de paramètre t_0 correspond à la solution $u(\cdot, t_0)$ de l'équation parabolique :

$$\frac{\partial u}{\partial t}(\mathbf{x}, t) = |Du|(\mathbf{x}, t) \operatorname{curv} u(\mathbf{x}, t),$$

avec condition initiale $u(\cdot, 0) = u(\cdot)$, où Du est le gradient de u et $\operatorname{curv} u$ est la courbure de la ligne de niveau passant par \mathbf{x} , c'est-à-dire $\operatorname{div} \frac{Du}{|Du|}$. L'invariance géométrique globale de la famille d'ensembles \mathcal{B} est équivalente à la même invariance géométrique du filtre morphologique, ce qui permet par exemple à Catté *et al.* [14] de proposer un schéma invariant par changement de contraste pour cette évolution par courbure moyenne.

La classification de [3] permet aux auteurs de présenter le seul filtre morphologique régulier et invariant affine, dont l'EDP associée est :

$$\frac{\partial u}{\partial t} = |Du|(\operatorname{curv} u)^{1/3}.$$

L'opérateur d'évolution de courbe associé a été découvert indépendamment à la même époque par Sapiro and Tannenbaum [81]. Un schéma géométrique a été par la suite découvert par Moisan [56]. Il apparaît que ce filtre, appelé l'AMSS, Affine invariant Morphological Scale-Space, n'est pas invariant sous une transformation projective, et qu'aucune invariance géométrique supplémentaire ne peut être atteinte. Donc, pour obtenir l'invariance projective, Faugeras et Keriven [20] doivent abandonner le principe du maximum, ce qui entraîne une instabilité numérique du filtre résultant. Parmi les filtres réguliers, le filtre le plus invariant est donc l'AMSS.

Une nouvelle espèce de filtres morphologiques, abandonnant la condition de régularité, et motivée par l'étude de quelques "stack filters", voir Salembier [73], a été présentée par Serra et Salembier dans [85, 80]. Ils les appellent les filtres connexes. Ils reposent sur les zones plates, les composantes connexes des ensembles isoniveaux, et construits en fusionnant des zones plates adjacentes, donnant à leur

réunion le niveau de gris de l'une des zones plates dont elle est construite. Cela a stimulé la découverte de nombreux filtres intéressants pour le traitement d'image et de séquence d'images, voir Salembier [74], Salembier et Garrido [76], Salembier, Oliveras et Garrido [78], d'un algorithme d'estimation du mouvement, voir Salembier et Sanson [79], et d'un algorithme de compression d'image, voir Salembier *et al.* [75, 77]. Les filtres connexes ont la propriété importante de conserver les bords et les jonctions en T dans l'image sans les déplacer.

Ces articles sont particulièrement intéressants pour notre travail, puisqu'ils reposent sur des manipulations d'un arbre. En effet, l'image est partitionnée en ses composantes connexes d'ensembles isoniveaux, et une segmentation par fusion de telles régions est définie. La restriction est qu'à chaque étape seulement deux régions fusionnent, et le processus de fusion continue jusqu'à ce qu'il ne reste plus qu'une région, l'image entière. Les étapes de cette segmentation peuvent se représenter par un arbre d'inclusion binaire. Notons que la construction de l'arbre d'inclusion suit de très près ces lignes, sauf qu'à chaque étape plusieurs régions peuvent fusionner.

Les filtres proposés par Salembier et les coauteurs reposent sur la reconstruction de l'image après la suppression de quelques nœuds dans l'arbre, reposant sur un certain critère. Dans [76], les critères monotones sont présumés plus robustes, mais les auteurs expliquent comment un critère non monotone peut toutefois être transformé en critère monotone : une modification du critère a lieu de manière à contredire la conservation ou la suppression de nœuds un nombre minimum de fois ; un algorithme de programmation dynamique permet une telle modification du critère. En d'autres termes, ils affirment que seuls les filtres correspondant à un élagage de l'arbre sont stables. De cette manière, même des filtres insensibles à l'inversion du contraste sont proposés. Néanmoins, tous ces filtres reposent sur une segmentation antérieure, et cette segmentation doit être morphologique pour que le filtre soit morphologique. Dans cette thèse, la segmentation est donnée par la construction de l'arbre d'inclusion des formes.

En remarquant que les filtres connexes représentent une large classe de filtres, Meyer [51, 52] distingue parmi eux ceux qui préservent l'ordre des niveaux de gris des pixels voisins, ce qu'il appelle les "flattenings" monotones, et les spécialise encore plus en "levelings", des filtres qui satisfont la condition suivante, particulièrement élégante en simplicité et concision : g est un leveling de l'image f si pour tous pixels voisins P et Q , $g(P) > g(Q) \Rightarrow f(P) \geq g(P) > g(Q) \geq f(Q)$. Meyer et Maragos donnent des exemples d'espace-échelle reposant sur des levelings dans [53, 43].

Un type particulièrement intéressant de leveling est l'ouverture d'aire, ainsi que la fermeture d'aire, voir Vincent [88, 89]. En effet, elles sont morphologiques, sup inf

et inf sup avec comme éléments structuraux \mathcal{B} la famille des ensembles connexes d'aire donnée t contenant l'origine \mathbf{O} . l'ouverture d'aire aplanit les maxima d'aire insuffisante. Les images filtrées ont des maxima régionaux (resp. minima) d'aire au moins t après l'ouverture d'aire (resp. fermeture). Ballester *et al.* [5] étudient dans un cadre continu les sections monotones maximales d'une image filtrée par ouverture et fermeture d'aire.

4.2 Filtre de grain

Suivant la direction de Vincent, Masnou dans [49] propose un moyen d'éviter les inconvénients de l'ouverture et la fermeture d'aire de Vincent, à savoir que chacun traite d'un type spécifique d'extrema. La solution de Masnou repose sur l'inclusion des intérieurs de lignes de niveau, comprises comme des courbes de Jordan, le filtre proposé s'appelle le filtre de grain. Toutefois, il n'est pas vrai que les frontières des lignes de niveau sont faites de courbes de Jordan disjointes. Essayant pourtant d'avoir une justification rigoureuse pour le filtre de grain, Ambrosio *et al.* [4] développent une nouvelle notion de connexité pour les ensembles de niveau d'une image à variation bornée (BV, for bounded variation). Pour une définition des fonctions BV, voir Evans et Gariepy [18]. Dans leur cadre, la connexité est adaptée aux fonctions BV, qui sont définies à un ensemble de mesure nulle près. Ils montrent qu'avec leur définition, la frontière d'une composante connexe d'ensemble de niveau se compose d'un nombre fini ou dénombrable de courbes de Jordan rectifiables disjointes (dans le sens de la théorie de la mesure), et Masnou dans [48] a pu démontrer la propriété essentielle du filtre de grain, son invariance par inversion du contraste, pour les fonctions régulières (de classe C^n dans \mathbb{R}^n) qui, après changement de contraste éventuel, sont dans la classe BV, appelées faiblement à variation bornée (WBV). Ballester *et al.* ont pu étendre le même résultat aux fonctions continues dans WBV.

Alors que le cadre BV est puissant pour l'étude des fonctions, voir en particulier Rudin, Osher et Fatemi [72], il existe des indices laissant penser que les images naturelles n'entrent pas dans le cadre BV, voir Alvarez, Gousseau et Morel [2].

Dans cette section, nous reformulons le filtre de grain comme un opérateur sup inf classique agissant sur les fonctions semicontinues supérieurement, et montrons l'invariance relativement à une inversion de contraste pour les fonctions continues. La notion de connexité est la notion topologique classique ; nous ne travaillons pas dans le cadre BV. A cette différence près, c'est une extension des résultats mentionnés ci-dessus.

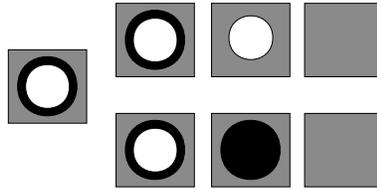


FIG. 4.1 – Différence entre l’ouverture et la fermeture d’aire, et le filtre de grain. Gauche : image originale. Ligne du haut : espace-échelle de l’image à travers des ouvertures et des fermetures d’aire alternées ; l’anneau noir étant plus petit que le disque blanc, il disparaît avant le disque. Ligne du bas : espace-échelle de l’image à travers le filtre de grain ; l’anneau noir est considéré comme un disque noir occlus par le disque blanc, donc il contient le disque blanc et disparaît après celui-ci.

4.2.1 Description

Le filtre de grain est une variante de l’ouverture et de la fermeture d’aire qui évite les inconvénients de ces levelings : ils sont différents et ne commutent pas. En particulier, cela signifie que ces filtres traitent différemment les ensembles de niveau supérieurs et inférieurs. Comme il a été souvent expliqué dans les chapitres précédents, un traitement égal des ensembles de niveau supérieurs et inférieurs est la principale motivation de cette thèse. Le filtre de grain est fait de telle sorte à opérer sur les formes, pas directement sur les composantes connexes d’ensembles de niveau. Grossièrement, l’image filtrée par le filtre de grain d’aire t de l’image u s’obtient par le processus suivant : c’est l’image dont les formes sont les formes de u d’aire au moins t . En d’autres termes, l’arbre d’inclusion de l’image filtrée par le filtre de grain de u , $T_t u$, est l’arbre d’inclusion de u dans lequel tous les nœuds d’aire moindre que t sont supprimés.

Notons la relation étroite avec l’ouverture et la fermeture d’aire : l’ouverture (resp. la fermeture) d’aire de l’image u est l’image dont les ensembles de niveau supérieurs (resp. inférieurs) sont composés des composantes connexes des ensembles de niveau supérieurs (resp. inférieurs) de u d’aire au moins t . Néanmoins, l’ouverture et la fermeture d’aire ne prennent pas en compte la notion d’inclusion, voir la Figure 4.1.

Comme les formes sont des traits morphologiques, ce filtre est invariant par changement de contraste, et étant donné que les formes sont insensibles à une inversion de contraste, nous nous attendons à ce que ce filtre soit autodual. Nous montrerons ces propriétés après avoir défini proprement le filtre de grain. Nous donnons deux versions du filtre de grain : l’opérateur ensembliste et l’opérateur fonctionnel.

L'opérateur ensembliste

Comme nous appliquerons le filtre de grain aux images semicontinues supérieurement, dont les ensembles de niveau supérieurs sont fermés, l'opérateur ensembliste correspondant au filtre de grain est défini sur les ensembles fermés.

Définition 4.1 *Pour un réel $t > 0$, nous définissons l'opérateur ensembliste T_t , transformant des fermés de X en sous-ensembles de X , par :*

$$\forall F \text{ fermé } \subset X, \quad T_t(F) = \bigcup \left\{ \text{sat}(C) \setminus C' : C = \text{cc}(F), \mu(\text{sat}(C)) \geq t, \right. \\ \left. C' \text{ trou de } C, \mu(C') > t \right\}.$$

Notons que la relation étroite avec les égalités dans le Corollaire 2.41, donnant la reconstruction d'un ensemble de niveau à partir de ses formes.

L'opérateur fonctionnel

Comme tout filtre morphologique, il peut être formulé comme un opérateur inf sup ou sup inf, et comme dans le cas de l'ouverture et de la fermeture d'aire, les éléments structurants peuvent être pris indépendamment de u . Il faut bien comprendre que bien que ces éléments soient indépendants de u , le résultat est déterminé uniquement par les formes de u , et nous pourrions aussi bien prendre des formes de u comme éléments structurants. Cette formulation, indépendante de u , nous permet de prouver plus facilement les propriétés de ce filtre.

Définition 4.2 *Les éléments structurants du filtre de grain d'aire t , dont la famille est notée \mathcal{B}_t , sont les ensembles B vérifiant :*

1. B est connexe et fermé ;
2. $\mathbf{0} \in \text{sat}(B)$;
3. $\mu(\text{sat}(B)) \geq t$;
4. $\mathbf{0} \notin B \Rightarrow \mu(\text{cc}(X \setminus B, \mathbf{0})) \leq t$.

Le filtre de grain d'aire $t > 0$ appliqué à l'image u est défini par :

$$T_t u(\mathbf{x}) = \sup_{B \in \mathcal{B}_t} \inf_{\mathbf{y} \in \mathbf{x} + B} u(\mathbf{y}). \quad (4.1)$$

La quatrième propriété vérifiée par les éléments de \mathcal{B}_t est que si l'origine $\mathbf{0}$ est dans un trou, ce trou est d'aire au plus t .

Que cet opérateur fonctionnel T_t soit associé à l'opérateur ensembliste \mathbb{T}_t de la Définition 4.1 sera montré plus loin, après avoir prouvé que l'opérateur ensembliste est monotone et semicontinu supérieurement.

4.2.2 Lien entre opérateur ensembliste et opérateur fonctionnel

Examinons l'effet du filtre de grain sur les ensembles de niveau d'une image semicontinue supérieurement u . Nous supposons que nous sommes dans le cadre de la Section 2.5, i.e., soit $X = \mathbb{R}^n$ et l'image u est constante hors d'un certain ensemble fermé, ou X est la fermeture d'un domaine de Jordan.

Nous prouverons que les ensembles de niveau supérieurs de $T_t u$ sont les images des ensembles de niveau supérieurs de u par l'opérateur ensembliste \mathbb{T}_t . En d'autres termes,

$$\forall \lambda \in \mathbb{R}, \quad [T_t u \geq \lambda] = \mathbb{T}_t([u \geq \lambda]). \quad (4.2)$$

Cela se fait classiquement en trois étapes, voir Guichard et Morel [26] :

1. \mathbb{T}_t est monotone : $A \subset B \Rightarrow \mathbb{T}_t(A) \subset \mathbb{T}_t(B)$;
2. \mathbb{T}_t est semicontinu supérieurement : pour toute suite décroissante d'ensembles compacts F_n , $\mathbb{T}_t(\bigcap_n F_n) = \bigcap_n \mathbb{T}_t(F_n)$;
3. $\mathcal{B}_t = \{F : \mathbf{0} \in \mathbb{T}_t(F)\}$.

Le troisième point n'est pas tout à fait vrai, nous avons seulement une inclusion, pourtant ce sera suffisant pour notre propos.

Avant tout ceci, la première chose à prouver est que \mathbb{T}_t transforme un compact en compact.

Proposition 4.3 *Pour les opérateurs de saturation des Formules (2.4) et (2.5), \mathbb{T}_t transforme des ensembles compacts en ensembles compacts.*

Preuve.

1. Considérons une suite de points \mathbf{x}_n dans $\mathbb{T}_t(K)$ où K est un compact. Supposons que cette suite converge vers $\mathbf{x} \in X$. Nous montrerons que $\mathbf{x} \in \mathbb{T}_t(K)$. Ceci prouvera que $\mathbb{T}_t(K)$ est fermé, et comme il est visiblement borné, le résultat sera prouvé.

2. Tout \mathbf{x}_n appartient à une composante $\mathbb{T}_t(K_n)$ de $\mathbb{T}_t(K)$ (voir le Lemme 4.4, plus bas), K_n étant une composante connexe de K . Supposons que la famille des $\mathbb{T}_t(K_n)$ est en fait finie pour $n \in \mathbb{N}$. Alors nous pouvons extraire une sous-suite telle que tous les éléments appartiennent à $\mathbb{T}_t(K_n)$ et comme $\mathbb{T}_t(K_n)$ est fermé (son complémentaire est l'extérieur de K_n union certains trous, tous ces ensembles sont

ouverts), donc $\mathbf{x} \in T_t(K_n)$. Pour le reste de la preuve, nous supposons que les $T_t(K_n)$ sont infinis.

3. Il est clair que $\text{sat}(T_t(K_n)) = \text{sat}(K_n)$. Seul un nombre fini d'entre eux peuvent être deux à deux disjoints, comme chacun a une aire au moins t . Donc nous pouvons supposer qu'ils se rencontrent tous (après éventuellement avoir extrait une sous-suite), donc ils forment une suite d'ensembles soit décroissante soit croissante (après une éventuelle nouvelle extraction de sous-suite).

4. Si les $\text{sat}(K_n)$ sont décroissants, leur intersection est un ensemble $\text{sat}(K')$, où K' est une composante connexe de K et $\mu(\text{sat}(K')) \geq t$, d'après le Théorème 2.39. Il est facile de voir que \mathbf{x} appartient à $\partial\text{sat}(K') = \partial K' \subset K' \subset T_t(K')$.

5. Si les $\text{sat}(K_n)$ sont croissants, la limite inférieure des K_n est non vide (car elle contient \mathbf{x}), donc leur limite supérieure est un continu, inclus dans K , par le théorème de Zoratti (voir la formulation exacte dans [35]). Soit K' la composante connexe de K la contenant. Tous les $\text{sat}(K_n)$ appartiennent à la même composante connexe C de $X \setminus K'$. Donc cette composante connexe est d'aire au moins t . Nous montrerons que c'est un trou de K' (en d'autres termes, pas son extérieur), prouvant que $\text{sat}(K')$ est d'aire au moins t , et donc que $T_t(K') \neq \emptyset$, donc $\mathbf{x} \in T_t(K)$.

6. Ce dernier point est dû à la forme particulière de l'opérateur de saturation que nous utilisons. Si $X = \mathbb{R}^n$, comme les K_n sont bornés, chacun est séparé d'un voisinage de l'infini par K' , donc ils sont inclus dans un trou de K' . Si X est la fermeture d'un domaine de Jordan, si K' ne rencontre pas le cadre, la démonstration est similaire au cas \mathbb{R}^n . Si K' rencontre le cadre sans le contenir, si aucun des $\text{sat}(K_n)$ ne rencontre le cadre, leur union C non plus. Donc C est un trou de K' . Si un certain $\text{sat}(K_n)$ rencontre le cadre, chaque $\text{sat}(K_n)$ suivant aussi. Puisqu'aucun d'eux n'est X , cela implique que leur aire est au plus la moitié de celle de l'image. Nous concluons que C , dont l'aire est la borne supérieure des aires des $\text{sat}(K_n)$, a aussi une aire pas plus grande que t . Donc C ne peut être l'extérieur de K' . \square

T_t est monotone

Considérons deux ensembles fermés A et B tels que $A \subset B$. Alors les composantes connexes de A sont incluses dans les composantes connexes de B . Si C est une composante connexe de A dont la saturation est d'aire au moins t , il existe une composante connexe C' de B contenant C , et $\text{sat}(C')$ a une aire au moins t puisqu'elle contient $\text{sat}(C)$. Si H' est un trou de C' d'aire plus grande que t , H' est une composante connexe de $X \setminus C'$ qui est incluse dans $X \setminus C$. Donc H' est incluse

dans une composante connexe H de $X \setminus C$, et $\mu(H) > t$. Donc

$$\text{sat}(C) \setminus H \subset \text{sat}(C) \setminus H'. \quad (*)$$

H peut être un trou de C ou l'extérieur de C . Dans tous les cas, nous avons $T_t(A) \subset \text{sat}(C) \setminus H$, et l'inégalité $(*)$ donne

$$T_t(A) \subset \text{sat}(C) \setminus H'.$$

En prenant l'intersection des $\text{sat}(C) \setminus H'$ pour tous les H' trous de C' et d'aire plus grande que t , nous obtenons $T_t(A) \subset T_t(B)$. T_t est donc monotone.

T_t est semicontinu supérieurement

Si $(F_n)_{n \in \mathbb{N}}$ est une suite décroissante d'ensembles fermés, leur intersection F est un fermé, donc $T_t(\bigcap_n F_n)$ est bien défini. Comme pour tout $n \in \mathbb{N}$, $F \subset F_n$, et que T_t est monotone, $T_t(F) \subset \bigcap_n T_t(F_n)$.

Pour l'inclusion inverse, nous avons besoin d'une hypothèse supplémentaire : si aucun des F_n n'est borné, F contient un voisinage de l'infini, i.e., le complémentaire d'un compact. Ce n'est pas restrictif pour notre propos. En effet, si X est la fermeture d'un domaine de Jordan, il n'y a pas d'hypothèse supplémentaire, et si $X = \mathbb{R}^n$, nous travaillons sur des images constantes sur un voisinage de l'infini, de telle sorte que ses ensembles de niveau non bornés contiennent un voisinage de l'infini en commun. Dans ce cas, ce voisinage contient une bande $B = \{\mathbf{x} : a \leq \|\mathbf{x}\| \leq b\}$ d'aire t ; si nous substituons à un F_n non borné l'ensemble $F_n \cap \{\mathbf{x} : \|\mathbf{x}\| \leq b\}$, la composante connexe non bornée de F_n est la composante de $F_n \cap \{\mathbf{x} : \|\mathbf{x}\| \leq b\}$ qui rencontre $\{\mathbf{x} : \|\mathbf{x}\| = b\}$, union $\{\mathbf{x} : \|\mathbf{x}\| > b\}$. Cette composante de $F_n \cap \{\mathbf{x} : \|\mathbf{x}\| \leq b\}$ est connexe et d'aire plus grande que t , donc dans ce qui suit, nous pouvons supposer que tous les F_n sont compacts.

Le lemme suivant sera utilisé plusieurs fois :

Lemme 4.4 *Si $F \subset X$ est compact et $(C_i)_{i \in I}$ sont ses composantes connexes, alors*

$$T_t(F) = \bigcup_{i \in I} T_t(C_i).$$

$\forall i \in I$, $T_t(C_i)$ est soit vide soit un continu et les composantes connexes de $T_t(F)$ sont les $T_t(C_i)$ non vides.

Preuve.

1. Comme T_t est monotone et $\forall i \in I, C_i \subset F$ nous avons

$$\bigcup_{i \in I} T_t(C_i) \subset T_t(F).$$

2. Réciproquement, si $\mathbf{x} \in F$, nous avons par la Définition 4.1 un $i \in I$ tel que \mathbf{x} est dans $\text{sat}(C_i)$, dont l'aire est plus grande que t , mais pas dans un trou d'aire plus grande que t de C_i . Donc $\mathbf{x} \in T_t(C_i)$.

3. Soit $i, j \in I, i \neq j$. Supposons que $T_t(C_i) \cap T_t(C_j) \neq \emptyset$. Ceci implique $\text{sat}(C_i) \cap \text{sat}(C_j) \neq \emptyset$. Si nous considérons la fonction caractéristique de F , $u = \chi_F$, $\text{sat}(C_i)$ et $\text{sat}(C_j)$ sont des formes de χ_F , donc elles sont emboîtées. Supposons par exemple $\text{sat}(C_i) \subsetneq \text{sat}(C_j)$. Soit $\mathbf{x} \in \partial \text{sat}(C_i)$. D'après le Lemme 2.36, il existe une forme $\text{sat}(\text{cc}([u < 1], \mathbf{y}))$ telle que

$$\mathbf{x} \in \text{sat}(\text{cc}([u < 1], \mathbf{y})) \subsetneq \text{sat}(C_j).$$

Comme $\partial \text{sat}(C_i)$ est connexe et dans F , il est inclus dans un trou de $\text{cc}([u < 1], \mathbf{y})$, donc sa saturation, qui est $\text{sat}(C_i)$, est incluse dans le même trou. D'autre part, $\text{cc}([u < 1], \mathbf{y})$ est une composante connexe de $X \setminus F$, donc inclus dans un trou H de $\text{sat}(C_j)$. Cela donne : $T_t(C_i) \subset \text{sat}(C_i) \subset H$ et $\mu(H) > \mu(\text{sat}(C_i)) \geq t$, car $H \setminus \text{sat}(C_i)$ est ouvert, donc contient une boule ouverte dont l'aire est non nulle. Donc $T_t(C_i) \cap T_t(C_j) = \emptyset$, contrairement à l'hypothèse.

4. $T_t(C_i)$ est fermé car égal à C_i (ou \emptyset si $\mu(C_i) \leq t$) moins une union d'ouverts. Etant inclus dans un compact ($\text{sat}(F)$), c'est un compact. $T_t(C_i)$ est la réunion des C_i et de quelques composantes connexes de leur complémentaire (celles d'aire plus petite que t). Donc il est connexe.

5. Considérons un sous-ensemble $J \subset I$ tel que $\forall j \in J, T_t(C_j) \neq \emptyset$. Nous prouvons que si le cardinal de J est au moins 2, $D = \bigcup_{j \in J} T_t(C_j)$ n'est pas connexe. Comme $D' = \bigcup_{j \in J} C_j$ n'est pas connexe, il existe un ensemble ouvert et fermé E' dans D' différent de \emptyset et D' . E' est l'union de C_j (puisque chacun est connexe), donc écrivons

$$E' = \bigcup_{k \in K} C_k,$$

avec $\emptyset \neq K \subsetneq J$. Soit

$$E = \bigcup_{k \in K} T_t(C_k).$$

Nous montrons que E est ouvert et fermé dans D , ceci prouvera que D n'est pas connexe, et donc que les composantes connexes de $T_t(F)$ sont les $T_t(C_i)$ non vides.

6. Il est clair que $\emptyset \neq E \subsetneq D$. Il existe un ouvert U' de X tel que $U' \cap D' = E'$. Soit $U = U'' \cup E$, où U'' est l'union des composantes connexes de U' qui rencontrent un C_k , $k \in K$. Alors U est ouvert dans X , puisqu'il peut être écrit comme U'' , qui est une union d'ouverts, à savoir des composantes connexes de U' (car X est localement connexe), union des trous des C_k , qui sont tous ouverts. De plus, si $j \in J \setminus K$, nous avons $T_t(C_j) \cap E = \emptyset$, et aussi $T_t(C_j) \cap U'' = \emptyset$: supposons que ce n'est pas le cas,

$$T_t(C_j) \cap U'' \neq \emptyset. \quad (*)$$

Il existe une composante connexe O de U' rencontrant un C_k , $k \in K$ (par définition de U''), et C_j . En effet, si elle rencontre un trou H d'aire au plus t de C_j , H ne contient pas C_k , sinon comme $H \setminus \text{sat}(C_k)$ est ouvert et non vide, donc de mesure non nulle, cela impliquerait $\mu(H) > \mu(\text{sat}(C_k)) \geq t$. Donc O est connexe et rencontre deux composantes connexes différentes de $X \setminus C_j$ (H et celle contenant C_k), donc $O \cap C_j \neq \emptyset$. Ceci implique $U' \cap C_j \neq \emptyset$, contredisant le fait que $U' \cap D' = E'$. Donc (*) est faux, et nous concluons $T_t(C_j) \cap U'' = \emptyset$, et donc $U'' \cap (D \setminus E) = \emptyset$, prouvant $U \cap D = E$, donc que E est ouvert dans D .

7. L'application du même argument à $D \setminus E$ à la place de D prouve que E est fermé dans D . \square

Continuons la démonstration de la semicontinuité de T_t . Soit $\mathbf{x} \in \bigcap_n T_t(F_n)$. Pour chaque n , \mathbf{x} appartient à quelque (unique) $\text{sat}(C_n)$, où C_n est une composante connexe de F_n , $\mu(\text{sat}(C_n)) \geq t$, et pas à un trou de C_n d'aire plus grande que t . Pour $n \geq 1$, C_n est inclus dans une composante connexe C de F_{n-1} et $T_t(C_n) \subset T_t(C)$. Donc $T_t(C) \cap T_t(C_{n-1}) \neq \emptyset$ et d'après le Lemme 4.4, nous obtenons $C = C_n$, et donc la suite de continus $(C_n)_{n \in \mathbb{N}}$ est décroissante.

Donc leur intersection C est un continu d'après le théorème de Zoratti. Nous affirmons que $\mathbf{x} \in T_t(C)$. En effet, si \mathbf{x} appartient à C , c'est évident, et si \mathbf{x} appartient à un trou H de C , il y a un N tel que \mathbf{x} n'appartient pas à C_n pour tout $n \geq N$. Si pour un tel n_0 , \mathbf{x} appartenait à l'extérieur de C_{n_0} , puisque l'extérieur de C contient l'extérieur de C_{n_0} , il serait dans l'extérieur de C . Donc \mathbf{x} appartient à une suite de trous H_n dans C_n pour $n \geq N$. Nous montrons maintenant que $H = \bigcup_{n \geq N} H_n$. Comme $C \subset C_n$, nous obtenons $H_n \subset H$, de telle sorte que

$$H \supset \bigcup_{n \geq N} H_n.$$

Soit $\mathbf{y} \in \partial \bigcup_{n \geq N} H_n$, U un voisinage de \mathbf{y} et V un voisinage connexe de U tel que $\bar{V} \subset U$ et que \bar{V} soit compact. Expriment que $\mathbf{y} \in \overline{\bigcup_{n \geq N} H_n}$, nous avons :

$$\exists n_0 \geq N, \quad H_n \cap V \neq \emptyset$$

et par monotonie des H_n ,

$$\forall n \geq n_0, \quad H_n \cap V \neq \emptyset. \quad (*)$$

Ecrivons que $\mathbf{y} \in \overline{\bigcap_{n \geq n_0} X \setminus H_n}$,

$$\forall n \geq n_0, \quad V \cap X \setminus H_n \neq \emptyset. \quad (**)$$

Des assertions (*) et (**), la connexité de V donne

$$\forall n \geq n_0, \quad V \cap \partial H_n \neq \emptyset.$$

Comme $\partial H_n \subset C_n$, nous obtenons $V \cap C_n \neq \emptyset$ et donc $\bar{V} \cap C_n \neq \emptyset$ pour $n \geq n_0$. Comme $\bar{V} \cap C_n$ est fermé dans le compact \bar{V} , c'est un compact, et par complétude de X , nous obtenons $\bar{V} \cap C \neq \emptyset$, ainsi $U \cap C \neq \emptyset$. Puisque cela vaut pour tout voisinage de \mathbf{y} , $\mathbf{y} \in \bar{C} = C$. Cela prouve que

$$\partial \bigcup_{n \geq N} H_n \subset C$$

et comme $C \cap H = \emptyset$, $(\partial \bigcup_{n \geq N} H_n) \cap H = \emptyset$. Donc $\bigcup_{n \geq N} H_n$ est fermé dans H et comme c'est un ouvert dans H (union d'ouverts), la connexité de H implique

$$H = \bigcup_{n \geq N} H_n.$$

Du fait que $\mathbf{x} \in T_t(C_n)$, il s'ensuit que $\mu(H_n) \leq t$, et donc $\mu(H) = \mu(\bigcup_{n \geq N} H_n) \leq t$, de telle sorte que $\mathbf{x} \in T_t(C)$, et comme $C \subset F$, nous avons $\mathbf{x} \in T_t(F)$.

\mathcal{B}_t est une base de $\{F : \mathbf{O} \in T_t(F)\}$

Cela signifie que chaque élément de \mathcal{B}_t est dans $\{F : \mathbf{O} \in T_t(F)\}$ et que chaque élément de $\{F : \mathbf{O} \in T_t(F)\}$ contient un élément de \mathcal{B}_t . La première assertion est une conséquence directe de la définition de T_t , et la seconde vient du fait que si $\mathbf{O} \in T_t(F)$, il y a une composante connexe C de F contenant \mathbf{O} ou \mathbf{O} est dans un

trou de C d'aire au plus t , exprimant que $C \in \mathcal{B}_t$.

Liens entre opérateurs ensembliste et fonctionnel

Si nous notons $\mathcal{F}_t = \{F : \mathbf{O} \in T_t(F)\}$, les deux premiers points montrent, grâce au théorème de Maragos :

$$\{\mathbf{x} \in X, \sup_{B \in \mathcal{F}_t} \inf_{\mathbf{y} \in \mathbf{x}+B} u(\mathbf{y}) \geq \lambda\} = T_t([u \geq \lambda])$$

et comme $\mathcal{F}_t \supset \mathcal{B}_t$,

$$\sup_{B \in \mathcal{F}_t} \inf_{\mathbf{y} \in \mathbf{x}+B} u(\mathbf{y}) \geq T_t u$$

et si $B \in \mathcal{F}_t$, $\exists B' \in \mathcal{B}_t$ tel que $B' \subset B$, nous avons

$$\inf_{\mathbf{y} \in \mathbf{x}+B} u(\mathbf{y}) \leq \inf_{\mathbf{y} \in \mathbf{x}+B'} u(\mathbf{y})$$

et le membre de droite est au plus $T_t u(\mathbf{x})$, de telle sorte que

$$\sup_{B \in \mathcal{F}_t} \inf_{\mathbf{y} \in \mathbf{x}+B} u(\mathbf{y}) = T_t u(\mathbf{x}).$$

Cela montre que l'ensemble de niveau supérieur λ de $T_t u$ est l'image par l'opérateur ensembliste T_t de l'ensemble de niveau supérieur λ de u .

4.2.3 Propriétés

Dans cette section, nous examinons les propriétés du filtre de grain défini dans l'Equation (4.1).

Monotone

C'est le principe de comparaison globale bien connu : si $u \leq v$, $T_t u \leq T_t v$. Cela vient facilement de la croissance de T_t :

$$[T_t u \geq \lambda] = T_t([u \geq \lambda]) \subset T_t([v \geq \lambda]) = [T_t v \geq \lambda].$$

Invariance par changement de contraste

Si g est une fonction réelle croissante (au sens large) et semicontinue supérieurement,

$$\forall t \quad g \circ T_t = T_t \circ g.$$

Notons que g n'a pas besoin d'être strictement croissante, c'est-à-dire que des zones plates sont autorisées. C'est l'invariance par changement de contraste au sens fort : g n'est pas forcément continue, ni strictement croissante.

Cette propriété est la conséquence directe de l'égalité déjà démontrée

$$\forall \lambda, \quad [T_t u \geq \lambda] = T_t([u \geq \lambda]),$$

puisque

$$[g \circ T_t u \geq \lambda] = [T_t u \geq g^{(-1)}(\lambda)] = T_t([u \geq g^{(-1)}(\lambda)]) = T_t([g \circ u \geq \lambda]) = [T_t \circ g u \geq \lambda].$$

Invariance par transformation spéciale affine

L'opérateur commute avec les transformations spéciales affines. Cela signifie que si A est une transformation affine de déterminant 1, c'est-à-dire une transformation affine conservant l'aire,

$$T_t(u \circ A) = (T_t u) \circ A.$$

Cela vient du fait que si L est la partie linéaire de A , L est continue et conserve l'aire, de telle sorte qu'elle transforme un ensemble connexe en un ensemble connexe de la même aire. Par conséquent,

$$B \in \mathcal{B}_t \Leftrightarrow L(B) \in \mathcal{B}_t$$

et S est une forme de u si et seulement si $L(S)$ est une forme de $u \circ L$.

Nous pouvons aussi étendre cette propriété pour traiter de transformations affines inversibles, mais ne conservant pas forcément l'aire. Si A est une telle transformation affine, un ensemble connexe d'aire a devient un ensemble connexe d'aire $a \times |\det A|$. Cela donne facilement :

$$\forall A \in \text{GA}(\mathbb{R}^n), \quad T_t(u \circ A) = (T_{t/|\det A|} u) \circ A.$$

Cette propriété pourrait probablement se généraliser à des transformations plus générales : supposons que nous remplaçons la mesure μ par une application croissante Φ des sous-ensembles de X (ordonnés partiellement par inclusion) dans \mathbb{R} avec quelque régularité : Si les X_n sont croissants, $\Phi(\bigcup X_n) = \sup_n \Phi(X_n)$; si nous avons un groupe de transformations continues G de X telles que $\Phi(G(C)) = \Phi(C)$ pour tout $C \subset X$, alors T_t serait invariant sous toute transformation de G . Nous pourrions par exemple imaginer d'étendre cette propriété aux transformations pro-

jectives, bien que cela demande de trouver un Φ croissant. L'existence d'un tel Φ est une question ouverte.

Pas de régularité

Cela signifie que le filtre de grain n'est pas gouverné par une EDP, ou en d'autres termes que nous ne pouvons pas parler de l'évolution infinitésimale de u par T_t . Selon la classification d'Alvarez *et al.* dans [3], il n'y a qu'un espace-échelle régulier, morphologique et invariant par transformation spéciale affine : l'AMSS (Affine Morphological Scale-Space). La propriété que le filtre de grains ne partage pas avec l'AMSS est la régularité. En fait, si nous pouvions parler d'une évolution asymptotique par T_t , elle serait gouvernée par l'équation triviale

$$\frac{\partial u}{\partial t} = 0.$$

En effet, supposons que u est C^2 , et \mathbf{x} un point où $\nabla u(\mathbf{x}) \neq 0$. Alors

$$\exists t > 0, \quad \forall h \leq t, \quad T_h u(\mathbf{x}) = u(\mathbf{x}).$$

Comme u est C^2 , le gradient de u ne s'annule pas sur un voisinage U de \mathbf{x} . Donc l'ensemble $U \setminus [u = u(\mathbf{x})]$ a deux composantes connexes, $U \cap [u > u(\mathbf{x})]$ et $U \cap [u < u(\mathbf{x})]$, éventuellement après réduction de l'ouvert U . Ces deux composantes connexes sont ouvertes, donc de mesure non nulle, notons t le minimum de ces deux aires, moins un petit $\epsilon > 0$. Donc $\text{cc}([u \geq u(\mathbf{x})], \mathbf{x})$ est d'aire au moins t , de telle sorte que $\mathbf{x} \in [T_h u \geq \lambda]$. Par ailleurs, \mathbf{x} n'appartient pas à $[T_h u \geq \mu]$ pour $\mu > \lambda$, parce que s'il est dans un trou d'une composante connexe de $[u \geq \mu]$, ce trou contiendrait $U \cap [u < \lambda]$, donc serait d'aire au moins t .

Causalité

L'espace-échelle basé sur le filtre de grain est causal. Cela signifie que chaque échelle peut être déduite de toute échelle inférieure par un opérateur de transition. Cela veut également dire qu'aucune information n'est introduite, donnant un espace-échelle, comme l'illustre le schéma classique de la Figure 4.2 :

$$\forall s, t, \quad s \leq t, \quad \exists T_{t,s} \text{ t.q. } T_t = T_{t,s} \circ T_s.$$

Dans ce cas, l'opérateur de transition a une forme très particulière, c'est l'opérateur lui-même : $T_{t,s} = T_t$.

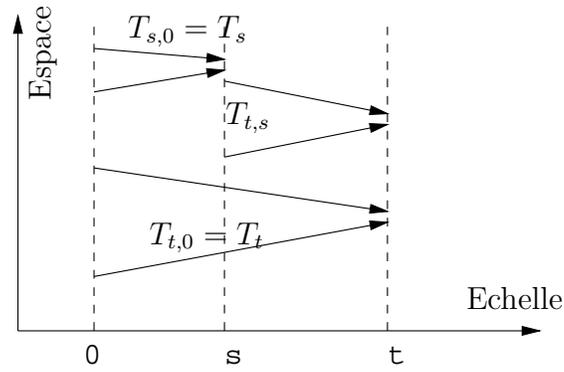


FIG. 4.2 – Schéma classique de l'espace-échelle provenant d'un filtre causal. Chaque ligne verticale représente l'image (l'espace) à une certaine échelle. L'image à l'échelle s (resp. t) s'obtient de l'image à l'échelle 0 par l'opérateur T_s (resp. T_t). L'image à l'échelle t peut aussi être déduite de l'image à l'échelle s par l'opérateur de transition $T_{t,s}$.

Cela revient à montrer que pour tout λ ,

$$[T_t u \geq \lambda] = [T_t \circ T_s u \geq \lambda]$$

ou de façon équivalente

$$T_t([u \geq \lambda]) = T_t \circ T_s([u \geq \lambda]).$$

Séparons les composantes connexes de $[u \geq \lambda]$ en trois parties :

$$[u \geq \lambda] = \bigcup_{i \in I_t} C_i \cup \bigcup_{i \in I_s} C_i \cup \bigcup_{i \in I_0} C_i$$

où $\mu(\text{sat}(C_i)) \geq t$ si $i \in I_t$, $s \leq \mu(\text{sat}(C_i)) < t$ si $i \in I_s$, et $\mu(\text{sat}(C_i)) < s$ si $i \in I_0$. Grâce au Lemme 4.4, nous pouvons écrire :

$$\begin{aligned} T_s([u \geq \lambda]) = & \bigcup_{i \in I_t} \{ \text{sat}(C_i) \setminus \bigcup H, H \text{ trou de } C_i, \mu(H) \leq s \} \\ & \cup \bigcup_{i \in I_s} \{ \text{sat}(C_i) \setminus \bigcup H, H \text{ trou de } C_i, \mu(H) \leq s \}, \end{aligned}$$

les termes de ces unions étant les composantes connexes. A nouveau grâce au Lemme 4.4, l'effet de T_t sur cet ensemble est l'union des images par T_t de chaque composante.

Version light : certaines figures sont absentes

Si $i \in I_s$, nous obtenons

$$\text{sat}(\text{sat}(C_i) \setminus \bigcup H) \subset \text{sat}(\text{sat}(C_i)) = \text{sat}(C_i),$$

de telle sorte que sa mesure est inférieure à t , et

$$T_t(\text{sat}(\text{sat}(C_i) \setminus \bigcup H)) = \emptyset.$$

Si $i \in I_t$, nous avons

$$\text{sat}(\text{sat}(C_i) \setminus \bigcup_{\mu(H) > s} H) = \text{sat}(C_i)$$

et donc d'aire au moins t . Les trous de $\text{sat}(C_i) \setminus \bigcup H$ sont précisément les H , trous de C_i , d'aire au moins s . Ceux qui sont remplis par T_t sont ceux d'aire au moins t . Donc

$$T_t(\text{sat}(C_i) \setminus \bigcup_{\mu(H) > s} H) = T_t(\text{sat}(C_i) \setminus \bigcup_{\mu(H) > t} H)$$

et en prenant l'union sur tous les $i \in I_t$, nous avons :

$$T_t([u \geq \lambda]) = T_t \circ T_s([u \geq \lambda]),$$

le résultat annoncé.

Idempotence

Une conséquence de la forme particulière de l'opérateur de transition est, en prenant $s = t$, $T_{t,t} = T_t$, de telle sorte que

$$T_t = T_t \circ T_t$$

i.e., T_t est idempotent.

Cela distingue fortement le filtre de grain (cette propriété est aussi valide pour l'ouverture et la fermeture d'aire) de tout espace-échelle régulier : dans ce dernier cas, l'itération du filtre simplifie d'autant plus l'image et est équivalente à appliquer le filtre à une échelle plus grande (structure de semi-groupe), alors que dans notre cas, cela n'a aucun effet.

Autodualité

L'avantage le plus intéressant du filtre de grain sur les autres filtres morphologiques est peut-être son autodualité. Cette propriété est très importante : elle signifie que les objets sombres sont traités de la même manière que les objets clairs. Ceci peut s'exprimer de deux manières équivalentes : si u est une fonction continue,

$$T_t(-u) = -T_t(u) \quad (4.3)$$

ou de façon équivalente

$$\sup_{B \in \mathcal{B}_t} \inf_{\mathbf{y} \in B} u(\mathbf{y}) = \inf_{B \in \mathcal{B}_t} \sup_{\mathbf{y} \in B} u(\mathbf{y}). \quad (4.3')$$

C'est à la fois une ouverture et une fermeture. Que ces propriétés sont équivalentes est facile à montrer :

$$\begin{aligned} (4.3) &\Leftrightarrow \sup_{B \in \mathcal{B}_t} \inf_{\mathbf{y} \in B} (-u(\mathbf{y})) = - \sup_{B \in \mathcal{B}_t} \inf_{\mathbf{y} \in B} u(\mathbf{y}) \\ &\Leftrightarrow - \inf_{B \in \mathcal{B}_t} \sup_{\mathbf{y} \in B} u(\mathbf{y}) \\ &\Leftrightarrow (4.3'). \end{aligned}$$

Remarque 4.1. Cette autodualité n'est pas valable pour l'ouverture et la fermeture d'aire : la fermeture d'aire agit sur les maxima, alors que l'ouverture d'aire agit sur les minima. Pour agir sur les deux, Vincent propose d'alterner la fermeture d'aire C_t et l'ouverture d'aire O_t , c'est-à-dire d'appliquer $C_t \circ O_t$ ou $O_t \circ C_t$. Malheureusement, ces deux opérateurs sont différents, comme le montre la Figure 4.3.

Nous allons prouver l'Equation (4.3). Nous transformons cette équation en celle équivalente :

Lemme 4.5 *L'Equation (4.3) est équivalente à*

$$\forall \lambda \in \mathbb{R}, \quad T_t([u \leq \lambda]) = [T_t u \leq \lambda] \quad (4.4)$$

Preuve. En écrivant l'Equation (4.4) en $-\lambda$ à la place de λ , nous avons

$$\forall \lambda \in \mathbb{R}, \quad T_t([-u \geq \lambda]) = [-T_t u \geq \lambda],$$

et de par l'invariance par changement de contraste de T_t :

$$\forall \lambda \in \mathbb{R}, \quad ([T_t(-u) \geq \lambda]) = [-T_t u \geq \lambda].$$

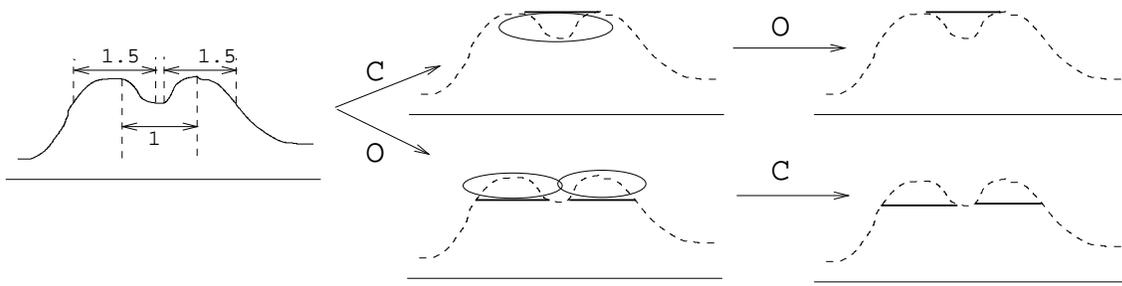


FIG. 4.3 – Le résultat de l’alternance de la fermeture et de l’ouverture d’aire sur une fonction dépend de l’ordre dans lequel nous appliquons les opérateurs. Gauche : fonction originale. Haut : nous appliquons d’abord l’ouverture d’aire O_2 puis la fermeture d’aire C_2 . Bas : nous appliquons d’abord la fermeture d’aire C_2 puis l’ouverture d’aire O_2 . Les zones où la fonction devient constante sont entourées.

C’est-à-dire que les ensembles de niveau supérieurs de $T_t(-u)$ et de $-T_t u$ sont les mêmes, ce qui revient à l’Equation (4.3). \square

Considérons $\lambda, \mu \in \mathbb{R}$ tels que $\lambda < \mu$.

Evidemment, $[u \leq \lambda] \cap [u \geq \mu] = \emptyset$. Si C et C' sont des composantes connexes de, respectivement, $[u \leq \lambda]$ et $[u \geq \mu]$, grâce au Lemme 2.21, $\text{sat}(C)$ et $\text{sat}(C')$ sont soit emboîtés soit disjoints. Considérons le cas où ils sont emboîtés.

Supposons d’abord que $\text{sat}(C) \subset \text{sat}(C')$. Alors C est dans un trou H de C' . Si $\mu(\text{sat}(C)) \geq t$, nous avons $\mu(H) \geq t$, et comme C est fermé (rappelons que u est supposée *continue*), $H \setminus C$ est ouvert et non vide, donc contient une boule, dont l’aire est strictement positive. Ceci implique $\mu(H) > t$.

Supposons maintenant que $\text{sat}(C') \subset \text{sat}(C)$. De la même manière que ci-dessus, si $\mu(\text{sat}(C')) \geq t$, il est inclus dans un trou H de C et $\mu(H) > t$.

Nous déduisons de ces considérations

$$T_t([u \leq \lambda]) \cap T_t([u \geq \mu]) = \emptyset.$$

Comme $T_t([u \geq \mu]) = [T_t u \geq \mu]$, en prenant la réunion sur tous les $\mu > \lambda$, nous déduisons

$$T_t([u \leq \lambda]) \cap [T_t u > \lambda] = \emptyset$$

et donc $T_t([u \leq \lambda]) \subset [T_t u \leq \lambda]$.

Il reste à montrer l’inclusion inverse, qui demande plus de travail. Elle peut

s'écrire

$$(X \setminus T_t([u \leq \lambda])) \cap [T_t u \leq \lambda] = \emptyset$$

ou encore

$$T_t([u \leq \lambda]) \cup \bigcup_{n \geq 1} [T_t u \geq \lambda_n] = X,$$

avec λ_n une suite décroissante de limite λ , et comme $[T_t u \geq \lambda_n] = T_t([u \geq \lambda_n])$, cela s'écrit :

$$\forall \mathbf{x} \in X, \quad \forall n \geq 1, \mathbf{x} \notin T_t([u \geq \lambda_n]) \Rightarrow \mathbf{x} \in T_t([u \leq \lambda]). \quad (4.5)$$

Nous allons prouver cette assertion, dont l'argument principal repose dans le lemme suivant :

Lemme 4.6

$$\forall \lambda \in \mathbb{R}, \forall t > 0, \quad X = T_t([u \leq \lambda]) \cup T_t([u \geq \lambda]).$$

Preuve. Soit $\mathbf{x} \in X$.

1. Selon le Théoreme 2.39, toutes les formes au niveau λ de X ayant une aire au moins t ont une intersection S qui est aussi une forme au niveau λ . D'après la remarque suivant la preuve du Théoreme 2.39, si S est de type inférieur, S est d'aire au moins t . Si S est de type supérieur, il peut s'écrire comme une intersection de formes supérieures, et grâce au théorème de Lindelöf, comme intersection d'une suite décroissante de tels éléments, donc son aire est au moins t .

2. Supposons que S est de type supérieur et que \mathbf{x} appartient à un trou H de la composante connexe C d'ensemble de niveau supérieur sur laquelle S repose. Alors nous pouvons écrire

$$H = \bigcup_{G \in G_{\lambda, \mathbf{x}}, G \subset H} G. \quad (*)$$

En effet, que H contienne le membre de droite H' est évident. Si cette inclusion est stricte, comme H est connexe et H' est ouvert (union d'ouverts), H' n'est pas fermé dans H , donc $H \cap \partial H' \neq \emptyset$. Soit \mathbf{y} dans cet ensemble. Dans tout voisinage connexe U de \mathbf{y} il y a des points de $[u \geq \lambda]$, sinon U serait dans une composante connexe de $[u < \lambda]$ et comme il existe $G \in G_{\lambda, \mathbf{x}}$, $G \subset H$ rencontrant U , nous aurions $G \cup H$ connexe, ce qui contredirait le fait que G est une forme. Donc \mathbf{y} est le point limite d'une suite dans $[u \geq \lambda]$, donc $u(\mathbf{y}) \geq \lambda$. D'après le Lemme 2.36, il existe une forme inférieure G au niveau λ contenant \mathbf{y} dans un trou et contenue dans H . Comme G est ouvert, G contient un voisinage de \mathbf{y} , donc rencontre un $G' \in G_{\lambda, \mathbf{x}}$ contenu dans H . Donc G et G' sont emboîtés, entraînant $G \supset G'$. Donc $G \in G_{\lambda, \mathbf{x}}$, impliquant

$G \subset H'$, contrairement aux hypothèses. Cela montre la validité de l'Equation (*). Comme H est une union d'ouverts, grâce au théorème de Lindelöf, il peut s'écrire comme l'union d'un nombre au plus dénombrable d'entre eux. Donc nous avons $\mu(H) \leq t$. Cela montre que $\mathbf{x} \in T_t(S) \subset T_t([u \geq \lambda])$.

3. Si $S = \text{sat}(C)$ est de type inférieur et \mathbf{x} appartient à un trou H de C , alors H est une forme supérieure. Par définition de S , nous devons avoir $\mu(H) \leq t$. Donc, \bar{C} est connexe, par continuité de u , $\bar{C} \subset [u \leq \lambda]$ et $\text{sat}(\bar{C}) \supset S \ni \mathbf{x}$. De plus, si \mathbf{x} est dans un trou de $\text{sat}(\bar{C})$, ce trou est inclus dans H , donc son aire est au plus t . Cela montre que $\mathbf{x} \in T_t(\bar{C}) \subset T_t([u \leq \lambda])$. \square

Pour prouver l'Equation (4.5), prenons $\mathbf{x} \in X$ tel que pour tout n , $\mathbf{x} \notin T_t([u \geq \lambda_n])$. D'après le Lemme 4.6, nous avons $\mathbf{x} \in \bigcap_n T_t([u \leq \lambda_n])$. Avec la semicontinuité supérieure de T_t , nous obtenons

$$\mathbf{x} \in T_t\left(\bigcap_n [u \leq \lambda_n]\right) = T_t([u \leq \lambda]).$$

Préservation de la continuité

Si u est continue, alors $T_t u$ l'est aussi. En effet

$$[T_t u \geq \lambda] = T_t([u \geq \lambda])$$

montrant que $[T_t u \geq \lambda]$ est fermé, donc $T_t u$ est semicontinue supérieurement et de la même manière,

$$[T_t u \leq \lambda] = T_t([u \leq \lambda])$$

prouvant que $[T_t u \leq \lambda]$ est fermé, et $T_t u$ est aussi semicontinue inférieurement.

Remarque 4.2. Cela devient faux si $X = \mathbb{R}^n$ et que l'image n'est pas constante dans un voisinage de l'infini. Considérons la fonction distance f à l'ensemble fermé $C_\infty \cup C_1 \cup C_2 \cup \dots$, défini par :

$$\begin{aligned} C_n &= [-n, 0] \times \left\{-\frac{1}{n}\right\} \cup \{-n\} \times \left[-\frac{1}{n}, 2\right] \cup \left[-n - \frac{1}{2}, -n + \frac{1}{2}\right] \times [2, 3] \\ C_\infty &= \mathbb{R} \times \{0\}. \end{aligned} \quad (4.6)$$

Alors pour tout n , nous avons $\mu(C_n) = 1$ alors que $\mu(C_\infty) = 0$. Pour tout $t < 1$,

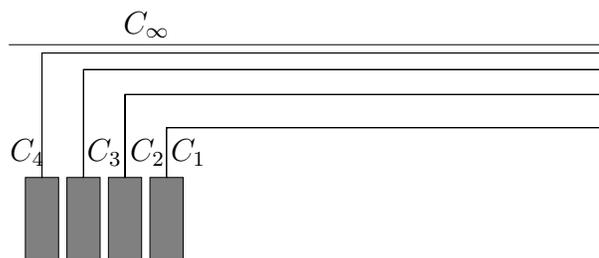


FIG. 4.4 – La fonction distance à l'ensemble $C_\infty \cup \bigcup C_n$, défini dans l'Equation 4.6, ne reste pas continue sous l'action de T_t .

nous avons :

$$T_t(\{f \leq 0\}) = T_t(C_\infty \cup \bigcup_n C_n) = \bigcup_n C_n,$$

qui n'est pas fermé, donc $T_t f$ n'est pas semicontinue inférieurement (voir la Figure 4.4).

Bien que T_t préserve la semicontinuité et la continuité, il ne préserve pas la différentiabilité. En fait, un image différentiable devient simplement continue par le filtre de grain. Cela distingue fortement un tel filtre d'un filtre régulier : il ne lisse pas l'image, et donc ne permet pas de calculer des dérivées. A la place, il sélectionne l'information dans l'image originale qui est considérée comme utile.

Conservation des jonctions en **T**

Alors que le filtre de grain ne lisse pas les lignes de niveau de l'image, il conserve des jonctions en **T**, en enlève d'autres, mais ne les détruit pas. Les jonctions en **T** sont les points limites de la partie commune de deux lignes de niveau (donc une image continue ne présente pas de jonction en **T**, puisque les lignes de niveau sont des ensembles isoniveaux, donc disjointes). Ce sont des indices forts d'une occlusion, considérée comme l'opération fondamentale sur les images dans [9]. Comme les jonctions en **T** sont détruites par les espaces-échelles réguliers, des auteurs ont presque considéré que l'espace-échelle est une perturbation en analyse d'image, voir [10].

4.2.4 Expériences

Calcul

Il est bien sûr hors de question de parcourir toutes les composantes connexes dont la saturation rencontre un pixel donné P et est d'aire suffisante, pour calculer $T_t u(P)$ comme dans la définition, par un inf sup. A la place, nous avons un moyen

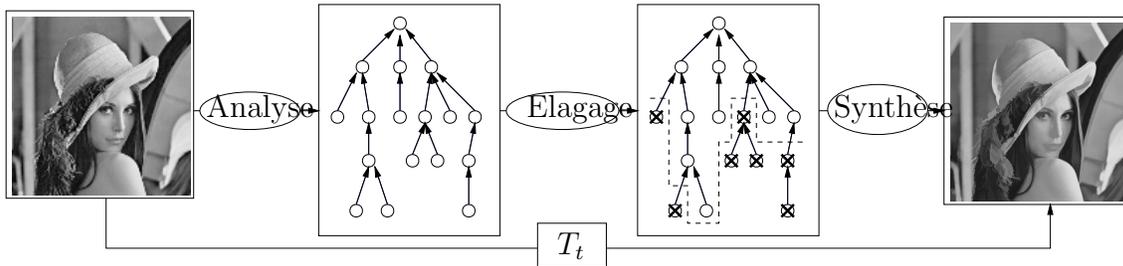


FIG. 4.5 – Illustration de l'élagage d'arbre correspondant à l'image filtrée par le filtre de grain $T_t u$. De gauche à droite : l'image originale u ; analyse de u par la FLST, donnant son arbre de formes (illustration schématique) ; l'élagage, où tous les nœuds correspondant à des formes d'aire inférieure à t sont supprimées ; synthèse de l'arbre élagué, par l'algorithme de reconstruction 1, donnant l'image filtrée par le filtre de grain $T_t u$.

facile de le calculer pourvu que nous soyons capable d'extraire l'arbre d'inclusion de l'image u . En effet, le filtre de grain correspond à un élagage de l'arbre. Nous enlevons de l'arbre toutes les formes d'aire inférieure à t et nous reconstruisons l'image. Notons que si une forme S doit être enlevée, toutes les formes du sous-arbre de racine S le doivent aussi, car leur aire est aussi inférieure à t . C'est la propriété définissant l'élagage d'arbre. Ceci est illustré dans la Figure 4.5, où l'analyse et la synthèse sont la FLST et sa reconstruction associée.

Mais une implémentation plus efficace du filtre de grain peut être réalisée. Elle consiste en une légère modification de la FLST de manière à calculer directement le filtre de grain. En effet, la FLST travaille en supprimant successivement des formes dans l'image u et en les stockant. Cela se fait dans un mode de croissance de région. C'est-à-dire que les parents sont supprimés après leurs enfants. C'est exactement ce que fait le filtre de grain. La seule procédure à modifier est `ExtractionBranche` (voir l'Algorithme 4) pour ajouter une autre condition de rupture :

```

if  $\#R \geq t$  then
    Fin ← vrai
end if

```

Nous pouvons aussi nous passer du stockage de la forme. Alors l'image u après avoir appliqué l'algorithme modifié ainsi est $T_t u$.

Si nous voulons calculer directement l'arbre d'inclusion de image filtrée par le filtre de grain, $T_t u$, au lieu de calculer l'arbre complet puis de supprimer des formes, il suffit de modifier la procédure originale `ExtractionBranche` de l'Algorithme 4

pour ajouter une condition avant le stockage de la forme dans l'arbre : que son aire soit au moins t .

Exemples

Nous montrons dans la Figure 4.6 l'image Lenna à diverses échelles et dans la Figure 4.7 les lignes de niveau correspondantes. Notons que les lignes de niveau restantes ne bougent pas.

Nous montrons dans les Figures 4.8 et 4.9 les images d'un paysage à diverses échelles, et dans les Figures 4.10 et 4.11 leurs lignes de niveau.

Débruitage

Les filtres morphologiques sont particulièrement adaptés au bruit impulsionnel. Dans notre cas, le bruit impulsionnel produit probablement des petites formes, donc pour s'en débarrasser, l'application du filtre de grain à une échelle suffisante serait une bonne solution. La Figure 4.12 montre le filtre de grain appliqué à une image avec des niveaux variés de bruit impulsionnel. Remarquons que la plus grande partie du bruit semble supprimée, mais que les lignes de niveau sont bruitées. C'est inévitable avec un tel filtre, puisque les lignes de niveau de l'image filtrée sont présentes dans l'image originale. Néanmoins, à une échelle suffisamment large, le bruit semble avoir disparu.

D'autres filtres basés sur la taille des grains existent et devraient être comparés au filtre de grain. Par exemple, l'ouverture et la fermeture d'aire supposent un ordre entre eux. L'absence d'autodualité est un problème pour les images texturées en particulier. Une autre possibilité est un filtre médian suivi d'un leveling. Le problème avec cette stratégie est que le filtre de grain détruirait les jonctions en T. Enfin, un élagage de l'arbre de partition binaire, tel qu'exposé dans [76], donnerait peut-être de bons résultats, mais suppose de définir précisément le critère de fusion de régions, qui devrait être de préférence invariant par changement de contraste.

4.3 Quantification adaptative

4.3.1 Description

Bien qu'il soit garanti que le nombre de formes dans l'image ne peut dépasser le nombre de pixels, ce nombre peut tout de même être trop élevé pour des algorithmes complexes d'analyse d'image. Typiquement, le nombre de formes peut atteindre

FIG. 4.6 – Espace-échelle de l'image Lenna (256×256) induit par le filtre de grain. Les échelles (c'est-à-dire l'aire minimum des formes conservées) sont de gauche à droite et de haut en bas : 1 (i.e., l'image originale), 5, 25, 125, 625 et 3125.

Version *light* : certaines figures sont absentes

FIG. 4.7 – Lignes de niveau des images de la Figure 4.6, c'est-à-dire des images Lenna aux échelles 1, 5, 25, 125, 625 et 3125. Les lignes de niveau sont tracées après quantification des images tous les 4 niveaux de gris.

Version *light* : certaines figures sont absentes

FIG. 4.8 – Espace-échelle d’une image (630×350). De haut en bas : échelles 1, 5 et 25.

Version *light* : certaines figures sont absentes

FIG. 4.9 – (suite de la Figure 4.8) De haut en bas : échelles 125, 625 et 3125.

Version *light* : certaines figures sont absentes

FIG. 4.10 – Lignes de niveau des images de la Figure 4.8 après une quantification tous les 4 niveaux de gris.

FIG. 4.11 – Lignes de niveau des images de la Figure 4.9 après une quantification tous les 4 niveaux de gris.

Version *light* : certaines figures sont absentes

FIG. 4.12 – Suppression de bruit impulsionnel par le filtre de grain. Les trois colonnes montrent l'espace-échelle dérivé du filtre de grain de l'image originale (première colonne), de l'image avec un bruit impulsionnel de 10% (deuxième colonne) et 20% (troisième colonne). Les échelles, de la première ligne à la cinquième, sont respectivement 1, 5, 10, 20 et 40.

la moitié du nombre de pixels pour une image texturée. Nous voudrions rejeter une partie de l'information jugée non pertinente. Un bon candidat pour cela est l'ensemble des petites formes. Les enlever de l'image, c'est-à-dire appliquer le filtre de grain, est efficace, comme nous l'avons montré, mais peut s'avérer insuffisant, comme les expériences de la section suivante le prouvent. Nous voudrions réduire encore ce nombre.

Une solution simple serait de quantifier l'image : cela réduit le nombre de niveaux de gris dans l'image, et peut-être le nombre de formes dans la même proportion. Toutefois, une quantification aveugle peut enlever des traits importants dans l'image. Nous voudrions une quantification adaptée au contenu de l'image. Dans [24], Froment propose de ne garder que les lignes de niveau ayant le nombre maximum de jonctions en T et rapporte de très bons résultats de compression. L'idée sous-jacente est que même pour des objets n'en occluant aucun autre, le fond n'est jamais uniforme dans les images naturelles : il y a presque toujours des gradients de réflexion lumineuse, des petites taches dans le fond, etc. Néanmoins, il est clair que certaines lignes importantes dans l'image peuvent ne pas avoir de nombreuses jonctions en T. L'approche que nous proposons ici est liée à celle-là, dans le sens que le critère de suppression repose sur des observations heuristiques sur les images naturelles¹.

4.3.2 Dégradés

La lentille de la caméra lisse l'image avant le processus d'échantillonnage. C'est pourquoi ce qui devrait normalement apparaître comme des discontinuités bien nettes dans l'image présente un dégradé (voir la Figure 4.13). L'effet d'un dégradé est une accumulation de formes emboîtées proches. L'idée de notre quantification adaptative est de garder seulement une forme pour représenter ce dégradé.

La forme que nous gardons dans le dégradé peut être arbitraire, mais du principe que les objets dans le monde où nous vivons sont plutôt réguliers, une bonne forme à garder serait la plus régulière. Notre critère de régularité est L^2/S où L est la longueur de la frontière de la forme et S son aire. La forme dans le dégradé ayant le L^2/S minimal est conservée, les autres formes sont supprimées. De plus, pour assurer que la forme reste visible, nous changeons son niveau de gris à la moyenne des niveaux de gris des pixels dans la gradation. Cette moyenne des niveaux de gris est présente uniquement dans un but de confort visuel.

Nous devons préciser la signification du terme dégradé. Considérons la relation

¹Nous appelons images naturelles les clichés d'une caméra, analogique ou numérique, par opposition aux images synthétiques.

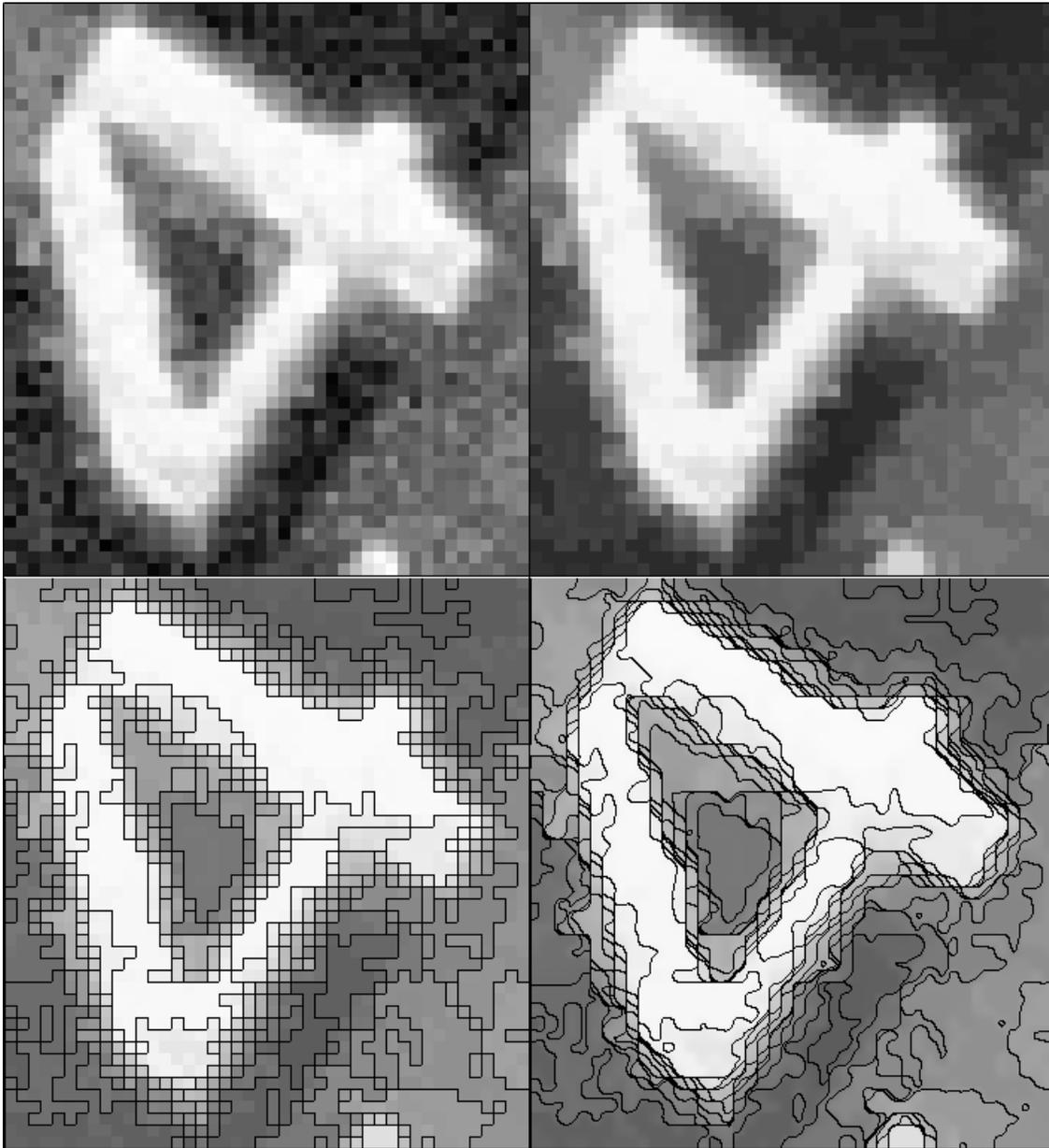


FIG. 4.13 – Les objets visuels présentent un dégradé à leurs frontières dans les images naturelles. Haut : détail agrandi (50×50) d'une image (gauche) et la même partie de l'image après un filtre de grain d'aire 20 pixels (droite). Bas : quelques-unes de ses lignes de niveau (gauche) et les mêmes lignes après lissage avec l'AMSS, de telle sorte à mieux les distinguer (droite). Noter l'accumulation de lignes de niveau à la frontière de la forme significative.

suivante \simeq entre deux formes S_1 et S_2 :

- S_1 est le parent de S_2 ;
- S_1 n'a pas d'autre fils que S_2 ;
- S_1 et S_2 sont du même type.

Ces conditions semblent raisonnables pour établir que S_1 et S_2 sont dans le même dégradé. La deuxième condition empêche une forme qui contient plusieurs fils d'être dans le même dégradé que l'un d'eux. Nous pourrions aussi ajouter une autre condition : que l'aire de S_1 et l'aire de S_2 ne diffèrent pas plus qu'un certain pourcentage, pour assurer que les formes sont vraiment proches. Ce pourcentage de tolérance deviendrait donc un paramètre de la méthode. Bien sûr, les trois (ou quatre) conditions ci-dessus ne définissent pas une relation d'équivalence, il n'est néanmoins pas difficile de se rendre compte que si $S_1 \simeq S_2 \simeq \dots \simeq S_n$ et $S'_1 \simeq S'_2 \simeq \dots \simeq S'_m$ avec $S_n = S'_1$, alors

$$S_1 \simeq \dots \simeq S_n = S'_1 \simeq \dots \simeq S'_m.$$

Donc les suites maximales de formes en relation par \simeq sont disjointes. Nous appellerons dégradés ces suites maximales et les formes n'appartenant pas à une telle suite maximale. Nous verrons qu'il est toutefois utile d'isoler en tant qu'un dégradé en soi les formes qui ont au moins un enfant d'un type différent (voir la Figure 4.14). Dans un dégradé, la forme la plus régulière sera appelée la forme représentative du dégradé.

4.3.3 Suppression d'un nœud

Maintenant que la définition d'un dégradé est établie, nous expliquons comment nous pouvons supprimer le nœud correspondant à une forme non représentative de son dégradé. Enlever le nœud revient simplement à reconnecter les enfants à son parent. C'est ce qu'illustre la Figure 4.15. Pourvu que le nœud supprimé ne soit pas la racine, la structure résultante reste un arbre d'inclusion.

Nous devons aussi préciser ce qui arrive aux pixels d'une forme S dont le nœud correspondant est supprimé. Le plus simple est de convenir que la plus petite forme les contenant est le parent P de S . Avec cette convention, il est très facile de manipuler l'arbre correspondant : nous ajoutons un champ booléen attribué à chaque nœud, **supprimé**, disant si le nœud est considéré comme supprimé ou non. La seule chose à faire pour supprimer un nœud dans l'arbre est de l'enregistrer dans le champ **supprimé** pourvu que nous respections les conventions suivantes : le parent d'une forme S est le parent de S dans l'arbre original pourvu que celui-ci ne soit pas supprimé, ou le parent de celui-ci s'il est supprimé, et ainsi de suite (nous remontons

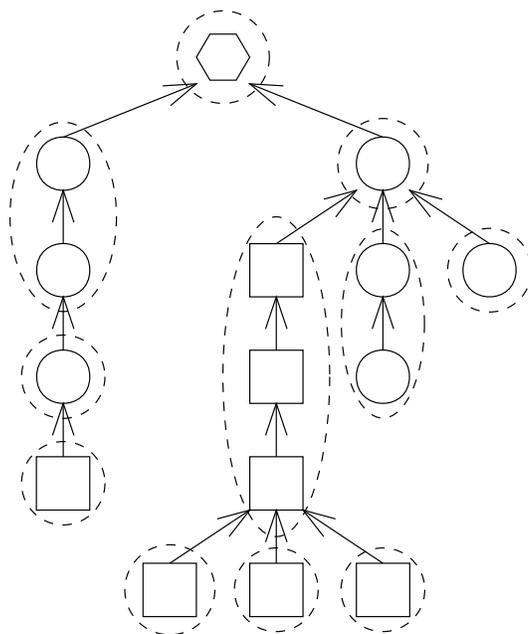


FIG. 4.14 – Partition d'un arbre d'inclusion en dégradés. La racine est représentée par un pentagone, les formes de type inférieur par un cercle et les formes de type supérieur par un carré. Les dégradés sont représentés par des ellipses en pointillé.

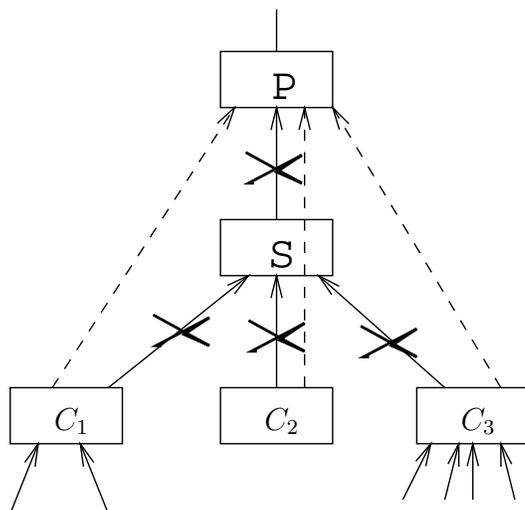


FIG. 4.15 – Enlever un nœud S de l'arbre correspond à l'opération suivante : reconnecter ses enfants, C_1 , C_2 et C_3 , à son parent P .

dans l'arbre à partir de S jusqu'à rencontrer un nœud non supprimé) et la plus petite forme associée au pixel est la plus petite forme le contenant dans l'arbre original si celle-ci n'est pas supprimée, ou le premier ancêtre non supprimé. Notons qu'il est légèrement plus difficile de trouver les enfants d'une forme dans l'arbre ainsi modifié, car nous avons plusieurs ramifications à explorer jusqu'à rencontrer des nœuds non supprimés.

La question fondamentale concernant la suppression d'un nœud dans l'arbre d'inclusion est : l'arbre modifié est-il l'arbre d'inclusion de l'image reconstruite ? L'image reconstruite s'obtient comme d'habitude : le niveau de gris du pixel est le niveau de gris associé à la plus petite forme le contenant. La Figure 4.16 montre que nous ne pouvons pas supprimer tout nœud si nous voulons remplir cette condition. Les effets dramatiques provoqués par la suppression de cette forme sont dus au fait que cette forme a un enfant d'un type différent. Même si le parent P de la forme S était du même type que S , nous devrions comparer le niveau de gris de P à celui de l'enfant de S pour assurer la condition de consistance.

Au contraire, il est sûr de supprimer une feuille de l'arbre. Plus généralement, les nœuds suivants sont garantis supprimables :

1. un nœud sans enfant ;
2. Un nœud S avec enfants du même type que S .

Les conditions sont suffisantes, mais pas nécessaires, pour qu'une forme soit supprimable. Toutefois, cela suffit pour être sûr que la suppression de toutes les formes sauf une d'un dégradé est sûr : en effet, si un dégradé est fait de plusieurs formes, la plus petite n'a pas d'enfant, ou des enfants du même type ; les autres formes du dégradé ont un enfant unique, qui est dans le dégradé, donc qui est du même type. De même, prendre la moyenne des niveaux de gris des pixels dont la plus petite forme est dans le dégradé et l'attribuer à la forme représentative du dégradé est sûr : il est compris entre les niveaux de gris de la plus petite et de la plus grande forme du dégradé, donc il ne peut y avoir d'inversion de contraste.

4.3.4 Expériences

Les expériences concernant le filtre de grain montrent que l'information visuellement importante reste présente dans le filtre de grain à petite échelle. La Figure 4.17 montre que le nombre de formes dans l'image Lenna décroît fortement pour de petites aires. Même à petite échelle, l'application du filtre de grain réduit d'un montant considérable le nombre de formes. Par exemple, à l'échelle 3 pixels, presque la moitié des formes disparaissent.

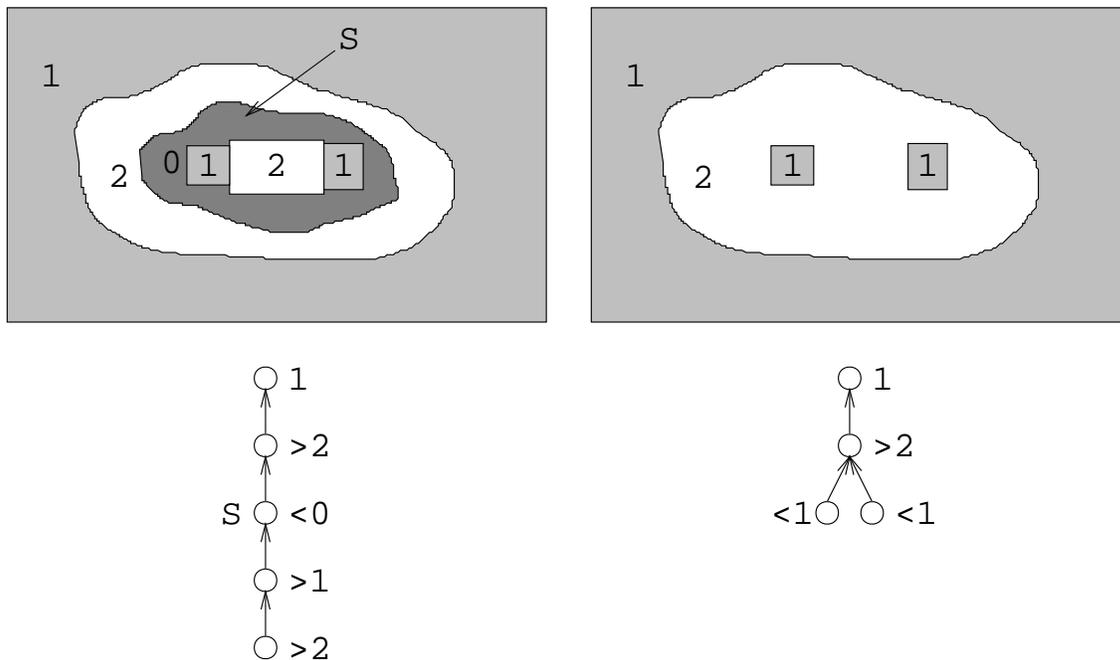


FIG. 4.16 – La suppression aveugle d'un nœud dans l'arbre peut donner un arbre qui n'est pas l'arbre d'inclusion de son image reconstruite. Gauche : l'image originale et son arbre, dont nous enlevons le nœud S . Droite : l'image reconstruite après la suppression du nœud S et son arbre d'inclusion associé. Notons que la suppression de S a des effets dramatiques sur l'arbre de l'image reconstruite : cela supprime une forme au niveau 2, coupe la forme supérieure au niveau 1 en deux formes et ces formes deviennent de type inférieur.

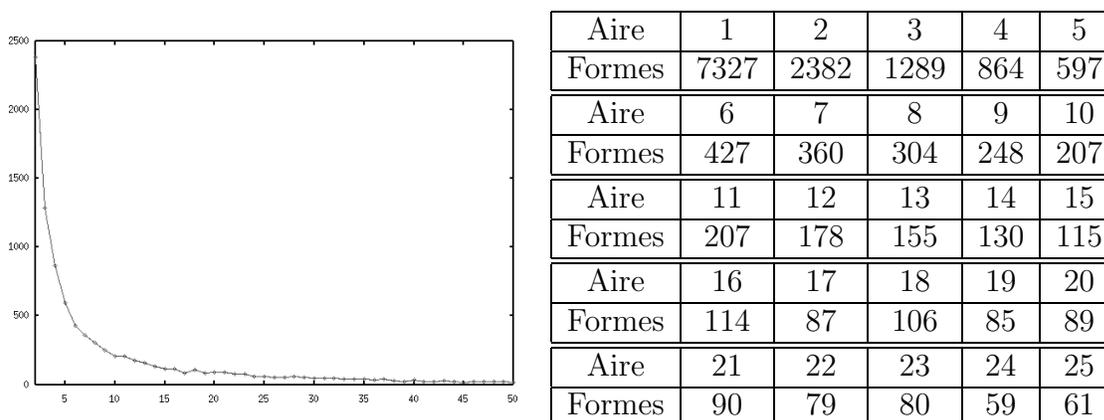


FIG. 4.17 – Nombre de formes de taille donnée dans l’image Lenna (256×256). Seules les aires de 1 à 50 apparaissent. Le nombre total de formes dans l’image est 19 324.

Nous pouvons observer le même phénomène pour une image hautement texturée, dont les statistiques apparaissent dans la Figure 4.18.

La réduction du nombre de formes induite par la quantification adaptative est modeste si le filtre de grain n’a pas été appliqué avant. La raison en est que les petites formes (de 1 ou 2 pixels) sont très fréquentes dans l’image, de telle sorte que les formes ayant un seul enfant sont rares, car la plupart des formes ont un enfant de petite aire. Donc le nombre de dégradés dans l’image n’est pas beaucoup plus bas que le nombre de formes, expliquant la mauvaise performance de cette quantification en termes de compression. Toutefois, dès que le filtre de grain est appliqué avant, le nombre de formes peut décroître fortement.

La Figure 4.19 montre le nombre de formes restantes dans l’image Lenna après filtre de grain suivi de quantification adaptative, et la Figure 4.20 quelques images correspondantes. Notons que l’image quantifiée est une version sommaire de l’image filtrée par le filtre de grain, mais que peu de détails manquent (sauf les dégradés), bien que la quantité d’information soit beaucoup plus faible. Les Figures 4.21 et 4.22 montrent le même résultat pour une image de la “Vue de Delft” de Vermeer.

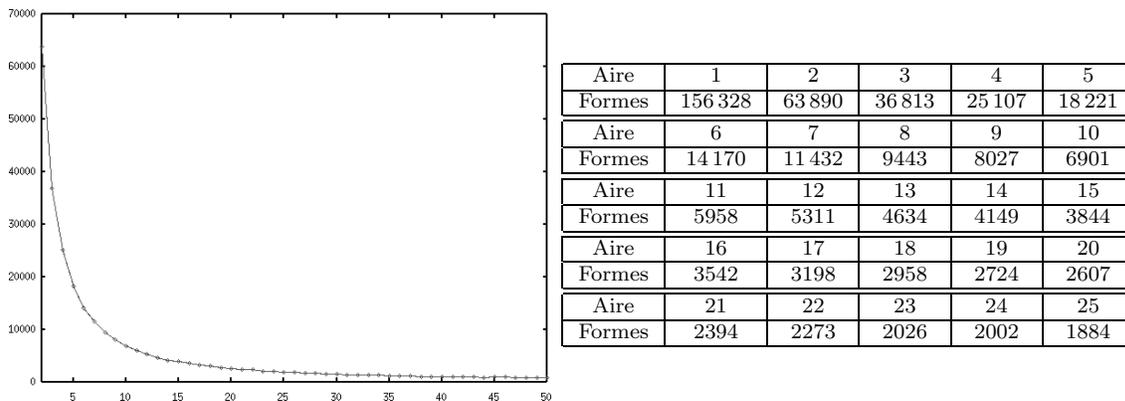


FIG. 4.18 – Le nombre de formes d’aire donnée dans l’image du tapis (715×1024). Seules les aires de 1 à 50 apparaissent. Le nombre total de formes dans l’image est 484 961.

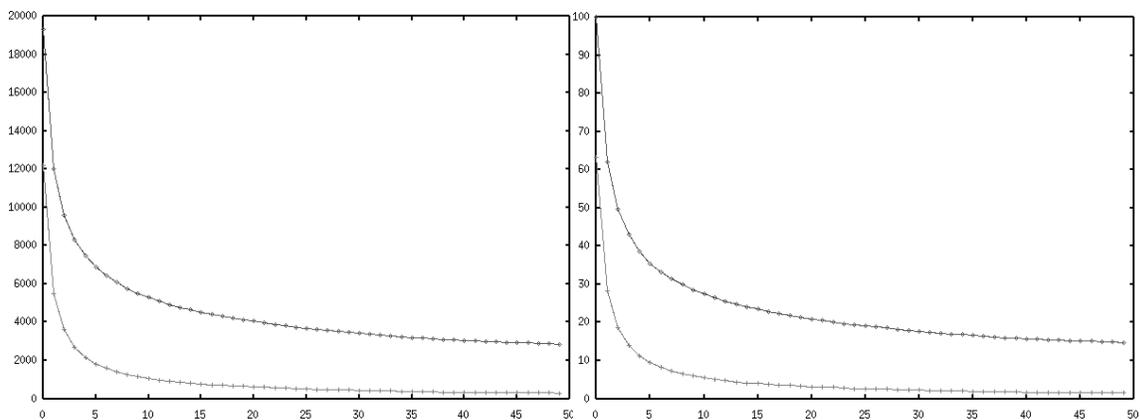


FIG. 4.19 – Décroissance du nombre de formes avec l’image Lenna (256×256) après le filtre de grain suivi d’une quantification adaptative, par rapport à l’aire du filtre de grain. Gauche : le nombre absolu de formes, après application du filtre de grain (courbe sombre), ou filtre de grain suivi de quantification adaptative (courbe claire). Droite : les mêmes, en termes de pourcentages de formes restantes relativement au nombre initial de formes.

FIG. 4.20 – Filtre de grain (gauche) suivi de quantification adaptative (droite) de l'image Lenna (256×256). Les échelles du filtre de grain sont, de haut en bas, 5, 10 et 20.

Version *light* : certaines figures sont absentes

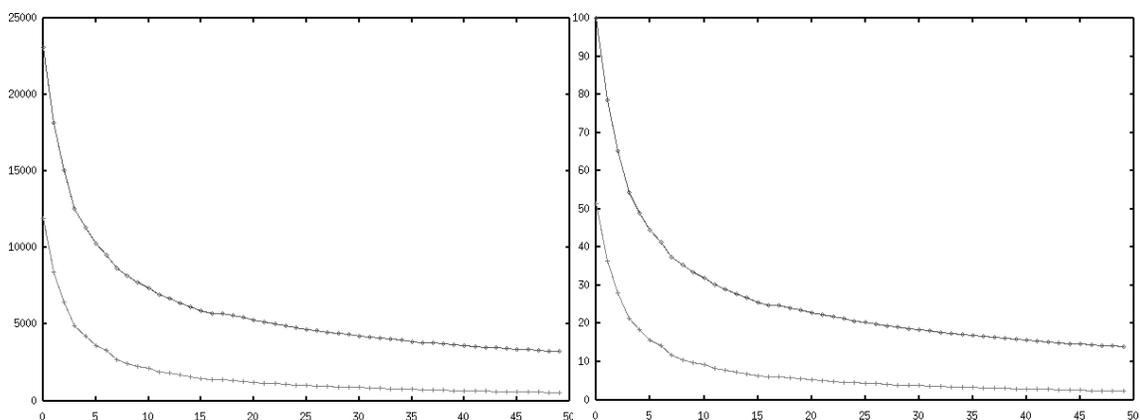


FIG. 4.21 – Décroissance du nombre de formes avec l'image de Delft (464×384) après le filtre de grain suivi par la quantification adaptative, par rapport à l'aire du filtre de grain. Gauche : le nombre absolu de formes, après application du filtre de grain (courbe sombre), ou filtre de grain suivi de quantification adaptative (courbe claire). Droite : les mêmes, en termes de pourcentages de formes restantes relativement au nombre initial de formes.

FIG. 4.22 – Filtre de grain (gauche) suivi de quantification adaptative (droite) de l'image de Delft (256×256). Les échelles du filtre de grain sont, de haut en bas, 5, 10 et 20.

Version *light* : certaines figures sont absentes

Deuxième partie

Recalage d'images

Chapitre 5

Introduction

5.1 Généralités

Le problème du recalage d'images 2D peut se poser de la façon suivante : étant données deux images qui diffèrent globalement par une transformation géométrique, comment retrouver cette transformation. La transformation est dans une classe d'applications géométriques dépendant d'un petit nombre de paramètres. Les plus classiques sont : translation (deux paramètres), rotation-translation (trois paramètres, translation plus angle de rotation) et similitude (quatre paramètres, angle de rotation, facteur de zoom et translation). La rotation-translation permet de retrouver la pose d'un objet plat lorsque la distance de la caméra à l'objet reste constante, la caméra pointant perpendiculairement au plan de l'objet, et la similitude quand la distance de l'objet à la caméra peut changer, mais la caméra reste pointée perpendiculairement au plan de l'objet.

Lorsque la direction de la caméra change, ces modèles sont insuffisants et une transformation affine (six paramètres) est nécessaire, pourvu que la caméra soit loin (comparée à la distance focale) de l'objet, et sinon, une projection perspective complète (huit paramètres) est nécessaire.

Si l'objet observé n'est pas plat, le changement ne peut être modélisé seulement par une transformation géométrique avec peu de paramètres (sauf quand la forme de l'objet lui-même peut se modéliser avec peu de paramètres), et nous sortons du domaine du recalage d'images pour entrer dans celui de la stéréovision, voir Faugeras [19].

Etant donné le petit nombre de paramètres décrivant la transformation géométrique, ce problème est à première vue facile à régler, mais comme souvent en traitement d'image, la modélisation du problème ne correspond que de loin aux

conditions expérimentales. Outre le fait que les images observées sont échantillonnées (et souvent pas à la fréquence de Nyquist) et des versions quantifiées des images analogiques (ce qui n'est pas un problème négligeable), et outre la présence de bruit d'observation, les conditions d'observation sont souvent les suivantes :

1. les conditions d'éclairage peuvent avoir changé entre les deux photos, provoquant un changement de contraste global, voire local ;
2. les images observées se composent en général de plusieurs objets, certains d'entre eux n'obéissant pas au mouvement global, conduisant en particulier à des occlusions.

Il existe *grosso modo* deux sortes de techniques de recalage, nous renvoyons à la revue de Brown [7] et aux références s'y trouvant. La plupart des techniques reposent sur une corrélation globale ou locale, et conduisent à des calculs rapides dans le domaine de Fourier, comme par exemple DeCastro et Morandi [17], ou Yaroslavsky et Eden [94, chapter 9]. Un deuxième type de méthode (comme Wang et Bhattacharya [91]) calcule d'abord des traits significatifs de l'image, qui doivent être suffisamment stables pour apparaître dans les deux images.

5.2 Méthodes de corrélation

Les méthodes de corrélation recouvrent en fait une grande classe de possibilités, partageant la caractéristique commune de reposer sur une mesure de différence entre les images. Supposons que nous avons une mesure donnant la distance entre deux images. Alors le déplacement entre les images est celui qui minimise cette distance (qui l'annule, dans le cas idéal). Le problème est donc la minimisation d'une fonction dépendant des paramètres du déplacement.

La distance utilisée la plus classique est donnée par la norme L^2 . La distance L^2 entre u et v peut être développée en la somme des normes de u et v moins deux fois le produit croisé $u \cdot v$. Comme les normes de u et v ne dépendent pas des paramètres du déplacement (et sont égales dans le cas idéal), grâce à un changement de variable dans les intégrales, le problème est la maximisation du terme

$$\arg \max_A \int u \circ A(\mathbf{x})v(\mathbf{x}) d\mathbf{x}$$

où A appartient à la classe des déplacements admissibles. Cette quantité à maximiser est la corrélation de $u \circ A$ et v . Des variantes de ceci sont classiques, comme de prendre la corrélation de $u \circ A - \int u$ et $v - \int v$ pour être insensible à un décalage global du

contraste, ou d'utiliser une mesure autre que la mesure de Lebesgue dans l'intégrale, de manière à mettre en valeur certaines parties de l'image par exemple.

Si nous nous restreignons aux translations, il est bien connu que les deux paramètres de la translation apparaissent dans la phase de la transformée de Fourier, et donc peuvent facilement être isolés ; si $v(\mathbf{x}) = u(\mathbf{x} - \mathbf{t})$, le théorème de la Translation de Fourier donne la relation suivante entre leurs transformées de Fourier, voir DeCastro et Morandi [17] par exemple :

$$\mathcal{F}v(\xi) = e^{-i\xi \cdot \mathbf{t}} \mathcal{F}u(\xi),$$

où $\mathcal{F}u(\xi) = \int u(\mathbf{x})e^{-i\mathbf{x} \cdot \xi} d\mathbf{x}$. Le vecteur de translation inconnu \mathbf{t} peut être isolé par la formule :

$$e^{-i\xi \cdot \mathbf{t}} = \frac{\overline{\mathcal{F}u(\xi)}\mathcal{F}v(\xi)}{|\mathcal{F}u(\xi)|^2}$$

et

$$\mathcal{F}^{-1} \left(\frac{\overline{\mathcal{F}u(\xi)}\mathcal{F}v(\xi)}{|\mathcal{F}u(\xi)|^2} \right) = \delta_{\mathbf{t}}.$$

Ceci représente un image uniformément à 0, exceptée en \mathbf{t} , où elle doit être 1. En pratique, les coordonnées du maximum de cette image sont prises comme le vecteur de translation. Ceci requiert seulement trois transformées de Fourier, en faisant un algorithme de choix quand le temps de calcul compte.

La technique Fourier peut aussi traiter d'un angle de rotation et d'un facteur de zoom, c'est-à-dire retrouver une similitude, comme l'expliquent Reddy et Chatterji [65]. Donnons-en un aperçu pour le cas de la rotation-translation : $v(\mathbf{x}) = u(R_{-\theta}\mathbf{x} - \mathbf{t})$.

Prendre le module de la transformée de Fourier de chaque membre donne :

$$|\mathcal{F}v(\xi)| = |\mathcal{F}u(R_{-\theta}\xi)|$$

c'est-à-dire les modules des transformées de Fourier subissent la même rotation. Par ce procédé, la translation n'apparaît que dans la phase, donc disparaît quand nous prenons le module. Donc le centre de rotation est maintenant l'origine dans le domaine de Fourier et il suffit de prendre une version polaire du module pour transformer cette rotation en translation. S'il y a de plus un facteur de zoom, la même technique est valable en prenant la transformation log-polaire du module des transformées de Fourier. La translation en x dans le log du facteur de zoom et la translation en y l'angle de rotation. Notons toutefois que prendre la transformation polaire ou log-polaire exige une interpolation *dans le domaine de Fourier*. L'inter-

polation à utiliser est loin d'être évidente, et peut affecter le résultat.

Quoi qu'il en soit, qu'elle soit calculée par la transformée de Fourier ou pas, la méthode de corrélation est globale, et s'il y a des mouvements secondaires dans les images, la manière dont ils affectent le résultat est la suivante : le mouvement calculé est une sorte de moyenne de tous les mouvements dans les images. Une façon de l'éviter est de faire de la corrélation locale, c'est-à-dire de calculer la corrélation restreinte à des fenêtres. De plus, un changement de contraste peut modifier le résultat de manière complexe.

Cela fait de la méthode par corrélation un méthode facile mais aveugle : elle peut être sensible à plusieurs facteurs avec des effets inconnus.

5.3 Correspondances de caractéristiques

La méthode par correspondance de caractéristiques est plus complexe à implémenter que la méthode par corrélation, à cause de la variété des caractéristiques possibles et du nombre de moyens de les mettre en correspondance. Les caractéristiques peuvent être des lignes, des bords, des coins, etc. Voir par exemple Wang *et al.*, [90]. La seule contrainte est que ces points soient suffisamment robustes, et leur localisation suffisamment exacte. Ces caractéristiques peuvent être locales (comme les coins, les points d'inflexion de courbes), ou semi-locales (segments).

La corrélation locale peut être considérée dans un certain sens aussi comme une méthode de mise en correspondance de points : un voisinage (une fenêtre) est prise autour de chaque point et la corrélation est restreinte à ce voisinage.

5.4 Aperçu de la méthode

Par contraste avec les méthodes par corrélation locale, la méthode que nous proposons ne fixe pas de voisinage *a priori* pour la mise en correspondance. Les méthodes semi-locales permettent de tirer avantage de caractéristiques suffisamment globales pour une estimation précise du déplacement, mais assez locales pour soulever l'espoir que nombre d'entre elles dans les deux images ne sont pas altérées par occlusion. En effet, nos caractéristiques sont les formes. Elles dépendent de l'image et il y en a de toutes tailles. Nous essayons de mettre en correspondance toute forme avec une autre dans la seconde image. Pour que tout le recalage soit invariant par changement de contraste, il y a deux conditions :

1. les caractéristiques extraites ne dépendent pas du contraste dans l'image ;

2. la mise en correspondance de caractéristiques ne prend pas en compte leur contraste.

La première condition est déjà remplie, puisque les formes ne dépendent pas du contraste. La deuxième nous défend de prendre en compte leur niveau de gris dans la mise en correspondance. Donc, seule l'information géométrique fournie par la forme doit être utilisée. De plus, la correspondance doit se faire indépendamment du déplacement (inconnu), c'est-à-dire doit être indépendante de toutes les transformations dans la classe des déplacements possibles. Des invariants basés sur les moments sont faciles à obtenir pour les classes de déplacements classiques, voir Reiss [66]. Deux formes sont considérées en correspondance si leurs invariants sont proches.

Une fois que nous avons un ensemble de correspondances entre formes entre les deux images, nous les faisons voter pour un déplacement, c'est-à-dire pour les paramètres définissant le mouvement. L'ensemble des paramètres ayant reçu le nombre maximum de votes doit être le déplacement dominant global. Une fois trouvé, le résultat peut être amélioré par un post-traitement : nous pouvons faire une moyenne des déplacements donnés par l'ensemble des correspondances compatibles avec le mouvement dominant, c'est-à-dire les gagnants du vote. Cela assure que les fausses correspondances, ou les correspondances en rapport avec des mouvements secondaires, ne se mélangent pas avec les correspondances dominantes, et si tout va bien la moyenne résultante peut être précise. C'est un avantage du recalage par mise en correspondance de caractéristiques : les caractéristiques peuvent être discriminées en fonction de leur vote.

Chapitre 6

Recalage d'images invariant par changement de contraste

Les parties principales de ce chapitre ont été publiées dans [57].

6.1 Correspondances

6.1.1 Choix des caractéristiques

Comme nous l'avons expliqué dans le chapitre précédent, notre choix est de trouver des correspondances de caractéristiques entre les images. Comme le changement de contraste pouvant arriver entre les images est l'une de nos principales préoccupations, nous devons sélectionner avec soin nos caractéristiques pour qu'elles ne dépendent pas elles-mêmes du contraste. De bons candidats pour cela sont les ensembles de niveau, ou mieux (car moins globales) les composantes connexes des ensembles de niveau ; les lignes de niveaux satisfont également les conditions. Comme dans cette thèse nous avons spécifiquement construit une représentation d'image invariante par changement de contraste, l'arbre d'inclusion des formes, nous sommes tenté de l'utiliser. En fait, au stade actuel, l'information d'inclusion n'est pas utilisée, ou de façon très sommaire. Mais nous devons garder à l'esprit que c'est une information structurante de haut niveau, et qu'elle pourrait améliorer de façon significative l'efficacité du recalage. Quoi qu'il en soit, les caractéristiques que nous utilisons ici sont les formes des images, c'est-à-dire les composantes connexes des ensembles de niveau dont nous remplissons les trous. Comme nous l'avons expliqué dans la première partie de cette thèse, nous avons bon espoir que quelques-unes de ces formes représentent de vrais objets.

6.1.2 Descripteurs

Le succès d'une méthode de recalage d'images reposant sur des correspondances de caractéristiques est fonction de l'invariance dans ces caractéristiques et dans les règles de correspondance. Après avoir choisi des caractéristiques invariantes par changement de contraste, nous essayons de trouver des descripteurs invariants par changement de contraste pour les apparier. Mais ces descripteurs doivent aussi être invariants par rapport à tout déplacement admissible. Cela veut dire que si nous cherchons une similitude, les descripteurs des formes doivent être invariants par similitude.

Nous choisissons des descripteurs globaux et élémentaires des formes : leurs moments. Les moments d'ordre n d'un sous-ensemble S de \mathbb{R}^2 sont :

$$m_{ij} = \iint_S x^i y^j dx dy = \iint_{\mathbb{R}^2} \mathcal{X}_S(x, y) x^i y^j dx dy \quad (6.1)$$

où $i + j = n$. Il y a $n + 1$ moments d'ordre n . Le moment d'ordre 0 est l'aire de S , les moments d'ordre 1 sont $m_{10} = m_{0,0}\bar{x}_S$ et $m_{01} = m_{0,0}\bar{y}_S$, les coordonnées du barycentre S multipliées par son aire.

Remarque 6.1. Comme les moments sont des caractéristiques *additives*, ils peuvent se calculer de façon efficace pour toutes les formes, comme nous l'avons expliqué dans la Section 3.4.2.

Mis à part m_{00} , l'aire de la forme, aucun des moments n'est invariant par rapport à la translation. Pour cela, nous devons utiliser les moments *centraux*, qui sont définis par

$$\mu_{ij} = \iint_S (x - \bar{x}_S)^i (y - \bar{y}_S)^j dx dy = \iint_{\mathbb{R}^2} \mathcal{X}_S(x, y) (x - \bar{x}_S)^i (y - \bar{y}_S)^j dx dy \quad (6.2)$$

Notons que $\mu_{10} = \mu_{01} = 0$. Les moments centraux d'ordre n sont des fonctions polynômiales des moments d'ordre jusqu'à n .

L'invariance par translation ne suffit pas pour les applications que nous visons. Nous aurions besoin au moins d'ajouter l'invariance par rotation, et éventuellement l'invariance par zoom. D'autre part, chaque invariance ajoutée rend certains moments inutilisables. Par exemple, la condition d'invariance par translation annule les moments d'ordre 1, qui deviennent inutiles pour la correspondance. Cet effet nous force à considérer des moments d'ordre supérieur. C'est une limite de cette méthode, puisqu'il est bien connu que les moments d'ordre plus élevé sont plus sensibles au bruit. C'est pourquoi nous nous restreignons à des déplacements aussi simples que la rotation (avec centre inconnu) ou la similitude, et nous ne considérerons pas de

moments d'ordre plus grand que 3.

La matrice d'inertie d'un sous-ensemble S de \mathbb{R}^2 est la matrice symétrique 2×2 des moments d'ordre 2 :

$$I_S = \begin{pmatrix} \mu_{20} & \mu_{11} \\ \mu_{11} & \mu_{02} \end{pmatrix}. \quad (6.3)$$

Si nous considérons la fonction d'inertie $i : \mathbb{S}^1 \rightarrow \mathbb{R}$ qui transforme un angle α en l'intégrale des distances au carré des points de S à la droite passant par le barycentre et d'orientation α , i est une forme quadratique du vecteur $(\cos \alpha, \sin \alpha)^T$, de matrice I_S . Si S subit une rotation d'angle α , la nouvelle matrice d'inertie de S devient :

$$I_{R_\theta S} = R_\theta S R_{-\theta} \quad (6.4)$$

où R_θ est la matrice de rotation d'angle θ . Cela montre que la matrice d'inertie n'est pas un invariant par rotation, mais que les deux nombres $\det I_S$ et $\text{tr } I_S$, respectivement le déterminant et la trace de la matrice d'inertie, le sont.

Si nous admettons d'aller jusqu'à l'ordre 3 pour les moments, deux autres invariants par rotation-translation peuvent être construits :

$$\begin{aligned} r_4 &= (\mu_{30} - 3\mu_{12})^2 + (\mu_{0,3} - 3\mu_{21})^2 \\ r_5 &= (\mu_{30} + \mu_{12})^2 + (\mu_{0,3} + \mu_{21})^2. \end{aligned}$$

Pour résumer, nous associons à chaque forme son vecteur de descripteurs invariants par rotation :

$$\begin{aligned} r &= (r_1, r_2, r_3, r_4, r_5) \\ r_1 &= m_{00} \\ r_2 &= \text{tr } I_S \\ r_3 &= \det I_S \\ r_4 &= (\mu_{30} - 3\mu_{12})^2 + (\mu_{0,3} - 3\mu_{21})^2 \\ r_5 &= (\mu_{30} + \mu_{12})^2 + (\mu_{0,3} + \mu_{21})^2. \end{aligned}$$

Si nous nous attendons à un bruit important dans les images, nous ferions mieux d'abandonner les descripteurs r_4 et r_5 comme ils deviendraient probablement non significatifs.

De plus, si nous nous attendons à un changement d'échelle entre les images, l'aire m_{00} n'est plus un invariant. Néanmoins, toutes ces caractéristiques sont des invariants relatifs, ce qui veut dire que nous pouvons les diviser par l'aire élevée à

une certaine puissance pour obtenir des invariants absolus par rapport à un zoom. Dans un cas de recalage par similitude, le vecteur de descripteurs est :

$$\begin{aligned} s &= (s_1, s_2, s_3, s_4) \\ s_1 &= r_2/m_{00}^2 \\ s_2 &= r_3/m_{00}^4 \\ s_3 &= r_4/m_{00}^5 \\ s_4 &= r_5/m_{00}^5. \end{aligned}$$

A nouveau, s_3 et s_4 devraient n'être utilisés que quand les images ne sont pas trop bruitées. La normalisation par l'aire élevée à une certaine puissance rend probablement ces caractéristiques moins stables que ceux invariants par rotation, donc si nous pensons qu'il n'y a pas de zoom, il est préférable d'utiliser les descripteurs invariants par rotation plutôt que ceux invariants par similitude.

Reiss [66] explique comment construire d'autres invariants à partir de moments d'ordre plus élevé.

6.1.3 Recherche des correspondances

Etant données deux images u_1 et u_2 à recalcr, nous extrayons les formes S_1, \dots, S_k et S'_1, \dots, S'_l , et calculons leur vecteur de descripteurs associé, c_1, \dots, c_k et c'_1, \dots, c'_l , qui peuvent être des invariants par rotation-translation ou par similitude. Puisqu'ils sont composés d'invariants, nous pouvons les comparer. Nous dirions que la forme S_i de u_1 et la forme S'_j de u_2 sont *en correspondance* si leurs vecteurs de descripteurs associés, respectivement $c_i = (c_i^1, \dots, c_i^m)$ et $c'_j = (c'_j^1, \dots, c'_j^m)$ vérifient :

$$\forall p \in \{1, \dots, m\} \quad \frac{1}{t_p} \|c_j^p\| \leq \|c_i^p\| \leq t_p \|c_j^p\|, \quad (6.5)$$

où $t = (t_1, \dots, t_m)$ se compose de seuils de tolérance, $t_p \geq 1$ pour tout p . Plus t_p est proche de 1, moins nous sommes tolérant à propos d'une erreur dans le descripteur c^p .

Si S_i et S'_j se correspondent, nous notons cette correspondance $C = (S_i, S'_j)$.

Remarque 6.2. Il faut noter que l'Equation (6.5) est clairement réflexive et symétrique mais *pas* transitive (sauf si $t = (1, \dots, 1)$), ce qui est ce que nous attendons d'une relation de similitude : toute forme peut se transformer en toute autre forme par un nombre suffisant de petites perturbations, chaque forme intermédiaire étant similaire à la précédente.

6.2 Votes

Une fois que l'ensemble des correspondances $\mathcal{C} = (C_1, \dots, C_q)$ entre les images u_1 et u_2 est établi, nous sommes en mesure de déterminer le déplacement. Toutefois, avec les critères simplistes de correspondance que nous utilisons, nous nous attendons à trouver un grand nombre de correspondances erronées. C'est pourquoi nous ne voudrions pas faire participer toutes les correspondances dans la détermination du déplacement, mais plutôt trouver un moyen de sélection automatique des « bonnes » correspondances.

Un bon moyen pour cela est d'introduire une procédure de vote : les correspondances votent pour un déplacement, c'est-à-dire pour un jeu de paramètres, et le jeu de paramètres ayant reçu le plus grand nombre de votes est notre estimation du déplacement. C'est le principe d'une procédure de vote, mais ce n'est pas exactement ainsi que nous faisons : en fait, chaque correspondance est en théorie suffisante pour déterminer un déplacement, que ce soit une rotation-translation ou une similitude, en comparant les moments (par exemple, le zoom serait le rapport des aires, l'angle de rotation serait l'angle entre les directions des vecteurs propres de la matrice d'inertie, la translation le vecteur déterminé par les barycentres). Mais à nouveau, nous ne faisons pas assez confiance aux valeurs des moments pour estimer directement les paramètres. Comme nous avons dit, ces moments sont juste assez bons pour faire une sélection préliminaire des correspondances. Toutefois, les barycentres, puisque basés sur des moments d'ordre 1, sont probablement assez sûrs. De cette manière, une correspondance est juste suffisante pour estimer une translation, mais pas plus. Au contraire, avec *deux* correspondances, nous avons deux points et leurs images déplacées, donc c'est juste ce dont nous avons besoin pour estimer une similitude, et plus que nécessaire pour estimer une rotation-translation.

Si nous considérons deux correspondances (S_i, S'_j) et $(S_{i'}, S'_{j'})$ de barycentres respectifs $\mathbf{x}_i, \mathbf{y}_j, \mathbf{x}_{i'}$ et $\mathbf{y}_{j'}$, les paramètres de la similarité, s, θ et \mathbf{t} sont donnés par :

$$s = \frac{\|\mathbf{y}_{j'} - \mathbf{y}_j\|}{\|\mathbf{x}_{i'} - \mathbf{x}_i\|} \quad (6.6)$$

$$\sin \theta = \frac{(\mathbf{x}_{i'} - \mathbf{x}_i) \wedge (\mathbf{y}_{j'} - \mathbf{y}_j)}{\|\mathbf{x}_{i'} - \mathbf{x}_i\| \cdot \|\mathbf{y}_{j'} - \mathbf{y}_j\|} \quad (6.6')$$

$$\cos \theta = \frac{(\mathbf{x}_{i'} - \mathbf{x}_i) \cdot (\mathbf{y}_{j'} - \mathbf{y}_j)}{\|\mathbf{x}_{i'} - \mathbf{x}_i\| \cdot \|\mathbf{y}_{j'} - \mathbf{y}_j\|} \quad (6.6'')$$

$$\mathbf{t} = \mathbf{y}_j - sR_\theta \mathbf{x}_i (= \mathbf{y}_{j'} - sR_\theta \mathbf{x}_{i'}) \quad (6.6''')$$

Si nous estimons seulement une rotation-translation, nous devrions vérifier que

$s = 1$, permettant toutefois une certaine tolérance. Si ce n'est pas le cas, les deux correspondances sont considérées incompatibles et elles ne votent pas ensemble. Notons que ces formules impliquent des divisions par les quantités $\|\mathbf{x}_{i'} - \mathbf{x}_i\|$ et $\|\mathbf{y}_{j'} - \mathbf{y}_j\|$, qui peuvent parfois presque s'annuler. Nous devrions donc vérifier qu'elles sont suffisamment grandes, de telle sorte que le vote soit assez précis. Sinon, nous devrions les empêcher de voter. Une autre vérification avant les votes serait de vérifier que les rapports d'aires de formes correspondantes correspond approximativement au facteur de zoom estimé s .

Si ces conditions sont remplies, la paire de correspondances (S_i, S'_j) et $(S_{i'}, S'_{j'})$ est autorisée à voter. Cela signifie qu'un compteur à un emplacement dans un espace de paramètres 3-D (pour la rotation-translation) ou 4-D (pour la similitude) est incrémenté. Toutefois, trouver un maximum dans un tel histogramme peut prendre du temps. Nous nous restreindrions plutôt à des histogrammes 1-D ou 2-D. La partie de translation, t , peut être estimée après s et θ , de telle sorte que nous pouvons séparer l'estimation de la partie linéaire et de la partie affine. Dans une première étape, nous faisons voter toutes les paires de correspondances (en fait seulement celles qui sont compatibles) pour le zoom (dans le cas de la similitude) et la rotation, trouvons le compteur maximum dans cet histogramme, puis dans une seconde étape votons en un histogramme 2-D pour la translation en utilisant la partie linéaire estimée du déplacement. Ainsi, seuls des histogrammes 1-D et 2-D sont utilisés.

Notons que cette procédure de votes a déjà été proposée par Chang *et al.* dans [15].

6.3 Précision

Comme nous travaillons avec des images numériques, les positions des formes sont précises au mieux au pixel près¹. Ceci empêche la procédure de votes de donner une estimation des paramètres meilleure qu'un pixel. Dans quelques applications, c'est loin d'être suffisant. Une précision d'un dixième de pixel ou d'un centième de pixel est nécessaire. Si les erreurs dues à la numérisation sont distribuées suivant une gaussienne, une régression linéaire des paramètres estimés donnerait un résultat plus précis. Dans cette régression, qui est une moyenne de différentes estimations, nous ne devons pas inclure les fausses estimations, dues à des correspondances erronées. Donc, nous l'estimons seulement avec les correspondances qui sont compa-

¹Cette restriction est aussi valable pour les méthodes de corrélation. Pour avoir une meilleure précision, il est nécessaire d'interpoler les images, ou de façon équivalente la surface de corrélation, voir Tian et Huhns [87].

tibles avec le déplacement estimé. Cette estimation « bootstrap » est probablement bien meilleure.

Nous commençons par sélectionner les correspondances qui sont compatibles avec le déplacement gagnant de l'élection. La correspondance (S_i, S'_j) est considérée comme compatible si

$$\|\mathbf{y}_j - sR_\theta \mathbf{x}_i - \mathbf{t}\| \leq \epsilon.$$

Le seuil ϵ peut être estimé à partir de la forme des histogrammes près de leur maximum. Des pics assez forts signifient que les votes sont précis, et ϵ peut être petit, alors que des pics plus arrondis indiquent que nombre de votes ne sont pas précis, et un ϵ plus grand serait mieux adapté. Plus précisément, ce que nous faisons est non seulement de trouver le maximum dans l'histogramme pour chaque paramètre, mais aussi de sélectionner un mode autour de ce maximum, le mode correspondant au plus grand intervalle autour du maximum où l'histogramme reste concave ; alors nous considérons que la correspondance est compatible avec le mouvement s'il y a un jeu de paramètres dans chaque mode tel que \mathbf{x}_i serait transformé exactement en \mathbf{y}_j . Ce moyen de faire évite d'introduire un paramètre supplémentaire dans l'algorithme, ϵ .

Nous notons l'ensemble des correspondances compatibles \mathcal{C}_c . Alors nous voulons résoudre le problème de minimisation suivant :

$$\arg \min_{s, \theta, \mathbf{t}} \sum_{(S_i, S'_j) \in \mathcal{C}_c} \|sR_\theta \mathbf{x}_i + \mathbf{t} - \mathbf{y}_j\|^2. \quad (6.7)$$

Le problème est apparemment non linéaire à cause du fait que θ apparaît sous la forme de $\cos \theta$ et $\sin \theta$, mais nous pouvons changer les variables inconnues pour le rendre linéaire :

$$\arg \min_{s_1, s_2, \mathbf{t}} \sum \left\| \begin{pmatrix} s_1 & -s_2 \\ s_2 & s_1 \end{pmatrix} \mathbf{x}_i + \mathbf{t} - \mathbf{y}_j \right\|^2. \quad (6.8)$$

Notons $\mathbf{S} = (s_1 \ s_2)^T$ et prenons en compte l'égalité

$$\begin{pmatrix} s_1 & -s_2 \\ s_2 & s_1 \end{pmatrix} \mathbf{x}_i = A^{(i)} \mathbf{S} \text{ where } A^{(i)} = \begin{pmatrix} x_i & -y_i \\ y_i & x_i \end{pmatrix},$$

nous pouvons réécrire l'Equation (6.8) en

$$\arg \min_{\mathbf{S}, \mathbf{t}} \sum \|A^{(i)} \mathbf{S} + \mathbf{t} - \mathbf{y}_j\|^2. \quad (6.9)$$

Pour résoudre l'Equation (6.9), nous calculons les dérivées partielles relativement

aux paramètres et les égalons à 0. Cela donne le système

$$\begin{aligned} \sum A^{(i)T} A^{(i)} \mathbf{S} + \left(\sum A^{(i)} \right)^T \mathbf{t} &= \sum A^{(i)T} \mathbf{y}_j \\ \sum A^{(i)} \mathbf{S} + N \mathbf{t} &= \sum \mathbf{y}_j \end{aligned}$$

si N est le nombre de correspondances compatibles avec les modes. Après quelques manipulations algébriques, nous obtenons

$$\left\{ \begin{aligned} \left[\sum A^{(i)T} A^{(i)} - \frac{1}{N} \left(\sum A^{(i)} \right)^T \left(\sum A^{(i)} \right) \right] \mathbf{S} &= \sum A^{(i)T} \mathbf{y}_j \\ &- \frac{1}{N} \left(\sum A^{(i)} \right)^T \left(\sum \mathbf{y}_j \right) \\ \mathbf{t} &= \frac{1}{N} \left[\sum \mathbf{y}_j - \sum A^{(i)} \mathbf{S} \right] \end{aligned} \right.$$

ce qui permet de calculer le vecteur \mathbf{S} puis le vecteur \mathbf{t} .

Pour calculer S , nous devons inverser la matrice 2×2

$$\sum_i A^{(i)T} A^{(i)} - \frac{1}{N} \left(\sum_i A^{(i)} \right)^T \left(\sum_i A^{(i)} \right)$$

qui d'après l'inégalité de Cauchy-Schwarz est singulière si et seulement si tous les $A^{(i)}$ sont proportionnels, c'est-à-dire que tous les \mathbf{x}_i sont alignés.

6.4 Complexité

Si k est le nombre de formes de la première image et l de la deuxième, nous pouvons théoriquement avoir jusqu'à $k.l$ correspondances et la complexité de l'élection est $(k.l)^2$ car nous devons prendre en compte toutes les paires de correspondances. Même si nous supprimons les formes trop petites (moins de 20 pixels par exemple), le nombre de formes dans chaque image peut être de l'ordre de dizaines de milliers. Donc l'élection devient dans ces conditions inabordable. Toutefois, c'est une configuration du pire cas. En général, le nombre de correspondances est bien inférieur à $k.l$; il est heureusement plutôt de l'ordre de grandeur de $\max(k, l)$. Mais même ainsi, l'élection peut être trop gourmande en temps de calcul pour nous.

Pour réduire encore le nombre de correspondances, la solution la plus simple est de quantifier les images avant d'extraire les formes. Le nombre de formes est presque automatiquement divisé par le pas de quantification. Mais cette quantification peut

perdre des objets importants dans l'image. Une meilleure idée est de faire quelque chose de similaire à la quantification adaptative présentée dans la Section 4.3 : regrouper les formes en « objets », liés de près aux branches de l'arbre. En effet, à cause du lissage fait par la lentille de la caméra, des objets dans la scène correspondent souvent à plusieurs formes emboîtées. L'idée est de regrouper ces formes emboîtées dans le même « objet » et faire voter les objets à la place des formes directement, avec un poids correspondant au nombre de formes incluses dans les objets, puisque des objets composés de nombreuses formes représentent probablement des choses importantes dans la scène. Nous ne risquons guère de perturber les histogrammes de votes en empêchant les votes de paires de formes correspondantes dans le même « objet ». En fait, des formes dans le même objet sont très proches, donc leur vote commun donné par les Equations (6.6), (6.6'') et (6.6') serait très imprécis.

6.5 Extensions

6.5.1 Réduction du nombre de correspondances

En fait, le nombre de correspondances de petites formes est énorme et de plus elles ne présentent pas de « forme » caractéristique ! C'est plus particulièrement vrai des formes d'un pixel. Ces formes s'apparient avec toute autre forme de la même taille, et la complexité de l'élection devient beaucoup trop grande. C'est pourquoi elles ne sont pas prises en compte (que les images subissent le filtre de grain avant ou que tout simplement nous ne les incluons pas dans les dictionnaires de formes) lorsque nous cherchons les correspondances. Néanmoins, rien ne les empêche d'être incluses dans l'étape ultérieure de minimisation par les moindres carrés (6.7). En effet, une fois que le déplacement est estimé grossièrement, le nombre de correspondances compatibles de petites formes est grandement réduit, et le résultat de la minimisation des moindres carrés peut être plus précise.

Pourtant, pour l'élection, de complexité N^2 si N est le nombre de votes, N doit ne pas être trop grand pour les calculs soient faisables ; typiquement il ne doit pas dépasser quelques milliers pour que le recalage puisse se faire en quelques secondes. La suppression de petites formes peut être insuffisante pour réduire le nombre de correspondances. Si c'est le cas, une stratégie doit être inventée pour réduire *a priori* leur nombre. Par exemple une contrainte un-vers-un pourrait être ajoutée (c'est l'un de nos axes de recherche actuels).

Un moyen facile de réduire le nombre de correspondances revient à utiliser la même idée que pour la quantification adaptative, rassembler les formes en dégradés.

Toutefois, le nombre de formes dans un dégradé peut être important, donc nous devons le conserver. De plus, un seuil supérieur d'aire entre une forme et ses enfants doit exister, de manière à ne pas rassembler dans le même dégradé des formes d'aires vraiment différentes. Ces remarques impliquent des petites modifications aux notions de dégradé, et pour les distinguer, nous appellerons plutôt *sections* la nouvelle notion. Une section se compose d'un ensemble monotone de formes, son poids est le nombre de formes, et nous pouvons aussi calculer son centre, compris comme le centre de la plus grande forme contenue. Nous voudrions faire voter les correspondances de sections, puisque le nombre de sections est sûrement plus petit de manière significative que le nombre de formes. Pour cela, nous devons définir deux choses : une correspondance de sections et son poids.

Correspondances de sections

La définition d'une correspondance de sections est tout à fait naturelle : nous disons que deux sections \mathcal{S}_1 et \mathcal{S}_2 se correspondent si une forme de \mathcal{S}_1 s'apparie avec au moins une forme de \mathcal{S}_2 . Donc, le nombre de correspondances de sections ne peut dépasser le nombre de correspondances de formes. La construction de ces correspondances de sections est facile : nous ordonnons les correspondances de formes avec l'ordre lexicographique induit par un ordre total des formes de la première image et un ordre total des formes de la deuxième image. Les ordres des formes doivent être n'importe quel ordre de leur section les contenant, lequel ordre est arbitraire, par exemple l'adresse de leur structure de données. Alors nous construisons une correspondance de sections pour chaque intervalle de correspondances de formes dont les sections les contenant sont égales.

Donc une correspondance de sections est un ensemble de correspondances de formes, dont les sections les contenant sont les mêmes.

Poids d'une correspondance de sections

Nous devons attribuer un poids aux correspondances de sections. Cela dépend du nombre de formes dont celles-ci sont composées, mais nous pouvons faire mieux : nous pouvons imposer une cohérence interne dans les correspondances de sections. La contrainte de cohérence est la suivante : deux formes de la première image, et plus généralement, deux correspondances de formes dans la même correspondance de sections sont considérées incompatibles si les aires de la forme dans la première image sont ordonnées différemment que les aires de leurs formes correspondantes respectives dans la deuxième image. Si nous notons $\#S$ l'aire d'une forme S , les

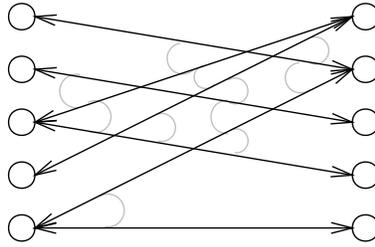


FIG. 6.1 – Incompatibilités de correspondances de formes dans une correspondance de sections. Si les cercles de gauche représentent les formes de la section dans la première image, ordonnées de haut en bas par inclusion, les cercles de droite les formes de la section dans la seconde image, ordonnées pareillement, et les lignes les correspondances de formes, les arcs de cercle gris relient des correspondances de formes incompatibles.

correspondances de formes (S_1, S_2) et (S'_1, S'_2) sont incompatibles si $\#S_1 - \#S_2$ et $\#S'_1 - \#S'_2$ ont des signes différents (ou l'un est zéro mais pas l'autre). C'est ce qu'illustre la Figure 6.1. Nous voulons enlever des correspondances de formes de manière à garder seulement des correspondances compatibles, et un nombre maximum d'entre elles. Ce nombre est le poids de la correspondance de sections. Les correspondances incompatibles dans la Figure 6.1 sont celles qui se rencontrent ou ont une extrémité en commun.

Ceci peut s'interpréter comme un problème de graphe. Chaque correspondance de formes peut être représentée par un nœud et une arête entre deux nœuds signifie leur incompatibilité. Cela revient donc à sélectionner un nombre maximum d'arêtes tel qu'aucune paire d'entre elles n'est liée, c'est-à-dire une clique maximale du graphe. La solution n'est pas unique, mais une solution nous suffit. La recherche de cliques maximales dans un graphe est notablement NP-complet, c'est-à-dire revient à énumérer toutes les possibilités. Heureusement, une meilleure solution existe dans notre cas.

Le fait que les formes dans chaque section soient totalement ordonnées (par inclusion) est essentiel. Etant donnée une correspondance de sections $(\mathcal{S}_1, \mathcal{S}_2)$ composée de correspondances de formes

$$C_1 = (S_{i_1}, S'_{j_1}), \dots, C_k = (S_{i_k}, S'_{j_k}),$$

nous pouvons les ordonner totalement par la relation :

$$C_m \leq C_n \Leftrightarrow \begin{cases} \#S_{i_m} < \#S_{i_n} & \text{or} \\ \#S_{i_m} = \#S_{i_n} & \text{and } \#S'_{j_m} \geq \#S'_{j_n} \end{cases}. \quad (6.10)$$

Il s'agit de l'ordre lexicographique induit par les aires des formes dans l'image de gauche comme clé primaire, et l'opposé des aires des formes dans l'image de droite comme clé secondaire. Supposons que, éventuellement après permutation des indices, $C_1 < C_2 < \dots < C_k$. Notre algorithme attribue une « hauteur » h_m à tous les C_m , représentant le nombre maximum de correspondances mutuellement compatibles inférieures à C_m , qui sont toutes compatibles avec C_m . Bien sûr le poids de $(\mathcal{S}_1, \mathcal{S}_2)$ est

$$\max_{m=1, \dots, k} h_m.$$

Nous initialisons la hauteur de chaque correspondance de formes à 0. Nous trouvons d'abord les correspondances de hauteur 1, puis 2 et ainsi de suite, jusqu'à ce que toutes les correspondances aient leur hauteur calculée.

Pour trouver les correspondances de hauteur 1, nous bouclons sur les C_m , conservant les aires (a_1, a_2) des formes de la dernière correspondance de hauteur 1 trouvée. Clairement, C_1 est de poids 1, donc nous initialisons a_1 et a_2 à, respectivement, $\#S_{i_1}$ et $\#S'_{j_1}$. C_2 est de hauteur 1 si et seulement si $\#S_{i_2} = a_1$, ou $\#S_{i_2} > a_1$ mais $\#S'_{j_2} \leq a_2$, car sinon C_1 et C_2 sont compatibles. Donc si C_2 est de hauteur 1, nous l'enregistrons dans h_2 et mettons à jour a_1 et a_2 à $\#S_{i_2}$ et $\#S'_{j_2}$. De même, C_3 est de hauteur 1 si et seulement si C_3 n'est pas compatible avec C_1 et C_2 , ce qui revient à des comparaisons de $\#S_{i_3}$ à a_1 et de $\#S'_{j_3}$ à a_2 . Cette opération est généralisée dans l'assertion suivante :

Proposition 6.1 *Soit $C_1 = (S_{i_1}, S'_{j_1}), \dots, C_k = (S_{i_k}, S'_{j_k})$ les correspondances de sections dans une correspondance de sections donnée, telle que $C_1 < \dots < C_k$ pour l'ordre défini dans (6.10). Pour tout m dans $\{1, \dots, k\}$ et h un nombre entier, si nous savons que $h_m \geq h$, nous avons*

$$h_m = h \Leftrightarrow \begin{cases} \forall n < m, & h_n < h & \text{ou} \\ \#S_{i_m} = a_1, & \text{ou } \#S_{i_m} > a_1 \text{ et } \#S'_{j_m} \leq a_2 & \text{où } (a_1, a_2) = (\#S_{i_n}, \#S'_{j_n}), \\ & & n = \max\{p : h_p = h\} \end{cases}$$

Nous utiliserons le lemme suivant :

Lemme 6.2 (Transitivité de la compatibilité par rapport à l'ordre) *Les*

correspondances de formes $C < C' < C''$ étant dans une même correspondance de sections, si d'une part C et C' sont compatibles et d'autre part C' et C'' le sont, alors C et C'' sont également compatibles.

Preuve.

1. Nous écrivons $C = (S_1, S'_1)$, $C' = (S_2, S'_2)$ et $C'' = (S_3, S'_3)$. Alors l'ordre de C et C' donne $\#S_1 < \#S_2$ (sinon il y a égalité et $S_1 = S_2$, contredisant la compatibilité), et l'ordre de C' et C'' , $\#S_2 < \#S_3$. Nous déduisons $\#S_1 < \#S_3$.

2. La compatibilité de C et C' implique alors $\#S'_1 < \#S'_2$, et la compatibilité de C' et C'' , $\#S'_2 < \#S'_3$, impliquant $\#S'_1 < \#S'_3$.

3. Nous avons prouvé que $\#S_3 - \#S_1$ et $\#S'_3 - \#S'_1$ sont des nombres positifs, donc C et C'' sont compatibles. \square

La preuve de la Proposition 6.1 est alors :

Preuve.

1. Supposons d'abord $h_m = h$, et que $\exists n < m, h_n = h$. Prenons le plus grand tel n , et a_1, a_2 les aires de formes correspondantes dans C_n . Comme $C_n < C_m$ par hypothèse, nous déduisons $\#S_{i_m} \geq a_1$. Puisque $h_n = h$, nous avons une famille croissante F de h correspondances, la dernière étant C_n , qui sont deux à deux compatibles. Si C_m était compatible avec C_n , d'après le Lemme 6.2, C_m serait aussi compatible avec toute correspondance dans F , d'où $h_m \geq h + 1$. Nous déduisons alors $\#S_{i_m} = a_1$, ou $\#S_{i_m} > a_1$ mais $\#S'_{j_m} \leq a_2$.

2. Réciproquement, si pour tout $n < m$, $h_n < h$, nous avons $h_m \leq 1 + \max_{n < m} h_n \leq h$, impliquant $h_m = h$ car par hypothèse $h_m \geq h$. Si nous sommes dans le second cas de l'alternative, nous avons que C_n et C_m sont incompatibles. Supposons que $h_m > h$, alors il y a une forme C_p de hauteur h compatible avec C_m , avec $C_p < C_m$. Par définition de n , nous avons $C_p \leq C_n$ et en fait pas égalité car C_n et C_m ne sont pas compatibles. Donc $C_p < C_n$. Mais $\#S_{i_p} < \#S_{i_m}$, donc si $\#S_{i_m} = a_1$, nous avons $\#S'_{j_p} \geq a_2$ (sinon C_p et C_n seraient compatibles), alors que $a_2 > \#S'_{j_m}$ (venant de $C_n < C_m$), ce qui impliquerait $\#S'_{j_p} > \#S'_{j_m}$, contredisant la compatibilité de C_p et C_m . Si $\#S_{i_m} > a_1$, nous avons par hypothèse $\#S'_{j_m} \leq a_2$, et deux cas doivent être examinés : si $\#S_{i_p} < a_1$, l'incompatibilité de C_n et C_p donne $\#S'_{j_p} \geq a_2$, et *a fortiori*, $\#S'_{j_p} \geq \#S'_{j_m}$, montrant que C_p et C_m sont incompatibles ; sinon, $\#S_{i_p} = a_1$, impliquant $\#S'_{j_p} > a_2$ et donc $\#S'_{j_p} > \#S'_{j_m}$, prouvant aussi l'incompatibilité. Dans tous les cas, une contradiction apparaît, nous concluons que l'hypothèse $h_m > h$ est fautive. \square

C'est le principal résultat nous permettant d'écrire l'algorithme d'extraction de la hauteur de chaque correspondance de formes, illustré dans l'Algorithme 7. La complexité de cet algorithme est clairement $O(k^2)$, l'étape de tri prenant seulement $O(k \log k)$. Le calcul pour toutes les correspondances de sections est d'ordre

$$k_1^2 + k_2^2 + \dots + k_c^2$$

où c est le nombre de correspondances de sections, avec

$$k_1 + k_2 + \dots + k_c = N,$$

le nombre de correspondances de formes. Donc la complexité est majorée par N^2 . En pratique, presque tous les k_i sont petits, et le calcul global des poids est une étape négligeable en termes de temps écoulé, comparé aux autres étapes.

Algorithm 7 Calcul des hauteurs des correspondances de formes dans une correspondance de sections

Require: $C_1 < \dots < C_k$ {Les correspondances de formes ordonnées}

$\forall i = 1, \dots, k, h_i \leftarrow 0$

for $h = 1$ à k **do** {Détermination des correspondances de hauteur h }

$a_1 \leftarrow +\infty, a_2 \leftarrow +\infty$

for $p = h$ à k **do**

if $h_p = 0$ **then** {Hauteur de C_p pas encore connue}

if $\#S_{i_p} \leq a_1$, ou $\#S_{i_p} > a_1$ mais $\#S'_{j_p} \leq a_2$ **then**

$h_p \leftarrow h$ {La hauteur de C_p est trouvée}

$a_1 \leftarrow \#S_{i_p}, a_2 \leftarrow \#S'_{j_p}$

end if

end if

end for

end for

Si une famille maximale de correspondances de formes dans chaque correspondance de section est désirée, cela s'obtient immédiatement une fois les hauteurs calculées : partir de la dernière correspondance, descendre jusqu'à trouver une correspondance de hauteur maximale w , puis continuer la boucle jusqu'à atteindre une

correspondance compatible de hauteur $w - 1$, et ainsi de suite. La complexité est $O(k)$.

Votes

Le poids du vote pour le zoom/rotation de deux correspondances de sections est choisi comme le produit des poids des sections : nous considérons que chaque correspondance de formes dans la première correspondance de sections vote avec chacune dans la seconde correspondance de sections, et que tous ces votes sont identiques. C'est une hypothèse raisonnable, car les formes dans une section sont très proches.

6.5.2 Autres déplacements globaux

La même méthode se généralise à d'autres déplacements globaux sans beaucoup de changements, tels que le recalage affine ou le recalage projectif planaire. Les différences avec le schéma expliqué plus haut sont les suivantes :

- Les invariants utilisés pour l'appariement doivent être modifiés pour être des invariants relativement à la classe de mouvement étudié. De nombreux invariants relativement à des transformations affines ou projectives planaires existent, voir Reiss [66].
- Le procédure de vote doit être modifiée, car deux points et leurs points correspondants ne sont pas suffisants. Nous en avons besoin de trois pour un déplacement affine et de quatre pour une projection planaire.

Le premier point exige d'utiliser des moments d'ordre supérieur, qui sont plus sensibles au bruit et à l'échantillonnage. La stabilité des invariants ainsi déduits doit être regardée de près.

Le deuxième point n'est pas un problème théorique, mais pratique : si nous avons besoin de trois points et de leurs correspondants pour voter, l'élection devient de complexité $O(N^3)$, et si quatre points sont nécessaires, $O(N^4)$. Pour un grand nombre de correspondances N , cela devient irréaliste. La meilleure solution serait d'avoir un moyen d'éliminer *a priori* un grand nombre de correspondances.

6.5.3 Utilisation de l'information d'inclusion

Une information géométrique forte concernant l'image est codée dans l'arbre, sans être utilisée ici : l'ordre des formes, c'est-à-dire leur inclusion. En effet, si les formes A et B sont dans l'image 1 avec $A \subset B$, leurs formes correspondantes A' et

B' doivent être emboîtées dans le même ordre. Ceci nous conduirait vers le recalage structurel, comme trouver des isomorphismes de sous-arbres. Mais cela ne revient pas à trouver les isomorphismes maximaux de sous-arbres entre les deux arbres d'inclusion, car nous connaissons un nombre de correspondances possibles entre les nœuds des arbres.

6.5.4 Gestion des occlusions

L'un des principaux problèmes en analyse d'image est peut-être dû aux occlusions. L'opération d'occlusion peut être considérée comme l'événement le plus élémentaire dans les images, comme l'addition des ondes pour les sons.

Dans un sens faible, les formes sont adaptées aux éventuelles occlusions : puisque nous remplissons les trous, et en supposant que les objets sont pleins, une forme se débarrasse de son objet occluant. Malheureusement, un type d'occlusion plus courant est quand l'objet occluant recouvre une partie du contour de l'objet occlus. Ceci peut impliquer que la forme détectée ne corresponde qu'à une partie de l'objet, ou pire, que nous ne puissions isoler l'objet en tant que forme, si par exemple c'était un objet clair sur fond sombre et l'objet occluant est encore plus clair. Même si la partie visible de l'objet est une forme, ses moments sont biaisés et il a de fortes chances de ne pouvoir voter correctement.

Une solution à ce problème serait de considérer les lignes de niveau des images. Bien sûr, les lignes de niveau sont aussi perturbées par les occlusions, donc nous devons considérer des morceaux minimaux de lignes de niveau, minimaux dans le sens qu'ils ont besoin de porter de l'information, mais que tout morceau plus petit n'en porterait pas. Ceci revient à considérer des parties des bords des formes à la place des formes elles-mêmes.

Comme ces lignes sont quantifiées et bruitées, nous devons les lisser. Mais la méthode lissage ne doit pas perturber l'estimation ultérieure du mouvement. En d'autres termes, le mouvement estimé entre deux courbes lissées doit être exactement le même que celui entre les deux courbes originales. Cela veut dire que le lissage doit commuter avec les transformations géométriques admissibles. Comme le lissage de lignes le plus invariant possible est affine invariant, voir Alvarez *et al.* [3], il n'y a pas d'espoir de trouver un recalage projectif planaire par une telle méthode². Une fois lissée, des points caractéristiques affine invariants doivent être détectés. Les points de courbure maximale, c'est-à-dire les coins, bien qu'affine invariants, ont besoin

²Mais si nous avons plusieurs estimations affines assez locales, il serait possible de les considérer comme tangentes à la transformation projective et donc de l'estimer.

de seuils pour être détectés, et comme leur courbure change après transformation affine, ils devraient être évités. Au contraire, les points d'inflexion sont de bons candidats : leur nombre décroît durant le lissage affine invariant. Leur localisation précise est difficile à déterminer, puisque par définition la courbe est presque droite en leur voisinage, mais la direction de la tangente en de tels points est très stable. D'autres directions affine invariantes stables sont celles des bitangentes. A partir de ces directions, les courbes peuvent être segmentées en parties, chaque partie étant codée par quelques descripteurs, et nous pouvons suivre la même démarche pour la mise en correspondance et les votes. Ceci est un travail en cours, la méthode générale et des premiers résultats sont exposés par Lisani *et al.* dans [39].

Chapitre 7

Expériences de recalage

Dans ce chapitre, nous présentons quelques exemples réels et synthétiques de recalage. Les expériences réelles sont volontairement choisies difficiles, chacune mettant en œuvre son propre type de difficulté. Toutefois, ces difficultés ne sont pas des cas particuliers et arrivent fréquemment. Cela montre que le recalage d'images n'est pas un tâche triviale, et que toutes ces difficultés doivent être prises en compte lorsque nous développons une stratégie pour le traiter.

7.1 Estimation de pose

Le problème classique de l'estimation de pose est de trouver l'orientation d'un objet plat vu d'une distance fixe et perpendiculairement à son plan. Le but est donc de retrouver une rotation-translation. C'est typiquement la première tâche à accomplir en inspection de qualité industrielle automatique dans les chaînes de montage. Un objet doit être comparé à un modèle pour vérifier sa conformité. Des petits déplacements ne sont pas rares dans le processus, et ils doivent être compensés avant la comparaison avec le modèle.

7.1.1 Quelle heure est-il ?

Notre première expérience concerne des images de montre prises à deux moments différents (comme les aiguilles en témoignent) et avec des poses différentes, voir la Figure 7.1. Le cadran de la montre a été posé sur le verre d'un scanner de qualité moyenne et deux clichés sont pris, la montre étant déplacée et l'heure changée entre les deux. Le bon point pour notre algorithme est la multitude de marques distinctes sur le cadran, de toutes tailles. Un grand nombre d'entre elles ne sont pas occluses

FIG. 7.1 – Deux images de la même montre à un instant différent et avec une position légèrement différente.

par les aiguilles, donnant l'espoir d'obtenir un bon recalage. Néanmoins, cet exemple est aussi un défi, pour les raisons suivantes.

- Les images sont assez bruitées, c'est particulièrement visible sur les zones qui devraient être uniformes.
- L'image de droite est un peu floue.
- Les aiguilles bougent avec un mouvement complètement différent du déplacement global, et occluent des formes.
- De nombreuses formes sont présentes plusieurs fois dans l'image, et quelques-unes un grand nombre de fois, particulièrement les graduations, mais aussi quelques chiffres, donnant des mouvements concurrents.
- L'autosimilarité structurelle en rotation pourrait donner un grand nombre de votes pour des rotations autour du centre de la montre, cachant le mouvement dû au déplacement.

Les images sont de taille approximative 500×500 . Nous ne gardons que les formes d'au moins 20 pixels et ne rencontrant pas le cadre (car les formes rencontrant le cadre représentent des objets occlus par le cadrage de l'image). A cause du grand nombre de formes significatives, nous limitons la distance de deux formes extraites des deux images à 50 pixels au plus pour les autoriser à s'apparier. Sans cette limitation, le nombre de correspondances serait très élevé, rendant l'étape de vote impossible à accomplir en temps raisonnable. Sous ces conditions, le nombre

FIG. 7.2 – Images reconstruites des formes de chaque image de la Figure 7.1 qui ont au moins une forme correspondante dans l’autre image. Les formes rencontrant le cadre de l’image sont volontairement supprimées.

de correspondances de formes est 171 015, impliquant 20 308 formes dans l’image de gauche et 19 247 formes dans l’image de droite. Quand les formes sont rassemblées en sections, ceci revient à 7887 correspondances de sections, impliquant 4100 sections de l’image de gauche et 3568 dans l’image de droite. Quand les incompatibilités de correspondances de formes à l’intérieur d’une correspondance de sections sont résolues (comme l’explique la Section 6.5.1), il reste 49 515 correspondances de formes. Les images représentées par les arbres d’inclusion, dont nous ne conservons que les formes qui ont un correspondant dans l’autre image, sont visibles dans la Figure 7.2.

Le vote s’effectue de la manière suivante : nous votons d’abord pour le zoom/rotation (en considérant toutes les paires de correspondances de sections), déterminons le zoom/rotation dominant, l’appliquons à l’image de gauche (avec un centre de rotation arbitraire) puis votons pour la translation. L’échantillonnage est choisi de 1 pixel pour la translation, et les zooms sont échantillonnés régulièrement entre $\frac{1}{1,10}$ et 1, 10 (nous n’autorisons qu’au plus 10% de taux de changement d’échelle entre les images) en 145 échantillons et l’angle de rotation en 1450 échantillons entre $-\frac{\pi}{2}$ et $\frac{\pi}{2}$. Ces échantillonnages sont choisis de telle sorte que le vote de deux correspondances de sections a la précision d’un échantillon en moyenne. Notons que cette précision dépend de façon cruciale de la distance des (centres des) formes dans la même

image. Comme la répartition des formes dans les images est *a priori* inconnue, cette distance moyenne ne peut être calculée, mais typiquement devrait être une fraction de la plus grande distance dans l'image, sa diagonale. Le point important à noter ici est que l'échantillonnage de l'angle de rotation et du facteur de zoom dépend de la taille de l'image : nous ne pouvons prévoir la même précision dans les paramètres pour les petites et les grandes images, mais ce n'est pas une limitation ; en effet, le point important n'est pas la précision absolue des paramètres, mais la précision du recalage, en termes de nombre de pixels.

Avec de tels échantillonnages, le mouvement dominant correspond un zoom presque nul (facteur 1,00006), une petite rotation ($0,44^\circ$ dans le sens des aiguilles d'une montre) et une translation de $(-13, -1)$ lorsque le centre de rotation est le coin haut-gauche de l'image. Les histogrammes 2-D de votes apparaissent dans la Figure 7.3. Le graphe de ces histogrammes autour de leur maximum est dans la Figure 7.4. Le nombre de votes pour le zoom/rotation est bien plus élevé car cela correspond à des votes de *paires* de correspondances, alors que le vote pour la translation provient de correspondances simples. Nous inspectons la précision de la manière suivante : nous appliquons le mouvement dominant à l'image de gauche et la superposons à l'image de droite (nous prenons la moyenne de deux pixels superposés), voir la Figure 7.5. Notons que l'image résultat est bonne, sauf que chaque aiguille apparaît deux fois, car les aiguilles bougent avec un mouvement différent.

Une inspection rapprochée des images de vote témoigne de la quasi invariance par rotation de ces images : les deux principaux pics secondaires dans l'image correspondent précisément à un décalage d'une marque d'une minute, c'est-à-dire les marques numérotées (voir la Figure 7.6). La Figure 7.7 montre le recalage correspondant à ces pics secondaires. Les décalages de plusieurs marques n'apparaissent pas car la distance devient plus grande que 50 pixels. Une fois que le zoom-rotation est fixé, cette quasi invariance se retrouve dans le vote pour la translation, sous la forme de cercles concentriques.

Bien que globalement satisfaisant, ce recalage n'est pas un succès total. Quand nous essayons d'atteindre la précision subpixelique, comme expliqué dans le chapitre précédent, nous obtenons un effet stroboscopique partiel dû aux marques dans le coin bas-droite de l'image, alors que le coin haut-gauche est correctement recalé. La Figure 7.8 montre le recalage calculé et la Figure 7.9 les formes participant à l'estimation aux moindres carrés. Le nouveau facteur de zoom est 1,0003, la rotation $0,42^\circ$ dans le sens des aiguilles d'une montre et le vecteur de translation $(-12, 9; 0, 0)$. Regarder de près les images recalées montre que la précision n'a pas été vraiment améliorée, et que de petits décalages restent présents. Le mouvement

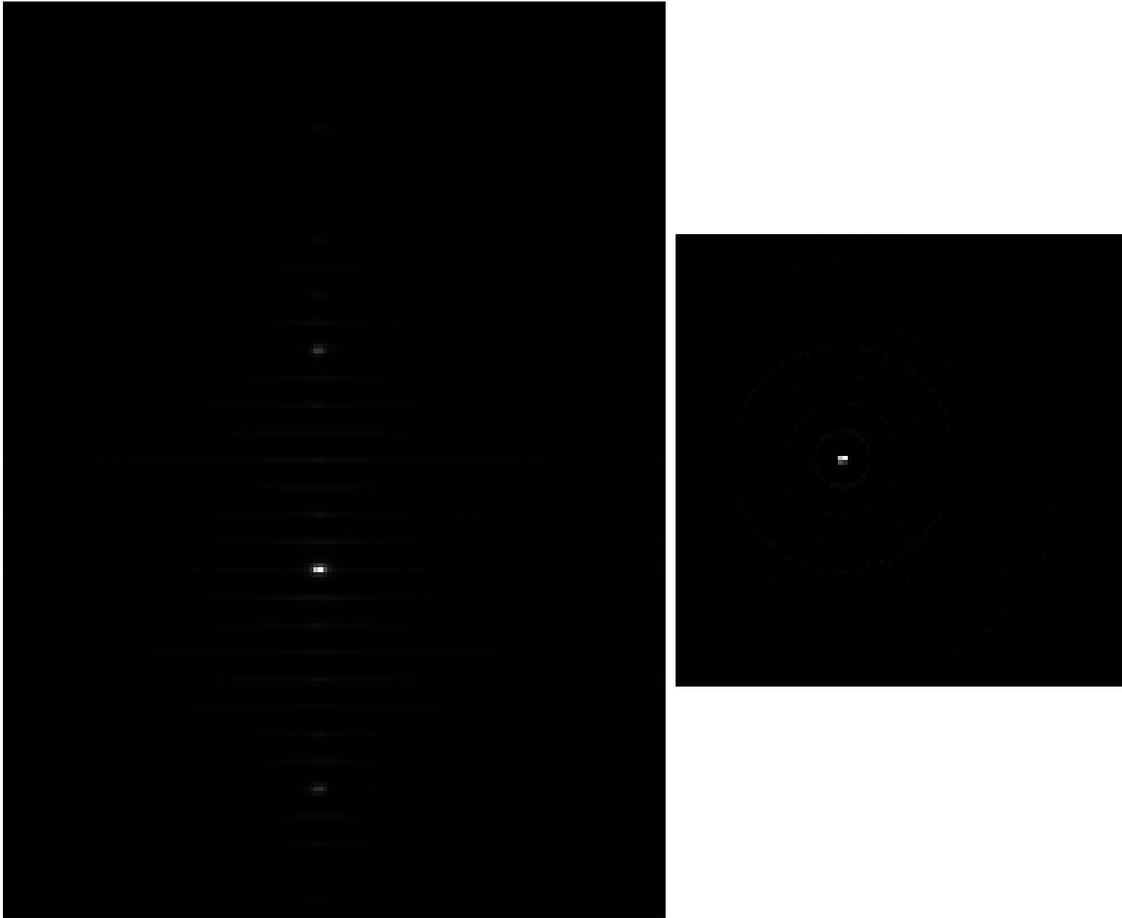


FIG. 7.3 – Votes pour les paramètres de la similitude dans les images de montre de la Figure 7.1. Gauche : l’histogramme des votes pour le facteur de zoom (axe horizontal) et l’angle de rotation (axe vertical), le niveau de gris étant proportionnel au nombre de votes. Droite : l’histogramme des votes pour la translation en x (axe horizontal) et y (axe vertical). Notons que les pics dans ces histogrammes sont nets et non ambigus, inspirant confiance dans le mouvement estimé.

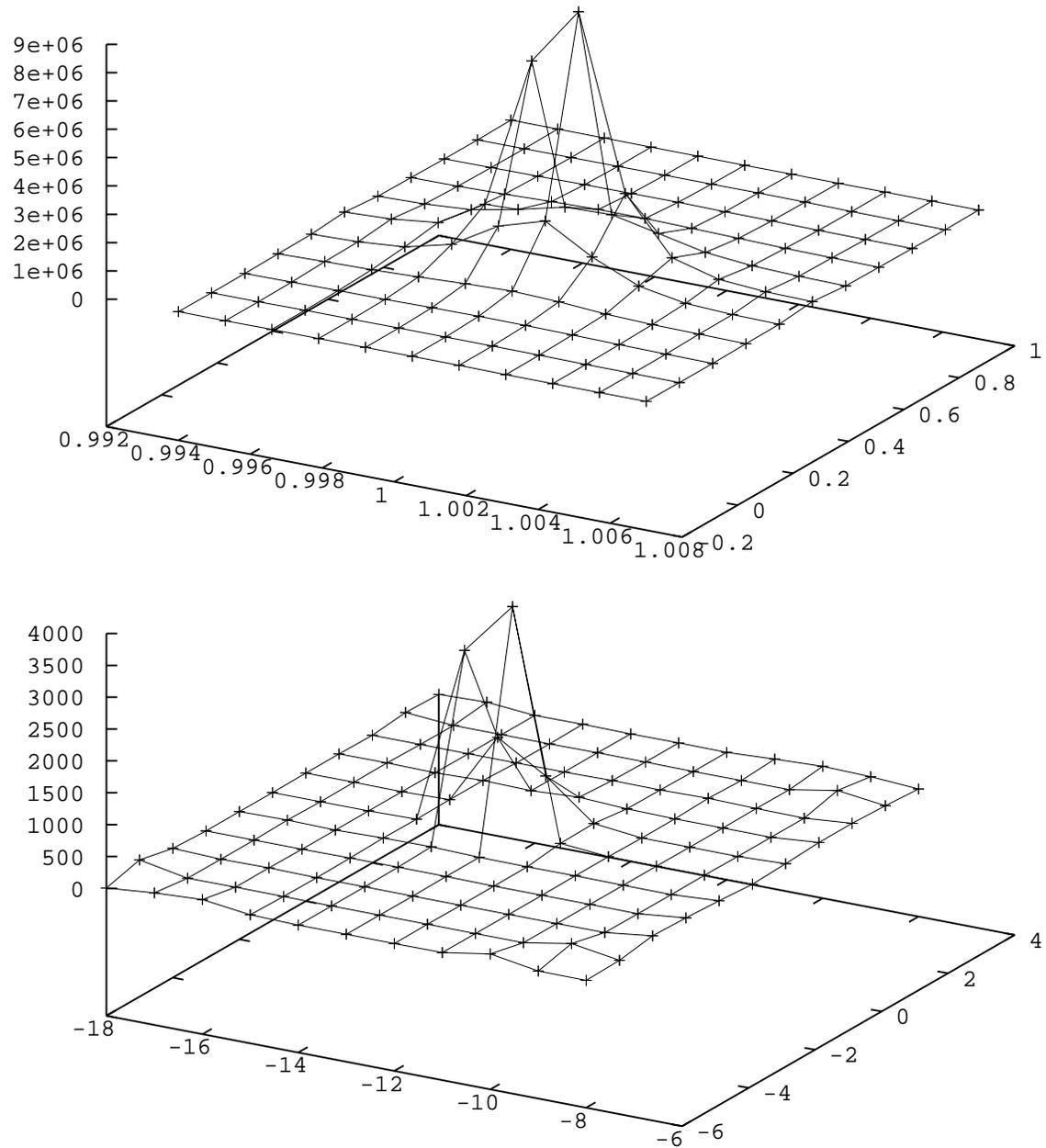


FIG. 7.4 – Les graphes des histogrammes de votes de la Figure 7.3 autour de leur maximum.

FIG. 7.5 – Recalage des images de la Figure 7.1. Les deux images sont superposées. Gauche : superposition sans recalage. Droite : superposition après application du mouvement estimé à la première image.

estimé comprend des fausses correspondances : le mouvement calculé n'est pas le bon. La raison en est que des formes dans la partie inférieure droite sont considérées comme s'appariant correctement avec plusieurs autres formes, et la moyenne donne un mauvais résultat. Cela se passe dans la partie en bas à droite est que le centre de rotation a été choisi comme le coin haut-gauche de l'image, puisque le choix est arbitraire. Une tolérance d'erreur dans l'angle de rotation donne un petit déplacement pour les formes près du centre, ne sélectionnant que les bonnes correspondances, alors qu'elle autorise un plus grand déplacement loin de lui (dans la partie inférieure droite), rendant des correspondances venant de l'effet stroboscopique compatibles avec le mouvement dominant. Cet effet peut arriver dans toutes les images avec une forte auto-similarité. La solution serait de sélectionner de façon plus intelligente les correspondances compatibles avec le mouvement calculé.

Les images de la Figure 7.9 peuvent s'interpréter comme une sorte d'intersection d'images. L'image de gauche est reconstruite à partir des formes qui ont été reconnues au bon endroit dans l'image de droite, et inversement. C'est très proche de ce qui est proposé par Ballester *et al.* dans [6] : ils extraient les composantes connexes des biniveaux ($[\lambda \leq u(\mathbf{x}) \leq \mu]$) et cherchent un correspondant au même endroit dans l'autre image. Les correspondances sont établies d'une manière très similaire à la nôtre, en comparant des moments, le périmètre, etc. Chaque composante

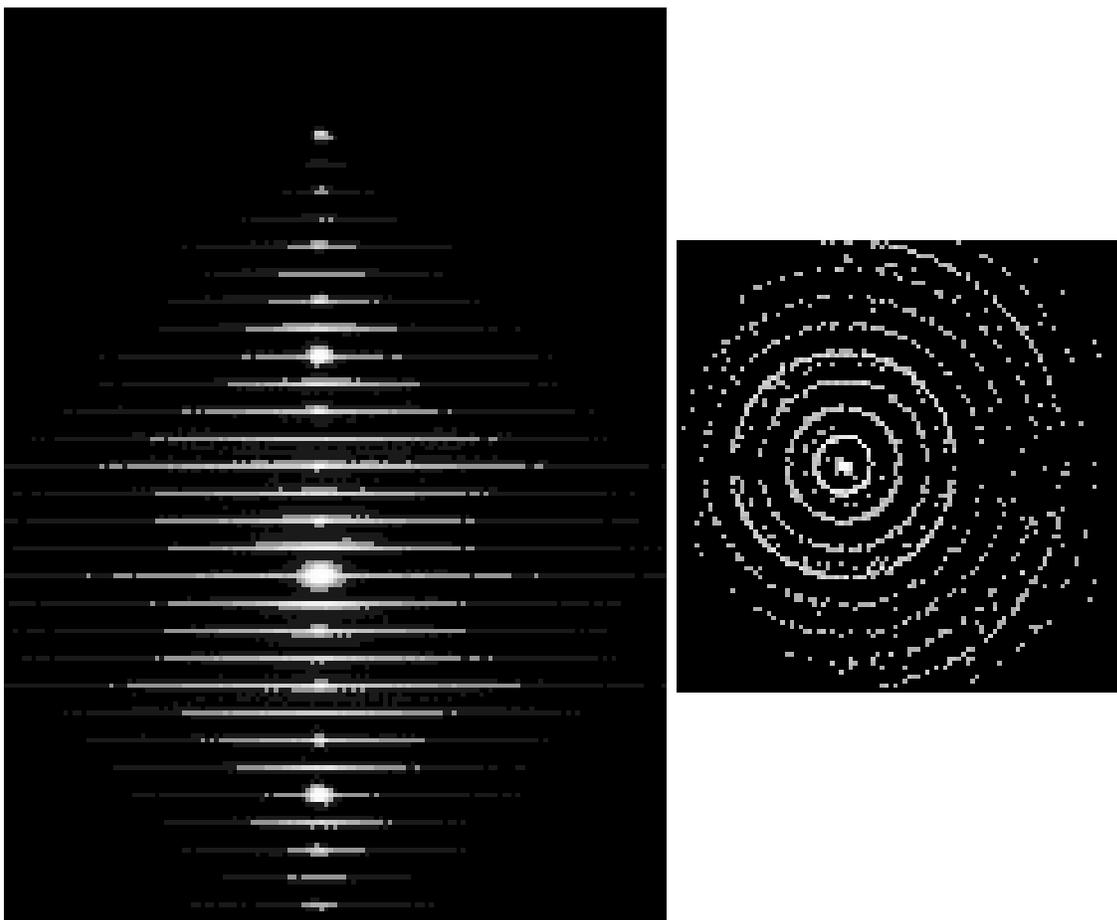


FIG. 7.6 – Les votes pour les paramètres de la similitude dans le recalage des montres de la Figure 7.1, comme dans la Figure 7.3, mais avec un changement de contraste non linéaire rehaussant les pixels sombres. Cela rend des pics secondaires visibles. Notons en particulier les deux principaux pics symétriques dans l’histogramme pour le zoom-rotation (gauche). Ils correspondent à des rotations d’une minute dans le sens des aiguilles d’une montre et dans le sens inverse.

FIG. 7.7 – Recalages correspondant aux principaux pics secondaires dans l’histogramme des votes pour le zoom-rotation des images de la Figure 7.1. Ils correspondent à une rotation de 6° dans le sens des aiguilles d’une montre (image de droite) et dans le sens inverse relativement au mouvement dominant. Notons que les marques sont parfaitement recalées dans les deux images, montrant que les pics secondaires correspondent à leurs votes.

FIG. 7.8 – Recalage de l’image de la Figure 7.1 obtenu par une estimation aux moindres carrés des correspondances compatibles avec le mouvement dominant. Le résultat est assez décevant, du fait du mélange de mouvements concurrents incompatibles.

connexe de biniveau ayant trouvé un correspondant dans l’autre image est conservé, et l’image est reconstruite à partir d’eux. Cela diffère de notre « intersection » principalement parce que, outre le fait qu’aucun mouvement n’est attendu dans leur cas, les biniveaux ne suffisent pas à reconstruire une image sans ambiguïté. Ils n’ont pas de structure d’arbre, et ils définissent pour chaque image deux intersections : une pour laquelle chaque pixel prend le maximum du seuil inférieur des composantes connexes de biniveau restantes le contenant, l’autre le minimum du seuil supérieur. La structure d’arbre d’inclusion des formes nous permet de reconstruire *une* image d’intersection pour chaque image originale.

7.1.2 Une autre expérience de montre

Nous montrons un autre exemple de recalage d’images de montre (voir la Figure 7.10) prises dans les mêmes conditions que dans l’expérience précédente (l’aiguille des secondes n’apparaît pas droite car elle bougeait durant l’acquisition). Les images sont de taille 532×529 .

Au contraire de l’exemple précédent, le nombre de marques est bien plus réduit, et l’invariance relativement à la rotation est très forte. Cela rend l’estimation de la

FIG. 7.9 – Images reconstruites des formes de chaque image de la Figure 7.1 impliquées dans une correspondance participant à l'estimation aux moindres carrés. Le résultat semble assez cohérent, sauf quelques formes qui apparaissent dans un image et pas dans l'autre.

FIG. 7.10 – Jeu d'images d'une montre, utilisées pour expérimenter la méthode de recalage.

FIG. 7.11 – Logarithme des spectres de Fourier dans les images de la Figure 7.10. Aucune direction privilégiée n'apparaît, à l'exception des axes de coordonnées, produits par les effets de bord. Cela montre la quasi invariance de l'image relativement à la rotation, expliquant l'échec du recalage par corrélation.

rotation difficile. En effet, la méthode de corrélation échoue dans son estimation de l'angle : l'explication réside dans l'invariance rotationnelle, comme le montrent les spectres de Fourier (voir la Figure 7.11). Rappelons que l'estimation de la rotation est simplement une corrélation le long de cercles des spectres de Fourier (module des coefficients de Fourier). Aucune direction privilégiée n'apparaît dans le spectre de Fourier, donc le résultat de corrélation est influencé par les effets de bord (produisant les axes horizontal et vertical passant par l'origine forts). Cela donne comme estimation une rotation nulle.

Nous avons mené l'expérience avec les mêmes paramètres que dans l'exemple précédent. Nous obtenons 5072 formes dans l'image de gauche et 4960 formes dans l'image de droite ayant une correspondance, avec un nombre total de correspondances de formes se montant à 9713. La Figure 7.12 montre les formes ayant un correspondant dans l'autre image. Nous avons 1930 objets dans l'image de gauche et 1936 dans celle de droite, et seulement 4101 de ceux-ci restent compatibles avec leur correspondance d'objets. Les histogrammes de votes apparaissent dans la Figure 7.13. Notons que dans ce cas aussi les pics sont nets et le mouvement dominant non ambigu.

Le facteur de zoom estimé est très proche de 1 (1,0008), l'angle de rotation 2,34°

FIG. 7.12 – Images reconstruites des formes des images de la Figure 7.10 ayant une forme correspondante dans l’autre image.

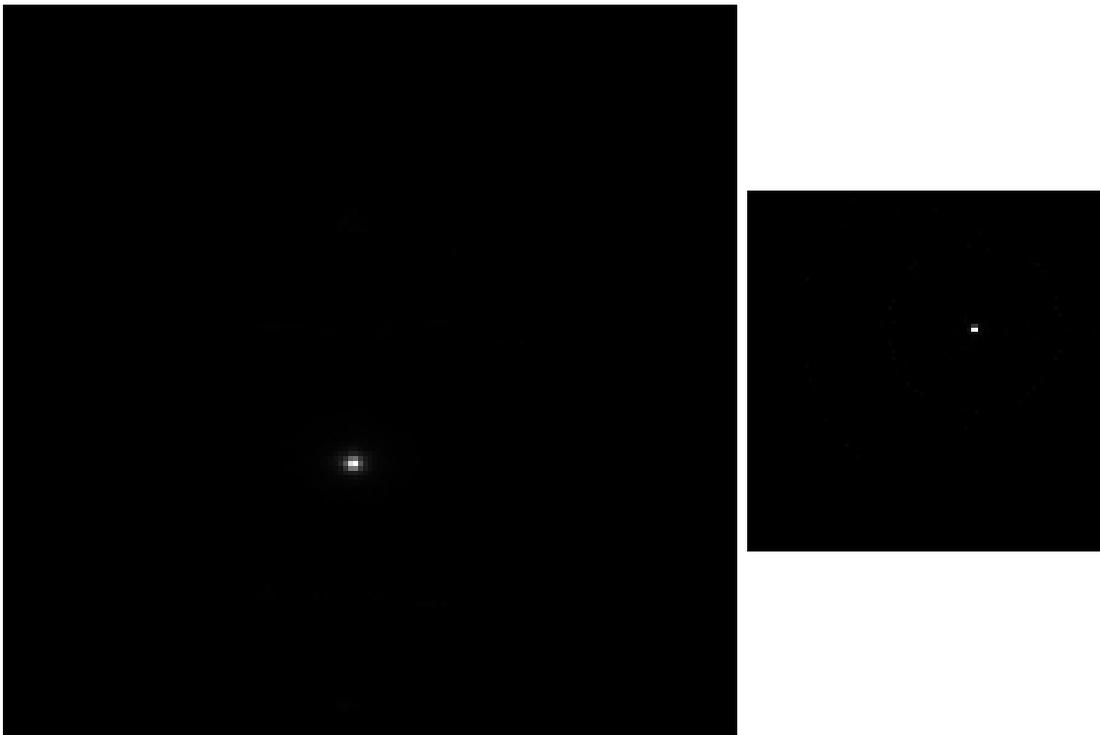


FIG. 7.13 – Votes pour les paramètres de la similitude dans le recalage des images de la Figure 7.10. Gauche : vote pour le facteur de zoom (axe horizontal) et angle de rotation (axe vertical). Droite : vote pour le vecteur de translation.

FIG. 7.14 – Résultats de la superposition des images de la Figure 7.10 avant (gauche) et après (droite) avoir appliqué la similitude estimée. Les images originales sont presque recalées en translation, mais pas en rotation (les lettres sont floues et à peine lisibles). Le recalage donne une bonne correspondance.

dans le sens des aiguilles d’une montre et le vecteur de translation $(13, -12)$ quand le centre de rotation est choisi comme le coin haut-gauche. La superposition des images avant et après recalage apparaît dans la Figure 7.14. Les images originales sont presque recalées en translation mais pas en rotation. Le recalage estimé est visuellement très précis.

L’estimation aux moindres carrés *a posteriori* donne des résultats très proches : un facteur de zoom de 1,0002, la même rotation et comme vecteur de translation $(13, 1; -11, 6)$. Les formes ayant une forme correspondante dans l’autre image compatible avec le mouvement dominant, c’est-à-dire les formes impliquées dans des correspondances participant à l’estimation aux moindres carrés, apparaissent dans la Figure 7.15. Notons que toutes les formes visuellement importantes semblent être présentes, avec l’exception notable des aiguilles, qui ne participent pas au mouvement dominant.

7.2 Similitude

Nous effectuons une estimation de similitude complète sur les images de la Joconde de la Figure 7.16. L’image de gauche, de taille 374×562 , est une photographie

FIG. 7.15 – Formes de chaque image de la Figure 7.10 ayant une forme correspondante dans l’autre image, dont la correspondance est compatible avec le mouvement dominant estimé.

scannée. L’image de droite, de taille 560×864 , est extraite d’une base de données du World Wide Web. Ces images sont un vrai défi du point de vue du recalage, pour les raisons suivantes :

- Il y a un fort changement de contraste entre les images.
- Le facteur de zoom est important.
- Les images sont de mauvaise qualité. De plus, elles viennent de sources différentes, chacune ajoutant son propre type de bruit.
- Les bords significatifs et les détails clairement visibles sont rares. Le style de De Vinci repose sur des dégradés et des contrastes lisses, les lignes de niveau deviennent très bruitées dans l’image numérique.
- Il n’est pas certain que le mouvement global soit une similitude. Une petite différence dans l’angle duquel l’image a été photographiée pourrait donner une petite transformation projective.

Il y a 925 sections dans l’image de gauche, et 8920 dans l’image de droite. Le nombre de correspondances de formes est 6718, ce qui se traduit par 3842 correspondances de sections. Des parties des histogrammes de votes apparaissent dans la Figure 7.17. Notons que les pics ne sont pas très pointus. Les paramètres estimés pour le facteur de zoom sont 1,54, une rotation de $1,34^\circ$ dans le sens des aiguilles d’une montre, et une translation de $(-18, 5; -0, 4)$.

FIG. 7.16 – Images de la Joconde utilisées pour tester le recalage par similitude.

Version *light* : certaines figures sont absentes

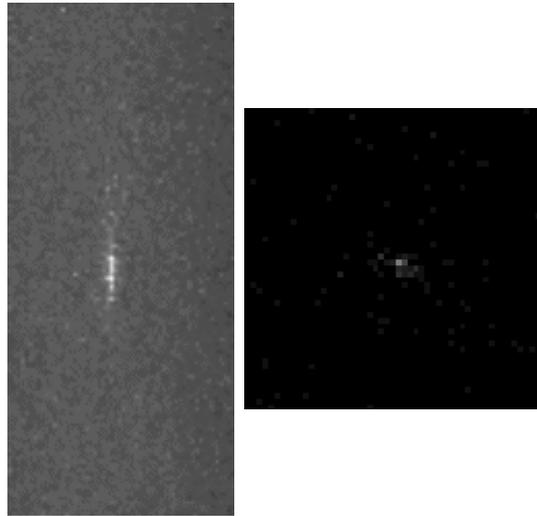


FIG. 7.17 – Histogrammes de votes, autour de leur maximum, dans le recalage des images de la Joconde de la Figure 7.16. Gauche : votes pour le zoom (axe horizontal)/rotation (axe vertical). Droite : votes pour la translation.

La superposition des images est montrée dans la Figure 7.18. Le résultat est globalement correct, bien que de petits décalages soient observés dans la partie inférieure. Quand nous essayons de les recaler à la main, nous pouvons noter qu’aucune similitude ne donne un résultat parfait. La transformation est en fait un peu plus complexe, affine ou peut-être projective.

7.3 Précision

Nous nous intéressons dans cette section à la précision du recalage calculé, et particulièrement à la question de savoir si la précision subpixelique est atteinte par cette méthode. Toutefois, nous voulons rendre l’expérience « réaliste ». Cela signifie que les conditions dans lesquelles les images que nous recalons sont créées doivent être conformes à ce que font les capteurs réels. En particulier, nous ne traitons pas d’images de fonctions de forme analytique simple. Donc, notre procédure est la suivante : nous prenons une image de grande taille et la sous-échantillons d’un certain facteur ; la deuxième image est créée en tradant l’image originale d’un petit montant (typiquement, pas un multiple du pas d’échantillonnage) puis en la sous-échantillonnant. De cette manière, le décalage entre les deux images est vraiment une fraction de pixel. Nous comparons la translation estimée à ce décalage.

Le point crucial dans cette procédure est que nous ne lissions pas l’image avant

FIG. 7.18 – Superposition des deux images de la Figure 7.16 après application du recalage par similitude estimé.

Version *light* : certaines figures sont absentes

de l'échantillonner : nous créons volontairement de l'aliasage. La raison est que presque tous les capteurs n'échantillonnent pas l'image continue lissée par la lentille à la fréquence de Nyquist. La plupart, si ce n'est toutes, des images que nous traitons ne sont pas échantillonnées dans les conditions du théorème de reconstruction de Shannon. Si c'était le cas, le recalage par corrélation n'atteindrait pas seulement la précision subpixelique, comme rapporté dans [87], mais aurait une précision théorique infinie. Ce serait la méthode parfaite. Mais dans les cas réels, les images ne sont pas échantillonnées à la fréquence de Nyquist, donc leur interpolation par des sinus cardinaux ne correspond pas à l'image originale définie sur un domaine continu, donc la précision du recalage par corrélation peut être assez mauvais, d'autant plus que la fréquence d'échantillonnage est en-dessous de la condition de Nyquist.

L'image utilisée pour conduire nos investigations est une image satellite du capteur Spot 2, de taille 6000×6000 . Nous l'échantillons d'un facteur dix en x et y , c'est-à-dire que nous ne gardons qu'un pixel sur 100 dans l'image originale. C'est notre image de gauche, voir la Figure 7.19. Notons que ce n'est pas une image facile à traiter, elle est assez oscillante. L'image de droite est créée en décalant l'image originale de n pixels avant échantillonnage, $n = 1, \dots, 9$. La vraie translation dans notre expérience de recalage est donc $n/10$.

Dans les expériences, le nombre de formes d'aire plus grande que 20 pixels et ne rencontrant pas le cadre de l'image est environ 20 000 dans chaque image. Ceci induit à peu près 30 000 correspondances de formes. Les formes sont regroupées en sections, dont le nombre est approximativement 8000 dans chaque image, et nous avons 8500 correspondances de sections. Les erreurs dans la translation estimée sont reportées dans la Figure 7.20. Comme prévu, l'erreur est d'autant plus importante que le décalage réel est proche du demi-pixel. Pourtant, les erreurs restent faibles, et nous pouvons parler d'une précision globale de l'ordre du dixième de pixel.

Mais cette précision dépend probablement du nombre de formes dans l'image. Comme les estimations sont les résultats d'une moyenne, nous pouvons nous attendre à une erreur d'autant plus petite que l'image est riche en détails, au contraire du recalage par corrélation, pour lequel la précision est indépendante de la taille de l'image (si nous négligeons les effets de bord).

FIG. 7.19 – Image Spot 2 que nous utilisons pour tester la précision de notre méthode de recalage. Sa taille est 600×600 , construite par échantillonnage d'un facteur 10 suivant chaque axe de l'image originale de taille 6000×6000 .

t_x	t'_x	$ t_x - t'_x $	t_y	t'_y
0,1	0,078	0,022	0	0,013
0,2	0,138	0,062	0	0,000
0,3	0,230	0,070	0	0,040
0,4	0,289	0,111	0	0,138
0,5	0,457	0,043	0	0,049
0,6	0,531	0,069	0	0,163
0,7	0,767	0,067	0	0,046
0,8	0,834	0,034	0	0,001
0,9	0,912	0,012	0	0,008

FIG. 7.20 – Erreur dans la translation estimée pour divers décalages. Voir le texte pour les détails. t est le vrai vecteur de translation, t' est la translation estimée.

Chapitre 8

Conclusion

8.1 Points forts de cette thèse

Les principales nouveautés de cette thèse sont les suivantes :

- Une représentation des images basée sur la géométrie, complète et non redondante, avec une structure facile à utiliser, l'arbre.
- Des informations importantes sont codées dans l'arbre, comme les extrema et les trous.
- Un algorithme rapide pour calculer cet arbre.
- Une définition du filtre de grain et une preuve de son autodualité pour les images continues.
- Une quantification adaptative de l'image préservant les principales informations topologiques.
- Un recalage d'images contraste-invariant, non perturbé par des déplacements secondaires, et de bonne précision même dans des cas défavorables.

8.2 Extensions possibles

Parmi les extensions possibles de ce travail, nous distinguons particulièrement les suivantes :

- Utilisation de l'arbre pour la compression. La quantification adaptative montre que les principales lignes des images ne sont pas très nombreuses, et des méthodes de compression telles que dans [24] seraient possibles.
- Comparaison d'images par les structures d'arbre (par exemple, détection de coupures dans les flots vidéo).

- Véritable gestion des occlusions. Des premiers résultats dans cette direction sont présentées dans [39].

8.3 Problèmes ouverts

L'introduction de la structure d'arbre des formes pour représenter les images soulève de nouveaux problèmes.

- A partir d'une famille d'ensembles connexes ayant une structure d'arbre d'inclusion, comment attribuer à chacun un niveau de gris tel que l'image reconstruite ait comme arbre associé précisément cet arbre ? Cela nous permettrait de définir une nouvelle notion de changement de contraste local, c'est-à-dire un changement de contraste qui n'affecte pas la structure d'arbre, et de dire quand deux images sont équivalents modulo un tel changement de contraste local.
- Trouver un changement de contraste local optimal (dans le sens défini ci-dessus) utilisant l'arbre.
- Etudier la stabilité de l'arbre. A savoir : deux images légèrement différentes ont-elles des structures d'arbres proches ? Deux images similaires ont-elles des arbres similaires ? Une réponse, partielle, est positive, car comme nous l'avons vu, les formes d'aire suffisante semblent stables, comme les bons résultats du filtre de grain en attestent.
- Construction de caractéristiques invariantes stables pour les formes en vue des correspondances utilisées dans le processus de recalage. De nombreux articles sur des caractéristiques géométriques invariantes existent, mais à notre connaissance, il n'y en a pas de définitives.

Bibliographie

- [1] A.V. Aho, J.E. Hopcroft, and J. Ullman. *Data Structures and Algorithms*. Addison-Wesley, 1983.
- [2] L. Alvarez, Y. Gousseau, and J.M. Morel. Scales in natural images and a consequence on their BV norm. In *Proc. of the 2nd Workshop on Scale-Space Theories in Computer Vision*, pages 247–258, Corfu, Greece, 1999.
- [3] L. Alvarez, F. Guichard, P.L. Lions, and J.M. Morel. Axioms and fundamental equations of image processing : Multiscale analysis and P.D.E. *Archive for Rational Mechanics and Analysis*, 16(9) :200–257, 1993.
- [4] L. Ambrosio, V. Caselles, S. Masnou, and J.M. Morel. The connected components of sets of finite perimeter. Preprint.
- [5] C. Ballester, V. Caselles, and J.M. Morel. Topological description of topographic maps and applications. En préparation, 2000.
- [6] C. Ballester, E. Cubero-Castan, M. González, and J.M. Morel. Contrast invariant image intersection. Preprint CEREMADE, 1998.
- [7] L.G. Brown. A survey of image registration techniques. *ACM Computing Surveys*, 24(4) :325–376, Décembre 1992.
- [8] J. Canny. A variational approach to edge detection. In *National Conference on Artificial Intelligence*, pages 54–58, Washington DC, Août 1983.
- [9] V. Caselles, B. Coll, and J.M. Morel. A Kanisza program. *Progress in Nonlinear Differential Equations and their Applications*, (25), 1996.
- [10] V. Caselles, B. Coll, and J.M. Morel. Is scale-space possible? In *Proc. of the 1st Workshop on Scale-Space Theories in Computer Vision*, Utrecht, the Netherlands, 1997.
- [11] V. Caselles, B. Coll, and J.M. Morel. Topographic maps and local contrast changes in natural images. *International Journal of Computer Vision*, 33(1) :5–27, Septembre 1999.

- [12] V. Caselles, J.L. Lisani, J.M. Morel, and G. Sapiro. Shape preserving local contrast enhancement. In *Proceedings of International Conference of Image Processing*, pages I :314–xx, 1997.
- [13] V. Caselles, J.L. Lisani, J.M. Morel, and G. Sapiro. Shape preserving local histogram modification. *IEEE Transactions on Image Processing*, 8(2) :220, Février 1999.
- [14] F. Catté, F. Dibos, and G. Koepfler. A morphological scheme for mean curvature motion and applications to anisotropic diffusion and motion of level sets. *SIAM*, (32), Décembre 1995.
- [15] S.H. Chang, F.H. Cheng, W.H. Hsu, and Z. Wu. Fast algorithm for point patten matching : Invariant to translations, rotations and scale changes. *Pattern Recognition*, 30(2) :311–320, 1997.
- [16] J.L. Cox and D.B. Karron. Digital Morse theory. Manuscript available from <http://www.casi.net>, 1998.
- [17] E. DeCastro and C. Morandi. Registration of translated and rotated images using finite Fourier transforms. *PAMI*, 9(5) :700–703, Septembre 1987.
- [18] L.C. Evans and R.F. Gariepy. *Measure Theory and Fine Properties of Functions*. Studies in Advanced Mathematics. CRC Press, 1992.
- [19] O. Faugeras. *Three-Dimensional Computer Vision : A Geometric Viewpoint*. MIT Press, Cambridge, Massachusetts, 1993.
- [20] O. Faugeras and R. Keriven. Some recent results on the projective evolution of 2d curves. In *Proceedings of IEEE International Conference on Image Processing*, volume 3, pages 13–16, Washington, Octobre 1995.
- [21] C. Fiorio. A topologically consistent representation for image analysis : the frontiers topological graph. In *Proceedings of the 6th Conference of Discrete Geometry for Computational Imagery*, Lyon, France, 1996.
- [22] C. Fiorio. Border map : A topological representation for nd image analysis. In *Proceedings of Conference of Discrete Geometry for Computational Imagery*, pages 242–257, Marne la Vallée, France, Mars 1999.
- [23] C. Fiorio. Topological operators on the frontiers topological graph. In *Proceedings of Conference of Discrete Geometry for Computational Imagery*, pages 207–217, Marne la Vallée, France, Mars 1999.
- [24] J. Froment. A compact and multiscale image model based on level sets. In *Proc. of the 2nd Workshop on Scale-Space Theories in Computer Vision*, pages 152–163, Corfu, Greece, 1999.

- [25] M. Gangnet, J.C. Hervé, T. Pudet, and J.M. Van Thong. Incremental computation of planar maps. *Digital Pattern Recognition Letters*, (5), 1989.
- [26] F. Guichard and J.M. Morel. Image iterative smoothing and P.D.E.'s. Book in preparation, 2000.
- [27] H. Hahn. Über die Komponenten offener Mengen. *Fund. Math.*, 2 :189–192, 1921.
- [28] T.S. Huang, G.J. Yang, and G.Y. Tang. A fast two-dimensional median filtering algorithm. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 27 :13–18, Février 1979.
- [29] R.A. Hummel. Representations based on zero-crossings in scale-space. In *Proceedings of the IEEE International Conference of Computer Vision and Pattern Recognition*, pages 204–209, 1986.
- [30] J.J. Koenderink. The structure of images. *Biological Cybernetics*, 50 :363–370, 1984.
- [31] T.Y. Kong and A. Rosenfeld. Digital topology : Introduction and survey. *Computer Vision, Graphics and Images Processing*, 48(3) :357–393, Décembre 1989.
- [32] T.Y. Kong and A. Rosenfeld. If we use 4- or 8-connectedness for both the objects and the background, the Euler characteristic is not locally computable. *Pattern Recognition Letter*, 11 :231–232, 1990.
- [33] V.A. Kovalevsky. Finite topology as applied to image analysis. *Computer Vision, Graphics and Image Processing*, 46(2) :141–161, Mai 1989.
- [34] A.S. Kronrod. On functions of two variables. *Uspehi Mathematical Sciences*, 5(35) :24–134, 1950. (en russe).
- [35] C. Kuratowski. *Topologie, I et II*. Jacques Gabay, 1992.
- [36] C.N. Lee, T. Poston, and A. Rosenfeld. Holes and genus of 2d and 3d digital images. *Graphical Models and Image Processing*, 55(1) :20–yy, Janvier 1993.
- [37] C.N. Lee and A. Rosenfeld. Computing the Euler number of a 3d image. In *Proceedings of International Conference of Computer Vision*, pages 567–571, 1987.
- [38] P. Lienhardt. Topological methods for boundary representation : A survey. *Computer Aided Design*, 23(1) :59–81, 1989.
- [39] J.L. Lisani, L. Moisan, P. Monasse, and J.M. Morel. Affine invariant mathematical morphology applied to a generic shape recognition algorithm. In *Proceedings of International Symposium of Mathematical Morphology*, San Francisco, California, Juin 2000.

- [40] R. Lumia. A new three-dimensional connected components algorithm. *Computer Vision, Graphics and Image Processing*, 23(2) :207–217, Août 1983.
- [41] R. Lumia, L.G. Shapiro, and O.A. Zuniga. A new connected components algorithm for virtual memory computers. *Computer Vision, Graphics and Image Processing*, 22(2) :287–300, Mai 1983.
- [42] S. Mallat. *A Wavelet Tour of Signal Processing*. Academic Press, New York, 1998.
- [43] P. Maragos and F. Meyer. Nonlinear P.D.E.s and numerical algorithms for modeling levelings and reconstruction filters. In *Proc. of the 2nd Workshop on Scale-Space Theories in Computer Vision*, pages 363–374, 1999.
- [44] P. Maragos and R.W. Schafer. Morphological filters. part I : Their set-theoretic analysis and relations to linear shift-invariant filters. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 35 :1153–1169, 1987.
- [45] P. Maragos and R.W. Schafer. Morphological filters. part II : Their relations to median, order-statistic, and stack filters. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 35 :1170–1184, 1987.
- [46] D. Marr. *Vision : A Computational Investigation into the Human Representation and Processing of Visual Information*. W.H. Freeman and Co., 1982.
- [47] D. Marr and E.C. Hildreth. Theory of edge detection. *Proceedings of the Royal Society of London*, B-207 :187–217, 1980.
- [48] S. Masnou. *Filtrage et Désocclusion d’Images par Méthodes d’Ensembles de Niveau*. PhD thesis, Université Paris IX-Dauphine, Paris, France, 1998.
- [49] S. Masnou. Image restoration involving connectedness. In *Proceedings of the 6th International Workshop on Digital Image Processing and Computer Graphics*, volume 3346, Vienna, Austria, 1998. SPIE.
- [50] G. Matheron. *Random Sets and Integral Geometry*. John Wiley, N.Y., 1975.
- [51] F. Meyer. *Mathematical Morphology and Its Application to Signal and Image Processing*, chapter From Connected Operators to Levelings. Kluwer Academic Publishers, h. heijmans and j. roerdink edition, 1998.
- [52] F. Meyer. *Mathematical Morphology and Its Application to Signal and Image Processing*, chapter The Levelings. Kluwer Academic Publishers, H. Heijmans and J. Roerdink edition, 1998.
- [53] F. Meyer and P. Maragos. Morphological scale-space representation with levelings. In *Proc. of the 2nd Workshop on Scale-Space Theories in Computer Vision*, pages 187–198, 1999.

- [54] Y. Meyer. *Wavelets : Algorithms and Applications*. SIAM, Philadelphia, 1993.
- [55] J. Milnor. *Morse Theory*. Number Study 51 in Annals of Mathematics Studies. Princeton University Press, 1969.
- [56] L. Moisan. Affine plane curve evolution : A fully consistent scheme. *IEEE Transactions on Image Processing*, 7(3) :411–420, Mars 1998.
- [57] P. Monasse. Contrast invariant image registration. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, volume 6, pages 3221–3224, Phoenix, Arizona, 1999.
- [58] P. Monasse and F. Guichard. Scale-space from a level lines tree. In *Proceedings of the 2nd Workshop on Scale-Space Theories in Computer Vision*, Lecture Notes in Computer Science, 1682, pages 175–186, Corfu, Greece, Septembre 1999. to be republished as an article in *Journal of Visual Communication and Image Representation*.
- [59] P. Monasse and F. Guichard. Fast computation of a contrast-invariant image representation. *IEEE Transactions on Image Processing*, Mai 2000.
- [60] J.M. Morel and S. Solimini. *Variational Methods in Image Processing*. Birkhäuser, 1994.
- [61] D. Mumford and J. Shah. Optimal approximations by piecewise smooth functions and variational problems. *Communications on Pure and Applied Mathematics*, XLII(5) :577–685, 1988.
- [62] M.H.A. Newman. *Elements of the Topology of Plane Sets of Points*. Dover, 1992.
- [63] M. Nitzberg and D. Mumford. The 2.1-D sketch. In *Proceedings of the 3^d International Conference on Computer Vision*, pages 138–144, Osaka, Japan, 1990.
- [64] S. Osher and L.I. Rudin. Feature-oriented image enhancement using shock filters. *SIAM Journal of Numerical Analysis*, 27(4) :919–940, 1990.
- [65] B.S. Reddy and B.N. Chatterji. An FFT-based technique for translation, rotation, and scale-invariant image registration. *IEEE Transactions on Image Processing*, 5(8) :1266–1271, Août 1996.
- [66] T.H. Reiss. *Recognizing Planar Objects Using Invariant Image Features*, volume 676 of *Lecture Notes in Computer Science*. Springer Verlag, 1993.
- [67] A. Rosenfeld. Connectivity in digital pictures. *Journal of the ACM*, 17(1) :146–160, Janvier 1970.

- [68] A. Rosenfeld. Arcs and curves in digital pictures. *Journal of Applied and Computational Mathematics*, 20(1) :81–87, Janvier 1973.
- [69] A. Rosenfeld. Adjacency in digital pictures. *Information and Control*, 26, 1974.
- [70] A. Rosenfeld and A.C. Kak. *Digital Picture Processing*. Academic Press, 1982.
- [71] A. Rosenfeld and J.L. Pfaltz. Sequential operations in digital picture processing. *Journal of Applied and Computational Mathematics*, 13(4) :471–494, Octobre 1966.
- [72] L.I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D*, (60), 1990.
- [73] P. Salembier. Morphological multiscale segmentation for image coding. *IEEE Transactions on Signal Processing*, 38(3) :359–386, 1994.
- [74] P. Salembier. Region-based filtering of images and video sequences : A morphological viewpoint. Preprint, 2000.
- [75] P. Salembier, P. Brigger, J.R. Casas, and M. Pardas. Morphological operators for image and video compression. *IEEE Transactions on Image Processing*, 5(6) :881–898, Juin 1996.
- [76] P. Salembier and L. Garrido. Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval. *IEEE Transactions on Image Processing*, 9(4) :561, Avril 2000.
- [77] P. Salembier, F. Meyer, P. Brigger, and L. Bouchard. Morphological operators for very low bit rate video coding. In *Proceedings of International Conference of Image Processing*, page 19P1, 1996.
- [78] P. Salembier, A. Oliveras, and L. Garrido. Antiextensive connected operators for image and sequence processing. *IEEE Transactions on Image Processing*, 7(4) :555–570, Avril 1998.
- [79] P. Salembier and H. Sanson. Robust motion estimation using connected operators. In *Proceedings of International Conference of Image Processing*, pages I :77–xx, 1997.
- [80] P. Salembier and J. Serra. Flat zones filtering, connected operators and filters by reconstruction. *IEEE Transactions on Image Processing*, (4) :1153–1160, 1995.
- [81] G. Sapiro and A. Tannenbaum. Affine invariant scale-space. *International Journal of Computer Vision*, 11(1) :25–44, Août 1993.
- [82] R. Sedgewick. *Algorithms in C++ : Fundamentals, Data Structures, Sorting, Searching*. Addison-Wesley, 3 edition, 1999.

- [83] J. Serra. *Image Analysis and Mathematical Morphology*. Academic Press, New York, 1982.
- [84] J. Serra. Introduction to mathematical morphology. *Computer Vision, Graphics and Image Processing*, 35(3) :283–305, Septembre 1986.
- [85] J. Serra and P. Salembier. Connected operators and pyramids. In *Proceedings of SPIE Conference on Image Algebra and Mathematical Morphology*, volume 2030, pages 65–76, San Diego, California, 1993.
- [86] B.M. ter Haar Romeny, editor. *Geometry-Driven Diffusion in Computer Vision*. Kluwer Academic Publishers, 1994.
- [87] Q. Tian and M.N. Huhns. Algorithms for subpixel registration. *Computer Vision, Graphics and Image Processing*, 35(2) :220–233, Août 1986.
- [88] L. Vincent. Grayscale area openings and closings, their efficient implementation and applications. In J. Serra and Ph. Salembrier, editors, *Proceedings of the 1st Workshop on Mathematical Morphology and its Applications to Signal Processing*, pages 22–27, Barcelona, Spain, 1993.
- [89] L. Vincent. Morphological area openings and closings for grey-scale images. In *Proceedings of the Workshop Shape in Picture : Mathematical Description of Shape in Gray-Level Images*, pages 197–208, Driebergen, The Netherlands, 1994. Springer, Berlin.
- [90] C.Y. Wang, H. Sun, S. Yada, and A. Rosenfeld. Some experiments in relaxation image matching using corner features. *Pattern Recognition*, 16(2) :167–182, 1983.
- [91] Y. Wang and P. Bhattacharya. Hierarchical stereo correspondence using features of gray connected components. In *Proceedings of IEEE International Conference on Image Processing*, pages 264–267, Santa Barbara, California, 1997.
- [92] M. Wertheimer. Untersuchungen zur Lehre der Gestalt, ii. *Psychologische Forschung*, (4) :301–350, 1923.
- [93] A.P. Witkin. Scale-space filtering. In *International Joint Conference on Artificial Intelligence*, pages 1019–1022, Karlsruhe, 1983.
- [94] L. Yaroslavsky and M. Eden. *Fundamentals of Digital Optics*. Birkhäuser, 1996.