Approche interpixel en analyse d'images : une topologie et des algorithmes de segmentation

Christophe Fiorio

Numéro d'identification : 95MON20179

Académie de Montpellier

Université Montpellier II

— **Sciences et Technique du Languedoc** —

Thèse

présentée à l'Université des Sciences et Techniques du Languedoc
pour obtenir le diplôme de DOCTORAT

Spécialité	: **Informatique**
Formation Doctorale	: **Informatique**
École Doctorale	: **Sciences pour l'Ingnieur**

Approche interpixel en analyse d'images, une topologie et des algorithmes de segmentation

par

Christophe Fiorio

Soutenue le 24 Novembre 1995 devant le Jury composé de :

M. Michel Chein, Professeur, Univ. Montpellier II, LIRMM, ... Président
M. Jean Françon, Professeur, Univ. L. Pasteur Strasbourg, LSIIT, ... Rapporteur
M. Jens Gustedt, Technische Universität Berlin, IFP Digitale Filter, ... Rapporteur
M. Walter Kropatsch, Professeur, Technische Universität Wien, PRIP, ... Rapporteur
M. Ehoud Ahronovitz, Maître de Conférence, Univ. Montpellier II, LIRMM, ... Examinateur
M. Jean-Pierre Aubert, Maître de Conférence, Univ. Montpellier II, LIRMM, ... Examinateur
M. Philippe Montesinos, Maître Assistant, EMA–EERIE, LGI2P, ... Examinateur
M. Michel Habib, Professeur, Univ. Montpellier II, LIRMM, ... Directeur de Thèse

Remerciements

Cette thèse est l'aboutissement de trois ans d'études. Elle symbolise également trois ans d'une vie. Avant de tourner la page, j'aimerais remercier certaines personnes et leur exprimer ma sympathie, soit pour l'intérêt qu'elles ont porté à mon travail, soit simplement pour leur « contribution » à cette tranche de vie.

Je remercie vivement Michel Chein pour avoir accepté de présider le jury de ma thèse et pour le compliment qu'il m'a fait après la soutenance.

J'ai beaucoup apprécié la lecture attentive et les commentaires aussi bien instructifs que constructifs de M. Jean Françon. J'ai vraiment regretté qu'il ne puisse venir assister à ma soutenance à cause des grèves. Mais les commentaires et compliments me concernant qu'il a adressé aux membres du jury m'ont particulièrement touché et je lui en suis reconnaissant.

Cette thèse doit beaucoup à Jens Gustedt. Il a d'abord été le premier à s'intéresser à mes travaux. Les discussions que nous avons eues alors, ont débouché sur une collaboration fructueuse. Je pense qu'il a été le véritable initiateur de mes travaux et sans lui j'aurais sans doute eu beaucoup plus de mal à trouver une voie de recherche. Je lui adresse toute ma gratitude et ma reconnaissance.

Je voudrais également exprimer toute mes remerciements à M. Walter Kropatsch pour s'être intéressé à mes travaux, puis pour m'avoir fait l'honneur d'accepter d'être rapporteur. Ses nombreuses remarques ont beaucoup contribué à la version finale de ce document. Ses suggestions m'ont ouvert de nouvelles perspectives pour mes travaux. Pour cela je le remercie également.

Travailler avec Philippe Montésinos m'a beaucoup apporté, tant au niveau relationnel, que scientifique. Sans lui je n'aurais pas une connaissance aussi approfondie de l'aspect traitement du signal de l'analyse d'images. Je l'en remercie et lui dois donc beaucoup.

Jean-Pierre Aubert est une personne admirable. J'ai un respect certain pour l'homme ainsi que pour le scientifique. Sans lui notre travail sur la topologie des images ne serait sûrement pas aussi rigoureux. Je le remercie encore d'avoir quelque peu délaissé sa logique temporelle pour s'intéresser à mes pixels et autres voxels.

Ehoud Ahronovitz est arrivé à point, au moment où le doute s'installait en moi. Son oreille attentive (et patiente !), ses remarques, ainsi que son soutien ne sont pas étrangersà l'aboutissement de ces travaux. En cela je le remercie grandement. Par ailleurs je voudrais souligner les qualités humaines extraordinaires du personnage. J'ai beaucoup apprécié l'ambiance de travail mais aussi les nombreuses discussions (pas forcément sérieuses d'ailleurs) que nous avons eues. J'ai également particulièrement été touché par son dévouement et par l'amitié qu'il m'a témoignée. Je voudrais m'excuser une dernière fois auprès de lui pour l'avoir obligé à rester (très) tard le soir ou même à travailler le week-end.

Michel Habib a bien voulu être mon directeur de thèse. Je lui en suis reconnaissant. J'avais déjà entrevu ses connaissances et compétences scientifiques. J'ai découvert sa simplicité et sa chaleur humaine. J'ai beaucoup apprécié la confiance qu'il m'a témoignée ainsi que l'autonomie dont j'ai pu bénéficier quant à la conduite de mes travaux. D'autre part, il a su par quelques remarques me remotiver quand il le fallait. Tout cela a constitué un très bon apprentissage de la recherche et je lui en serai toujours redevable. Enfin l'intérêt qu'il a porté à mes résultats sont une source de motivation énorme pour la poursuite de mes recherches.

Je remercie également tous mes compagnons de cafétéria, et parmi ceux-là je tiens à exprimer ma profonde amitié envers Philippe Baldy pour les nombreux moments partagés. Je voudrais également remercier Philippe Charnier, ex-membre de l'équipe image, sans qui je n'aurais pas pu être appelé par Raoul Médina, « service technique officieux ». Je remercie d'ailleurs ce dernier pour sa bonne humeur, ses plaisanteries et ses e-mails fameux. En parlant de messages fameux, je voudrais remercier Jean-Charles Régin, non pas pour son côté « boute-en-train », mais pour m'avoir fait découvrir le Vin.

Je tiens également à remercier l'équipe d'enseignants avec qui j'ai pu travailler, et particulièrement Thérèse Libourel qui est une personne remarquable que j'apprécie énormément.

Enfin je remercie particulièrement mon épouse Sylvie pour avoir supporté mes horaires particuliers, mes sautes d'humeurs et pour le soutien moral qu'elle a su

m'apporter.

Sommaire

Chapitre 1

Contexte de l'étude

Cette étude s'inscrit dans les différents travaux menés par l'équipe Image du **D**épartement d'**I**nformatique **F**ondamentale du LIRMM[1] et plus particulièrement du groupe **Al**gorithmique **Co**mbinatoire et **Lo**gique du DIF. Les centres d'intérêt de ce groupe sont la théorie des ordres, le maniement des structures discrètes, notamment les graphes et les treillis, et plus particulièrement la conception d'algorithmes. Le traitement d'images est une préoccupation récente de ce groupe. Dans ce cadre j'ai participé aux débuts de l'équipe Image ainsi qu'à ses premières publications. L'orientation de mes travaux, la façon d'aborder les problèmes et même celle de les traiter ont donc fortement été influencées par cet environnement, cette culture.

Dans ce travail j'aborde essentiellement les problèmes de la représentation des images dans le domaine de l'analyse d'images, à travers une approche dite « *interpixel* ». Il s'agit d'images de tout type, binaires, en niveaux de gris, en 2D ou 3D, et de leur représentation dans les diverses étapes du processus d'analyse. L'approche proposée consiste à ne plus considérer une image comme un ensemble de pixels ou voxels mais comportant aussi des éléments liant les pixels ou voxels entre eux. Ceci m'a amené à étudier la topologie des images. J'ai cherché à définir une topologie, adaptée aux espaces discrets que sont les images, mais restant le plus proche possible de la topologie classique dans $\mathbb{R}^n$. Cette première étude a permis de proposer un modèle formel d'images, la *topologie-étoile* « *star-topologie* » [AAF95], s'intégrant parfaitement à l'approche interpixel. Je montre également que

[1] Laboratoire d'Informatique, de Robotique et de Micro-électronique de Montpellier.

cette vision n'est pas en contradiction avec les approches classiques de détection de contours en proposant une adaptation du filtre de R. Deriche [Der87] en inter-pixel. Ce détecteur peut parfaitement s'intégrer dans le cadre d'une coopération régions-contours, grâce, notamment, aux algorithmes de segmentation en régions et d'étiquetage des composantes connexes [FG96] présentés dans ce document. Enfin la représentation par graphe d'adjacence est étendue à celle de graphe topologique des frontières afin de prendre en compte la topologie des images.

Ces travaux font l'objet de plusieurs collaborations entreprises par notre équipe et nous permettent de valider sur des cas concrets nos algorithmes et méthodes. De plus les discussions avec nos partenaires nous ont amenés à nous poser des questions nouvelles et ont permis l'émergence de problèmes théoriques intéressants et difficiles, comme ceux relatifs à la perception et la représentation des images. Le chapitre 2 situe ces problèmes dans le domaine très vaste de l'analyse d'images. Du fait de ces collaborations, nous travaillons plus particulièrement dans le domaine de l'imagerie médicale et celui très particulier de l'analyse d'images à la volée [ACF94]. Dans ce dernier domaine une application au contrôle qualité de la fabrication du coton est en cours de développement.

La collaboration entreprise avec l'équipe « modélisation » du CIRAD[2] m'a aussi amené à travailler sur différents types d'images médicales. En effet cette équipe spécialisée dans la synthèse d'images et plus particulièrement la modélisation et la visualisation de la pousse des végétaux a décidé de lancer un projet de modélisation d'images médicales en trois dimensions. Les applications et types d'images à traiter sont multiples et en particulier comprennent une phase importante d'analyse afin d'extraire les données à visualiser. Le chapitre 2 de ce document donne un aperçu des difficultés liées à l'analyse d'images en général et des outils associés : méthodologie, segmentation, ...

Le besoin d'analyser des surfaces et des volumes a créé un terrain favorable à l'approche interpixel présentée dans ce document. Cette approche s'appuie sur le besoin de déterminer clairement les bords et frontières entre différents objets voisins ; on ne peut oser ici encore dire adjacent, car ce terme revêt une signification très précise en analyse d'images et mérite quelques développements. C'est ainsi que nous avons lancé cette collaboration portant sur la représentation et la

[2]Centre Internationale de Recherche en Agronomie pour le Développement

segmentation d'images.

La représentation devant s'appuyer sur un modèle cohérent, il m'a semblé utile de me pencher sur la représentation topologique des images. Cette démarche s'est ensuite étendue à un travail d'équipe et nous a amenés à proposer une topologie que nous avons baptisée « *topologie-étoile* » qui permet à la fois :

- d'obtenir une représentation cohérente des images en levant des ambiguïtés liées à la connexité, l'adjacence, et en général aux frontières des images ;
- d'être très proche de la topologie classique dans $\mathbb{R}^n$, ce qui nous permet de récupérer facilement les théorèmes de la géométrie classique.

Le chapitre 3 détaille cette partie de mon travail en partant de la topologie et en allant aux problèmes spécifiques à l'analyse d'images.

D'autre part notre équipe devrait à long terme s'orienter vers le développement d'un système de vision complet et donc effectuer des recherches dans le domaine de la reconnaissance. En effet notre culture algorithmique de graphes nous incline à ne pas perdre de vue cet aspect du problème de la vision par ordinateur. Il faut donc également que les algorithmes développés permettent le passage à une structure de données de plus haut niveau, une structure de graphe adéquate par exemple. Je propose, dans le chapitre 4, une structure de données appelée « *le graphe topologique des frontières* » [Fio96], ainsi que l'algorithme permettant de l'extraire de l'image. Cette structure de données est une extension du graphe d'adjacence et permet de mieux rendre compte de la topologie de l'image.

L'analyse d'images à la volée présente un autre domaine dans lequel nous travaillons avec le CIRAD cité déjà précédemment. Dans un domaine totalement différent, celui du contrôle qualité où l'image défile sans interruption, il s'agit de mettre en œuvre des algorithmes rapides qui n'aboutissent pas à une explosion de l'espace mémoire. Une première étude a permis de valider nos idées [Cha95, Kam95] : on pourrait balayer une image et construire le graphe d'adjacence de façon continue tout en libérant l'espace relatif aux objets entièrement traités, le graphe d'adjacence, lui aussi, se voyant régulièrement augmenter puis diminuer en taille. Actuellement cette étude se poursuit avec une adaptation de l'algorithme `ScanLine` présenté au chapitre 5. En fait cet algorithme fait partie d'une démarche qui nous permet de diversifier les applications sans pour autant développer des algorithmes spécifiques à un type particulier d'images.

On peut présenter le problème ainsi : soit trouver des outils d'analyse utilisables

quelle que soit l'image à traiter, c'est ce qui a été fait avec l'adaptation en interpixel et en subpixel des détecteurs de contour du type Canny-Deriche [Can86, Der87], soit avoir un algorithme capable de s'adapter à l'application ; c'est le principe de l'Oracle, fonction d'évaluation utilisée par les algorithmes `MergeSquare` et `ScanLine` mais dont il est indépendant. Ces algorithmes sont le résultat d'une étude menée conjointement avec J. Gustedt. Le but était de montrer l'efficacité des méthodes combinatoires pour la segmentation d'images, en opposition avec les techniques habituelles, généralement issues du domaine du traitement du signal. Par ailleurs plutôt que de demander un réglage de trop nombreux paramètres, difficiles à maîtriser et encore plus difficiles à associer, nous pensons qu'il vaut mieux n'en laisser que très peu à l'utilisateur. C'est ainsi qu'en plus des deux algorithmes de segmentation présentés dans le chapitre 5 nous proposons une méthode « d'unification » des multiples paramètres que pourraient engendrer de tels algorithmes permettant la combinaison de nombreux critères locaux et globaux.

D'une façon générale, on assiste aujourd'hui à des méthodes alliant des techniques orientées contours et régions. Les travaux présentés ci-dessus ont permis la mise en place d'algorithmes efficaces dans le domaine de la segmentation en régions liés à la représentation interpixel. Ils m'ont naturellement amené à chercher les améliorations que pourrait apporter une technique liée au filtrage (approche dite « contour »). La difficulté ici provient du besoin de filtres ne travaillant plus sur les pixels de bord mais sur les éléments de bord eux-mêmes situés entre les pixels, conformément à notre approche. C'est ainsi que j'ai développé, en collaboration avec P. Montésinos du LGI2P[3] un détecteur de contour en interpixel, présenté dans le chapitre 6. Nous montrons qu'on obtient alors une meilleure précision dans la localisation des contours. Nous pensons de plus pouvoir mieux détecter les contours marquant une forte variation sur une faible largeur du signal.

En fin de compte, on peut présenter cette thèse comme un travail qui nous permet de fournir **des algorithmes génériques d'analyse bas-niveau d'images** variées avec en final, **une représentation**, s'appuyant sur **un modèle formel** bien établi, permettant le passage à des traitements de plus haut niveau. L'ordre de présentation adopté dans ce document est en fait une remise en forme des recherches effectuées pendant ces trois années. La ligne directrice choisie corresponda

[3]Laboratoire de Génie Informatique et d'Ingéniérie de Production

à l'apport essentiel de ces travaux, c'est à dire le développement d'un formalisme topologique combinant et adaptant différentes approches existantes en analyse d'images.

Ces travaux ont donné lieu à plusieurs publications ou articles en cours de soumission. Ainsi l'étude sur une topologie adaptée à l'analyse d'images, présentée au chapitre 3, a été publiée au « $5^t h$ colloquium Discrete Geometry for Computer Imagery » [AAF95] Les travaux, concernant l'utilisation du graphe d'adjacence pour l'analyse d'images à la volée, qui ont donné lieu aux développement du chapitre 4, ont été publiés dans les actes de la conférence « Machine Vision Application » [ACF94] qui s'est déroulée au Japon en 1994. L'étude présentée au chapitre 4 a été publiée dans les actes de la conférence Discrete Geometry for Computer Imagery [Fio96]. Quant aux algorithmes présentés au chapitre 5 ils seront publiés prochainement dans la revue internationale « Theoretical Computer Science » [FG96].

Chapitre 2

Analyse d'image : aperçu et problématique

L'analyse d'images s'inscrit dans une problématique plus générale appelée la *vision par ordinateur*. Mais qu'est-ce que la vision par ordinateur ? Pour R.M. Haralick et L.G. Shapiro [HS92], la vision est la science définissant les bases théoriques et algorithmiques permettant d'extraire, d'analyser automatiquement des informations sur notre environnement à partir d'une image, un ensemble d'images ou une séquence d'images, sur une machine spécialisée ou non. Cette définition laisse imaginer combien est vaste ce domaine de recherche. En effet les applications répondant à une telle définition sont nombreuses et vont de la vision pour un robot autonome à la détection de défauts sur un élément, en passant par la reconnaissance d'objets bien définis et caractérisés. Il est donc impératif de réfléchir à la manière d'aborder ce vaste problème.

2.1 Problématique de la vision.

Mais qu'est-ce que la vision et comment fonctionne-t-elle chez l'homme ? D. Marr ([Mar82a] page 31) définit la vision comme « un processus qui produit à partir d'images provenant du monde extérieur une description utile à l'observateur et non perturbée par de l'information non pertinente ». C'est bien le but recherché dans la vision par ordinateur, l'observateur pouvant être un opérateur humain ou un autre

processus automatisé de décision. Le fonctionnement de la vision chez l'homme reste complexe, mais nous voudrions porter l'attention sur trois caractéristiques principales relevées par M. Salotti [Sal94] dans les travaux de W. Freeman [Fre91]

1. Très grande rapidité de la reconnaissance visuelle : en fait, on sait maintenant que cette reconnaissance est due à un enchaînement d'étapes simples, sans remise en cause des résultats intermédiaires.
2. Spécialisation des traitements : en effet des expériences ont montré que certaines zones du cerveau étaient sensibles à la couleur alors que d'autres sont sensibles à la forme. On peut donc en déduire qu'un groupe de neurones donné va être dédié à une tâche particulière du processus de reconnaissance.
3. Capacité d'apprentissage énorme : nous sommes capables de mémoriser en quelques secondes un nouveau visage.

Le premier point évoqué ici nous semble important et nous y reviendrons plus tard. La spécialisation des traitements est un fait acquis dans le domaine de la vision et du traitement d'images. En effet les algorithmes existants sont nombreux, variés et surtout très spécialisés. Ainsi, dans la recherche des éléments significatifs et informatifs de l'image, on peut rencontrer des algorithmes de détection de coins (par exemple [DG93]), de courbes, d'arêtes,... Mais également des algorithmes spécifiques à un problème particulier comme le traitement d'images aériennes [Mon90], médicales, ... Quant au troisième point, on est loin d'atteindre de telles performances. En effet les méthodes de reconnaissance actuelles supposent qu'une première analyse de l'image a déjà été faite et a permis de dégager les éléments intéressants de l'image (voir par exemple [SB93]). Or ces opérations prennent déjà plusieurs secondes voire des minutes. Ensuite, en supposant que cette première étape s'est déroulée idéalement, il reste à effectuer la reconnaissance proprement dite. Et là on se trouve confronté à des problèmes difficiles liés aux méthodes d'appariements, de recherche de motifs et d'isomorphisme de sous-graphe [Vos92]. Nous n'aborderons pas le problème de la vision et de la reconnaissance dans cette thèse. Pour avoir une idée des problèmes posés par la reconnaissance d'objets on pourra se reporter à l'étude de C.G. Perrot et L.G.C. Hamey [PH91], et pour la vision en général, aux ouvrages classiques maintenant comme : [YF86, HS92, HM93].

Revenons à la première caractéristique, évoquée ci-dessus, de la vision chez l'homme. Il est dit que la reconnaissance est en fait un enchaînement d'étapes

simples et sans remise en cause de l'une à l'autre. Dans la vision par ordinateur cette façon de procéder risque de devenir obligatoire. En effet les algorithmes existants sont très spécialisés et l'aboutissement final qu'est la vision ou plus modestement la reconnaissance d'un objet va nécessiter plusieurs étapes permettant de passer successivement de l'icone (les données de l'image digitale) à une information identifiée et exploitable. Un tel enchaînement de processus différents nécessite de préparer les transitions entre les différentes étapes, mais aussi de définir ces étapes. En résumé, il faut élaborer une méthodologie afin que chacun puisse se situer et définir correctement le but recherché.

2.2 Essais de méthodologie.

En fait plusieurs méthodologies ont été proposées dans la littérature, certaines semblant s'opposer à d'autres, toutes ne suivant pas la même démarche (pour une étude du problème de la méthodologie en vision par ordinateur, le lecteur pourra se reporter à l'étude de J.M. Jolion [Jol94]). D. Marr [Mar82b], le premier, a proposé vers la fin des années 70 un cadre de base pour la vision qu'il considère comme un système de traitement de l'information. De ce paradigme se dégagent trois grandes étapes qui se retrouvent pratiquement dans toutes les méthodologies existantes et qui font pratiquement l'unanimité : segmentation, reconstruction et reconnaissance. Afin de détailler un peu ces trois étapes nous allons présenter la méthodologie prônée dans [HS92]. Elle se décompose en cinq parties :

1. Conditionnement.

   On considère qu'une image est le résultat d'une transformation perturbée par des variations qui amplifient, atténuent et modifient l'information initiale. Ce modèle est directement issu des méthodes de traitement du signal. Le conditionnement consistera donc à supprimer le bruit introduit par le capteur. Il fait appel à des techniques de filtrage, mais également à des méthodes de morphologie mathématique, ou encore à des traitements sur la répartition de l'intensité lumineuse dans l'image à partir de l'histogramme des niveaux de gris, ...

2. Étiquetage.

On suppose ici que ce que l'on recherche est structuré en différents types d'information et que chaque structure peut être représentée par un ensemble de pixels connexes. L'étiquetage consiste alors à associer une étiquette à chaque pixel, c'est à dire à déterminer quel type d'information véhicule ce pixel. On regroupe sous ce label les processus de seuillage, de détection d'arêtes, de coins, d'appartenance à une forme primitive, ...

3. Groupement.

   On cherche maintenant à identifier des ensembles maximaux de pixels connectés ayant la même étiquette. Les étiquettes des pixels peuvent être symboliques ; on a alors réellement une opération de recherche de composantes connexes. Mais on peut également réduire l'opération d'étiquetage à sa plus simple expression et prendre pour étiquette l'information pixel elle-même. Dans les images en niveaux de gris cela revient alors à déterminer un ensemble de pixels ayant des niveaux de gris semblables ; ce traitement est appelé *segmentation en région*.

4. Extraction.

   L'opération de regroupement a permis de déterminer un nouvel ensemble d'entités. Mais elles sont nues : il faut maintenant déterminer les propriétés de ces entités (barycentre, surface, orientation, distribution des niveaux de gris, ...) en fonction de leur type (région, arête, coin, ...) et également les relations spatiales et topologiques les reliant les unes aux autres.

5. Appariement.

   On a repéré certaines informations dans l'image, il faut maintenant associer une sémantique, c'est à dire retrouver l'organisation perceptuelle de ces entités et les mettre en correspondance avec le modèle recherché, la description symbolique de l'objet.

On retrouve bien le paradigme de D. Marr, c'est à dire segmentation (représentée ici par conditionnement, étiquetage et groupement), reconstruction (extraction) et reconnaissance (définie ici comme l'appariement). Il faut également remarquer qu'en pratique la segmentation n'est pas toujours décomposée en trois parties comme suggéré par cette méthodologie. Les algorithmes de détections d'arêtes, par exemple, regroupent le conditionnement et l'étiquetage.

2.3 De la définition d'une image.

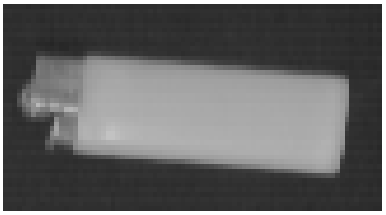
Il apparaît que définir une méthodologie et une méthode générale pour l'analyse d'images est actuellement très difficile, tant sont nombreux et différents les problèmes abordés. De même il est délicat d'élaborer un système complet de vision par ordinateur. Nous allons alors nous intéresser plus particulièrement à la première phase, c'est à dire la segmentation. Le but d'un processus de segmentation est d'extraire les attributs caractérisant les entités représentées dans l'image, ou encore de décrire l'image à l'aide d'indices visuels simples.

Mais avant d'aborder le problème même de la segmentation, il faut savoir quelles sont les données dont on dispose, c'est à dire préciser ce qu'est une image. Quand on dit image, sans référence à l'informatique, on parle d'une représentation spatiale d'un objet, d'une scène ou d'une autre image. Elle peut être réelle ou virtuelle, c'est à dire issue d'un procédé de synthèse. Et en analyse d'images ? Généralement on travaille sur des images digitales obtenues à partir du monde réel. On peut alors la représenter par une fonction $I(x, y)$ où x et y sont les coordonnées d'un point de l'image et $I(x, y)$ représente l'information observée. Cette valeur est généralement proportionnelle à l'énergie rayonnante reçue dans la bande de fréquences électromagnétiques de sensibilité du capteur dans une petite zone autour du point de coordonnées (x, y), c'est à dire un niveau de gris. On appelle ces images des images intensitées, ou niveaux de gris. Si le capteur possède plusieurs bandes de fréquences électromagnétiques on obtient une image couleur. Ce type d'image est bien évidemment le plus répandu et le plus connu. Mais il existe d'autres types d'images. Par exemple les *« range image »* où la valeur $I(x, y)$ représente la distance entre le point de coordonnées (x, y) et un objet dans « le monde réel ». On peut également obtenir à l'aide de capteurs tactiles une image dont les valeurs sont proportionnelles à la déformation de la surface autour, du point (x, y). Nous nous intéresserons uniquement dans ce document aux images intensitées. Elles sont généralement représentées par une matrice où chaque valeur est une valeur discrète d'intensité.

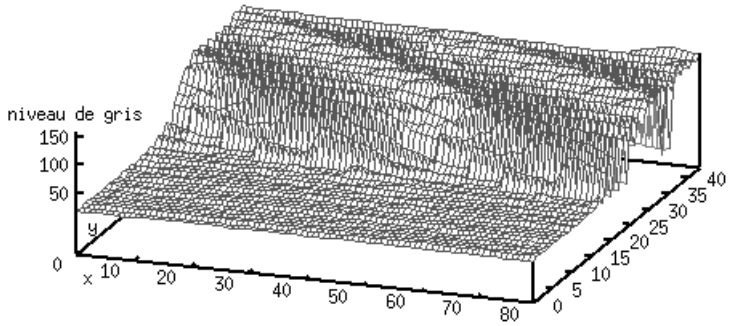
2.4 Des difficultés de la segmentation.

Maintenant que nous avons clairement défini la donnée de notre problème, c'est à dire une image, nous allons essayer d'expliquer le but et la problématique de la segmentation. Le problème principal en vision est que l'unité d'observation n'est pas l'unité d'analyse. En effet l'unité de travail est le pixel dont les propriétés sont position et intensité. Cette connaissance n'amène rien quant à la vision de l'image. Il n'est pas question par exemple de pouvoir décrire la forme, le positionnement, l'orientation d'un objet. La représentation d'un objet dans l'image (sa projection) correspond bien au point de vue de cet objet sous une certaine orientation. Comment il apparaît dans l'image est en fait une configuration spatiale des valeurs de plusieurs pixels, et une valeur particulière apporte peu d'information. Donc l'agrément entre la projection dans l'image et la position observée, qui est le but final d'un système de vision, demande que l'on soit capable d'inférer implicitement ou explicitement la position d'un objet à partir de la configuration spatiale des pixels. Il faudra en outre être capable de valider, confirmer, cette inférence. Or tous les auteurs sont d'accord sur un point au moins : une image ne contient pas assez d'informations brutes permettant une reconstruction non ambigüe et complète d'un objet. Au contraire elle est constituée d'un flot d'informations primaires souvent non significatives en elle-mêmes. Il faut donc éliminer l'information non pertinente, repérer des attributs caractéristiques des objets recherchés (arêtes, coins, courbes, frontières, ...), puis reconnaître ceux faisant partie d'un même objet. Et enfin inférer sur la forme et la position de l'objet en faisant correspondre les attributs observés avec ceux de l'objet réel. On pourra pour cela utiliser également la projection géométrique due au capteur. On retrouve ici sous une autre forme les cinq parties de la méthodologie présentée plus haut. On peut également remarquer que tout le processus de reconnaissance dépend essentiellement des attributs repérés dans l'image. La segmentation qui est la première étape de tout système de vision est donc primordiale. C'est d'ailleurs un problème très étudié en analyse d'images (pour s'en convaincre le lecteur pourra se reporter aux états de l'art suivant [Zuc76, HS85, PP93]).

La difficulté du problème vient du fait que les données initiales (les pixels) véhiculent peu ou pas d'information et c'est la vue d'ensemble de tous ces pixels qui permet de dégager une information pertinente. Ainsi si l'on regarde la figure 2.1



a. image originale



b. fonction $I(x, y)$

FIG. 2.1 – visualisation d'une image par sa fonction $I(x, y)$

on peut voir la représentation d'une fonction image. On n'y discerne pas aussi nettement l'objet que dans l'image originale. D'autre part la position de l'objet, son environnement, le type de capteur, le point d'observation, tous ces paramètres influent sur la difficulté du problème. Prenons le cas d'un rectangle à reconnaître. Un algorithme de détection de coins devrait nous donner les quatres coins de ce rectangle. Si on connaît la taille initiale, on peut alors déterminer la position et l'orientation relative du plan du rectangle par rapport à la caméra. Mais en fait le problème peut ne pas être aussi simple. Si l'environnement du rectangle est complexe, il pourrait y avoir certaines parties cachées, dont les coins ; selon l'éclairage, des ombres ou un halo lumineux pourraient en fausser la détection. On voit que même dans un cas aussi simple, la reconnaissance peut s'avérer délicate. La segmentation d'images nécessite donc des algorithmes performants. C'est pourquoi,

ils sont souvent dédiés à un problème particulier.

Une méthode générale n'existe donc pas à l'heure actuelle, à notre connaissance. En effet les algorithmes existants sont d'une part spécialisés, et d'autre part sont souvent développés de manière indépendante et ne préparent pas le terrain, ni ne fournissent les outils permettant le passage à l'étape suivante du processus de reconnaissance. Nous pensons qu'il est important de toujours garder à l'esprit que tout algorithme, toute méthode de segmentation a pour but d'être intégré dans un processus complet d'analyse. Il doit donc fournir les interfaces avec les étapes suivantes du processus de reconnaissance.

Ainsi après une présentation de notre approche originale de la segmentation d'images, *l'analyse interpixel*, nous définissons une topologie adaptée à l'analyse d'images. Afin de montrer que cette approche est compatible avec les méthodes de détection de contours classiques où les arêtes sont représentées par des pixels, nous avons développé un algorithme de détection de contour en interpixel. Puis nous revenons à une étude combinatoire du problème et proposons un algorithme linéaire de segmentation en régions. Cet algorithme permet d'utiliser des critères locaux (liés au pixels) et globaux (liés aux régions). Grâce au détecteur interpixel précédemment évoqué, une coopération région-contour est donc envisageable. De plus la structure définie dans cet algorithme est utilisée dans le cadre d'un autre algorithme linéaire, permettant la mise en commun de plusieurs segmentations en régions. Puis afin de réaliser effectivement la jonction entre la segmentation et la suite du processus de reconnaissance, nous proposons une représentation basée sur notre modèle formel : *le graphe topologique des frontières*.

Chapitre 3

Approche interpixel et modélisation des images

Après une série de travaux menés en analyse d'images [Jac92, ACF94, Cha95, FG96], nous nous sommes trouvés confrontés au besoin d'un modèle cohérent de représentation des images. En effet, nos études sur la segmentation et la représentation sous forme de graphes d'adjacence et/ou des frontières, nous ont confirmé que les approches classiques de la connexité, ou dans la description des objets et de leurs caractéristiques, étaient insuffisantes ou même incohérentes. Dès lors, une étude de la topologie associée aux images devenait indispensable. Comme nous l'avons déjà dit, en analyse d'images l'unité de travail est le pixel dont les caractéristiques sont position et intensité lumineuse. À partir de ce constat, la plupart des travaux se sont concentrés sur cette donnée. Or nous pensons que considérer l'image seulement comme un ensemble de pixels est non seulement restrictif, mais aussi pose certains problèmes de représentation des informations recherchées. Ainsi si l'on veut désigner la frontière entre deux régions de l'image, sur quels pixels doit-on la situer ? Certes des travaux existent (voir par exemple dans [Pav77] la notion de contour étendu, *extended boundary*) qui essaient de donner une solution, mais aucune n'est complètement satisfaisante, et surtout, comme on le précisera dans ce chapitre, elles entraînent des incohérences topologiques. C'est pourquoi nous avions déjà éprouvé le besoins de coder les objets dans une image par leurs bords [Ahr85]. Néanmoins aucun modèle formel ne venait étayer cette approche pratique. De plus les bords des pixels ne semblent pas être les seuls éléments à

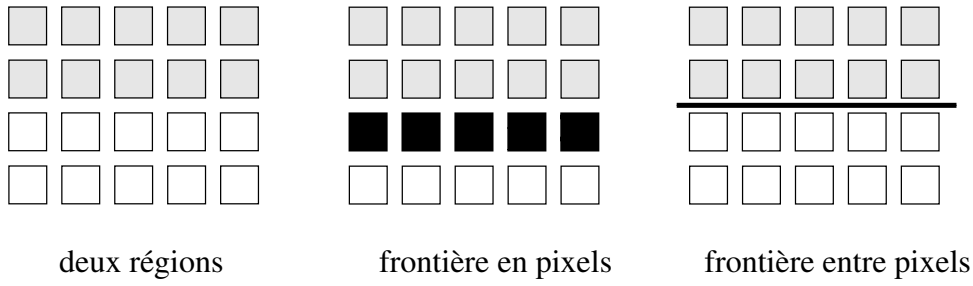


FIG. 3.1 – Deux régions et leur frontière commune

prendre en compte si l'on veut s'affranchir des incohérences topologiques, dues notamment aux définitions classiques de l'adjacence et de la connexité dans les images [KR89]. C'est dans cet état d'esprit que nous avons abordé l'étude des topologies « dites discrètes ». Les travaux de E. Khalimsky [KKM90b] et V.A. Kovalevsky [Kov89] nous ont confirmé la validité de notre approche du point de vue topologique. Dès lors la possibilité de définir une topologie, non seulement adaptée aux espaces discrets que sont les images, mais répondant aussi aux impératifs de l'analyse (définition de la notion d'ensemble homogène : les régions, de l'adjacence entre régions, ...) devenait envisageable.

3.1 Qu'est-ce que l'interpixel ?

Notre manière d'appréhender une image est différente puisque nous ne considérons pas une image comme seulement une matrice de pixels à k colonnes et l lignes, mais comme un espace dans lequel réside une information initiale : les pixels, mais où existe une autre information nécessaire à la cohérence de l'ensemble bien que non représentée : l'interpixel. Cette différence peut paraître subtile, voire futile, mais en fait elle change considérablement la vision des choses. En tout cas nous pensons qu'elle est naturellement plus proche de la réalité.

En effet si l'on présente à quelqu'un une image composée de deux régions distinctes (voir figure 3.1) et qu'on lui demande ou se situe la frontière, il répondra naturellement en montrant la ligne séparant ces deux régions. Or cette ligne n'est pas un ensemble de pixels. Doit-on alors considérer qu'il n'y a pas de frontière ?

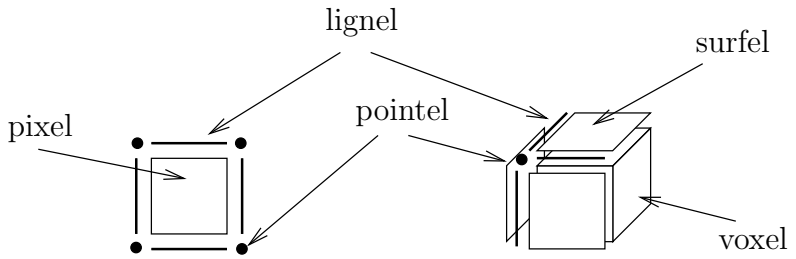


FIG. 3.2 – l'interpixel

Ou peut-être devons nous placer arbitrairement cette frontière dans une des deux régions. Les pixels désignés sont alors des pixels de l'intérieur de la région et aussi des pixels de la frontière, donc du bord ! Ne vaut-il pas mieux considérer qu'il existe une ligne entre les pixels représentant cette frontière ? D'une manière plus générale on dira que les pixels sont séparés par des *lignels* (lignes entre deux pixels correspondant à l'arête des pixels) et des *pointels* (sommets des pixels). Dans un espace à trois dimensions on trouvera également des surfels (face d'un voxel[1]) entre les voxels. La figure 3.2 présente la notion d'interpixel en deux et trois dimensions. On pourrait reprocher à cette approche de multiplier l'information, peut-être inutilement dans bien des cas. En fait cette information nous paraît indispensable si l'on veut avoir une définition de la connexité, des notions de frontière, bord, intérieur et extérieur aussi proche que possible de la géométrie euclidienne et de la topologie classique dans $\mathbb{R}^2$ et $\mathbb{R}^3$. Il paraît certes inutile de maintenir cette information pour l'intérieur d'une région, cet aspect est d'ailleurs abordé dans la section 3.3 de ce chapitre. En fait non seulement nous définissons un modèle topologique complet, mais nous cherchons à faire en sorte que les éléments utiles en analyse d'images, tels que, une région, la connexité, l'adjacence entre régions, soient définis à partir des pixels ou voxels, tout en tenant compte de l'interpixel. Ainsi une région sera définie comme un ensemble de pixels, mais on rajoutera des contraintes qui garantiront les propriétés topologiques en imposant la présence ou l'absence de certains éléments interpixels.

[1]Un voxel est à une image en trois dimensions ce que le pixel est à une image en deux dimensions, c'est à dire l'information de base. De la même manière qu'un pixel est représenté par un carré, le voxel sera représenté par un cube, puisqu'étant dans un espace à trois dimensions.

D'autre part, nous disposons d'outils algorithmiques puissants et efficaces [Cha95] pour coder les éléments extraits de l'image par un codage adapté, le codage interpixel [Jac92]. Ces outils nous donnent par exemple la possibilité, pour une image segmentée en régions, d'extraire le graphe d'adjacence des régions (défini dans [Ros74, Pav77]) avec le codage interpixel du contour de chaque région ainsi que des caractéristiques diverses telles que la surface, l'intensité moyenne, la boîte englobante, ... Nous disposons par exemple d'un algorithme d'extraction du graphe d'adjacence et de segmentation de l'image en régions. Cet algorithme est de plus linéaire en le nombre de pixels et donne de très bons résultats en pratique[2]. Nous sommes également capables, en dérivant cet algorithme et en nous servant de ce codage, de réaliser une segmentation « à la volée[3] » d'une image défilante. Cette application de contrôle-qualité [ACF94] fait l'objet de développement intéressant actuellement.

Le codage interpixel est un codage du type code de contour [Ced79, Dan82, Ahr85] dérivé du codage de Freeman [Fre74]. À la différence de H. Freeman, le codage interpixel code directement les frontières entre les régions, c'est à dire en interpixel. Il réalise également un chaînage des contours, mais ne nécessite que quatre directions au lieu de huit, et ceci quelque soit le type de connexité choisie[4]. La figure 3.3 donne un exemple de codage du contour d'une région par le codage de Freeman et par le codage interpixel. Pour une description plus complète des propriétés de ce codage nous invitons le lecteur à se reporter à [Cha95].

Un autre avantage de ce type d'approche est son adéquation avec les méthodes de représentation utilisée en synthèse [Lie92, BD94]. Certes, ceci n'apporte rien quant à la synthèse, mais est d'un grand intérêt d'un point de vue pratique, surtout dans le cas d'images en trois dimensions. En effet il ne faut pas oublier que tout processus d'analyse doit s'inscrire dans un cadre plus général tel qu'un logiciel d'aide au diagnostic par exemple. Ainsi pour une application médicale, il

[2] moins d'une seconde pour une image 512×512 sur une Sun Sparc 2

[3] On entend par à la volée, le traitement de l'image au fur et à mesure de son acquisition. On ne peut dans le cas évoqué ici attendre la capture de l'image finale puisqu'elle possède virtuellement une longueur infinie. Il faut donc réaliser le traitement à la volée.

[4] On rappelle que deux connexités classiques existent (voir par exemple [Cha79, CM91], la 4-connexité : un pixel est adjacent aux quatres pixels ayant une arête commune avec lui-même ; et la 8-connexité : on prend la 4-connexité et on rajoute les quatres pixels ayant un sommet commun.

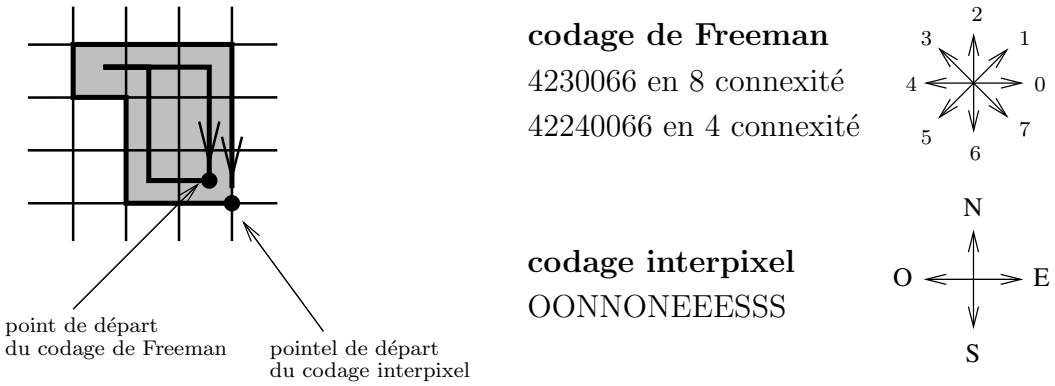


FIG. 3.3 – Codage de Freeman et codage interpixel.

est important de visualiser le résultat afin que le médecin puisse juger (voire corriger) les résultats obtenus. Donc, le fait de travailler directement en interpixel facilite grandement le passage à la représentation. Nous sommes donc en accord avec notre volonté exprimée dans le chapitre précédent de ne pas perdre de vue que la segmentation n'est que la première étape d'un processus plus général.

Certes certains argueront que les seuls éléments visibles de l'image sont les pixels, il n'y a donc pas lieu de traiter l'interpixel. De plus comment visualiser cet interpixel ? On ne peut rien afficher entre les pixels ! Mais ceci est un faux problème qui vient de la confusion entre élément d'information et élément de visualisation. Pourquoi ne pas considérer que nous avons des éléments d'informations divers (pixel, lignel et pointel dans le cas d'une image classique par exemple) qu'il faut afficher. Plusieurs choix sont alors possibles (voir figure 3.4) :

- ne pas représenter les interpixels n'appartenant pas à une frontière et représenter ceux appartenant à une frontière par le pixel à gauche ou au-dessus par exemple ;
- multiplier par quatre la taille de l'image et représenter chaque interpixel par un pixel ;
- multiplier la taille de l'image par seize en affichant quatre pixels pour un d'information et un pixel pour chaque interpixel, afin d'avoir l'information pixel occupant une surface plus grande que l'information interpixel dans l'image affichée.

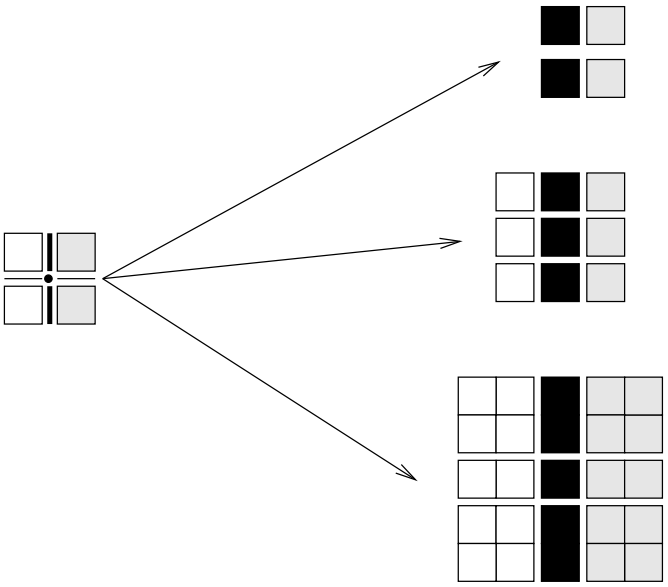


FIG. 3.4 – Visualisation de l'interpixel

Les solutions sont multiples. En fait il suffit de ne pas confondre élément d'information et élément d'affichage, ni étude et représentation. Jusqu'à présent ces deux points l'étaient, l'approche interpixel non seulement fait la différence, mais aussi prend en compte des éléments d'information supplémentaires : l'interpixel, c'est à dire en fait, le bord des éléments d'information primaire que sont les pixels, ou voxels.

3.2 Topologie et image.

3.2.1 prolégomènes à la topologie-étoile

Les pixels sont des ensembles indécomposables. Pour les modéliser, une approche consiste à les remplacer par des points situés en leur centre de gravité. On est alors amené à étudier une géométrie spécifique sur des ensembles de points isolés appelée géométrie discrète. Il faut alors redéfinir les propriétés et montrer à nouveau les théorèmes de la géométrie classique dont nous avons besoin dans le cadre de cette nouvelle géométrie. Nous pensons que le problème majeur posé par structure discrète d'une image est fortement lié à la notion de connexité, sur-

tout dans le domaine de l'analyse d'images. Aucun modèle concret ne propose actuellement de solutions à ce problème, satisfaisant tous les types d'images. Cette question a été largement étudiée et est toujours d'actualité (voir par exemple [Kov89, Her90, KKM90b, KKM90a, Lat93, AAF95]).

3.2.1.1 solutions déjà proposées

Nous présentons ici des solutions au problème de la connexité ; nous essayons également de vous montrer que ces solutions ne sont pas complètement satisfaisantes. Pour une étude plus complète de la topologie discrète, le lecteur pourra se reporter à [KR89].

Dans le cas binaire, une solution existe ; elle est difficilement transposable au cas des images en niveaux de gris. Cette solution, proposée par R.O. Duda, P.E. Hart et J.H. Munson [DHM67], consiste à utiliser deux connexités différentes, l'une pour le fond, l'autre pour l'image. Nous avons utilisé aussi cette approche dans la définition des contours interpixel pour les images binaires [Ahr85]. Si elle reste applicable pour les images en niveaux de gris dans le cas particulier où l'on ne considère qu'une unique région de l'image, le fond étant alors considéré comme homogène, on ne peut l'utiliser dès lors que l'on veut étudier l'image entière ou même seulement plusieurs régions.

La 6-connexité permet notamment d'éviter le paradoxe de la connexité, mais n'est pas utilisée car assez éloignée des considérations pratiques. En effet les images sont habituellement définies à l'aide de pavages carrés.

V.A. Kovalevsky [Kov89] propose de ne plus regarder l'image comme seulement un ensemble de pixels, mais comme un complexe cellulaire abstrait, c'est à dire un ensemble d'éléments hétérogènes de dimensions différentes correspondant à ceux que nous appelons, pointels, lignels et pixels. Cette approche n'est pas complètement nouvelle puisque déjà H. Elliot et L. Srinivasan en 1981 [ES81] proposaient d'utiliser les liens interpixels pour définir les frontières. Comme nous l'avons indiqué dans la section 3.1 nous avions également adopté cette représentation mais V.A. Kovalevsky est le premier à avoir proposé un modèle formel et un espace topologique utilisant ces éléments. De plus il a montré que tenir compte de ces éléments permettait d'éviter le paradoxe de la connexité et résolvait d'autres problèmes apparentés. Malheureusement, la représentation qu'il propose ne tire

pas pleinement partie de son modèle théorique. De plus il ne fait qu'aborder ces notions et ne cherche pas à les exploiter. En particulier il ne définit pas les notions de courbe, frontière ou contour ; il n'aborde pas non plus la question des images en 3 dimensions. E. Khalimsky, , R. Kopperman et P.R. Meyer présente dans [KKM90b] une solution du même type, mais les espaces topologiques proposés sont des produits cartésiens de droites discrètes. Leur modèle se restreint donc à un pavage carré du plan. Pas question par exemple d'envisager un pavage par des hexagones alors que le modèle que nous proposons le permet. On notera également qu'ils ne font, eux-aussi, qu'aborder l'application aux images.

G. Malandain [Mal93] utilise lui aussi la notion de complexe cellulaire. Il cherche à définir un modèle topologique complet et étudie notamment le théorème de Jordan dans le plan et dans l'espace. Toutefois il se restreint au cas binaire, et ce qu'il propose revient en fait à utiliser deux connexités différentes. Son travail a donc permis de justifier et de généraliser cette méthode au cas des images multidimensionnelles.

3.2.1.2 notre démarche

Les difficultés apparues pour définir une topologie dans des ensembles finis nous ont convaincus qu'il fallait utiliser une structure cellulaire où les éléments de base sont entourés des éléments « de bord » permettant de définir leurs liens. Nous avons choisi d'utiliser les complexes convexes [Lef30], dont les cellules sont des polytopes convexes et dont la topologie est définie à partir des étoiles ouvertes (*stars*). La justification de ce choix est donnée par le fait que le bord d'une cellule est constitué par un ensemble de faces de cette cellule, ce qui est bien le cas pour les structures classiques de la géométrie discrète. Nous proposons donc de suivre la voie ouverte par V.A. Kovalevsky [Kov89].

Notre démarche peut être appréhendée à partir de deux approches complémentaires :

- la donnée d'une relation d'ordre caractérisant une topologie sur un ensemble fini ;
- le passage au quotient de la topologie classique.

Concernant la première démarche, on rappelle qu'on ne peut trouver de topologie séparée au sens de Hausdorff sur un ensemble fini (c'est-à-dire où 2 points distincts admettent des voisinages disjoints), mais simplement T_0-séparée ce qui

signifie : étant donné deux points distincts, il existe un voisinage de l'un qui ne contient pas l'autre. Le théorème suivant va nous permettre d'établir la relation d'ordre caractérisant notre topologie.

Théorème 3.1 (Mac Kinsey,Tartski) *Il y a une bijection naturelle entre les espaces topologiques T_0-séparés finis et les ensembles ordonnés finis.*

On peut alors définir une relation d'ordre $\mathcal{R}$ de la manière suivante : si E est un espace topologique T_0-séparé, on définit la relation $\mathcal{R}$ par $\mathcal{R}(x, y)$ si et seulement si tout voisinage de x contient y. Réciproquement si E est un ensemble ordonné, la base d'ouverts définissant la topologie est déterminée ainsi : pour chaque élément x de E, on définit un ouvert de la base par l'ensemble des successeurs de x. L'ensemble de tous ces ouverts ainsi définis forme cette base. Ce résultat très ancien semble dû à Mac Kinsey et Tartski. On peut associer une notion de dimension de façon classique aux relations d'ordre (elle est égale à zéro pour les éléments minimaux, 1 pour les minimaux de l'espace obtenu après avoir enlevé les minimaux, etc). Dans le cadre de notre travail la relation $\mathcal{R}$ sera la relation de bornage (voir section 3.2.3). C'est à partir de ce constat que V.A. Kovalevsky a proposé d'utiliser les complexes cellulaires.

L'autre approche consiste à chercher une structure discrète qui récupère le plus possible les propriétés de la géométrie et de la topologie classiques. L'idée est de décomposer un sous-espace de $\mathbb{R}^n$ en objets bien définis et dénombrables afin d'obtenir un ensemble discret quotient de $\mathbb{R}^n$. Notre ensemble discret correspond alors à une partition d'un sous ensemble de $\mathbb{R}^n$, il suffit donc de lui associer la topologie quotient. Nous avons choisi de décomposer $\mathbb{R}^n$ en ensembles convexes et plus particulièrement en polytopes. Avant de rappeler quelques définitions, nous donnons un exemple de décomposition en dimension 3.

Classiquement en traitement d'images, l'unité d'information de base est un voxel dans une image 3D. Nous les représenterons par des petits cubes dont les côtés sont de longueur unité. Notre décomposition consiste alors en un pavage de $\mathbb{R}^3$ à partir des voxels, de leurs faces, de leurs arêtes et de leurs sommets. En prenant les composantes dans un repère cartésien, on obtient en dimension 3 la décomposition en voxels, surfels, lignels et pointels (voir figure 3.5) qui sont des sous-ensembles de $\mathbb{R}^3$ (les indices i, j, k qui suivent sont des entiers) :

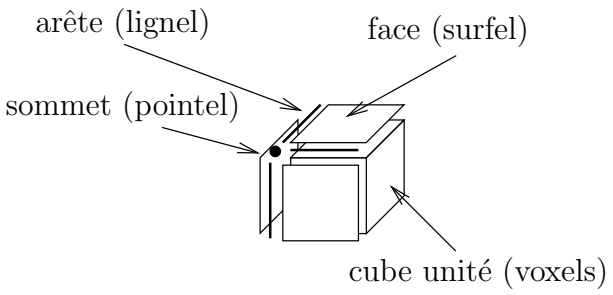


FIG. 3.5 – Voxel, surfel, lignel et pointel.

- Les voxels $V_{(i,j,k)} = \{(x,y,z) \,/\, x \in]i, i+1[\,, y \in]j, j+1[\,, z \in]k, k+1[\,\}$
- les facettes (surfel) :
  1. Les facettes horizontales de la forme : $\{(x,y,z) \,/\, x \in]i, i+1[\,, y \in]j, j+1[\,, z = k\}$
  2. Les facettes de face : $\{(x,y,z) \,/\, x \in]i, i+1[\,, y = j, z \in]k, k+1[\,\}$
  3. Les facettes de profil : $\{(x,y,z) \,/\, x = i, y \in]j, j+1[\,, z \in]k, k+1[\,\}$
- les arêtes (lignels) :
  1. Les arêtes de front : $\{(x,y,z) \,/\, x \in]i, i+1[\,, y = j, z = k\}$
  2. Les arêtes de bout : $\{(x,y,z) \,/\, x = i, y \in]j, j+1[\,, z = k\}$
  3. Les arêtes verticales : $\{(i,j,k)\}\{(x,y,z) \,/\, x = i, y = j, z \in]k, k+1[\,\}$
- Les sommets (pointels) : ensembles réduits à un point de la forme (i,j,k).

3.2.1.3 les ensembles convexes et les polytopes

Les définitions, théorèmes et propriétés qui suivent présentent les résultats de base qui nous sont nécessaires pour définir notre structure discrète : les complexes convexes. Ils sont tous extrait des livres de A. Brønstedt et de Grünbaum [Brø83, Gru67]. Commençons donc par la définition d'ensemble convexe :

Définition 3.1 *Soit E un sous-ensemble de* $\mathbb{R}^n$*. On dit que E est* convexe *si pour toute suite finie* $a_1, \ldots, a_p$ *de points de E et toute suite finie* $x_1, \ldots, x_p$ *de réels positifs ou nuls tels que* $x_1 + \ldots + x_p = 1$*, on ait :* $x_1 a_1 + \ldots + x_p a_p \in E$*.*

Le théorème suivant énonce une propriété importante des ensembles convexes. Cette propriété nous assure notamment que les faces d'un ensemble convexe (voir définition 3.6) soient bien des ensembles convexes elles-mêmes.

Théorème 3.2 *L'intersection d'ensembles convexes est un ensemble convexes.*

La notion d'hyperplan d'appui présentée à travers les définitions 3.2 à 3.5 va nous permettre de définir une face d'un ensemble convexe. Cette notion est primordiale car c'est à partir d'elle que nous définirons la relation de bornage (voir définition 3.13) dans les complexes convexes. Or cette relation est à la base de notre construction et notamment de la définition du plus petit ouvert contenant un élément.

Définition 3.2 *On appelle hyperplan H dans* $\mathbb{R}^n$ *un sous-espace affine de dimension n-1.*

Un hyperplan H peut être caractérisé par une équation de la forme $<x, u>= a$ où u est un vecteur perpendiculaire à H et où $<c, d>$ représente le produit scalaire de c par d.

Définition 3.3 *On appelle* demi-espace fermé *(resp.* ouvert*), l'ensemble des points de* $\mathbb{R}^n$ *tels que* $<x, u>\geqslant a$ *(resp.* $> a$*) ou* $<x, u>\leqslant a$ *(resp.* $< a$*).*

Définition 3.4 *On appelle dimension d'un convexe E, la dimension de l'espace vectoriel engendré par E.*

Définition 3.5 *On appelle* hyperplan d'appui *d'un ensemble convexe E de dimension n dans* $\mathbb{R}^n$*, un hyperplan P tel que tous les points de E soient dans un même demi-espace de P et tels que la distance de P à E soit nulle.*

En fait la notion d'hyperplan d'appui est une notion qui intuitivement se comprend, ainsi dans $\mathbb{R}^2$ par exemple les hyperplans d'appui (ici des droites) d'une ellipse sont ses tangentes.

On peut maintenant définir une face d'un ensemble convexe :

Définition 3.6 *On appelle* face *d'un ensemble convexe fermé E, l'intersection F de E avec un hyperplan d'appui. L'ensemble des faces de E est noté :* $\mathcal{F}(E)$.

On remarquera d'une part que les faces d'un ensemble convexe sont des intersections d'ensembles convexes, donc d'après le théorème 3.2 convexes elles-même. D'autre part nous avons les cas particuliers suivants :

1. si F est un point, c'est-à-dire de dimension 0, F est appelé un sommet ;
2. si sa dimension est 1, il est appelé une arête ;
3. si F est maximal, il est appelé une facette.

Par exemple si E est un cylindre, ses facettes sont les droites directrices. Si E est un cube, ses facettes sont ses côtés.

Après avoir défini une face d'un ensemble convexe, nous passons maintenant à la définition d'un point extrêmal :

Définition 3.7 *On appelle* point extrêmal *d'un ensemble convexe E, tout point qui n'est pas à l'intérieur d'un segment inclus dans E. L'ensemble des points extrêmaux de E est noté* $ext(E)$.

Toujours pour prendre un exemple dans le plan, les extrémités d'un segment sont des points extrêmaux.

Nous pouvons enfin présenter les polytopes qui seront les ensembles de base des complexes convexes.

Définition 3.8 *On appelle* polytope fermé *de dimension n, un sous-ensemble convexe compact de dimension n ayant un nombre fini de points extrêmaux. On appelle* polytope ouvert[5] *de dimension n l'intérieur du polytope fermé de dimension n pour la topologie dans* $\mathbb{R}^n$.

On citera les propriétés suivantes :

Propriétés 3.1 *1. Un ensemble* $K \in \mathbb{R}^n$ *est un polytope si et seulement si il est borné et s'il est l'intersection d'un nombre fini de demi-espaces.*

2. Les points extrêmaux d'un polytope sont ses sommets.

[5] *On remarquera que ce même polytope n'est pas un ouvert s'il est plongé dans un espace vectoriel de dimension* $m > n$. *Mais il est d'usage de toujours appeler ce polytope un* polytope ouvert. *On parle alors d'ouvert relatif pour exprimer qu'un polytope est ouvert dans l'espace affine engendré. De même un point du polytope ouvert est dit dans l'intérieur relatif du polytope fermé.*

3. *Une face de dimension k d'un polytope est un polytope de dimension k.*
4. *Si P est un polytope de dimension n, et F une face de P, toute face de F est une face de P.*
5. *Le polytope fermé est l'union du polytope ouvert et de ses faces propres.*
6. *Si P un polytope et P' une face de P alors pour tout point a de P et tout points b de P', $[a, b[\subseteq P$*

Remarque : Les points extrêmaux d'une face F de P sont l'intersection de F avec ext(P). Cependant, tout sous-ensemble de ext(P) n'est pas l'ensemble des points extrêmaux d'une face de P. Cela n'est pas vrai en particulier pour un carré : deux points extrémités d'une diagonale ne forment pas les points extrêmaux d'une face.

3.2.2 complexes convexes

En nous appuyant sur les résultats précédents nous allons maintenant définir les complexes convexes. Nous rappelons que l'idée est de décomposer un sous-espace de $\mathbb{R}^n$ en objets bien définis et dénombrables afin d'obtenir un ensemble discret quotient de $\mathbb{R}^n$. V.A. Kovalevsky propose dans [Kov89] d'ajouter des nouveaux éléments à l'ensemble des pixels afin de bien décrire les liens entre eux. Il a utilisé les complexes cellulaires abstraits. Nous proposons ici une structure plus contraignante mieux adaptée au contexte géométrique qui nous intéresse. Un complexe est un espace topologique muni d'une partition particulière dans laquelle les ensembles sont appelés *cellules* et sont des déformations de boules d'espace $\mathbb{R}^n$. Nous utiliserons les *complexes convexes*. Dans cette structure les cellules sont des polytopes convexes, qui sont des généralisations des pixels et des voxels.

Remarque : On a choisi de prendre des polytopes convexes comme cellules. D'autres choix peuvent être proposés :

- Prendre des simplexes engendrés par $n + 1$ points indépendants dans $\mathbb{R}^n$: triangle dans $\mathbb{R}^2$, tétraèdre dans $\mathbb{R}^3$. Cela permet d'avoir des définitions très simples des faces : tout ensemble fini de points extrèmaux est une face. Par contre dans les applications, on a souvent des carrés ou des hexagones.
- Les polytopes quelconques (pas nécessairement convexes). Cela permettrait d'avoir des étoiles pour cellules mais poserait des problèmes pour définir des faces.

– Les complexes cellulaires qui sont des surfaces qu'on peut obtenir à partir de polytopes par déformation. On a alors une grande généralité.

En fait on cherchait une structure contenant tous les cas classiques d'images et la plus riche possible, c'est à dire possédant des propriétés fortes. C'est pourquoi nous nous sommes restreints aux complexes convexes.

Les définitions de cette section ont, d'après Grünbaum [Gru67] été présentées dans le livre de S. Lefschetz [Lef30].

Définition 3.9 *On appelle* complexe convexe *une famille finie $\mathcal{C}$ de polytopes ouverts P_i de $\mathbb{R}^n$ appelés cellules du complexe tels que :*

– *Si $P_i \in \mathcal{C}$, alors toute face de $P_i \in \mathcal{C}$.*
– *Si $P_i \in \mathcal{C}$ et $P_j \in \mathcal{C}$,$i \neq j$, alors $\overline{P_i} \bigcap \overline{P_j}$ est vide*[6] *ou bien est une face commune de P_i et P_j.*

La dimension du complexe[7] *est par définition le maximum des dimensions des cellules du complexe.*

On appellera un polytope de dimension 0 un *pointel*, un de dimension 1 un *lignel* et un de dimension 2 un surfel. On pourra donner comme exemples de complexe convexe :

– l'ensemble des faces propres d'un polytope P, ce complexe est appelé le complexe bord de P et noté Bd(P) ;
– l'ensemble de toutes les faces de P noté $\mathcal{F}$(P) ;
– pour tout complexe, on note $\mathcal{S}_k(\mathcal{C})$ le complexe formé des polytopes de $\mathcal{C}$ de dimension inférieure à k ;

La figure 3.6 présente d'autres exemples (a., b. et c.) et contre-exemples (d., e. et f.) de complexes convexes. Ainsi 3.6.f. n'est pas un complexe convexe car ne contient pas que des polytopes, 3.6.e. non plus car les faces (symbolisée par les traits épais) ne sont pas dans le complexe. Enfin 3.6.d. ne peux l'être car les polytopes ne sont pas tous convexes, les faces n'appartiennent pas au convexe, et de plus l'intersection de deux polytopes du complexe n'est pas une face des deux polytopes.

[6] *On note $\overline{E}$ le fermé de l'ensemble E. Ici le fermé d'un ensemble s'entend au sens de la topologie classique dans $\mathbb{R}^n$.*

[7] *Désormais par abus de langage nous emploierons parfois le terme complexe en la place de complexe convexe.*

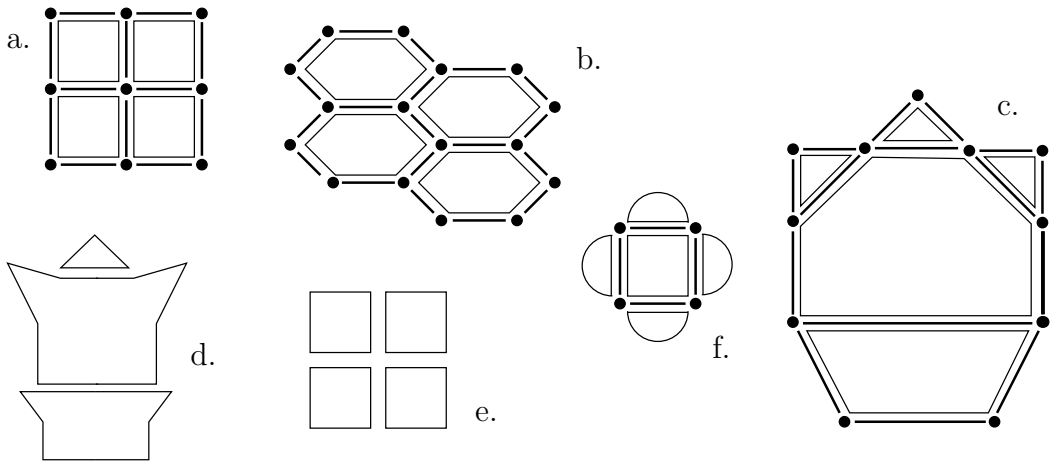


FIG. 3.6 – Exemples (a. b. c.) et contre-exemples (d. e. f.) de complexes convexes.

Évidemment dans le cas particulier des images on peut également définir un complexe convexe décrivant l'image. Pour les images classiques on retrouve alors les notions de cellules décrites par V.A. Kovalevsky . En effet, les polytopes de base seront les pixels (cellules de dimension 2) auxquels nous devons adjoindre leurs faces c'est à dire ce que nous avons appelé les liens et points interpixels. Il est à noter que dans le cas des images 3D, on retrouve le modèle couramment utilisé à base de voxels, faces, arêtes et sommets de voxels.

Définition 3.10 *On appelle* sous-complexe *L d'un complexe $\mathcal{C}$ un ensemble de cellules de $\mathcal{C}$ ayant une structure de complexe ce qui équivaut à : un ensemble de cellules telles que si c est une cellule de L, toute face de c est une cellule de L.*

Définition 3.11 *On appelle* complexe engendré *par un ensemble $\mathcal{K}$ de cellules d'un complexe donné, le plus petit complexe noté $\mathcal{C}(\mathcal{K})$ contenant ces cellules.*

La structure de polyèdre donnée par la Définition 3.12 va nous permettre de travailler avec une topologie induite par celle de $\mathbb{R}^n$ sur les structures discrètes que sont les complexes convexes.

Définition 3.12 *Si E est une famille de cellules, on appelle* polyèdre *de E, et on note $|E|$ l'union des points des cellules de E.*

3.2.3 relation de bornage et étoiles ouvertes des cellules

Un complexe $\mathcal{C}$ est un ensemble fini. Une T_0-topologie sur $\mathcal{C}$ est associée à une relation d'ordre (voir 3.2.1.2). Nous donnons ci-dessous une définition d'une topologie sur $\mathcal{C}$ et la relation associée qu'on appelle relation de bornage. Ces deux notions nous permettront de montrer le Théorème 3.9 décrivant le passage à la topologie quotient de celle induite par $\mathbb{R}^n$. Nous aurons ainsi décrit une topologie sur les ensembles discrets que sont les complexes convexes.

Définition 3.13 (relation de bornage et incidence) *Étant donnés deux polytopes P_1 et P_2, on dit que P_1 borne P_2 et on note $P_1 \leqslant_B P_2$ si P_1 est une face de P_2 ou si $P_1 = P_2$. Si P_1 borne P_2 ou P_2 borne P_1, on dit que P_1 et P_2 sont incidents.*

Proposition 3.3 *La relation de bornage $\leqslant_B$ définie sur l'ensemble des polytopes d'un complexe convexe est une relation d'ordre.*

Preuve :
La relation de bornage $\leqslant_B$ est une relation d'ordre partiel. En effet elle est réflexive d'après la définition, elle est antisymétrique car si F est une face de P, alors ext(F)$\in$ ext(P) et elle est transitive d'après la propriété 3.1. □

Cette relation possède en outre des propriétés particulières concernant notamment les minorants et majorants, ainsi on peut montrer la proposition suivante :

Lemme 3.4 *Deux cellules c et c' d'un complexe convexe $\mathcal{C}$ qui ont un minorant (resp. majorant) commun suivant la relation de bornage ont une borne inférieure (resp. supérieure).*

Preuve :
Si c et c' ont un minorant commun, c'est que l'intersection de $\overline{c}$ et $\overline{c'}$ est une face commune e. Les minorants communs sont alors les faces de e qui est donc la borne inférieure pour $\leqslant_B$. En procédant de proche en proche, on montre qu'un ensemble fini de cellules ayant un minorant commun admet une borne inférieure. Si deux cellules c et c' ont un majorant commun, l'ensemble M des majorants communs est fini et admet des minorants c et c' donc une borne inférieure qui est la borne supérieure de c et c'. □

Cett proposition nous permet d'envisager une structure de treillis pour la relation de bornage. Malheureusement deux cellules n'ont pas forcément de minorant

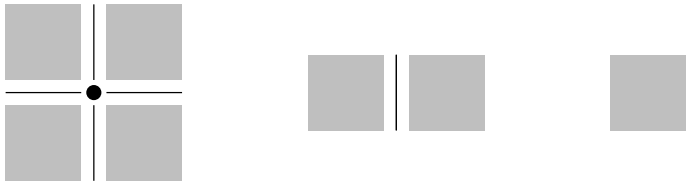


FIG. 3.7 – Étoile ouverte d'un pointel, lignel et surfel (pixel).

commun, respectivement de majorant commun. En fait il nous suffit de munir la relation de bornage d'un élément minimum et d'un élément maximum pour obtenir une structure de treillis, d'où la proposition suivante :

Théorème 3.5 *Si on ajoute un minimum et un maximum absolus, par exemple l'ensemble vide et l'espace $|\mathcal{C}|$ qui peuvent être considérés comme des cellules généralisées, on obtient un treillis.*

Preuve :
La borne inférieure de deux cellules est l'intersection de leurs fermetures. La borne supérieure est soit $\mathbb{R}^n$ lui-même, soit le minimum des majorants communs. □

Après avoir défini la relation de bornage que nous rattacherons par la suite naturellement à la notion de dimension d'une cellule, passons maintenant à la définition de l'étoile ouverte d'une cellule.

Définition 3.14 (étoile ouverte) *On appelle* étoile ouverte *(respectivement* complexe étoile *d'une cellule c dans le complexe $\mathcal{C}$, l'ensemble des cellules de $\mathcal{C}$ dont c est une face (respectivement le plus petit complexe contenant l'étoile ouverte de c). On note $star(c)$ l'étoile ouverte de c (voir exemple en 2D à la figure 3.7).*

Cette définition amène plusieurs remarques. D'abord concernant la relation de bornage, puisque c est une face des cellules de $star(c)$, alors c borne chaque cellule de $star(c)$. Autrement dit $star(c)$ est l'ensemble des majorants de c. Ceci peut être vérifié à l'aide de la Propriété 3.2.

Propriétés 3.2 *Si c borne e, $star(c) \supset star(e)$. Plus généralement si $star(c)$ et $star(c')$ ne sont pas disjoints, $star(c) \cap star(c')$ est l'étoile ouverte de la borne supérieure de c et c', que l'on note $star(c \vee_B c')$.*

Preuve :
Ce sont des propriétés classiques, la première des relations d'ordre (transitivité de la relation $\leqslant_B$), la seconde des treillis (l'intersection des majorants de deux éléments est l'ensemble des majorants de la borne supérieure de ces deux éléments.). □

Les étoiles ouvertes jouent un rôle très particulier pour la topologie des polyèdres et également celle des complexes convexes. En fait elles vont nous permettre de décrire le passage de la topologie sur les polyèdres à celle des complexes convexes.

3.2.4 de la topologie sur les polyèdres à la topologie-étoile

Étant donné un complexe $\mathcal{C}$, nous allons caractériser les ouverts du polyèdre $|\mathcal{C}|$ qui est la réunion des cellules de $\mathcal{C}$. Ceci nous permettra de définir sur $\mathcal{C}$ une topologie quotient.

Définition 3.15 *On définit sur $|\mathcal{C}|$ une relation d'équivalence* cell *: x et y sont* cell-équivalents *s'ils appartiennent à la même cellule ouverte. L'espace quotient de $|\mathcal{C}|$ par cette relation est l'espace $\mathcal{C}$ des cellules. On dit qu'un ensemble $E \in |\mathcal{C}|$ est saturé pour* cell *s'il est une réunion de cellules de $\mathcal{C}$. On note j l'application qui à un point de $|\mathcal{C}|$ associe sa classe d'équivalence, c'est à dire la cellule à laquelle il appartient.*

Proposition 3.6 *Soit $\mathcal{C}$ un complexe convexe.*

1. *Si c est une cellule de $\mathcal{C}$ et si un ouvert saturé du polyèdre $|\mathcal{C}|$ contient $|c|$, il contient $|star(c)|$.*
2. *Un sous-ensemble saturé de $\mathcal{C}$ est fermé si et seulement si c'est un sous-complexe de $\mathcal{C}$.*
3. *$|star(c)|$ est un ouvert.*

Preuve :

1. Soient $x \in |c|$ et e une cellule de $star(c)$. c borne e donc x est adhérent à $|e|$. Donc tout ouvert contenant x rencontre $|e|$ et par conséquent tout ouvert saturé contenant x contient $|e|$.

2. Soit K un sous-complexe de $\mathcal{C}$. K est l'union d'un ensemble fini de cellules c_i. L'union (finie) des fermetures est la fermeture de l'union donc si x est un point de $\overline{|K|}$, pour un i au moins, $x \in \overline{|c_i|}$ donc x est sur une face d'une cellule de K donc dans $|K|$. Réciproquement, si un ensemble fini K de cellules e_j est fermé, toute face d'une cellule de K est contenue dans K donc K est un sous-complexe de C.
3. Le complémentaire de $star(c)$ est un sous-complexe car si une cellule n'est pas bornée par c, aucune de ses faces n'est bornée par c. Le complexe C étant fini, d'après la partie 2 du présent théorème, c'est un fermé et $star(c)$ est un ouvert.

□

Propriétés 3.3 *Pour la topologie de $|\mathcal{C}|$, si c est une cellule de $\mathcal{C}$ alors $|star(c)|$ est le plus petit polyèdre ouvert contenant c.*

Preuve :
C'est une synthèse des parties 1 et 3 du théorème précédent. □

Maintenant que nous avons défini la notion de plus petit ouvert et celle d'ensemble ouvert et fermé sur les complexes cellulaires, nous allons pouvoir décrire une topologie sur les polyèdres, et ainsi caractériser une topologie sur les ensembles finis. Dans un premier temps caractérisons la nature même d'un polyèdre d'un complexe convexe de dimension n :

Proposition 3.7 *Pour tout complexe $\mathcal{C}$ de dimension n, $|\mathcal{C}|$ est fermé.*

Preuve :
$\mathcal{C}$ est une union finie de ses cellules fermées, donc il est fermé. □

On peut maintenant déduire du Théorème 3.6, la caractérisation de l'intérieur et de l'adhérence d'un sous-ensemble d'un polyèdre en fonction des étoiles ouvertes des cellules de ce sous-ensemble.

Proposition 3.8 *Étant donné un ensemble H de cellules du complexe $\mathcal{C}$, on a :*

1. *$\overline{|H|}$ est l'union des cellules e de H telles que $star(e) \subseteq H$*
2. *$\overline{|H|}$ est l'union des cellules de H et de leurs faces.*

Ceci nous permet de définir une topologie sur l'espace des complexes convexes, la topologie quotient de la topologie classique sur $\mathbb{R}^n$.

Définition 3.16 (Topologie-Étoile de $\mathcal{C}$) *Si $|\mathcal{C}|$ est muni de sa topologie classique (induite par celle de $\mathbb{R}^n$), la topologie quotient définie sur $\mathcal{C}$ a pour ouverts, par définition de la topologie quotient, les images par j (voir Définition 3.15) des ouverts de $|\mathcal{C}|$ qui sont saturés pour la relation cell*[8]*. Cette topologie est appelée la* topologie-étoile.

Des résultats précédents et de la Définition 3.16 de la topologie quotient nous déduisons le théorème fondamental suivant caractérisant la topologie-étoile :

Théorème 3.9 *Sur l'espace $\mathcal{C}$, les éléments sont les cellules du complexe $\mathcal{C}$. La topologie-étoile sur $\mathcal{C}$ est la topologie engendrée sur $\mathcal{C}$ par les étoiles ouvertes de ses cellules.*

Preuve :
La topologie engendrée par les étoiles ouvertes des cellules a pour ouverts les unions des étoiles ouvertes de cellules (dans la définition générale, on prend les unions quelconques et intersections finies mais ici les intersections d'étoiles ouvertes de cellules sont des étoiles ouvertes de cellules ou bien sont vides). D'après les deux théorèmes précédents, les ouverts saturés de $|\mathcal{C}|$ sont les unions d'étoiles ouvertes, donc les topologies sont confondues. □

Nous avons donc les propriétés suivantes sur les ensembles ouverts et fermés :
- Un ensemble E de cellules est un ensemble ouvert pour la topologie-étoile si et seulement si : $\forall c \in E, star(c) \subseteq E$.
- Un ensemble E de cellules est un ensemble fermé pour la topologie-étoile si et seulement si : $\forall c \in E,$ si $c \leqslant_B c'$ alors $c' \in E$.

Remarque : Cette topologie quotient est donc bien conforme au schéma présenté pour les topologies sur les ensembles finis qui sont associées aux relations d'ordre : un ouvert de base est bien l'ensemble des successeurs d'une cellule pour la relation de bornage.

Ce résultat est très important car il va nous permettre de récupérer des résultats de la topologie algébrique réelle qui sont transposables. Nous allons montrer plus loin comment on peut en tirer un théorème de Jordan.

[8] *Les ouverts de $|\mathcal{C}|$ qui sont saturés pour la relation cell sont des unions de cellules. Les ouverts dans l'espace quotient ainsi obtenu sont donc bien des ensembles de cellules.*

3.2.5 courbe, surface et théorème de Jordan

Du fait que l'on se trouve dans une structure quotient, on récupère un certain nombre de résultats classiques de la géométrie réelle.

Définition 3.17 ($\mathcal{C}$-courbe) *Une $\mathcal{C}$-courbe est une suite finie $c_0, ..., c_{2n}$ de cellules qui sont alternativement des sommets et des arêtes, telles que c_0 et c_{2n} soient des sommets et que les cellules d'indices successifs soient incidentes. Elle est dite simple si toutes ses cellules sont distinctes. Elle est dite fermée si $c_0 = c_{2n}$.*

Remarque : Une $\mathcal{C}$-courbe simple est un sous-complexe de dimension 1 de C.

Proposition 3.10 *Une $\mathcal{C}$-courbe (resp. fermée, simple) est l'image par j d'une courbe continue (resp. fermée, simple) de $\mathbb{R}^2$.*

Preuve :
Dire qu'une $\mathcal{C}$-courbe γ est l'image d'une courbe continue Γ de $\mathbb{R}^2$ signifie :

1. tous les points de Γ ont leur image dans γ ;
2. si $t < t'$, j(f(t)) a un indice dans γ inférieur ou égal à celui de j(f(t')).

Une $\mathcal{C}$-courbe est une suite de sommets et d'intervalles ouverts agencés de telle sorte que l'on obtienne une suite d'intervalles fermés qui se rencontrent en une extrémité. Cela donne une ligne brisée qui est une courbe continue. Plus précisément, nous donnons un paramétrage des points de $|\mathcal{C}|$ qui sont dans la courbe, c'est à dire une application de $I = [0, 1]$ dans $|\mathcal{C}|$. On peut prendre, par exemple, le paramétrage suivant, i variant de 0 à n :

- pour les sommets : $f(c_{2i}) = \frac{i}{n}$;
- pour les arêtes : l'application linéaire qui envoie l'intervalle ouvert $]\frac{i}{n}, \frac{i}{n+1}[$ dans l'intervalle ouvert $]c_{2i+1}] =]c_{2i}, c_{2i+2}[$

γ est fermée si Γ est fermée et simple si elle n'a pas de point double donc si Γ n'a pas de point double car les cellules sont disjointes. □

Définition 3.18 (ensemble connexe) *Un ensemble de cellules d'un complexe est dit* connexe *si quelles que soient les cellules c et c', il existe une suite $c = c_0, ..., c_p = c'$ telle que deux cellules d'indices consécutifs soient incidentes.*

Théorème 3.11 *Soit U un ensemble de cellules d'un complexe dans $\mathbb{R}^n$. Alors $|U|$ est connexe par arcs si et seulement si son image par j est connexe.*

Preuve :
Supposons que $|U|$ soit connexe par arcs. Montrons que U est connexe. Soient c et c' deux cellules de U. Soit $a \in |c|$ et $b \in |c'|$, par hypothèse il existe une courbe h joignant a et b. Par définition h est une application continue de I dans $|U|$ telle que $h(0) = a$ et $h(1) = b$. Nous allons construire une suite finie de cellules incidentes $c = c_0, c_1, ..., c_p = c'$. L'idée la plus simple est de prendre les cellules images par j des points de la courbe h. C'est ce que nous ferons en prenant quelques précautions, une même cellule pouvant être l'image de plusieurs points de h : on court-circuitera les boucles inutiles.

On construit une suite c_i de cellules et une suite d'indice t_i appelé indice de sortie de la courbe en c_i. Supposons que nous ayons construit une suite $c = c_0, c_1, ..., c_i$ de cellules toutes distinctes telles que deux cellules d'indices successifs soient incidentes et telles que pour tout $j < i$, il existe $t_j \in I$,la suite t_j étant croissante et vérifiant $t_j = sup(t \mid h(t) \in |c_j|)$. Si $c_i = c'$, le processus s'arrête.
Sinon,

- si $h(t_i) \notin |c_i|$, on note c_{i+1} la cellule telle que $h(t_i) \in |c_{(i+1)}|$. Comme h est continue, $h(t_i) \in \overline{|c_i|}$ et donc c_{i+1} borne c_i.
- si $h(t_i) \in |c_i|$, on choisit c_{i+1} parmi les cellules $k \in U$ telle que $inf(t \mid t \geqslant t_i$ et $h(t) \in |k|) = t_i$. Une telle cellule existe car sinon il y aurait des indices de I où la courbe ne serait pas définie. Comme h est continue, $h(t_i) \in \overline{|c_{i+1}|}$ et donc c_i borne c_{i+1}.

La suite c_i ainsi construite est finie (puisque toutes les cellules par construction sont distinctes) et vérifie les hypothèses requises.

Réciproquement soient a et b deux points de $|U|$; on doit montrer qu'il existe un chemin de a vers b. Soit c (resp c') la cellule telle que $|c|$ (resp $|c'|$) contienne a (resp. b). U étant connexe, il existe une suite finie de cellules $c = c_0, c_1, ..., c_p = c'$ telle que pour tout $i, 0 \leqslant i < p$ c_i soit incidente à c_{i+1}. Si c_i borne c_{i+1}, tout point de $|c_i|$ est dans l'adhérence de $|c_{i+1}|$. D'après la propriété 3.1, pour tout $y \in |c_{i+1}|$, il existe un chemin [x, y] tel que]x, y] $\in |c_{i+1}|$. On peut donc trouver une suite de points $x_i \in |c_i|$ et des chemins reliant les points d'indices successifs. L'union de ces chemins est un chemin de a vers b dans $|U|$. □

Remarque : Les précautions prises sur le nombre fini de cellules rencontrées sont nécessaires car parmi les fonctions réelles continues, on peut rencontrer des

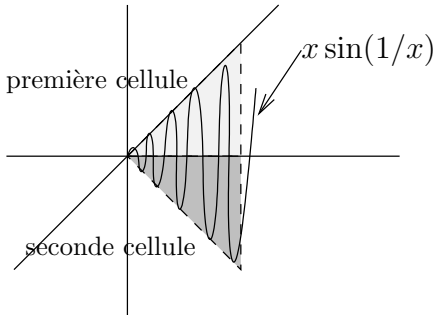


FIG. 3.8 – problème posé par certaines fonctions réelles continues.

fonctions telles que $x \to x\sin(1/x)$. Pour un tel exemple si l'on considère les deux cellules formées par les triangles dont les sommets sont $(0,0).(1,0),(1,1)$ et $(0,0).(1,0),(1,-1)$ (voir figure 3.8) la courbe représentant cette fonction passe une infinité de fois d'une cellule à l'autre.

Le théorème de Jordan classique se présente sous la forme suivante :

Théorème 3.12 (théorème de Jordan) *Toute courbe continue, fermée simple sépare le plan en deux composantes connexes : l'une bornée est appelée l'intérieur, l'autre est appelée l'extérieur.*

On admettra ce théorème de la géométrie classique qui admet diverses démonstrations qui ne sont pas élémentaires [GP74]. Nous en déduisons le théorème de Jordan dans l'espace des cellules par le passage au quotient.

Théorème 3.13 *Soit $\mathcal{C}$ un complexe convexe de dimension 2 et soit Γ une $\mathcal{C}$-courbe fermée simple. Γ sépare le plan en deux composantes connexes distinctes dont une bornée est appelée l'intérieur.*

Preuve :
Une $\mathcal{C}$-courbe fermée simple est l'image par j d'une courbe fermée simple Γ' dans $\mathbb{R}^2$. On applique le théorème de Jordan classique à Γ'. L'intérieur et l'extérieur de la courbe réelle sont des unions de cellules donc on peut appliquer le théorème 3.11 ce qui prouve le théorème. □

Ce théorème nous permet de définir la notion de « trou » que nous utiliserons plus loin :

Définition 3.19 (Trou) *Soit $\mathcal{C}$ un complexe convexe, on appelle* Trou*, on l'on note T, un ensemble connexe tel qu'il existe une courbe fermée simple contenant T en son intérieur.*

Notre démarche diffère de celle préconisée notamment par V.A. Kovalevsky [Kov89] ou J. Françon [Fra95, Fra96]. En effet ils utilisent uniquement des structures discrètes pour obtenir des résultats en géométrie discrète en s'appuyant en particulier sur la notion de complexe cellulaire abstrait. C'est une théorie mathématique nouvelle qu'ils développent actuellement.

Nous montrons que la structure que nous avons développée est le quotient d'une structure classique sur $\mathbb{R}^n$, les images étant supposée obtenues par « découpage » des surfaces de $\mathbb{R}^2$ (et plus généralement de $\mathbb{R}^n$). Nous obtenons ainsi des résultats par passage au quotient de ceux de la topologie algébrique classique. Cette démarche peut paraître moins directe, mais elle se justifie par son efficacité comme le montre notre démonstration du théorème 3.13 de Jordan pour les images 2D. De plus elle n'est pas nouvelle en mathématique, les nombres complexes sont bien utilisés pour prouver des théorèmes sur les équations algébriques réelles.

3.3 Et l'analyse d'images ?

3.3.1 introduction

Nous nous proposons maintenant d'appliquer la topologie-étoile définie dans les sections précédentes à l'analyse d'images. Ceci nous permettra d'obtenir un modèle de représentation cohérent et consistant, évitant le paradoxe de la connexité et les autres problèmes apparentés [Pav77].

Ce modèle a pour objectif d'être général, c'est à dire adapté non seulement aux images en niveaux de gris, mais aussi aux images multi-dimensionnelles. Mais il est clair que le fait de prendre en compte des cellules telles que les surfels, lignels et pointels est contraignant en termes d'analyse d'image. En effet la quantité d'informations fournies par les pixels ou les voxels est déjà considérable et en rajouter risque de devenir trop contraignant. Rappelons que le facteur de l'efficacité est à prendre en compte dans les applications de traitement d'images, et multiplier l'information ne peut que ralentir les traitements.

De plus l'information apportée par les lignels, pointels ou surfels n'est pas toujours pertinente. L'approche que nous présentons consiste donc à ne travailler principalement qu'avec les cellules de plus grande dimensions (les pixels ou les voxels[9]) et de ne prendre en compte les cellules de dimension inférieure à n que lorsqu'elles sont indispensables à la cohérence avec la topologie-étoile. Ainsi nous limitons l'utilisation de ce qui pourrait être considéré comme totalement artificiel : les lignels, pointels et surfels.

3.3.2 k-adjacence et k-connexité

Nous allons tout d'abord examiner les notions d'adjacence et de connexité. En effet ces notions sont primordiales puisque découlent d'elles celles de courbes, de contours, de frontières, ... Le fait de vouloir ne travailler principalement qu'avec les cellules de plus grande dimension nous oblige à spécialiser la notion de connexité donnée par la définition 3.18.

On dit habituellement que deux pixels ou voxels sont adjacents s'ils se touchent suivant un lignel, pointel ou surfel dans le cas des voxels. Nous obtenons ainsi la 4 et la 8-connexité dans le cas des images 2-D. Si l'ont étend directement cette notion à la topologie-étoile on obtient la définition suivante :

Définition 3.20 (k-adjacence) *Deux m-cellules e_1, e_2 dans $\mathbb{R}^n$, $m \leqslant n$, sont* k-adjacentes*, $k < m$, si et seulement si il existe une k'-cellule e telle que $m \geqslant k' \geqslant k$ et $e_1, e_2 \in star(e)$.*

Remarque : La 0-adjacence correspond à la 8-connexité classique et la 1-adjacence à la 4-connexité.

D'une manière générale, on dira que deux n-cellules sont adjacentes si et seulement si il existe une troisième cellule qui borne les deux premières (voir figure 3.9).

Contrairement aux modèles classiques, on ne peut se contenter de cette définition de l'adjacence entre cellules pour définir les notions de connexité et d'ensemble connexe. D'abord parce que cette définition de l'adjacence entre cellules correspond à celle habituellement utilisée et ne résout donc rien ; ensuite parce que les

[9]Nous appellerons ces cellules les *n-cellules* où n désigne la dimension maximale, c'est à dire 2 pour les pixels et 3 pour les voxels.

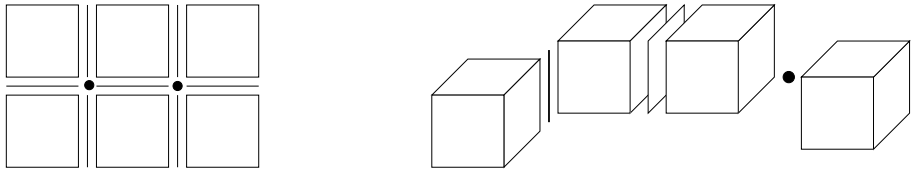


FIG. 3.9 – n-cellules adjacentes deux à deux

cellules de dimension non maximales ne sont pas réellement utilisées dans cette définition. De plus la notion de connexité est directement déduite de celle de l'adjacence.

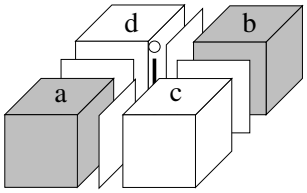


FIG. 3.10 – Connexité

Or une connexité induite par la relation d'adjacence ne correspond pas à ce que l'on recherche. En effet si l'on regarde la figure 3.10, les voxels gris sont adjacents par le lignel central, les voxels blancs par le pointel. On retrouve les problèmes du type "paradoxe de la connexité". Il est donc essentiel de définir une adjacence particulière permettant de définir la connexité. Attention il n'est pas question ici de remettre en cause l'adjacence de la définition 3.20, mais d'en donner une définition supplémentaire uniquement guidée par des considérations pratiques d'analyse d'images.

En fait nous allons définir une *adjacence dans un ensemble* qui permettra de préciser si deux n-cellules appartenant à un même ensemble sont adjacentes ou pas. On pourra ainsi en déduire la connexité de l'ensemble. L'idée est la suivante : Les voxels gris de la figure 3.10 sont "plus fortement" adjacents car liés par un lignel et non pas un pointel comme les voxels blancs. Nous allons donc définir la notion de *cellule jonction* et imposer que l'adjacence se fasse par ce type de cellules.

Définition 3.21 (k-jonction) *Soient deux n-cellules e, e' k-adjacentes ; on appelle k-Jonction*(e, e') *la cellule de plus grande dimension (supérieure à k) parmi l'ensemble des cellules bornant e et e'.*

Définition 3.22 (k-adjacence dans un ensemble) *Soit* $E \subset \mathbb{R}^n$ *un ensembles de cellules,* $e, e' \in E$ *sont dites* k-adjacentes *dans E si et seulement si e et* e' *sont k-adjacentes et 0-Jonction*$(e, e') \in E$.

Nous obtenons donc les définitions suivantes de la k-connexité et des ensembles k-connexes :

Définition 3.23 (k-connexité) *Deux m-cellules $e, e' \in E$ sont* k-connectées *si et seulement si il existe une suite de m-cellules de E $e = e_0, \ldots, e_p = e'$ telles que $\forall i = 1 \ldots p$, e_i est k-adjacente dans E à e_{i-1}.*

Définition 3.24 (ensembles k-connexes) *E, sous-ensemble de dimension d de $\mathbb{R}^n$ est* k-connexe *si et seulement si pour tout couple de cellules (e,e') de dimension d de E, e et e' sont k-connectées.*

Corollaire 3.14 *Soit E un ensemble k-connexe, alors E est k'-connexe, $\forall\ k' \leqslant k$.*

Preuve :
Si E est un ensemble k-connexe, alors il existe une suite de n-cellules de E $e = e_0, \ldots, e_p = e'$ telles que $\forall i = 1 \ldots p$, e_i est k-adjacente dans E à e_{i-1}.

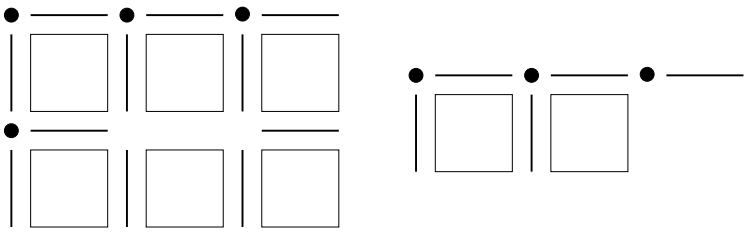
Or e_i est k-adjacente dans E à e_{i-1} signifie qu'il existe une k"-cellule e telle que $k'' \geqslant k$ et $e_i, e_{i-1} \in star(e)$. $k'' \geqslant k \Longrightarrow k'' \geqslant k'$ puisque $k' \leqslant k$. De même la k-$Jonction(e_i, e_{i-1})$ est aussi la k'-$Jonction(e_i, e_{i-1})$. Donc e_i est k'-adjacente dans E à e_{i-1} ; donc E est un ensemble k'-connexe. □

Remarque : Par abus de langage on emploiera désormais les termes Jonction, adjacence, et connexité pour 0-Jonction, 0-adjacence et 0-connexité.

Après avoir défini les notions de connexité, nous pouvons maintenant passer aux notions propres au traitement d'images, notamment celles de régions et d'adjacence entre régions.

3.3.3 le concept de région

Traditionnellement une région est définie comme un ensemble connexe de pixels ou de voxels dans le cas d'images 3-dimensions. Dans notre formalisme cela reviendrait à définir une région comme un ensemble connexe de n-cellules. Or si l'on regarde la figure 3.11.a, les pointels et le lignel manquants forment alors également une région. Même si du point de vue de la topologie-étoile cet ensemble de cellules a un sens, en terme d'analyse d'images, il ne serait pas raisonnable de laisser des régions définies uniquement à partir de cellules de dimension inférieure à n.



a – "trou" dans une région b – une région et son bord

FIG. 3.11 – Exemples de problèmes posés par les cellules de dimension inférieure à n dans les régions.

Le problème posé par la figure 3.11.b est différent, il vient du fait que nous voulons que les régions aient un bord, c'est à dire que les frontières entre les régions soient partagées entre elles. Mais il ne faut pas pour autant que la définition des régions autorise ce genre de "débordement". Il est clair que de tels ensembles, bien que connexes, ne correspondent pas à l'intuition que l'on se fait d'une région. Nous avons donc besoin d'une condition supplémentaire afin d'interdire de telles configurations.

Nous allons dans un premier temps définir les *k-hyper-surfaces*. De là on en déduira la définition d'une région et d'une image. On pourra vérifier qu'une k-hyper-surface, et donc à fortiori une région, interdit bien les cas comme ceux présentés à la figure 3.11 grâce aux propositions 3.17 et 3.18 ci-après.

Définition 3.25 (k-hyper-surface) R *ensemble de n-cellules de* $\mathbb{R}^n$ *est une* k-hyper-surface *si et seulement si* R *est k-connexe et* $\overset{\circ}{\overline{\mathrm{R}}} \subseteq \mathrm{R} \subseteq \overline{\overset{\circ}{\mathrm{R}}}$.

La figure 3.11.a est désormais impossible car $\overline{\mathrm{R}}$ impose de « combler » les trous, et prendre l'intérieur de $\overline{\mathrm{R}}$, c'est à dire $\overset{\circ}{\overline{\mathrm{R}}}$, n'y changera rien. Puisque on doit avoir $\overset{\circ}{\overline{\mathrm{R}}} \subseteq \mathrm{R}$, R ne peut contenir de « trous[10] ». De même pour la figure 3.11.b, prendre la fermeture de R, rajoutera des n-cellules autour des cellules de dimension inférieure à n, or ces cellules n'appartiennent pas à R et donc la condition $\mathrm{R} \subseteq \overline{\overset{\circ}{\mathrm{R}}}$ est contredite.

[10]de dimension inférieure à n.

D'une manière plus formelle les propositions 3.17 et 3.18 nous permettront de vérifier qu'une k-hyper-surface correspond bien aux critères que nous recherchons : un ensemble de cellules formant une région doit correspondre à une surface dans $\mathbb{R}^n$:

1. il ne peut y avoir de « trous » de dimension non maximale (inférieure à n) dans la surface ;
2. nos régions étant définies à partir des cellules de dimension n, toute cellule de dimension inférieure à n doit borner une cellule de dimension n de la région.

Pour pouvoir prouver ces deux propositions, nous avons besoins des Lemmes 3.15 et 3.16.

Lemme 3.15 $x \in \overset{\circ}{\mathrm{R}} \Rightarrow star(x) \subseteq \overset{\circ}{\mathrm{R}}$

Preuve :
$x \in \overset{\circ}{R} \Leftrightarrow star(x) \subseteq \overline{R}$. Or $\forall\, y \in star(x)$, $\Big(star(y) \subseteq star(x)\Big) \Rightarrow star(y) \subseteq \overline{R}$.
De plus $star(y) \subseteq \overline{R} \Leftrightarrow y \in \overset{\circ}{\overline{R}}$, donc $\forall\, y \in star(x)$, $y \in \overset{\circ}{\overline{R}} \Rightarrow star(x) \subseteq \overset{\circ}{\overline{R}}$. □

Lemme 3.16 $\forall\, x \in \overset{\circ}{\mathrm{R}},\ dim(x) = n \Rightarrow x \in \mathrm{R}$

Preuve :
$x \in \overset{\circ}{R}$ et $dim(x) = n \Leftrightarrow x \in \overline{R}$ donc $\exists\, y \in star(x)$ tel que $y \in R$.
Or $dim(x) = n \Rightarrow star(x) = \{x\}$ donc $y = x \Rightarrow x \in R$ □

Proposition 3.17 *Soit* R *un ensemble de cellules k-connexes, les deux propositions suivantes sont équivalentes :*

(1) $\overset{\circ}{\mathrm{R}} \subseteq \mathrm{R}$

(2) $\forall\, x \in \mathbb{R}^n,\ \Big(star(x) \setminus \{x\}\Big) \subseteq \mathrm{R} \Rightarrow x \in \mathrm{R}$

Preuve :
(1⇒2)
Montrons que si $\overset{\circ}{R} \subseteq R$ alors nous avons l'implication suivante : $star(x) \setminus \{x\} \subseteq R \Rightarrow x \in R$
On a $\forall\, x \in \mathbb{R}^n$, $star(x) \setminus \{x\} \subseteq R \Rightarrow x \in \overline{R}$. Deux cas se présentent à nous :

(1) $x \in \overset{\circ}{\overline{R}}$: Comme $\overset{\circ}{\overline{R}} \subseteq R$, $x \in \overset{\circ}{\overline{R}} \Rightarrow x \in R$ □

(2) $x \in \overline{R} \setminus \overset{\circ}{\overline{R}}$ c'est à dire $x \notin \overset{\circ}{\overline{R}}$ mais $x \in \overline{R}$.
$x \in \overline{R} \setminus \overset{\circ}{\overline{R}} \Rightarrow \exists\, y \neq x,\ y \in star(x)$ tel que $y \notin \overline{R}$.
Or $\overline{R} \supset R$ donc $y \notin R$. Ce qui est en contradiction avec l'hypothèse $star(x) \setminus \{x\} \subseteq R$. Donc $x \in \overline{R} \setminus \overset{\circ}{\overline{R}}$ est impossible □

.

(1⇐2)
Supposons que l'on ait $\forall\, x$, de dimension $d \leqslant n$, $star(x) \setminus \{x\} \subseteq R \Longrightarrow x \in R$. Soit $x \in \overset{\circ}{\overline{R}}$, montrons qu'alors $x \in R$.

Nous allons le montrer par récurrence sur la dimension de x.

1. D'après le lemme 3.16 la propriété est vraie pour x de dimension n
2. Supposons la propriété vraie au rang k, c'est à dire :

$$\forall x \in \overset{\circ}{\overline{R}},\ dim(x) \geqslant k \ \Rightarrow\ x \in R$$

Montrons que cette propriété est alors vraie au rang $k-1$:
Puisque la propriété est vraie au rang k, si $dim(x) \geqslant k$ alors $x \in R$.
Si $dim(x) = k-1$ alors :

$$\left.\begin{array}{r} x \in \overset{\circ}{\overline{R}} \overset{(Lemme\ 3.15)}{\Longrightarrow} star(x) \subseteq \overset{\circ}{\overline{R}} \\ \\ \forall y \in star(x) \setminus \{x\},\ dim(y) \geqslant k \end{array}\right\} \Rightarrow y \in \overset{\circ}{\overline{R}} \text{ et } dim(y) \geqslant k.$$

Puisque la propriété est vraie au rang k, $y \in R$ donc

$$\Big(\forall\, y \in star(x) \setminus \{x\},\ y \in R \Leftrightarrow star(x) \setminus \{x\} \subseteq R\Big) \Rightarrow x \in R$$

□

Proposition 3.18 *Soit* R *un ensemble de cellules k-connexes de* $\mathbb{R}^n$*, les deux conditions suivantes sont équivalentes :*

(1) $\mathrm{R} \subseteq \overset{\circ}{\overline{\mathrm{R}}}$

(2) Toute cellule de R borne une cellule de R de dimension n

Preuve :
(1⇒2)
$\forall x \in R, x \in \overline{\overset{\circ}{R}} \Rightarrow star(x) \bigcap \overset{\circ}{R} \neq \emptyset$, donc $\exists z \in star(x)$ tel que $z \in \overset{\circ}{R}$. $z \in \overset{\circ}{R} \Rightarrow star(z) \subseteq R$.
D'autre part il existe forcément une cellule de dimension n appartenant à l'étoile ouverte de z. Nous appellerons cette cellule y. Comme $star(z) \subseteq R,\ y \in R$.
De plus $z \in star(x) \Rightarrow star(z) \subseteq star(x)$, donc $y \in star(x)$, $y \in R$, y cellule de dimension n. □

(1⇐2)
$\forall x \in R$, $\exists y \in R$ de dimension n tel que $y \in star(x)$. Puisque y est de dimension n, $star(y) = y$. Or $y \in R$ donc $star(y) \subseteq R$ et $y \in \overset{\circ}{R}$. Comme $y \in star(x)$, $y \in star(x) \bigcap \overset{\circ}{R}$. Donc $star(x) \bigcap \overset{\circ}{R} \neq \emptyset$, soit d'après la définition de la fermeture d'un ensemble $x \in \overline{\overset{\circ}{R}}$ □

Remarque : On remarquera que la fermeture d'une 0-hyper-surface de dimension 1 correspond à la notion de courbe donnée par la définition 3.17.

Nous pouvons maintenant donner une définition pour les images et les régions :

Définition 3.26 (image) *Une* image *dans* $\mathbb{R}^n$ *est une 0-hyper-surface de dimension n.*

Définition 3.27 (région) *Une* région *d'une image* I *est une 0-hyper-surface de dimension n incluse dans* I.

Nous pouvons voir que les définitions d'une région et d'une image sont similaires. En fait on peut voir une image comme une région de $\mathbb{R}^n$, ou une région comme une image incluse dans une autre. Ces deux concepts sont en fait les mêmes. En particulier les images ne doivent plus être forcément rectangulaires. Ceci pourra nous permettre de traiter une région particulière comme une image.

De plus, on remarquera qu'une partie de la frontière peut appartenir à la région. En effet il est important de ne pas laisser les éléments formant les différentes frontières de côté ; rappelons que c'est grâce à cela que notre modèle s'affranchit des problèmes posés par les connexités habituelles. D'ailleurs ceci est imposé par la définition d'une segmentation telle que présentée par Pavlidis [Pav77], qui, on le remarquera, reste valable avec notre formalisme :

Définition 3.28 (segmentation) *Une* segmentation *associe un prédicat* P *d'homogénéïté à une image* I *définissant ainsi une partition de* I *en régions* $R_1, \ldots, R_k$ *telles que :*

1. $\mathrm{I} = \bigcup_{i=1}^{k} R_i$
2. $R_i \bigcap R_j = \emptyset,\ i, j \in \{1, \ldots, k\}$
3. $\forall\ i = 1, \ldots, k$ $\mathrm{P}(R_i)$ *est vrai.*
4. $\forall\ i \neq j, P(R_i \bigcup R_j)$ *est faux pour tout couple de régions* (R_i, R_j) *adjacentes.*

3.3.4 adéquation avec la topologique classique dans $\mathbb{R}^n$

Rappelons qu'un des buts de notre modèle est de pouvoir réutiliser les définitions et théorèmes de la topologie euclidienne dans $\mathbb{R}^n$. Vérifions que cela est vrai pour les notions d'*extérieur*, de *bord* et de *frontière*. Les définitions suivantes sont communément admises [Bro68] :

Pour tout sous-ensemble A d'un espace topologique X,

Extérieur : $\mathrm{Ext}(A) = \overset{\circ}{B}$ où $B = (X \setminus A)$

Bord : $\mathrm{Bd}(A) = A \setminus \overset{\circ}{A}$

Frontière : $\mathcal{F}r(A) = \mathrm{Bd}(A) \bigcup \mathrm{Bd}(X \setminus A) = \mathrm{Ext}(A) \setminus \overset{\circ}{A}$

Ces définitions restent valables dans notre formalisme en prenant $X = \mathbb{R}^n$.

Remarque : Il n'est pas possible de prendre $X = I$ car alors les bords (au sens littéral du terme) de l'image font partie de l'intérieur de l'image et donc une région qui serait située au bord de l'image n'aurait pas de frontière le long de ce bord. Son contour ne serait donc pas fermé ! Il en résulte qu'une image I est une partie strictement incluse dans l'espace topologique de $\mathbb{R}^n$.

3.3.5 adjacence entre régions

Maintenant que nous avons défini ce qu'est une région ainsi que son bord et sa frontière, examinons le problème de l'adjacence entre régions. En effet une représentation couramment utilisée est le graphe d'adjacence où les sommets du graphe représentent les régions et les arêtes la relation d'adjacence entre régions.

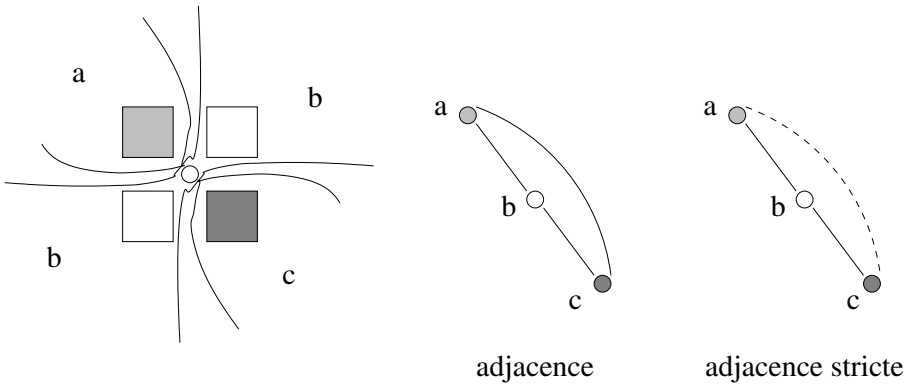


FIG. 3.12 – Adjacence et Adjacence stricte

Que se passe-t-il avec notre formalisme ? Communément on considère que deux régions sont adjacentes si elles ont une frontière commune, c'est à dire, au sens de notre espace topologique, que l'intersection de leur frontière est non-vide. Cette définition, dite *classique* de l'adjacence ne nous satisfait pas. En effet si l'on regarde la figure 3.12 on remarque que c'est le point interpixel central qui permet de connecter les deux pixels blancs qui forment donc une région unique (b). Si nous prenons la définition classique de l'adjacence ce point interpixel appartient à toutes les frontières et les trois régions présentes sont donc toutes adjacentes deux à deux. Or le graphe d'adjacence permet d'avoir une représentation abstraite de l'image. Ainsi lors du processus de segmentation, on ne travaillera plus que sur le graphe, et les fusions entre régions seront réalisées entre sommets voisins (reliés par une arête). Si l'on examine la figure 3.12, on peut voir que le fait de fusionner les deux régions (a) et (c) implique la séparation en deux de la région (b). Cette information n'est pas contenue dans le graphe. C'est pourquoi il nous semble indispensable de définir une nouvelle adjacence que nous appellerons *l'adjacence stricte*. En fait on peut voir que tout est lié à la position particulière du pointel central. Nous dénommerons ce type de cellules *cellules liantes*.

Définition 3.29 (cellule liante) *e est une* cellule liante *de R si et seulement si* $e \in Bd(R)$ *et il existe 2 n-cellules* $e_1, e_2 \in R$ *telles que* $e = Jonction(e_1, e_2)$.

On peut voir une cellule liante comme une **jonction du bord** de la région ou encore comme une cellule assurant la 0-connexité. En effet si cette cellule est en-

levée, la propriété de la 0-connexité est perdue. Nous pouvons maintenant préciser le rôle particulier qu'occupent ces cellules dans la notion d'adjacence et définir une *adjacence stricte* :

Définition 3.30 (adjacence stricte) *Deux régions R_1 et R_2 sont* strictement adjacentes *si et seulement s'il existe deux cellules e_1 et e_2 de dimension n, $e_1 \in R_1, e_2 \in R_2$, et s'il existe une k-cellule e de l'image, $(k < n)$, telle que $e = Jonction(e_1, e_2)$ et e n'est pas une cellule liante d'une troisième région.*

Théorème 3.19 *Soit R_1 et R_2 deux régions strictement adjacentes ; alors il existe e une k-cellule de l'image, $k < n$, telle que $e = Jonction(e_1, e_2)$ avec $e_1 \in R_1$, $e_2 \in R_2$. On a alors :*

$$\text{si } e \in R_1 \cup R_2 \text{ alors } R_1 \bigcup R_2 \text{ est une région.}$$

Preuve :
$R_1 \bigcup R_2$ est une région si et seulement si :

(1) $R_1 \bigcup R_2$ est connexe[11].

(2) $\overline{\overset{\circ}{R_1 \bigcup R_2}} \subseteq R_1 \bigcup R_2 \subseteq \overline{R_1 \overset{\circ}{\bigcup} R_2}$

Montrons d'abord que $R_1 \bigcup R_2$ est connexe. R_1 et R_2 sont connexes par définition. e borne e_1 et e_2 donc e est connexe à e_1 et e_2. Si $e \in R_1$ (resp. $e \in R_2$) alors toute cellule de R_1 (resp. de R_2) est connexe à e_1 (resp. à e_2) qui est connexe à e qui est connexe à e_2 (resp. à e_1) qui est connexe à toute cellule de R_2 (resp. de R_1). Donc par transitivité, $R_1 \bigcup R_2$ est connexe.

Reste à montrer que $\overline{\overset{\circ}{R_1 \bigcup R_2}} \subseteq R_1 \bigcup R_2 \subseteq \overline{R_1 \overset{\circ}{\bigcup} R_2}$.

Montrons d'abord $\overline{\overset{\circ}{R_1 \bigcup R_2}} \subseteq R_1 \bigcup R_2$.

D'après la proposition 3.17 $\overline{\overset{\circ}{R_1 \bigcup R_2}} \subseteq R_1 \bigcup R_2 \Leftrightarrow \forall\, x$ cellule de dimension inférieure à n de $\mathbb{R}^n$, $\Big(star(x) \setminus \{x\}\Big) \subseteq R_1 \bigcup R_2 \Rightarrow x \in R_1 \bigcup R_2$.

Supposons que x cellule de dimension inférieure à n de $\mathbb{R}^n$ vérifie $\Big(star(x) \setminus \{x\}\Big) \subseteq R_1 \bigcup R_2$. Montrons alors que $x \in R_1 \bigcup R_2$.

1. Si $\Big(star(x) \setminus \{x\}\Big) \subseteq R_1$ (resp. $\Big(star(x) \setminus \{x\}\Big) \subseteq R_2$) le problème est résolu puisque R_1 (resp. R_2) est une région, on a alors $x \in R_1$ (resp. $x \in R_2$).

[11] Rappel : par abus de langage connexe est employé pour 0-connexe.

2. Examinons le cas ou $\Big(star(x) \setminus \{x\}\Big)$ contient des cellules de R_1 et de R_2. Appelons x_1 et x_2 deux cellules de $\Big(star(x) \setminus \{x\}\Big)$ telles que x_1 et x_2 appartiennent respectivement à R_1 et R_2. On va montrer par l'absurde que $x \in R_1 \cup R_2$. Pour cela supposons que $x \notin R_1 \bigcup R_2$, c'est à dire que $x \in R_3$ une troisième région. D'après la proposition 3.18 $\exists y \in R_3$ de dimension n telle que $y \in star(x)$. Or $star(x) \setminus \{x\} \subseteq R_1 \bigcup R_2$ donc si $y \neq x$, $y \in \Big(R_1 \bigcup R_2\Big) \bigcap R_3$. Or $\Big(R_1 \bigcup R_2\Big) \bigcap R_3 = \emptyset$, donc $y = x$ et x est de dimension n ce qui est en contradiction avec l'hypothèse. Donc $x \in R_1 \bigcup R_2$.

Montrons maintenant que $R_1 \bigcup R_2 \subseteq \overline{R_1 \overset{\circ}{\bigcup} R_2}$. D'après la proposition 3.18 $R_1 \bigcup R_2 \subseteq \overline{R_1 \overset{\circ}{\bigcup} R_2} \Leftrightarrow \forall x \in R_1 \bigcup R_2$, $\exists y$ une cellule de dimension n, $y \in R_1 \bigcup R_2$ telle que $y \in star(x)$. Montrons que $\forall x \in R_1 \bigcup R_2$ on peut trouver une telle cellule y. C'est immédiat, toujours d'après la proposition 3.18. En effet $x \in R_1 \bigcup R_2$ est équivalent à $x \in R_1$ ou $x \in R_2$. Or R_1 et R_2 sont des régions et donc si $x \in R_1$ d'après la proposition 3.18 $\exists y \in R_1$ cellule de dimension n telle que $y \in star(x)$. Il en est de même si $x \in R_2$. □

Cette fois-ci les deux régions (a) et (c) de la figure 3.12 ne peuvent pas être strictement adjacentes car le point interpixel central est une cellule liante pour la région (b). On a donc le graphe d'adjacence G_2. Sur ce graphe il n'est pas possible de fusionner les deux régions (a) et (c) sans tenir compte de la région (b).

Mais lors de la segmentation et de la création de la région (b), la décision concernant le point interpixel central n'est pas forcément définitive. Il peut donc être important de garder sous une certaine forme les arcs d'adjacence. On pourra ainsi lancer une analyse plus fine : par exemple lors de l'utilisation d'une méthode de segmentation autorisant les retours en arrière (backtrack).

3.4 Conclusion et perspectives.

Le formalisme que nous avons présenté nous permet de travailler sur un modèle cohérent de représentation des images. Il s'appuie sur la notion de complexe cellulaire, reste lié à la topologie classique sur $\mathbb{R}^n$, mais prend aussi en compte les contraintes de l'analyse d'images.

La topologie-étoile, topologie engendrée par les étoiles ouvertes, nous permet de récupérer les résultats de la topologie algébrique réelle, pour les transposer à nos cellules. Dès lors, les courbes et surfaces de $\mathbb{R}^n$ se trouvent transposées

"naturellement" et nous permettent de proposer un théorème de Jordan adapté à la topologie-étoile.

Nous avons adopté cette topologie en la spécialisant afin de tenir compte des spécificités du domaine de l'analyse d'images : besoin de définir des régions, des bords, des frontières. Mais avant d'arriver au concept de région, il nous a fallu affiner encore les notions d'adjacence et de connexité. En effet, si une région peut grossièrement être vue comme un ensemble (homogène) de voxels ou pixels connexes, la topologie-étoile oblige à étudier de près les éléments de dimension inférieure permettant de réaliser la connexité. Cette approche peut paraître lourde car elle demande l'étude des cellules de dimension non maximale. En fait nos définitions spécialisées pour l'analyse d'images, nous permettent de ne prendre en compte ces cellules que lorsqu'elles sont relatives aux bords des régions. Ainsi les contraintes que nous ajoutons n'alourdissent pas exagérément les structures de données et les traitements.

Il faut maintenant étudier de plus près les notions que nous avons décrite dans le cadre d'un espace à 3 dimensions. Notamment, il serait intéressant de définir proprement la notions de surface et d'en déduire un théorème discret de Jordan dans un espace à 3 dimensions. On pourra remarquer que cette notion de surface est sous-jacente : c'est un ensemble de surfels, lignels, pointels adjacents. Reste à trouver une formulation correcte et essayer de définir correctement la notion de surface fermée.

Nous disposons d'une topologie adaptée, encore faut-il l'utiliser. Il est donc primordial de définir un (ou des) algorithmes de segmentation respectant cette topologie. Les algorithmes présentés dans le chapitre 5 travaillent déjà en interpixel et leur conformité avec les impératifs de la topologie-étoile devraient être vérifiée. On notera tout de même que le graphe topologique des frontières que nous présentons au chapitre 4 est bien conforme à notre topologie.

Dans un avenir plus lointain, il faudra définir d'autres notions que celle de région. En effet la notion d'objet semble indispensable car un objet dans l'image est rarement représenté par une seule région. De plus les régions composant l'objet ne sont pas forcément deux à deux adjacentes. En effet un objet peut être recouvert partiellement par un autre. Mais ceci fait partie des objectifs à long terme et sort du cadre de cette thèse.

Chapitre 4

Graphe topologique des frontières

4.1 Remarque préliminaire

Dans la topologie classique dans $\mathbb{R}^n$, et également dans la topologie-étoile, on appelle frontière d'un sous-ensemble A d'un espace topologique X, l'ensemble $\mathcal{F}r(A) = \mathrm{Bd}(A) \bigcup \mathrm{Bd}(X \setminus A)$ où $\mathrm{Bd}(A)$ est le bord de A (voir définition à la section 3.3.4). Or en analyse d'images on s'intéresse à la notion de frontière entre deux régions R_1 et R_2, c'est à dire à une partie connexe de $\mathcal{F}r(R_1) \bigcap \mathcal{F}r(R_2)$. Par la suite, et par abus de langage, le terme frontière désignera pour nous une frontière entre deux régions, c'est à dire une frontière en terme d'analyse d'images. Lorsqu'on parlera de la frontière $\mathcal{F}r(R)$ d'une région R, nous dirons « frontière au sens topologique du terme ».

4.2 Introduction

Dans le chapitre 3 nous avons présenté une modélisation topologique des images. Comme nous l'avons précisé dans l'introduction de ce document, nous voulons toujours garder à l'esprit que tout algorithme ou modèle de segmentation, doit pouvoir être intégré dans un processus plus général de reconnaissance. Une structure de données adaptée s'avère donc indispensable pour pouvoir réellement exploiter la topologie-étoile. Kovalevsky dans [Kov89] propose une représentation par liste de cellules. Afin de ne traiter que les données significatives de l'image, Kovalevsky

a introduit la notion de *block cells* : les éléments 2-dimension des *block cells* sont des sous-complexes ouverts de dimension 2 correspondant aux régions (au sens classique du terme c'est à dire ensemble de pixels deux à deux adjacents). Les frontières entre les régions ainsi définies deviennent les éléments 1-dimension et les points de jonction de ces frontières, les éléments 0-dimension. Cette structure ne nous convient pas pour deux raisons :

1. Nous préférons conserver une structure de graphe plutôt que d'utiliser des tableaux de listes. En effet on dispose alors de toute l'algorithmique de graphe pour aider à la résolution de certains problèmes.
2. Comme il est déjà dit dans l'article [Kov89], le squelette 1-dimension du complexe dual d'un *block cells* est équivalent au graphe d'adjacence. Or dans ce même article il apparaît que les graphes de voisinage, dont fait partie le graphe d'adjacence, ne respectent pas les axiomes de la topologie. Il semble donc que cette structure n'est pas plus de propriétés que le graphe d'adjacence, en tout cas du point de vue de la topologie des images.

Le deuxième argument justifie également le fait que le graphe d'adjacence classique n'est pas une structure de données satisfaisante. Il fallait donc chercher une structure de graphe permettant de coder l'image tout en décrivant sa topologie. Pour cela **deux points importants** devaient être respectés :

- coder les frontières et pas seulement l'adjacence ;
- garder la notion de « trou » dans une région, c'est à dire enregistrer les phénomènes d'inclusion de régions les unes dans les autres.

Nous avons élaboré une représentation que nous appelons le *graphe topologique des frontières*[1] [Fio96]. Cette représentation utilise une modélisation dérivée des cartes combinatoires que nous présentons à la section 4.4. Ces modélisations à base topologique sont déjà utilisées dans plusieurs logiciels de C.A.O. ou synthèse. Nous espérons, par nos recherches actuelles et futures, montrer l'intérêt de telles modélisations pour l'analyse d'images et ainsi valider notre représentation afin de pouvoir l'implanter dans de futurs logiciels. De fait, la question de trouver une bonne représentation pour l'analyse d'images est toujours d'actualité. En effet même si le graphe d'adjacence est connu depuis longtemps [Ros74, Pav77], il

[1] Nous aurions également pu l'appeler *carte des frontières* en référence au modèle des cartes combinatoire dont cette représentation est dérivée.

n'est pas complètement satisfaisant car certaines informations importantes sont manquantes. Ainsi W.G. Kropatsch [Kro94] utilise, dans le cadre de l'analyse hiérarchique à base de pyramides d'images, une structure de graphes planaires connexes duaux (voir 4.4). Nous expliquerons pourquoi cette structure ne nous convient pas et présenterons à la section 4.6 la nôtre. Nous prouverons également un algorithme réalisant l'extraction du graphe topologique des frontières en un seul balayage de l'image. Mais dans un premier temps voici quelques brefs rappels sur la théorie des graphes.

4.3 Brefs rappels sur la théorie des graphes

La terminologie et les définitions données dans cette section sont extraites des livres de C. Berge [Ber70] et M. Gondran et M. Minoux [GM95].

4.3.1 notions générales

Définition 4.1 (graphe) *Un* graphe $G = (V, E)$ *est déterminé par la donnée :*

- *d'un ensemble V dont les éléments sont appelés des* sommets.
- *d'un ensemble E dont les éléments sont des couples (u, v) de sommets. Si ces couples sont ordonnés on appelle ces éléments des* arcs *et le graphe est dit* orienté. *Sinon on appelle ces éléments des* arêtes *et le graphe est dit* non orienté.

Dans toute la suite les graphes que nous utiliserons seront implicitement considérés comme non orientés. Si nous devons utiliser des graphes orientés alors nous emploierons le terme de « graphe orienté » de manière explicite.

On appelle *boucle* une arête (u, v) telle que $u = v$. Un *multi-graphe* est un graphe pour lequel il peut exister plusieurs arêtes entre deux sommets u et v donnés. Un graphe est dit *simple* si :

- il est sans boucle ;
- ce n'est pas un multi-graphe, c'est à dire qu'il n'y a jamais plus d'une arête entre deux sommets quelconques.

On dit que deux sommets u et v sont adjacents s'il existe une arête $(u, v) \in E$. L'ensemble des sommets voisins d'un sommet u, noté $\Gamma_G(u)$, est l'ensemble de tous

les sommets v du graphe tels que u et v soient adjacents.

Définition 4.2 (degré) *Le* degré *du sommet v, noté $d_G(v)$ (ou $d(v)$ lorsqu'il n'y a pas d'ambiguïté) est le cardinal de l'ensemble $\Gamma_G(v)$.*

Un sommet est dit *isolé* si son degré est nul.

Définition 4.3 (chaîne) *Une* chaîne *de longueur q est une séquence de q arêtes :*

$$P = e_1, e_2, ..., e_q$$

telle que chaque arête e_r de la séquence $(2 \leqslant r \leqslant q-1)$ ait une extrémité commune avec l'arête e_{r-1} $(e_{r-1} \neq e_r)$ et l'autre extrémité commune avec l'arête e_{r+1} $(e_{r+1} \neq e_r)$. Les sommets e_1 et e_q sont appelés les extrémités *de la chaîne P. On dit que la chaîne P* joint *les sommets e_1 et e_q.*

Dans le concept orienté on parle de *chemin* et on impose la condition supplémentaire que tous les arcs soient orientés dans le même sens. Par abus de langage, on parlera également de chemin pour des graphes non orientés, le terme « chemin » signifie alors « chaîne ». On appelle *cycle* une chaîne $P = e_1, e_2, ..., e_q$, à une permutation près[2], telle que :

- la même arête ne figure pas deux fois dans la chaîne ;
- les deux sommets extrémités de la chaîne coïncident.

Un *cycle élémentaire* est un cycle minimal (pour l'inclusion), c'est à dire ne contenant strictement aucun autre cycle.

Il est facile de remarquer que la relation « $u = v$, ou $u \neq v$ et il existe une chaîne reliant u et v » est une relation d'équivalence. On peut alors donner la définition suivante :

Définition 4.4 (composante connexe) *Les classes d'équivalence de la relation « $u = v$, ou $u \neq v$ et il existe une chaîne reliant u et v » constituent une partition de V en graphes, appelés les* composantes connexes *de G.*

Un graphe sera dit connexe s'il ne forme qu'une seule composante connexe, c'est à dire si quelque soit u et v deux sommets, il existe une chaîne reliant ces deux sommets. La figure 4.1 montre un multi-graphe et présente la notion de boucle, de sommets adjacents et de composante connexe.

[2]Le cycle défini par la chaîne *a,b,c,a* est équivalent à celui défini par les chaînes *b,c,a,b* et *c,a,b,c*.

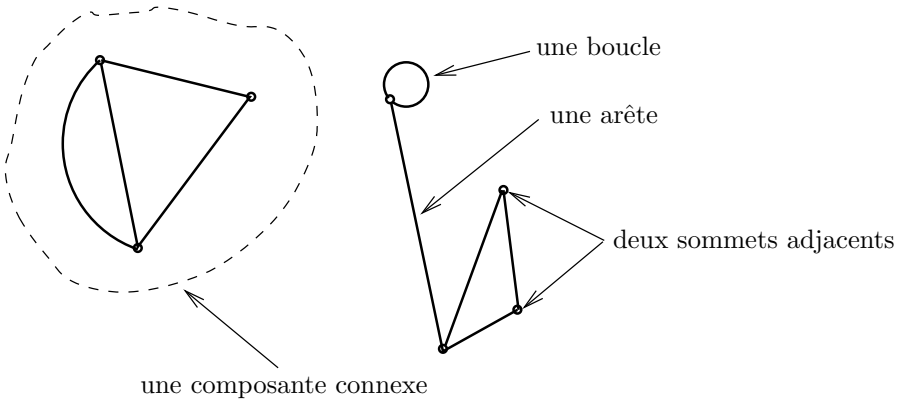


FIG. 4.1 – Un graphe

4.3.2 graphes planaires

Nous donnons ici brièvement les caractéristiques principales d'une classe particulière de graphe : les graphes planaires.

Définition 4.5 (graphe planaire) *Un graphe G est dit* planaire *lorsqu'il admet une représentation sur un plan par des points distincts figurant les sommets et des courbes simples figurant les arêtes, deux telles courbes ne se rencontrant pas en dehors de leurs extrémités.*

Les parties connexes (pour la topologie classique dans $\mathbb{R}^2$) du plan P délimitées par des arêtes du graphe G sont appelées les *faces*. L'ensemble des arêtes qui touchent une face constitue la *frontière* de la face. Deux faces sont adjacentes lorsque leurs frontières ont au moins une arête commune (et pas seulement des sommets communs). D'une manière générale, la frontière est constituée de cycles élémentaires disjoints (pour les arêtes), d'arêtes pendantes ou d'arêtes reliant deux cycles. À part pour la face extérieure, dite *face infinie*, il y a toujours un cycle contenant en son intérieur toutes les autres arêtes de la frontière, c'est le *contour* de la face. Ces différentes notions sont présentées à la figure 4.2.

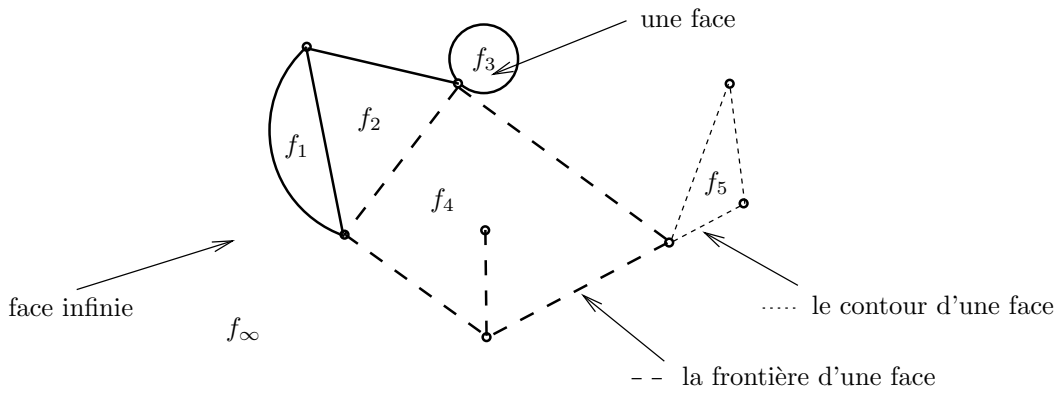


FIG. 4.2 – Un graphe planaire

4.4 Structure de graphes duaux

W.G. Kropatsch et H. Macho [KM95] ont présenté une structure de données à base de graphes planaires pour la représentation d'une image et l'utilisation de cette représentation dans un procédé d'analyse basé sur une pyramide d'images. Nous ne développons pas ce processus de traitement, pour de plus amples informations le lecteur pourra par exemple se reporter à [MMR91]. L'originalité de leur approche vient du fait qu'ils utilisent un multi-graphe qui est une extension du graphe d'adjacence des régions et son dual. Ce dernier permet d'effectuer certains contrôles afin de maintenir la cohérence de la représentation et d'éliminer les éléments d'information inutiles ou redondants.

Leur utilisation d'un multi-graphe est dûe à la même constatation que nous avons déjà faite : un graphe d'adjacence ne modélise pas la topologie de l'image. En effet dans le cas de deux régions adjacentes mais ayant deux frontières distinctes, le graphe d'adjacence ne permet pas de rendre compte de ce fait. D'où l'idée assez naturelle d'utiliser un multi-graphe où chaque arête symbolise l'existence d'une frontière entre les régions représentées par les sommets du graphe. On obtient ce qu'on pourrait appeler un multi-graphe d'adjacence.

Mais cette simple amélioration du graphe d'adjacence n'est pas suffisante, en effet les « trous » ne sont pas différenciés. Or il parait évident qu'en terme d'analyse

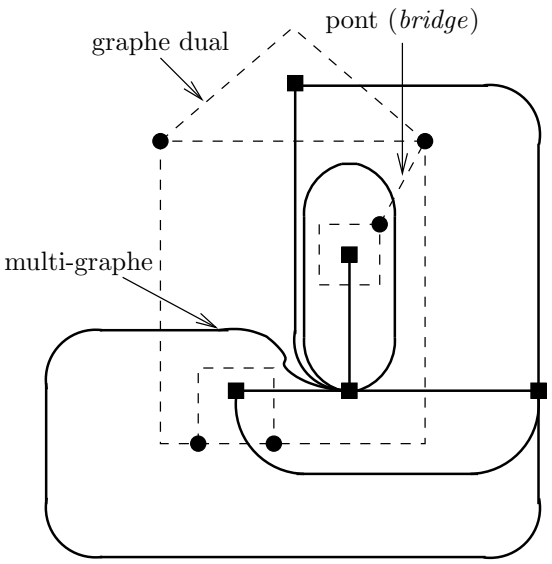


FIG. 4.3 – Le multi-graphe et son dual proposé dans [Kro94]

d'images cette information est importante. Le graphe ou le multi-graphe d'adjacence ne font pas de différences entre ces deux types d'adjacences. Comment savoir lorsque deux sommets sont reliés par une arête si les deux régions correspondantes sont voisines où si l'une est incluse dans l'autre ?

W.G. Kropatsch propose dans [Kro94] de rajouter une arête supplémentaire dans le dual, un pont (« *un bridge* »), afin de relier le « trou » à sa région englobante. Ce pont se traduit alors par une boucle dans le multi-graphe d'origine. Ce cas de figure est présenté à la figure 4.3 extraite de [Kro94].

Afin d'être complètement homogène, W.G. Kropatsch et H. Macho dans [KM95] rajoutent un pont entre l'image et son cadre (dans le cas de la figure 4.3, cela revient à rajouter un cadre symbolisant le bord de l'image et un pont entre ce cadre et la maison – le graphe dual –). Ils obtiennent ainsi une structure totalement homogène et rendant compte de la topologie de l'image. Un des avantages de cette structure c'est que les graphes sont planaires et connexes, facilitant ainsi les traitements. Cette propriété de connexité semble en outre primordiale pour leur processus d'analyse.

Parmi les inconvénients on citera l'obligationn de maintenir deux graphes.

D'autre part nous ne voulons pas ajouter des arêtes fictives tels que les ponts car notre but est de combiner notre représentation avec le codage interpixel afin, par exemple, d'étiqueter chaque frontière (donc chaque arête) par son codage interpixel. Quel code donner alors aux ponts ? Ces arêtes vont nécessiter alors un traitement particulier et nous perdrions donc l'homogénéïté de la structure. De plus l'extension de ce modèle pour la représentation d'images 3D parait difficile.

Ces considérations nous ont amenés à étudier d'autres modélisations permettant de rendre explicitement compte de la topologie d'une image et pouvant se généraliser à la dimension 3 (voire n).

4.5 Cartes combinatoires

Les cartes combinatoires que nous présentons dans cette section sont des modèles utilisés en C.A.O., modélisation et synthèse. En effet depuis longtemps les personnes travaillant dans ces domaines ont été confrontées au problème d'une représentation efficace et pratique. C'est pourquoi il nous a semblé intéressant d'étudier les représentations utilisées afin de bénéficier de toute l'expérience acquise dans ce domaine.

4.5.1 présentation générale

Les cartes combinatoires s'inscrivent dans le cadre des modèles de représentation par contours. Dans ce type de représentation les objets (les solides) sont définis par une subdivision en faces (et ces faces en sommets et arêtes[3]). Ce modèle topologique est complété par un modèle de plongement définissant la projection de l'objet dans $\mathbb{R}^2$ ou $\mathbb{R}^3$. Clairement le problème du plongement ne fait pas partie pour l'instant de nos préoccupations. En effet nous avons déjà développé dans notre équipe (voir [Cha95]) des algorithmes permettant de « redessiner » l'image à partir du codage interpixel des régions[4]. Nous avons donc comme seule contrainte de pouvoir insérer ce codage dans notre représentation.

[3]On retrouve déjà les notions de cellules de dimension 0, 1, 2 et 3 utilisées dans la topologie-étoile

[4]Le cas des images 3D fait partie de nos préoccupations actuelles et des perspectives de recherche. Il n'entre pas dans le cadre de cette thèse.

Y. Bertrand et J.F. Dufourd affirment dans [BD94] que les cartes et leurs extensions offrent un avantage décisif car elles sont définies à partir d'un seul élément de base, le *brin*, et qu'elles utilisent des opérateurs simples à mettre en œuvre : des involutions et des permutations. Ce qui implique une grande homogénéïté dans la représentation et une généralisation immédiate à une dimension n. Ceci nous a paru être un avantage significatif. De plus l'analogie avec les graphes planaires est immédiate, on se rapproche donc du modèle idéal souhaité. On montrera à la fin de cette section que nous ne pourrons pas l'utiliser directement. Le graphe topologique des frontières sera en fait basé sur une représentation directement dérivée des cartes combinatoires.

La notion de carte est apparue en 1960 [Edm60]. Les travaux depuis ont été nombreux, il n'est donc pas possible dans le cadre de cette thèse d'en faire une étude exhaustive. Le lecteur intéressé pourra se reporter à l'état de l'art de P. Lienhardt [Lie90, Lie91].

J.F. Dufourd propose dans [Duf91] un autre modèle à base de cartes combinatoires : les n-hypercartes ou n-h-cartes (qui sont une extension des hypercartes présentées par R. Cori [Cor75]). Y. Bertrand et J.F. Dufourd dans [BD94] montrent comment passer des n-h-cartes aux n-g-cartes et n-cartes. Cette autre modélisation est surtout intéressante du point de vue théorique car elle prouve l'équivalence entre tous les modèles de représentation basés sur les cartes topologiques.

Nous présentons ici le premier modèle de cartes combinatoires, les 2-cartes, ainsi que le modèle des cartes généralisées ou n-g-cartes. Ces dernières constituent la base d'un modeleur volumique interactif expérimental développé à l'Université Louis Pasteur[5] de Strasbourg. Elles présentent donc un intérêt certains d'un point de vue pratique.

4.5.2 2-cartes

Les 2-cartes ont été la première modélisation mathématique des représentations à bases topologiques [Edm60, Jac70, Cor75].

Rappelons dans un premier temps quelques notions de base : une *permutation* d'un ensemble fini E est une séquence ordonnée de tous les éléments de E, chaque élément apparaissant exactement une fois. Par exemple si $E = \{a, b, c\}$, il existe 6

[5]Université Louis Pasteur, 7 rue René Descartes, 67084 Strasbourg cedex, France

permutations de E :

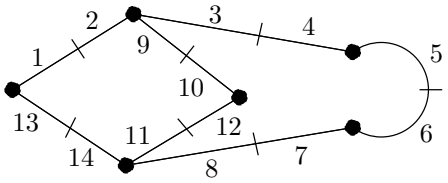
$$abc, acb, bac, bca, cab, cba$$

Une *involution* σ dans un ensemble E est une permutation qui satisfait $\sigma(\sigma(x)) = x$. Un point fixe x d'une fonction σ est tel que $\sigma(x) = x$.

Nous pouvons maintenant donner la définition d'une 2-carte :

Définition 4.6 (2-carte) *Une* 2-carte combinatoire $M = (D, \alpha_0, \alpha_1)$ *consiste en un ensemble fini D d'éléments appelés* brins *ou* demi-arêtes*, une involution α_0 sans point fixe, et une permutation α_1 dans D.*

Exemple :



La figure[6] *ci-contre montre une 2-carte M représentée dans le plan, où les brins sont numérotés et représentés par des demi segments ou courbes. On a :*
$D = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14\}$.
α_0 et α_1 sont données par la table suivante :

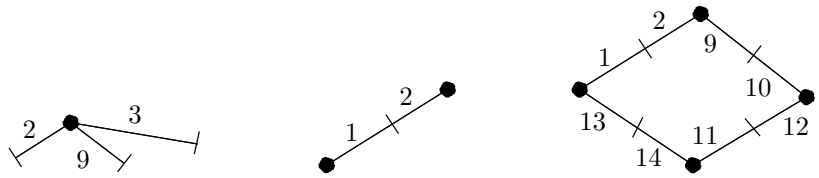
D	1	2	3	4	5	6	7	8	9	10	11	12	13	14
α_0	2	1	4	3	6	5	8	7	10	9	12	11	14	13
α_1	13	3	9	5	4	7	6	14	2	12	8	10	1	11

La notion d'orbite, présentée ci-dessus, ainsi que cette définition mathématique des 2-cartes va nous permettre de définir les éléments classiques de la théorie des graphes et plus particulièrement des graphes planaires : sommet, arête, face et composante connexe.

Définition 4.7 (orbite) *Soit une 2-carte $M = (D, \alpha_0, \alpha_1)$ et un ensemble $P = \{p_1, \ldots, p_k\}$ de permutations dans D, soit $< P >$ le groupe engendré par $\{p_1, \ldots, p_k\}$, l'orbite d'un brin $x \in D$ de M par P est l'ensemble :*

$$<P>(x) = \{y = \rho(x) \text{ où } \rho \in <P>\}$$

[6] *Dans cette figure, la « liaison » entre deux brins par α_0 est représentée par un petit trait fin entre deux demi arêtes ; $\alpha_1(x)$ est le brin y incident au même sommet et est le suivant dans le sens des aiguilles d'une montre autour de ce même sommet.*



(a) un sommet (b) une arête (c) une face

FIG. 4.4 – Exemples d'éléments de la théorie des graphes représentés par une 2-cartes.

La définition 4.7 d'une orbite permet de définir à partir des brins les éléments classiques de la théorie des graphes. Ainsi un sommet, une arête, une face et une composante connexe de M incidents à un brin x sont définis respectivement par $<\alpha_1>(x)$, $<\alpha_0>(x)$, $<\alpha_1^{-1} \circ \alpha_0>(x)$ et $<\alpha_0, \alpha_1>(x)$ (voir figure 4.4).

4.5.3 n-g-cartes

Récemment des besoins nouveaux tels que la modélisation de subdivisions d'espace de dimension supérieure à deux a amené l'étude de nouveaux modèles. P. Lienhardt a étendu les 2-cartes aux 3-cartes, définies en ajoutant une involution, permettant ainsi de représenter des objets 3D orientables. Il a également proposé, dans [Lie90], un modèle plus général : les cartes généralisées ou n-g-cartes (voir définition 4.8). Les cartes généralisées étant, comme leur nom l'indique, une généralisation des n-cartes permettant notamment d'introduire la notion d'orientabilité et donc de représenter des objets non orientables comme la bande de Moëbius (voir figure 4.5).

Définition 4.8 (n-g-carte) *Une* carte généralisée $G = (D, \alpha_0, \ldots, \alpha_n)$ *de dimension* n $(n \geqslant 0)$*, ou* n*-g-carte, est constituée d'un ensemble fini* D *d'éléments appelés brins, et d'involutions* $\alpha_0, \ldots, \alpha_n$ *dans* D *telles que* $\forall i, j, 0 \leqslant i \leq i+2 \leqslant j \leqslant n, \alpha_i \circ \alpha_j$ *est une involution.*

Exemple :

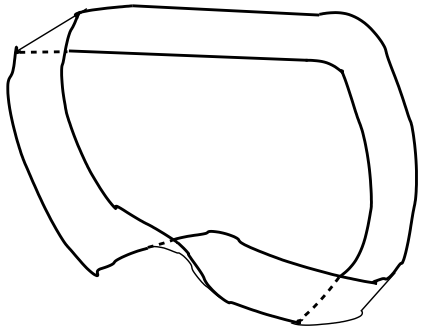
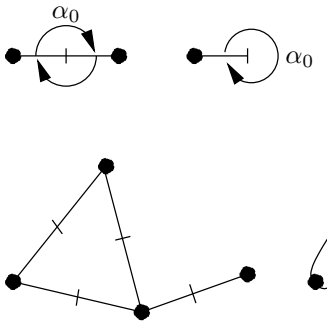
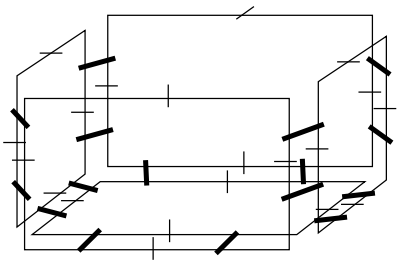


FIG. 4.5 – La bande de Mœbius : un objet non orientable.



Ci-contre une 0-g-carte contenant 3 brins.

Dans la 1-g-carte ci-contre α_0 est symbolisé par les traits fins sur les arêtes tandis que α_1 est représentée de manière implicite : deux brins sont en relation par α_1 s'ils sont incidents à un même sommet et si il se suivent dans le sens des aiguilles d'une montre autour de ce sommet.



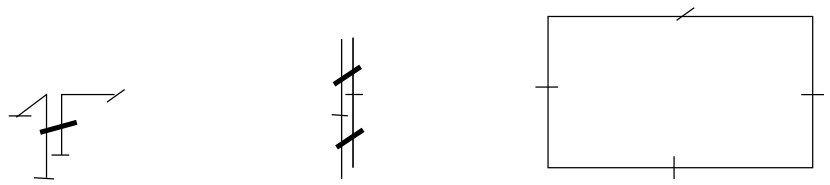
La figure ci-contre montre une 2-g-carte avec les mêmes conventions de représentation que précédemment. Les traits épais symbolisent l'involution α_2.

De la même manière que pour les 2-cartes, la notion d'orbite va nous permettre de définir les cellules d'une carte généralisée :

Définition 4.9 (cellule) *Soit une n-g-carte $G = (D, \alpha_0, \ldots, \alpha_n)$, la* cellule *de dimension k, ou $k-$cellule, incidente à un brin $x \in D$ de G est l'orbite :*

$$<\alpha_0, \ldots, \alpha_{k-1}, \alpha_{k+1}, \ldots, \alpha_n>(x)$$

La cellule simple *de dimension k, ou k-cellule simple, incidente à un brin x de G est $<\alpha_0, \ldots, \alpha_{k-1}>(x)$.*



(a) un sommet (b) une arête (c) une face

FIG. 4.6 – Sommet, arête et face dans une 2-g-cartes.

Les sommet, arête, face et volume incidents à un brin x dans G sont, respectivement, la 0-cellule, 1-cellule, 2-cellule et 3-cellule incidente à x. Un exemple pour une 2-g-carte est donné à la figure 4.6. On pourra comparer cette figure avec la Figure 4.4 présentant les même éléments mais pour une 2-carte.

Les n-cellules simples permettent de définir des faces simples, arêtes simples, ... Les bords de G sont définis par les points fixes de α_n. On dira alors que G est fermée si G n'a pas de points fixes par α_n.

Les cartes généralisée présentent plusieurs propriétés [BDFL92, Ber92], utiles pour la modélisation, mais également intéressantes dans le cadre d'une représentation pour l'analyse des images :

- de nombreuses propriétés topologiques peuvent être aisément calculées ;
- seulement deux opérations de base (une de consultation et une de construction) suffisent pour la manipulation de ces cartes (des opérations de plus haut niveau peuvent être déduites de celles-ci) ;
- structures de données immédiatement déduite de la définition même des cartes.

4.6 Graphe topologique des frontières

Nous avons donc choisi de nous appuyer sur les cartes combinatoires présentées à la section 4.5 pour la définition du graphe topologique des frontières. En effet nous y trouvons les avantages suivants :

- représentation topologique de l'image, s'intégrant directement dans le cadre

de notre approche interpixel ;
- lien direct vers une représentation à l'aide d'un graphe planaire ;
- expression algébrique de cette représentation à l'aide d'opérations simples à mettre en œuvre : permutations ou involutions ;
- représentation pouvant directement être utilisée dans le cadre d'une modélisation ultérieure des résultats obtenus. Ainsi dans le domaine de l'analyse d'images médicales, on travaille souvent avec des images qui sont des coupes, les résultats obtenus devant être finalement affichés en 3D.

Pourquoi définir une nouvelle représentation et pas seulement utiliser directement les cartes combinatoires ? Pour deux raisons :

1. Nous préférons qu'un sommet du graphe représente l'entité région plutôt que de symboliser des points particuliers du plan. De même les arêtes des cartes combinatoires représentent généralement des arêtes des polytopes à modéliser. Or nous voulons que les arêtes de notre graphe représentent les frontières entre les régions. Ce qui nous impose une première adaptation, qui pourrait être schématisée comme le passage au dual en termes de théorie des graphes avec éventuellement une réduction du nombre des arêtes.
2. À cause du problème récurrent des « trous ». En effet bien que les n-g-cartes permettent de calculer des caractéristiques topologiques des objets telles que le *genre*[7], le nombre de frontières et l'orientabilité des surfaces, la représentation adoptée ne correspond pas exactement à nos besoins. Ceci est lié à la première raison évoquée. Ainsi l'inclusion d'une face dans une autre (d'une région dans une autre) n'est pas directement codée dans la représentation. On peut pour palier à cela, tel que le propose M. Gangnet *et al.* [GHPT89], adjoindre à la carte, un arbre d'inclusion des contours. Nous préférons avoir une représentation unique.

On remarquera que l'utilisation de « ponts » dans la représentation par graphes duaux se rapproche de celle des cartes planaires et on pourrait très bien envisager une modélisation de cette représentation par les 2-g-cartes, α_2 permettant d'exprimer la présence d'un pont.

D'autre part, n'ayant pas besoin de représenter des objets non orientables, notre modèle s'appuiera plutôt sur celui des n-cartes, plus simple, que sur les

[7]Le nombre de trous.

cartes généralisées.

Ces considérations nous ont amenés à la définition du graphe topologique des frontières. C'est un multi-graphe planaire que nous modéliserons à l'aide d'une représentation dérivée des cartes combinatoires. Les sommets seront pour nous les régions et pour chacune d'elle, nous tiendrons à jour la liste des contours[8] de la région. Le premier de la liste étant toujours par convention le contour extérieur de la région, les autres, s'il en existe, correspondant aux contours intérieurs. Tous les contours d'une région s'exprimeront en terme de cycle d'une permutation sur les demi-arêtes incidentes à un même sommet de notre graphe topologique des frontières. Mais avant de passer à la définition formelle du graphe précisons les notions de *contours intérieurs*, *contour extérieur* et *frontière*.

4.6.1 définitions préalables

Par la suite le terme frontière sera toujours pris au sens de frontière entre deux régions. Il ne faut donc pas confondre avec la définition donnée à la section 3.3.4[9]. Il ne faut pas non plus faire la confusion avec la frontière d'une face d'un graphe planaire qui est d'ailleurs à rapprocher de la précédente. En fait, le terme *frontière* désignera une partie continue de la « frontière au sens topologique du terme » commune à deux régions adjacentes. On remarquera qu'il peut y avoir plusieurs frontières entre deux régions.

Définition 4.10 (frontière entre deux régions) *Soit $\mathcal{C}$ le complexe convexe définissant l'image. La* frontière $f_r(R_1, R_2)$ *entre deux régions R_1 et R_2 est un sous-complexe cellulaire connexe*[10] *de $\mathcal{C}$ contenant tous les éléments e tels que l'étoile ouverte $star(e)$ contienne au moins un élément de R_1 et au moins un élément de R_2, c'est à dire :*

$$f_r(R_1, R_2) = \{e \in C \,/\, star(e) \subseteq R_1 \cup R_2,\ star(e) \cap R_1 \neq \emptyset,\ star(e) \cap R_2 \neq \emptyset\}$$

Il est facile de voir que dans le cas des images 2D, une frontière est une $\mathcal{C}$-courbe, c'est à dire une séquence de lignels et pointels (voir définition 3.17), séparant deux

[8] La notion de contour sera précisée plus loin, (voir définition 4.12).

[9] C'est à dire : $\mathcal{F}r(R) = Bd(R) \bigcup Bd(X \setminus R)$ où R est une région et X est l'espace topologique de référence, par exemple $\mathbb{R}^n$.

[10] *Au sens de la définition 3.18, c'est à dire suite de cellules deux à deux incidentes.*

régions. En effet les régions sont des sous-complexes de dimension 2, de plus nous rappelons que pour nous les régions sont des ensembles disjoints ($R_1 \cap R_2 = \emptyset$) car issus d'une segmentation. L'étoile ouverte d'un pixel ne contenant qu'un seul élément, lui-même, il ne peut faire partie d'une frontière. Elle constituera donc un sous-complexe 1-dimension.

Définition 4.11 (jonction de frontières) *Soit $\mathcal{C}$ le complexe cellulaire convexe définissant l'image. Une* jonction *de frontières est un sous-complexe cellulaire connexe*[10] *$S \subset \mathcal{C}$ contenant tous les éléments e de $\mathcal{C}$ tels que l'étoile ouverte $star(e)$ contienne des éléments d'au moins 2 frontières.*

Dans le cas qui nous intéresse c'est à dire celui des images 2D, les jonctions de frontières sont des pointels. En effet les frontières étant des $\mathcal{C}$-courbes, les seules cellules dont l'étoile ouverte contient au moins 2 cellules 1-dimension sont les pointels. Nous pouvons désormais donner la définition 4.12 des contours d'une région :

Définition 4.12 (contours d'une région) *L'ensemble des contours, que l'on note $Co(R)$, d'une région R est donné par $\mathcal{F}r(R)$, c'est à dire la frontière de R au sens topologique du terme. Un contour d'une région sera alors défini comme une partie connexe*[10] *maximale de $Co(R)$.*

On notera ici une première différence avec la notion de contour pour les graphes planaires. En effet le terme de contour d'une région correspond en fait pour les graphes planaires à la définition des frontières d'une face. Nous ne pouvons utiliser la même définition car nous avons choisi de représenter les régions par les sommets et non pas les faces de notre graphe. D'autre part, une région pouvant contenir des « trous » nous sommes obligés de différencier **le** contour extérieur **des** contours intérieurs éventuels. Cette distinction n'existe pas en théorie des graphes car on ne se préoccupe pas de la topologie du graphe. Un graphe qu'on représenterait avec une composante connexe incluse dans une autre n'est pas différent si on le représente avec la même composante connexe « posée » à coté. Or dans le cas qui nous intéresse, c'est justement la topologie de l'image que l'on cherche à représenter.

Définition 4.13 (contour extérieur) *Le* contour extérieur *d'une région R est le sous-ensemble connexe $CoExt(R)$ contenant tous les éléments e de $Co(R)$ tels que la région R se trouve à sa droite si on le parcourt dans le sens négatif.*

Cette notion de contour extérieur correspond en fait à la définition du contour d'une face donnée à section 4.3.2 — le cycle contenant en son intérieur toutes les autres arêtes de la frontière de la face — Par contre la notion de contour intérieur n'existe pas dans la théorie des graphes, et ceci pour les raison déjà évoquées plus haut. Par opposition au contour extérieur, les contours intérieurs sont définis par :

Définition 4.14 (contour intérieur) *On appelle* contour intérieur *d'une région R, un sous-ensemble connexe*[10] *$CoInt(R)$ contenant tous les éléments e de $Co(R)$ tels que la région R se trouve à sa droite si on le parcourt dans le sens positif.*

4.6.2 définition et caractéristique

Comme nous l'avons déjà précisé, le graphe topologique des frontières se différencie du graphe d'adjacence par le fait que les arêtes ne représentent plus la relation d'adjacence mais l'existence d'une frontière entre les régions. Le graphe topologique des frontières est une représentation hybride issue des 2-cartes et du graphe d'adjacence. Il bénéficie ainsi des facilités offertes par la description algébrique des cartes combinatoires, de l'algorithmique de graphes, et enfin, reste une modélisation naturelle pour le domaine de l'analyse d'images.

Ainsi un graphe[11] topologique des frontières $G(V, D, \alpha, \sigma)$ où V désigne l'ensemble des sommets du graphe, D l'ensemble des demi-arêtes, σ une involution sur D et α une permutation sur D possède les caractéristiques suivantes :

1. Chaque région est représentée par un et un seul sommet.
2. Une demi-arête (brin) e est toujours incidente à un sommet R de V. On dit que e est incidente à R, c'est à dire une région, (voir représentation ci-contre). On utilisera la notation e^R si l'on veut préciser le sommet d'incidence de la demi-arête.

3. Une arête du graphe est un couple non ordonné $(e, \sigma(e))$ où $e \in D$. Soit E l'ensemble des arêtes, $(e_1^R, e_2^{R'}) \in E$ si et seulement si il existe une frontière entre R et R'. Une arête représente donc une frontière. On dira alors qu'une demi-arête symbolise la frontière "vue" du coté de la région dont elle est incidente.

[11] Par abus de langage on emploiera le terme de graphe à la place de multi-graphe.

4. α est une permutation sur l'ensemble D dont les cycles sont des permutations circulaires sur un ensemble de demi-arêtes incidentes à un même sommet. Chaque cycle correspond à un contour d'une région donnée et respecte l'ordre induit par la suite des frontières composant le contour.
5. $\infty \in D$ est un sommet particulier du graphe correspondant à l'extérieur de l'image. Celui-ci est donc vu comme une région englobant les régions de l'image.
6. À chaque sommet on associe la liste des cycles de α correspondant aux contours de la région correspondante à ce sommet. Le contour extérieur, s'il existe[12], sera toujours par convention le premier de la liste.

Cette définition nous donne des perspectives d'utilisation et d'application. Avant de les présenter donnons d'abord un exemple d'un graphe topologique des frontières.

4.6.3 exemple

Nous pouvons voir sur la Figure 4.7 un exemple d'image et du graphe topologique des frontières correspondant.

Cette figure utilise une représentation graphique du graphe (l'application α est symbolisée par les flèches). On peut également donner une représentation algébrique du graphe (voir Figure 4.8). Le graphe est alors décrit par les applications α et σ. Nous avons choisi de noter par des n-uplets les relations entre demi-arêtes en relation deux à deux. Ainsi pour σ le couple (e_1, e_2) signifie : $e_2 = \sigma(e_1)$ et $e_1 = \sigma(e_2)$. De même pour α le triplet (e_1, e_2, e_3) signifie : $e_2 = \alpha(e_1)$, $e_3 = \alpha(e_2)$ et $e_1 = \alpha(e_3)$. Il faut noter que les cycles de l'application α donnent la liste des frontières, composant un contour, ordonnée suivant l'ordre induit par le sens du contour. Pour que les notions de région et de contour extérieur et intérieur soient présentes dans cette représentation, les n-uplets de l'application α sont classés par sommet et le cycle correspondant au contour extérieur est toujours donné en premier.

On remarquera que la demi-arête issue de R_4 et appartenant à l'arête entre R_4 et R_5 n'est pas mise en relation par α avec les autres demi-arêtes issues de R_4.

[12] ∞ est l'unique sommet n'ayant pas de contour extérieur, mais un unique contour intérieur

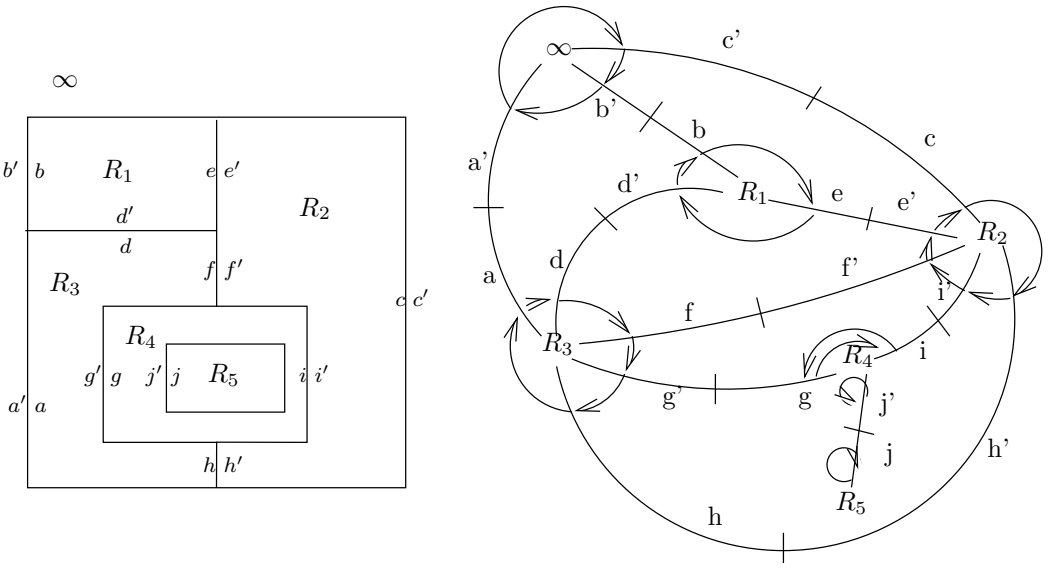


FIG. 4.7 – Un exemple de graphe topologique des frontières

σ : $(a,a')(b,b')(c,c')(d,d')(e,e')(f,f')(g,g')(h,h')(i,i')(j,j')$
α : ∞ : $()$; (a',c',b')
R_1 : (b,e,d')
R_2 : (c,h',i',f',e')
R_3 : (a,d,f,g',h)
R_4 : (i,g) ; (j')
R_5 : (j)

FIG. 4.8 – Représentation algébrique du graphe topologique des frontières de la Figure 4.7

En effet R_5 est "incluse" dans R_4, cette frontière appartient donc à un contour différent du contour extérieur de R_4. De plus il existe deux arêtes entre R_2 et R_3 puisqu'il y a deux frontières entre ces deux régions.

4.7 Algorithme d'extraction

4.7.1 schéma général

L'algorithme (voir Algorithme 1) consiste à parcourir l'image en construisant un graphe au fur et à mesure du balayage.

En supposant que chaque pixel on « connait » sa région, le balayage est effectué en déplaçant, de gauche à droite et de haut en bas, une fenêtre 2×2 sur l'image. La premiere fenêtre sera placée de telle sorte à ne couvrir que le premier pixel de l'image[13].Pour plus de détails sur cette méthode de balayage, le lecteur intéressé pourra se reporter à la thèse de P. Charnier [Cha95]. Nous nous intéressons à la configuration des frontières dans cette fenêtre. Il est facile de voir qu'il n'y a que 12 configurations possibles[14], nous les appelons *précodes*. En fonction du précode courant nous effectuons certaines opérations nous permettant de construire le graphe topologique des frontières. Il faut noter que dans toute la suite on fait l'hypothèse que pour chaque pixel on connaît sa région. Cette hypothèse est raisonnable puisque nous allons présenté au chapitre 5 deux algorithmes linéaires d'étiquetage des composantes connexes, selon le principe de l'union d'ensembles disjoints. Ceci nous permet notamment pour un précode donné de connaitre les régions présentes.

Une liste L (voir section 4.7.3 pour une description précise) nous permet de retrouver les arêtes du graphe correspondant aux frontières présentes dans le précode. Ces mêmes arêtes nous donnent les sommets correspondant aux régions de l'image visibles dans le précode. En fonction du précode, nous créons des sommets et des arêtes, nous maintenons à jour les relations α et σ entre demi-arêtes, fusionnons des arêtes et des sommets.

A la fin du balayage, le graphe obtenu est le graphe topologique des frontières (voir algorithme 1).

[13]On considère que l'image est entourée d'une région extérieure qui sera représentée dans le graphe par un sommet particulier.

[14]Les 4 configurations ne comportant qu'un seul lien interpixel ne sont pas valides en terme de frontière.

Algorithme 1: Extraction du *Graphe topologique des Frontières*

Données : une image
Résultat : le graphe topologique des frontières

1 Initialiser le graphe avec un sommet isolé ∞;
Initialiser D (ensemble des demi-arêtes) à vide;
Initialiser la liste L à vide;
pour chaque *précode de l'image* **faire**
    Exécuter le code associé au précode courant;
fin

4.7.2 configurations remarquables

Certaines configurations de frontières sont particulières. Elles sont présentes dans différents précodes. On notera tout particulièrement deux cas :

Définition 4.15 (Initiateur) *Un Initiateur est une configuration dans un précode où une frontière encadre le pixel en bas et à droite du précode.*

Définition 4.16 (Unificateur) *Un Unificateur est une configuration dans un précode où une frontière encadre le pixel en haut et à gauche du précode.*

4.7.3 la liste L

La liste L répertorie les frontières *actives*, c'est-à-dire connues mais non encore entièrement déterminées, ordonnées suivant l'ordre induit par un balayage gauche-droite de l'image. L'élément de la liste pointé par `Courant(`L`)` correspond à la frontière «attendue», c'est à dire à la frontière que l'on devrait rencontrer lors du prochain précode. La liste est remaniée à la rencontre des précodes Initiateur ou Unificateur.

Il faut noter qu'une même frontière peut-être présente plusieurs fois dans la liste. En effet elle sera enregistrée autant de fois que rencontrée lors du balayage. Chaque exemplaire d'une frontière est représenté dans la liste par une des demi-arêtes de l'arête correspondant à la frontière dans le graphe. La demi-arête n'est pas forcément la même pour chaque exemplaire et peut en outre changer lors de la progression de l'algorithme. Elle correspond au coté *actif* de la frontière. Par

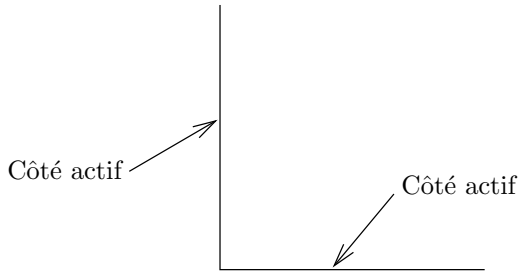


FIG. 4.9 – côté «actif» d'une frontière

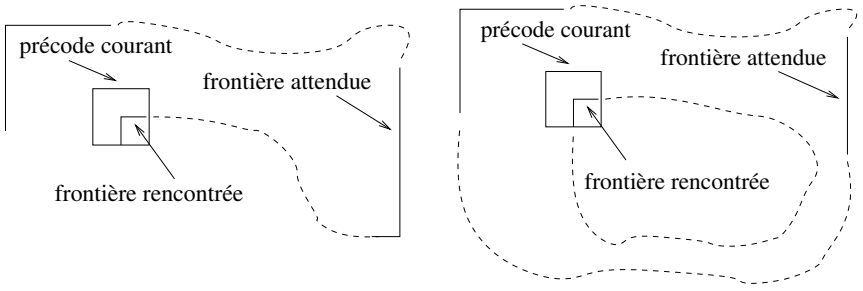
convention, le côté *actif* d'une frontière est celui situé à *gauche* pour une frontière *verticale* et en *haut* pour une frontière *horizontale*, comme le montre la figure 4.9.

4.7.4 l'application α

L'application α doit déterminer, pour chaque contour, une permutation circulaire sur les demi-arêtes, respectant l'ordre induit par la suite des frontières composant le contour. Cette détermination est possible dès le traitement du précode. En effet la définition des contours internes et externes (voir définitions 4.14 et 4.13) implique un sens de parcours du contour. Or les demi-arêtes symbolisent la frontière « vue du côté de la région ». Ainsi l'ensemble des demi-arêtes incidentes à un sommet symbolise les contours extérieurs et intérieurs de la région correspondante au sommet. La matière se trouvant à droite dans le sens de parcours correspondant au type de contour, l'application α doit respecter ce sens. Or en regardant un précode on peut déterminer ce sens et donc l'ordre que α doit induire sur les demi-arêtes correspondantes.

Ainsi si l'on prend l'exemple de la figure ci-contre l'ordre induit par α dans le traitement de ce précode (voir section 4.7.5.2) est indiqué par des flèches. On peut vérifier que la matière de la région est bien à droite des demi-arêtes correspondant aux frontières de cette même région.

Un problème se pose néanmoins : dans le cas d'un Initiateur, on ne peut pas savoir si la frontière rencontrée appartient à un contour différent ou non de la frontière attendue – c'est à dire pointée par `Courant(`L`)` – (voir Figure 4.10). Il faut néanmoins prendre une décision qui pourra être remise en cause. C'est

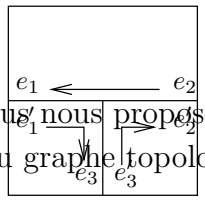


(a) même contour (b) contour différent

FIG. 4.10 – La frontière rencontrée appartient-elle au même contour que la frontière attendue ?

la liste L qui nous permet cela. Lors d'un précode Unificateur l'élément courant e de la liste L nous donne la frontière active, et le suivant e', celle qui devrait fermer le contour. Il suffit alors de vérifier si effectivement elles appartiennent au même cycle de α (c'est à dire au même contour). Pour cela il suffit de regarder si $e = \alpha(e')$. Sinon il faut faire l'union des deux contours, opération que nous dénommons `Unifier-Contour`.

4.7.5 Description détaillée de l'algorithme



Nous nous proposons de décrire de manière plus précise l'algorithme d'extraction du graphe topologique des frontières.

convention de nommage :

- soit e une demi-arête, on nomme alors e' la demi-arête telle que $e' = \sigma(e)$;
- La notation $\alpha(e_1) \leftarrow e_2$ signifie que désormais e_2 est l'image de e_1 par l'application α.

4.7.5.1 Les fonctions de l'algorithme

`créer(`e`,`R`)`

Créer une nouvelle demi-arête e incidente à R.

`Créer-Arête()`

On vient de détecter une nouvelle frontière entre deux régions. Il faut créer deux demi-arêtes et les relier par σ.

$(e, e')\leftarrow$ `Créer-Arête(`R, R'`)`
créer(e,R)
créer(e',R')
$e' = \sigma(e)$

`Créer-Sommet()`

Cette opération consiste à ajouter un nouveau sommet au graphe, comme son nom l'indique. Elle correspond toujours à un Initiateur. Ce nouveau sommet ne correspond pas forcément à la rencontre d'une nouvelle région de l'image. On vérifie donc que ce sommet n'a pas déjà été inséré dans le graphe. Si c'est le cas on ne crée pas de nouveau sommet, on se contente juste de le retourner.

$R\leftarrow$ `Créer-Sommet()`
si R *n'appartient pas déjà au graphe* **alors**
ajouter ce sommet R au graphe
le renvoyer comme résultat de la fonction
sinon
renvoyer R comme résultat

α`-Insérer(`e_1, e_2`)`

La notation choisie suppose qu'une des deux demi-arêtes nommées en paramètre est déjà en relation par α. Cette opération consiste alors à insérer l'autre demi-arête, venant d'être créée, **selon** α, c'est à dire en l'insérant dans la permutation circulaire en respectant l'ordre indiqué par le passage de paramètre.

α`-Lier(`e_1, e_2`)`

cette opération lie les deux demi-arêtes entre elles :

α-Lier(e_1, e_2)
$\alpha(e_1) \leftarrow e_2$
$\alpha(e_2) \leftarrow e_1$

L-Insérer$(e_1, ..., e_n)$

Les demi-arêtes passées en paramètre sont insérées dans cet ordre, dans la liste L avant Courant(L).

L-Retirer$(e_1, ..., e_n)$

Les demi-arêtes passées en paramètre sont retirées de la liste L une à une. Si une des demi-arêtes retirées est Courant(L), alors la suivante devient Courant(L).

L-Remplacer(e)

remplace dans la liste L l'élément Courant(L) par e :

L-Remplacer(e)
L-Insérer(e)
L-Retirer(*Courant(*L*)*)
Courant$(L) \leftarrow e$

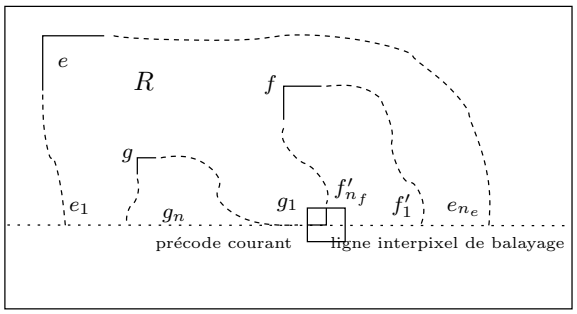
Unifier-Contour(e_1, e_2)

Dans le cas d'un Unificateur, il faut réunir les deux frontières en une seule. Mais les frontières e_1 et e_2 peuvent ne pas être issues du même contour[15]. En effet on a pu développer le contour auquel appartient e_1 comme un contour indépendant. Il faut alors réunir les deux contours. C'est ce que réalise Unifier-Contour(e_1, e_2) en insérant le cycle contenant e_2 avant e_1 dans le cycle contenant e_1 (voir exemples à la figure 4.11) :

Unifier-Contour(e_1, e_2)
$\alpha(\alpha^{-1}(e_1)) \leftarrow \alpha(e_2)$
$\alpha(e_2) \leftarrow e_1$
$\alpha(\alpha^{-1}(e'_2)) \leftarrow \alpha(e'_1)$
$\alpha(e'_1) \leftarrow e'_2$

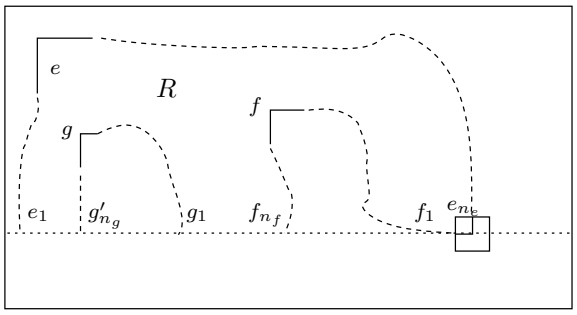
La notation $\alpha^{-1}(e_i)$ représente la demi-arête précédent e_i dans la permutation α. D'autre part, ici, la notation $\alpha(\alpha^{-1}(e_i)) \leftarrow \alpha(e_j)$ ne signifie pas suivant du précédent, c'est à dire lui-même, mais : « le précédent a désormais comme suivant $\alpha(e_j)$ » (voir la convention adoptée au début de la section 4.7.5).

[15] ceci pourra être vérifié par le test $\alpha(e_2) \neq e_1$



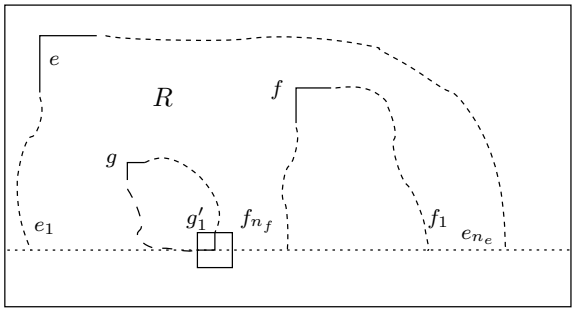
avant :
$R : (e_1 \dots e_{n_e})\ (g_1 \dots g_{n_g})(f_1 \dots f_{n_f})$ et
$L : e'_1 \leftarrow g_{n_g} \leftarrow g'_1 \leftarrow f_{n_f} \leftarrow f'_1 \leftarrow e_{n_e}$

après :
$R : (e_1 \dots e_{n_e})\ (g_1 \dots g_{n_g}\ f_1 \dots f_{n_f})$ et
$L : e'_1 \leftarrow g_n \leftarrow f'_1 \leftarrow e_{n_e}$ soit
$R : (e_1 \dots e_{n_e})\ (f_1 \dots f_{n_f}\ g_1 \dots g_{n_g})$



avant :
$R : (e_1 \dots e_{n_e})\ (g_1 \dots g_{n_g})\ (f_1 \dots f_{n_f})$ et
$L : e'_1 \leftarrow g_{n_g} \leftarrow g'_1 \leftarrow f_{n_f} \leftarrow f'_1 \leftarrow e_{n_e}$

après :
$R : (e_1 \dots e_{n_e} f_1 \dots f_{n_f})\ (g_1 \dots g_{n_g})$ et
$L : e'_1 \leftarrow g_{n_g} \leftarrow g'_1 \leftarrow f_{n_f}$



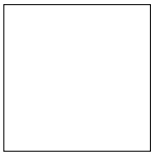
avant :
$R : (e_1 \dots e_{n_e})\ (g_1 \dots g_{n_g})\ (f_1 \dots f_{n_f})$ et
$L : e'_1 \leftarrow g_{n_g} \leftarrow g'_1 \leftarrow f_{n_f} \leftarrow f'_1 \leftarrow e_{n_e}$

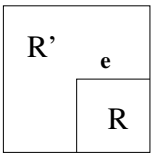
après :
$R : (e_1 \dots e_{n_e})\ (f_1 \dots f_{n_f})\ (g_1 \dots g_{n_g})$ et
$L : e'_1 \leftarrow f_{n_f} \leftarrow f'_1 \leftarrow e_{n_e}$

FIG. 4.11 – L'effet d'`Unifier-Contour` pour les contours de R.

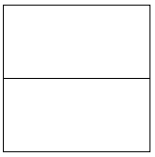
4.7.5.2 Le traitement des précodes

Nou présentons maintenant les traitements à effectuer en fonction de chaque précode. La notation $\equiv$ indique un fait sans entraîner de traitement quelconque.

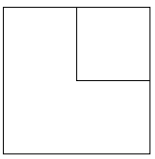
(0)  $\emptyset$

(1) 

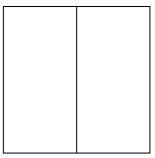
$R \leftarrow$ `Créer-Sommet()`;
$(e, e') \leftarrow$ `Créer-Arête(`R', R`)`;
α`-Lier(`e, e`)`;
α`-Lier(`e', e'`)`;
`L-Insérer(`e,e`)`;
si *premier précode* **alors** `Courant(`L`)` $\leftarrow e$;
sinon `Courant(`L`)` $\leftarrow$ `Prédécesseur(`*Courant(*L*)*`)`;

(2) 

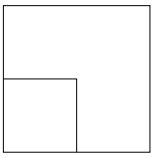
$\emptyset$

(3) 

`L-Remplacer(`σ*(Courant(*L*)*`)`;

(4) 

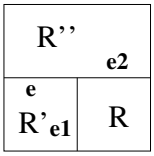
`Courant(`L`)` $\leftarrow$ `Suivant(`*Courant(*L*)*`)`;

(5) 

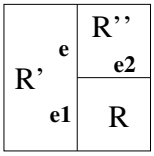
`L-Remplacer(`σ*(Courant(*L*))*`)`;
`Courant(`L`)` $\leftarrow$ `Suivant(`*Courant(*L*)*`)`;

Le traitement des précodes (6) et (7) sont indentiques, la seule différence réside dans la demi-arête e.

(6)



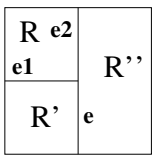
(7)



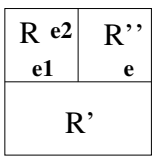
(6) :$e \equiv \sigma($`Courant(`L`)`$)$;
(7) :$e \equiv$`Courant(`L`)`;
$R \leftarrow$ `Créer-Sommet()`;
$(e_1, e'_1) \leftarrow$ `Créer-Arête(`R', R`)`;
$(e_2, e'_2) \leftarrow$ `Créer-Arête(`R'', R`)`;
α-`Lier(`e'_2, e'_1`)`;
α-`Insérer(`e, e_1`)`;
α-`Insérer(`e_2, e'`)`;
`L-Insérer(`e_1, e_2`)`;
`L-Retirer(`*`Courant(`*L`)`*`)`;
`Courant(`L`)`$\leftarrow e_2$;

Ici les deux traitement sont presque identiques, seule la dernière opération diffère.

(8)

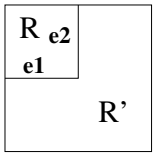


(9)



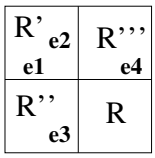
$e_1 \equiv$ `Courant(`L`)`;
$e_2 \equiv$ `Suivant(`*`Courant(`*L`)`*`)`;
si $\alpha(e_2) \neq e_1$ **alors** `Unifier-Contour(`e_1, e_2`)`;
`L-Retirer(`e_1, e_2`)`;
$(e, e') \leftarrow$ `Créer-Arête(`R'', R'`)`;
α-`Insérer(`e'_1, e'`)`;
α-`Insérer(`e, e'_2`)`;
(8) : `L-Insérer(`e'`)`;
(9) : `L-Insérer(`e`)` ; `Courant(`L`)` $\leftarrow e$;

(10)



$e_1 \equiv$Courant(L);
$e_2 \equiv$Suivant(*Courant*(L));
si $\alpha(e_2) \neq e_1$ **alors** Unifier-Contour(e_1, e_2);
L-Retirer(e_1, e_2);
si $e_1 \neq e_2$ **alors**
  $\alpha\left(\alpha^{-1}(e_2)\right) \leftarrow e_1$;
  $\alpha\left(\alpha^{-1}(e'_1)\right) \leftarrow e'_2$;
  $\sigma(e_1) \leftarrow e'_2$;
  Supprimer(e'_1, e_2);

(11)



$e_1 \equiv$Courant(L);
$e_2 \equiv$Suivant(*Courant*(L));
si $\alpha(e_2) \neq e_1$ **alors** Unifier-Contour(e_1, e_2);
L-Retirer(e_1, e_2);
$R \leftarrow$ Créer-Sommet();
$(e_3, e'_3) \leftarrow$ Créer-Arête(R'', R');
$(e_4, e'_4) \leftarrow$ Créer-Arête(R''', R);
α-Lier(e'_3, e'_4);
α-Insérer(e'_1, e_3);
α-Insérer(e_4, e'_2);
L-Insérer(e_3, e_4);
Courant$(L) \leftarrow e_4$;

4.8 Preuve et complexité de l'algorithme

4.8.1 validité de l'algorithme d'extraction

Nous allons donner dans cette section les idées principales de la preuve. Certaines parties plus faciles à vérifier, ne seront pas détaillées afin de ne pas surcharger le texte. Seuls seront détaillés les passages délicats. Pour prouver la validité de notre algorithme, il faut montrer :

1. qu'il se termine ;
2. qu'on obtient bien un graphe topologique des frontières.

Le premier point est immédiatement vérifié puisque l'algorithme parcourt l'image ligne à ligne, de la première à la dernière. Donc si l'image est finie et que le nombre d'opérations par précode est lui aussi fini, il se termine. Reste à vérifier le deuxième point. Pour cela il faut montrer que les 7 caractéristiques, énoncées à la section 4.6.2, du graphe topologique des frontières vérifiées.

1. « chaque région est représentée par un et un seul sommet » est vérifié. En effet, remarquons d'abord que la seule fonction créant des sommets est `Créer-Sommet()` ; de plus elle n'est appelée que dans un précode Initiateur. Or chaque région a au moins un pixel correspondant à un précode Initiateur, le pixel le plus en haut et le plus à gauche. Puisque c'est la première fois que l'on rencontre cette région on n'a donc pas pu rencontrer de précode Initiateur pour cette région et donc faire appel à `Créer-Sommet()`, le sommet correspondant n'existe donc pas encore dans le graphe . Comme on fait alors appel à `Créer-Sommet()`, le sommet est créé. Chaque région est donc représentée par au moins un sommet.

   Notons maintenant qu'il n'y a pas de fonction effaçant un sommet déjà créé, il faut donc montrer que l'on créé au plus un sommet par région. C'est bien le cas puisque avant d'en créer un, la fonction `Créer-Sommet` vérifie qu'il n'est pas déjà dans le graphe.

   On a donc un et un seul sommet dans le graphe par région dans l'image.
2. « une demi-arête e est toujours incidente à un sommet R ». Les demi-arêtes sont créées de cette manière (voir fonction `créer(`e`,`R`)`. Cette propriété est donc vérifiée par construction.
3. « une et une seule arête par frontière ». Ce point sera détaillé ci-dessous.
4. « σ met en relation les demi-arêtes issues d'une même arête » est vérifié car la seule fonction créant les arêtes est `Créer-Arête`, et elle met bien en relation par σ les deux demi-arêtes formant l'arête créée. De plus rien ne vient modifier $\sigma(e)$ pour toute demi-arête.
5. « α respecte l'ordre des frontières déterminant chaque contour ». Ce point sera également détaillé ci-dessous.

6. « ∞ est un sommet particulier du graphe » est vérifié puisque ce sommet est créé à l'initialisation (ligne 1 de l'algorithme 1).
7. « à chaque sommet est associée la liste de contours ». Cela est réalisé de manière implicite puisque pour chaque sommet on a la liste des arêtes incidentes, on peut alors retrouver chaque contour grâce à l'application α. La première arête créée sera toujours une arête du contour extérieur, donc il suffit à partir de cette arête de chercher le cycle selon α la contenant, et on a le contour extérieur. Il suffit alors de prendre une arête n'appartenant pas au contour extérieur (s'il y en a) pour trouver un contour intérieur, et ainsi de suite.

   On pourrait garder la liste des contours de chaque région de manière explicite, mais l'opération `Unifier-Contour` d'unification des contours ne pourrait plus se faire en temps constant, et l'on perdrait la complexité linéaire de l'algorithme (voir l'étude de la complexité à la section 4.8.2).

Les points 3 et 5 nécessitent un étude plus détaillée. Nous avons entre autre besoin de la notion d'image courante :

Définition 4.17 (image courante) *On appelle image courante la partie de l'image déjà examinée par l'algorithme (voir figure 4.12), on considère également que l'espace topologique contenant l'image (un sous-espace de $\mathbb{R}^n$) s'arrête au bord bas de cette image. Ainsi le bord bas de l'image (au sens littéral du terme) fait partie de l'intérieur de l'image ; il n'y a donc pas de frontière à cet endroit.*

Montrons maintenant qu'au cours de l'algorithme les invariants suivants sont respectés :

(4.8.1) Il y a une arête par frontière de l'image courante

(4.8.2) L'application α détermine pour chaque contour de l'image courante une permutation circulaire sur des demi-arêtes incidentes au sommet (correspondant à la région du contour) et respecte l'ordre induit par la suite des frontières composant le contour.

(4.8.1) et (4.8.2) sont vérifiés à l'initialisation. En effet il existe alors une seule région, l'extérieur, et on a bien un seul sommet : ∞. Il n'existe encore aucune frontière et l'ensemble D est initialisé à vide.

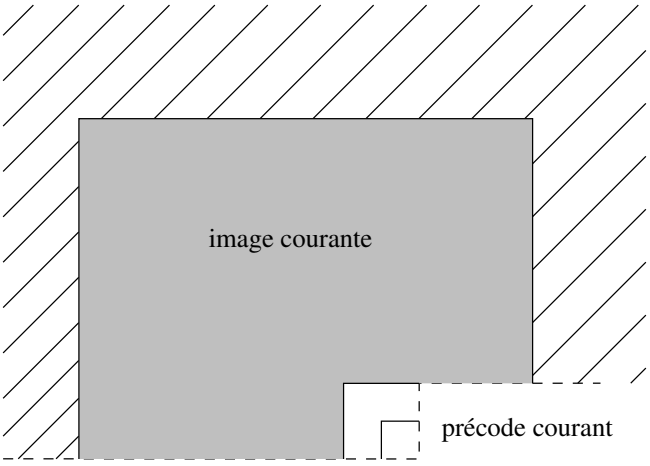


FIG. 4.12 – Image courante

Supposons qu'ils soient vérifiés à une étape de l'algorithme, c'est à dire pour une image courante donnée.

Montrons qu'alors à l'étape suivante ils seront toujours vrais. Pour cela il suffit de vérifier qu'après l'exécution du code associé au précode courant (voir le détail du code de chaque précode à la section 4.7.5.2), les invariants sont toujours valides dans la nouvelle image courante. Nous allons examiner les différents précodes et regarder si les invariants sont bien conservés.

Nous nous servirons des propriétés de la liste L, c'est à dire :

(4.8.3) L est la liste des frontières incomplètes de l'image courante (au sens ou on ne connait pas encore les extrémités des frontières) ordonnée suivant l'ordre de rencontre sur la ligne de balayage. Il y a un exemplaire par extrémité non connue.

(4.8.4) Les élément de L sont les demi-arêtes correspondant aux côtés actifs (voir définition à la section 4.7.3) des frontières.

(4.8.5) `Courant(`L`)` est le coté de la prochaine frontière qu'on s'attend à rencontrer lors du prochain précode.

Le lecteur pourra facilement vérifier que les traitements associés aux différents précodes permettent de conserver ces propriétés. De même on peut aisément vérifier que les invariants (4.8.1) et (4.8.2) sont conservés pour tous les traitements concernant la création d'arête lors de la découverte d'une nouvelle frontière. Les difficultés sont dues aux remises en questions de la structure déjà établie. Cette remise en question intervient lorsque l'on se rend compte que l'on avait créé deux arêtes pour une seule frontière. Les différents cas possibles sont présentés à la figure 4.11 donnée pour la description des opérations réalisées lors de `Unifier-Contour`. On peut voir sur cette figure que les propriétés de α sont bien conservées. Deux problèmes se posent néanmoins :

1. Quand doit-on faire appel à `Unifier-Contour` ?

2. N'a-t-on pas plus d'une arête pour une seule frontière.

L'appel à `Unifier-Contour` ne se fait que si $\alpha(e_2) \neq e_1$. En effet on ne doit pas faire appel à `Unifier-Contour` si les deux frontières correspondant aux deux demi-arêtes en question appartiennent en fait au même contour. Comme l'invariant (4.8.2) était vérifié avant ce précode, l'arête suivant e_2 dans le sens de parcours est forcément la suivante par l'application α. Donc si $\alpha(e_2) = e_1$ ces deux arêtes appartiennent au même contour. On fait alors appel à `Unifier-Contour`, c'est à dire lors du traitement des précodes (8) à (11).

Il pourrait sembler que dans le cas du précode (10) on ait deux arêtes pour la même frontière. En effet il n'y a pas de point de jonction de frontières dans ce précode. On a donc en fait une seule frontière et pour l'instant deux arêtes développées. Ces deux arêtes e_1 et e_2 peuvent-elle déja être une seule et même arête ? Oui si on vient en fait de développer un contour ne contenant qu'une frontière, c'est à dire un « trou » simple composé d'une seule région. Sinon il faut en supprimer une et remettre à jour l'application α. C'est ce qui est réalisé en fin de traitement de ce précode après le test, bien nécessaire, $e_1 \neq e_2$.

Avec tous ces éléments il est aisé de voir que les invariants sont bien conservés tout au long de l'algorithme. De ces invariants on en déduit immédiatement les propriétés 3 et 5 pour l'image finale. On a donc bien obtenu un graphe topologique des frontières avec les caractéristiques énoncées. □

4.8.2 étude de la complexité

Nous donnons ici les éléments permettant de vérifier que cet algorithme s'exécute en temps linéaire sous certaines conditions.

On représentera l'application α à l'aide de listes doublement chaînées, chacune représentant un cycle de la permutation, et donc un contour. Les cycles eux-mêmes pourront être rangés en listes rattachées à chaque sommet dont elles sont un contour. Toutes les opérations relatives à α concernent un élément particulier que l'on connait, soit parce qu'on vient de le créer, soit parce qu'on l'a sortie de la liste L. On ne fait alors qu'insérer avant ou après un autre élément. On peut également modifier son image ou dans le cas du précode (10) le supprimer, c'est à dire, entre autres, le retirer du cycle à laquelle il appartient. Comme un élément donné appartient à un et un seul cycle, que ces cycles sont gérés en listes, on peut considérer que toutes les opérations relatives à α s'exécutent en temps constant.

L'application σ ne pose aucun problème, on peut à chaque élément associer directement, par exemple à l'aide d'un pointeur, son image. De même que précédemment on peut alors considérer que toutes les opérations relatives à σ se déroulent en temps constant.

La liste L est une liste doublement chaînée et on ne s'intéresse qu'à l'élément courant, à son suivant et, éventuellement, à son prédécesseur. Ces opérations là s'exécutent également en temps constant.

À part les traitements relatifs aux sommets du graphe, tous les autres, ainsi que les différentes fonctions de l'algorithme, sont basés sur la liste L et les applications α et σ. Si les traitements relatifs aux sommets du graphe sont aussi en temps constant alors on pourra considérer que le traitement de n'importe quel précode s'exécute en temps constant.

À la création d'un sommet il faut vérifier que ce sommet n'existe pas déjà dans le graphe. D'après l'hypothèse que nous avons faite (étiquetage des composantes connexes du graphe), on a la liste de tous les sommets avant le début de l'algorithme. Il suffit donc de ranger ces sommets dans un tableau et de les marquer lorsqu'on les ajoute dans le graphe par l'opération `Créer-Sommet`.

La seule opération que nous n'avons pas encore examiné c'est l'extraction du précode lui-même. S'il est clair qu'à l'aide de quatre tests on peut déterminer la configuration des frontières, encore faut-il pour cela être capable de trouver la

région à laquelle appartient le pixel. Pour cela il suffit de reprendre la structure en arbre utilisé dans `ScanLine` par exemple et d'appliquer préalablement la fonction `MiseÀplat` sur toute l'image. Il est clair que globalement cette opération s'effectue en O(()n). Les pixels ayant alors un accès direct à leur région, l'extraction du précode se fera en temps constant. Dans ce cas notre algorithme s'exécute bien en O(n).

4.9 Perspectives

Dans un premier temps il faudrait étudier comment implanter différents opérations de modification du graphe à l'aide des applications σ et α. En fait certaines l'ont déjà été dans le cadre de l'étude de l'algorithme d'extraction (voir section 4.7). Ensuite pour pouvoir exploiter pleinement les possibilités offertes par une telle structure nous pensons la compléter par un codage interpixel des frontières. En effet ce codage nous permettra de connaître précisément la topologie des frontières. Les précodes nous donnent déjà avec précision, comment, et dans quel sens, compléter le codage de la demi-arête. Malheureusement on a alors deux codages[16] pour une même frontière (arête). Si l'on veut un codage unique, il faut alors « choisir » un côté (une demi-arête). Il semble qu'alors, lors de la fusion de deux frontières, il soit parfois nécessaire de « retourner » un des deux codes déjà établi d'une des deux frontières à unifier. Ce même genre de problème a déjà été résolu par P. Charnier dans sa thèse [Cha95]. Sa solution devrait pouvoir être adaptée à notre algorithme.

Notre structure nous permet déjà de faire la différence entre deux régions voisines (ayant une frontière de leur contour extérieur en commun) et une région incluse dans une autre, mais de manière implicite. Il serait bon de le faire de manière explicite, en rajoutant, par exemple, une involution supplémentaire mettant en relation une demi-arête du contour extérieur d'une région avec une demi-arête d'un contour intérieur d'une même région. Cette involution faciliterait ainsi les opérations de parcours sur le graphe.

Une évolution envisageable serait de ne plus se contenter de modéliser l'existence d'une frontière entre deux régions, mais plutôt de représenter la relation de stricte adjacence. Cela reviendrait à modifier la définition de frontière pour y

[16] Ces deux codages correspondent en fait aux codages vus de chaque côté de la frontière.

inclure les points de jonction, c'est à dire leurs extrémités. On perdrait alors la propritété de planarité du graphe. D'autre part un point de jonction peut mettre en relation jusqu'à quatre régions (dans le cas 2D) alors qu'une frontière met en relation actuellement seulement deux régions. La notion de graphe ou multi-graphe n'est alors plus suffisante et il faut envisager l'utilisation d'hyper-graphes (voir définition dans [Ber70] par exemple). Une telle structure serait néanmoins intéressante car elle devrait nous permettre de faire de la segmentation en régions de manière plus fine et plus précise, et une prise en compte totale de la topologie-étoile.

Une autre étude à envisager est l'extension de ce graphe aux images 3D. En effet le fait de nous être appuyé sur les cartes combinatoires nous permet d'envisager une bonne faisabilité. D'autant que nous avons déjà mis au point avec A. Seranno un algorithme d'extraction des surfaces régulières d'une image 3D, algorithme basé sur le même principe d'analyse de précodes [Ser95].

Quant aux applications, outre la possibilité de segmentation à l'aide du graphe et de l'algorithme de fusion de régions tels que ceux développés par P. Charnier [Cha95], on devrait pouvoir faire du chaînage paramétrable de contours (par paramétrage nous entendons : favoriser certains types de structures dans l'image, comme des cycles minimaux ou des chaînes maximales). En effet il suffirait de spécifier des algorithmes de parcours particuliers du graphe, entre autres grâce aux opérations α et σ. Nous pensons également que des traitements tels que la polygonisation pourraient être implantés facilement, mais ceci implique que le codage des frontières soit effectif.

Chapitre 5

Méthodes combinatoires en analyse d'images

Historiquement, cette étude se situe aux début de mes travaux. Le but recherché était double :

- montrer l'efficacité de méthodes combinatoires pour la segmentation d'images,
- trouver un algorithme efficace et rapide de pré-filtrage de l'image afin d'accélérer grandement le processus de segmentation mis au point par P. Charnier [Cha95]. Ce dernier est fortement pénalisé par la multitude de petites régions présentes dans l'image d'origine.

Finalement nous avons obtenu deux algorithmes [FG96], basés sur la même structure, réalisant une segmentation en régions tout en effectuant un étiquetage des composantes connexes de l'image. On notera également que ces algorithmes permettent une coopération régions-contours lors de la segmentation. Les méthodes de coopérations régions-contours ne sont pas nouvelles [BW91, KP91], mais généralement elles se contentent de faire une synthèse entre plusieurs traitements déjà réalisés (voir par exemple [VdGV91]. La nôtre réalise une segmentation en région en tenant compte de critère globaux (sur les régions) aussi bien que locaux, c'est à dire liés aux pixels. De plus elles n'effectuent pas un étiquetage des composantes connexes, et ne se soucient pas de la complexité des algorithmes.

5.1 Introduction

Dans ce chapitre nous rappelons brièvement en quoi consiste la segmentation en régions et l'étiquetage des composantes connexes. Nous décrivons également le problème de l'union d'ensembles disjoints, plus communément appelé *« Union-Find »* , sur lequel s'appuient nos deux algorithmes.

5.1.1 la segmentation en régions

La segmentation d'une image pourrait se résumer en l'extraction de ses caractéristiques essentielles. Comme on l'a vu dans le chapitre 6, on peut extraire les points de contour de l'image. Malheureusement les techniques de filtrage, existantes actuellement, ne permettent pas en elles-mêmes de déterminer des contours fermés. Or un objet est souvent décrit par ses faces, ce qui implique des contours fermés. Pour cela on cherche à définir les régions significatives de l'image. Elles déterminent alors des contours fermés. On parle de segmentation en régions (voir définition 3.28). C'est un aspect du traitement d'images qui a été, et qui est toujours beaucoup étudié (voir par exemple les états de l'art de S.W. Zucker [Zuc76] ou R.M. Haralick et L. Shapiro [HS85], ou des travaux plus récents tels que ceux de A. Veijanen, R. Adams et L. Bischof, ou encore P. Charnier [Vei94, AB94, Cha95] . La figure 5.1 présente deux exemples de segmentation en régions.

Les techniques usuelles en segmentation sont basées sur les approches par *fusion*, *division* ou même *division-fusion*. Ces deux manières de procéder sont duales. L'approche « division » consiste à diviser une région tant qu'elle ne correspond pas au critère d'homogénéïté choisi. La segmentation par « fusion » consiste à démarrer avec des petites régions (généralement les pixels eux-mêmes) puis à fusionner entre elles les régions adjacentes si leur union respecte le critère d'homogénéïté. L'approche « division-fusion » commence par une étape de division et pour pallier l'arbitraire de la méthode de division choisie (en quatre par exemple), on procède ensuite à une étape de fusion. Ces manières de procéder ne sont bien sûr pas les seules (voir par exemple [PCCB91].

Nous avons choisi d'implanter un algorithme de segmentation en régions par fusion (décrit pour la première fois par J.L. Muerle [MA68]). Nous commençons donc avec une image constituée d'autant de régions que de pixels, puis nous fu-

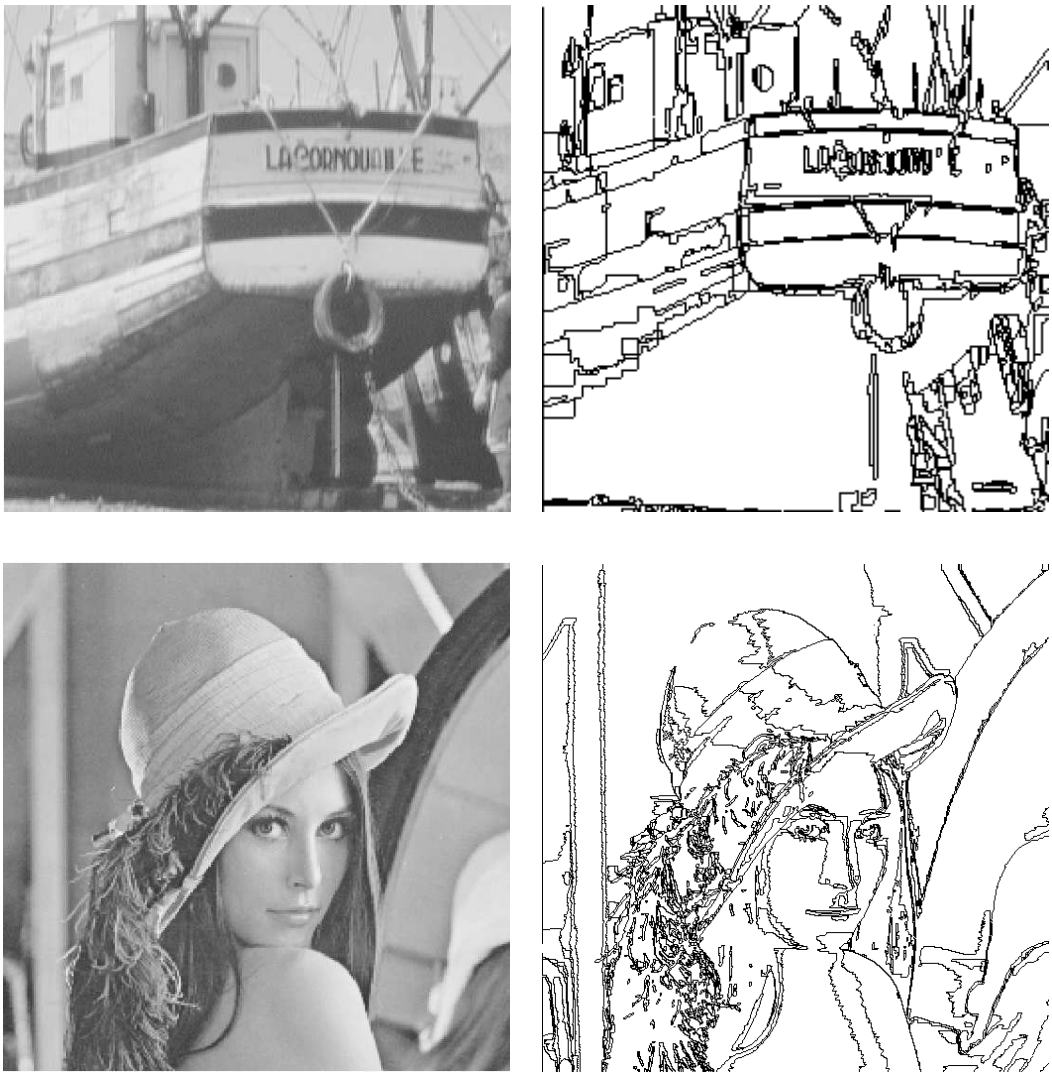


FIG. 5.1 – Exemples de segmentation : à gauche l'image originale, à droite les contours des régions obtenues après segmentation en régions par les algos `MergeSquare` et `ScanLine`.

sionnons ces régions deux à deux tant qu'elles ne sont pas optimales au sens du critère d'homogénéïté. Pour décider d'un tel regroupement nous choisissons une paire de pixels adjacents et regardons si les régions auxquelles ils appartiennent doivent être fusionnées. Le résultat et la complexité d'un tel algorithme dépendent énormément de l'ordre dans lequel les regroupements sont effectués. Nous pensons donc qu'une segmentation par croissance de régions devrait satisfaire les critères suivants :

(5.1.1) Chaque paire de pixels voisins doit être considérée au moins une fois, mais au plus un petit nombre de fois.

(5.1.2) La taille des différentes régions devrait être équilibrée au cours de l'algorithme.

Le critère (5.1.1) garantit que le test de regroupement sera effectué un nombre de fois proportionnel au nombre de pixels, cette condition étant nécessaire pour avoir un algorithme linéaire. De plus elle nous assure qu'aucun bord artificiellement créé ne subsiste entre les régions.

Le critère (5.1.2) est souhaitable afin que certaines régions ne prennent pas une trop grande importance avant que d'autres n'aient eu le temps de se former. Ce critère exclut les techniques simples de parcours de graphe comme le parcours en profondeur.

Ces critères étant fixés, il reste à déterminer une structure de données. La plus courante est le graphe d'adjacence qui associe à chaque sommet une région et où les arêtes symbolisent la relation d'adjacence. Malheureusement sur le plan pratique cette structure de données est relativement complexe à maintenir et de plus mène à des algorithmes non linéaires tels que présentés dans [Mon87]. En fait tel que l'a déjà fait remarquer M. Dillencourt et al. [DST92] la segmentation par croissance de régions nous ramène au problème de l'*union d'ensembles disjoints*. Nous avons nous aussi choisi d'examiner le problème sous cet angle mais contrairement à M. Dillencourt et al., nous ne nous restreignons pas aux images binaires. Nous proposons de plus deux algorithmes, dont l'un des deux peut être aisément parallélisé. Mais dans un premier temps présentons ce problème.

5.1.2 Le principe de l'Union-Find

5.1.2.1 définition du problème

Le problème de l'union d'ensembles disjoints consiste essentiellement à regrouper n éléments distincts dans une collection d'ensembles disjoints. Ici l'analogie avec notre problème de départ devient évidente : les éléments à regrouper sont les pixels et les ensembles disjoints, les régions. Pour réaliser cela deux opérations sont importantes :

- `Union` qui doit réaliser l'union effective de deux ensembles. Au départ cela consistera à regrouper deux pixels, puis ensuite des ensembles de pixels. Généralement chaque ensemble est identifié par un *représentant* qui est un certain membre de l'ensemble. Ce représentant sera par exemple le premier de la liste si l'ensemble est représenté par une liste chaînée, ou la racine d'un arbre s'il est représenté par une telle structure. Dans le cas d'un arbre (la structure généralement utilisée car donnant les meilleures performances) l'union de deux ensembles consistera alors à lier l'une des racines à l'autre (voir figure 5.2).
- `Chercher` (« find » en anglais) qui, étant donné un élément quelconque, permet de trouver le représentant de l'ensemble auquel appartient cet élément. Cette opération est primordiale puisqu'elle permet également de vérifier si deux éléments appartiennent au même ensemble.

5.1.2.2 complexité des algorithmes existants

Il n'existe pas à l'heure actuelle d'algorithme linéaire résolvant ce problème dans le cas général. La meilleure complexité connue est due à un algorithme de R.E. Tarjan [Tar75], elle est en $O(m.\alpha(m,n))$ où $\alpha(m,n)$ est l'inverse de la fonction d'Ackerman (voir définition à l'annexe C) et n, m, avec $n < m$, sont respectivement le nombre d'appels aux fonctions `Union` et `Chercher`. Il a de plus été prouvé (voir [Tar79, TvL84]) qu'une telle complexité était optimale pour les opérations sur des structures de données satisfaisant certaines conditions techniques. H.N. Gabow et R.E. Tarjan ont présenté un algorithme linéaire suivant certaines restrictions [GT84]. Dans le cas particulier des images binaires, M. Dillencourt et al. [DST92] ont également élaboré un algorithme linéaire pour l'« Union-Find ». C'est aussi

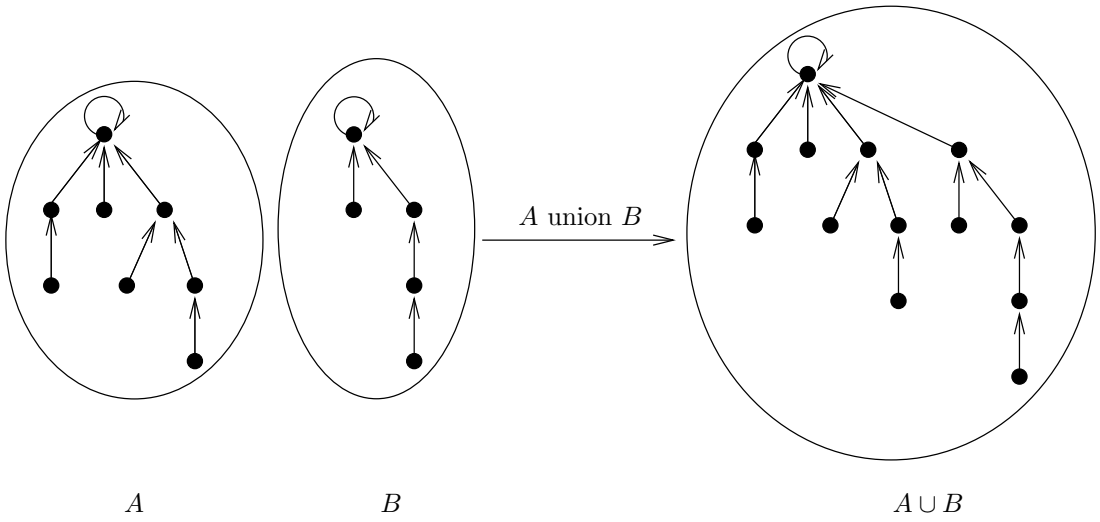


FIG. 5.2 – Union de deux ensembles représentés par des arbres

dans un cadre particulier, celui des images en niveaux de gris que J. Gustedt et moi-même avons présenté deux algorithmes linéaires pour ce problème [FG96].

Pour une présentation plus générale le lecteur pourra se reporter par exemple au livre de K. Mehlorn [Meh84], à l'état de l'art de Z. Galil et G.F. Italiano [GI91], ou l'article de M.J. Van Kreveld et M.H. Overmars [vKO93], pour des résultats plus récents.

5.1.2.3 méthodes habituelles pour la résolution du problème

Les implantations efficaces utilisent des arbres pour représenter les ensembles, la racine de chaque arbre étant l'élément représentant de l'ensemble correspondant. Pour réaliser l'opération `Union`, il suffit alors de lier la racine de l'un des deux ensembles en cause à l'autre racine, créant ainsi un nouvel arbre (voir figure 5.2). L'opération `Chercher` consiste simplement à « remonter » dans l'arbre depuis l'élément en question jusqu'à la racine de l'arbre. Pour atteindre la complexité en $O(m.\alpha(m,n))$ il faut appliquer les deux principes suivants dans la réalisation de ces opérations :

1. **l'union par rang** : il faut lors d'une union faire pointer la racine de l'arbre

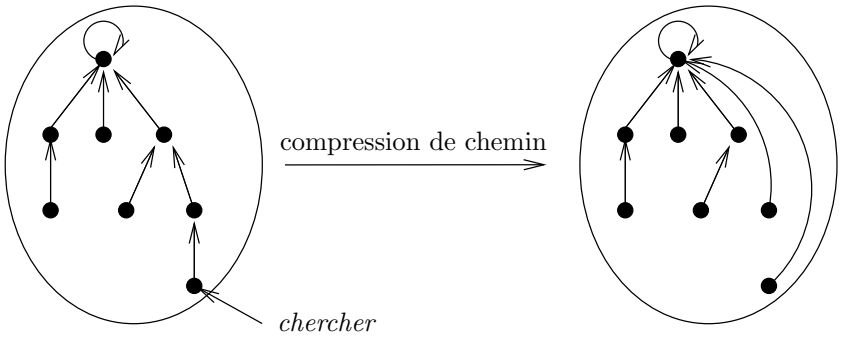


FIG. 5.3 – Compression de chemin lors de l'opération `Chercher`

contenant le moins de nœuds sur celle de l'arbre contenant le plus de nœuds. C'est ce qui a été réalisé sur l'exemple de la figure 5.2.

2. **la compression de chemin** : lors de l'opération `Chercher` il faut faire pointer chaque nœud de la route directe sur la racine comme la montre la figure 5.3.

On remarquera que ces deux opérations sont totalement indépendantes, en particulier la compression de chemin ne modifie en rien le nombre de nœuds de l'arbre et donc n'influence par l'union par rang.

5.2 Modélisation de la segmentation d'images par l'*Union-Find*

Nous allons maintenant montrer comment modéliser la segmentation par croissance de régions par l'union d'ensembles disjoints.

5.2.1 présentation générale

L'approche que nous avons choisie consiste donc à représenter chaque région par un arbre, les nœuds et feuilles de cet arbre étant les éléments de la région correspondante (voir figure 5.4). L'image devient alors une forêt d'arbres. L'élément racine joue un rôle particulier au sein de la région : c'est son représentant. Au

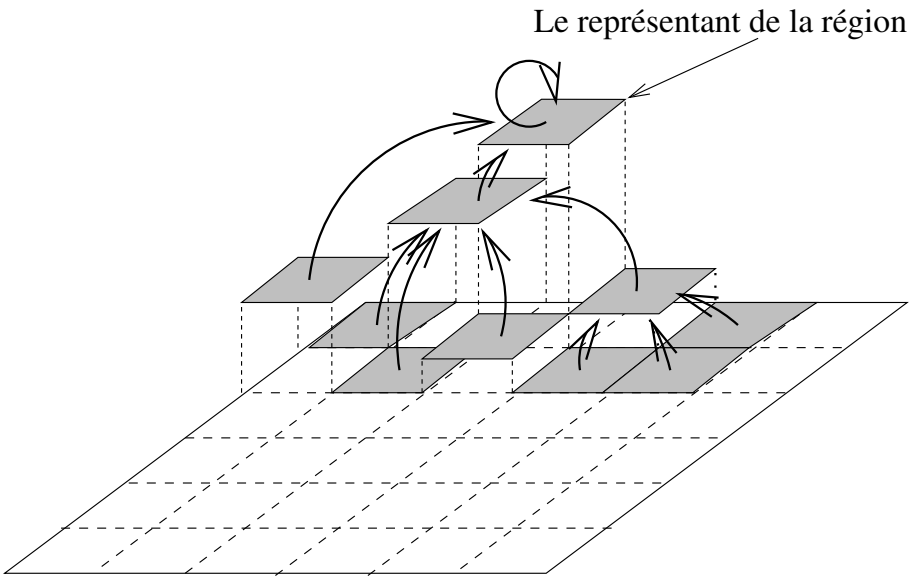


FIG. 5.4 – Représentation d'une région par un arbre.

départ du processus chaque pixel est considéré comme une région à part entière. Il suffit alors de parcourir l'image, d'examiner toutes les paires de pixels et de décider si l'on doit ou pas regrouper les deux régions auxquelles appartiennent les deux pixels de la paire considérée. Pour cela il suffit d'effectuer un `Chercher` afin de trouver le représentant de chacune des deux régions, s'ils sont différents (c'est à dire si les deux pixels n'appartenaient pas à la même région) regarder si les deux régions doivent être fusionnées, si oui le faire à l'aide de l'opération `Union`. Afin de prendre cette décision on s'aide d'un prédicat `Oracle`.

La complexité de l'algorithme dépendra donc des opérations `Union` et `Chercher`, mais aussi de l'efficacité du test effectué par le prédicat. Afin de garantir que l'`Oracle` utilisé n'augmente pas la complexité de nos algorithmes, nous avons utilisé de simples fonctions basées sur un seuil et des informations statistiques qui peuvent aisément être maintenues à jour par un nombre constant d'opérations, donc sans augmenter la complexité. Ces informations sont :

1. la valeur absolue de la différence des niveaux de gris[1] ;

[1]Toutes les définitions données dans ce chapitre sont données pour des images en niveaux de gris. Il est facile de voir qu'elle peuvent être généralisées à des images couleurs, il suffit de déterminer des critères pour l'`Oracle` n'augmentant pas la complexité. Pour ceux donnés ici,

2. la différence entre le niveau de gris minimum et maximum de la région après l'union ;

3. la variance de la région potentielle.

On peut bien sûr combiner les critères précédents à l'aide d'un opérateur booléen. Nous ne développerons pas dans cette section les aspects liés à la difficulté qu'entraine la combinaison de plusieurs critères ayant chacun son propre seuil, le lecteur pourra se reporter à la discussion proposée dans la section 5.3. Un autre critère qui n'a pas été utilisé par manque de temps mais qui est en cours de réalisation, est celui de la présence d'un lignel de contour entre les deux pixels considérés. Il suffit pour cela de pré-calculer une carte de lignels de contours à l'aide de l'opérateur de détection interpixel de contour présenté au chapitre 6.

Il est clair qu'un `Oracle` utilisant de tels critères peut être calculé en temps constant si à chaque `Union` on maintient à jour proprement les valeurs de niveaux de gris minimum, maximum, somme des niveaux de gris et somme du carré des niveaux de gris. Quant au coût de l'opération `Union` il est uniquement dû au coût de l'opération `Chercher`. En effet, lors d'une `Union` il faut d'abord chercher les deux représentants puis lier l'un des deux à l'autre. Cette dernière opération s'effectuant en temps constant le coût de l'`Union` sera donc dépendant du coût des deux opérations `Chercher` réalisées. Clairement le coût de l'opération `Chercher` est proportionnel à la longueur du chemin menant à la racine. On note le père d'un élément x et l'opérateur réalisant le saut à ce même père de la même façon : $\texttt{père}(x)$. On dit qu'un élément x a un *accès direct* à sa région si $\texttt{père}(x)$ est la racine de l'arbre, c'est à dire, le représentant de la région à laquelle appartient x. On appelle `ChercherComprimer` la fonction réalisant l'opération `Chercher` tout en effectuant la compression du chemin vers la racine. On définit alors la fonction `MiseÀplat` donnée par l'algorithme 2.

Cette fonction est utilisée par les deux algorithmes afin de diminuer le coût direct de l'opération `Chercher`.

Nous avons alors les propriétés, suivantes , communes aux deux algorithmes :

---

on peut par exemple considérer qu'avant tout calcul on transforme chaque couleur en niveau de gris en utilisant la formule définie par la Commission Internationale de l'Éclairage : $n = 0.28r + 0.59v + 0.11b$.

Algorithme 2: `MiseÀplat(`R`)` — où R est une région (un ensemble)

Données : Une région R

pour tout $x \in R$ **faire**
  `ChercherComprimer(`x`)`

(5.2.3) Après un appel à `ChercherComprimer(`x`)`, tous les éléments qui étaient sur le chemin de x à la racine ont un accès direct à leur région.

(5.2.4) `ChercherComprimer(`x`)` s'effectue en au plus l sauts[2], où l est la longueur du chemin de x vers la racine.

Supposons maintenant que nous avons une région R telle que tous ses éléments aient un accès direct à leur région, et que nous réalisions une série d'opérations `Union` et `Chercher` sur R. Il est clair qu'après plusieurs opérations `Union` tous les éléments de R n'ont plus un accès direct à leur région. Si nous faisons appel alors à `MiseÀplat(`R`)` nous obtenons :

(5.2.5) Après `MiseÀplat(`R`)` tous les éléments de R ont un accès direct à leur région.

(5.2.6) `MiseÀplat` s'exécute en au plus $2\left|R\right|$ sauts.

Quant à l'opération `Union`, chaque algorithme en utilise une différente tel qu'il sera décrit dans la section 5.2.2 suivante.

5.2.2 algorithmes linéaires de segmentation

Les deux algorithmes que nous proposons s'exécutent en temps linéaire. Ils utilisent tous deux des méthodes classiques de balayage de l'image. Ainsi `ScanLine` balaye l'image ligne à ligne, technique utilisée par exemple par O. Monga et B. Wrobbel-Daucourt dans [MWD87], alors que `MergeSquare` utilise une méthode de type arbre-quaternaire (« quad-tree »), voir par exemple [HP74].

Le premier algorithme parcourt l'image ligne par ligne. Pour chaque ligne il regarde pour chaque pixel s'il peut fusionner la région à laquelle le pixel appartient

[2] passage au père

avec celle du pixel à gauche et au-dessus. Après avoir fini une ligne il effectue un post-traitement consistant en l'appel de `MiseÀplat` pour chaque région de chaque pixel de la ligne, ceci afin de maintenir les propriétés de la structure de données.

Le second algorithme, dans sa variante récursive[3], procède en divisant l'image en 4 régions et récursivement chaque région en 4. À chaque étape de la récursivité on cherche à fusionner les régions des pixels des bords de ces carrés : pour chaque paire de pixels voisins du bord de deux carrés différents on effectue une opération `Union` si l'`Oracle` est d'accord. Cet algorithme peut être facilement parallélisé.

La complexité linéaire du premier algorithme est dûe au fait que nous limitons la hauteur des arbres représentant chaque région. La linéarité du second est prouvée par un amortissement du nombre d'opérations `Chercher`.

5.2.2.1 ScanLine

Cet algorithme est similaire à l'algorithme proposé par M. Dillencourt et al. qui est aussi linéaire mais qui est restreint aux images binaires (c'est à dire en noir et blanc) et qui ne permet pas la même généralisation aux graphes planaires que celle présentée dans [FG96].

L'algorithme 3 que nous avons appelé « `ScanLine` » parcourt l'image ligne par ligne et effectue des unions d'ensembles disjoints sur chaque région rencontrée. Dans la suite de cette section on supposera que l'image est un rectangle $w \times l$. Au début chaque région est constituée d'un et un seul pixel. On effectue d'abord le traitement pour la première ligne puis ensuite deux lignes par deux lignes (voir figure 5.5).

Pour chaque ligne on examine chaque pixel et on regarde à l'aide du prédicat `Oracle` si l'on doit fusionner les deux régions correspondantes aux pixels à gauche et au-dessus. Il est évident que nous ne traitons pas le premier pixel, et que pour la première ligne nous ne regardons que le pixel à gauche. Après avoir traité chaque pixel d'une ligne on fait appel à `MiseÀplat` pour chaque région de chaque pixel de la ligne.

L'opération `Union` doit être particularisée si l'on veut garantir la complexité : `Union` lie toujours la racine de la région ayant été rencontrée la première fois sur la ligne, à celle ayant été rencontrée plus tard. On réalise ainsi une sorte de

[3]On a alors l'hypothèse que l'image est un carré de taille $\sqrt{n} \times \sqrt{n}$.

Algorithme 3: `ScanLine`

Données : Un bitmap bm de taille $w \times l$.
Résultat : Une image segmentée en régions avec étiquetage des composantes connexes.

traitement spécial pour la première ligne;
pour *i=2 à l* **faire**
  traitement spécial pour le premier pixel de la ligne;
  pour *j=2 à w* **faire**
    gauche = `ChercherComprimer(`*bm[i,j-1]*`)`;
    dessus = `ChercherComprimer(`*bm[i-1,j]*`)`;
    courant = `ChercherComprimer(`*bm[i,j]*`)`;
    si *`Oracle(`*gauche,courant *`)`* **alors** `Union(`*gauche,courant* `)`;
    si *`Oracle(`*dessus,courant *`)`* **alors** `Union(`*dessus,courant* `)`;
  pour *j=2 à w* **faire** `MiseÀplat(`*`Chercher(`*bm[i,j]*`)`*`)`;

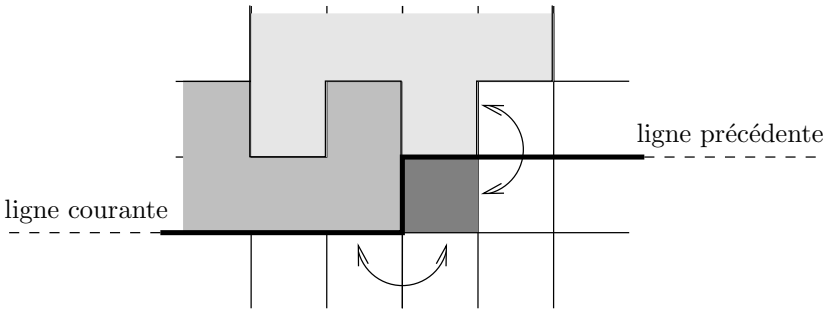


FIG. 5.5 – Balayage deux lignes par deux lignes.

parenthésage des régions rencontrées, de la plus « vieille » à la plus récente. Ce fait sera expliqué dans la preuve du lemme 5.3. Nous obtenons alors le résultat exprimé par le théorème 5.1 :

Théorème 5.1 *L'algorithme* `ScanLine` *examine toutes les paires de pixels voisins et s'exécute en temps linéaire en le nombre de pixels.*

Il est facile de voir que toutes les paires de pixels voisins sont examinées, ceci est dû au parcours utilisé. Pour prouver la complexité nous avons besoin de la

proposition 5.2 et du lemme 5.3.

Proposition 5.2 *Avant le traitement d'une ligne, chaque pixel de la ligne précédente a un accès direct à sa région.*

Preuve :
Ceci est garanti par la propriété (5.2.5) de `MiseÀplat`. □

La proposition 5.2 va nous permettre de prouver le lemme 5.3 suivant :

Lemme 5.3 *Chaque `ChercherComprimer` réalisé lors du traitement d'une ligne, effectue au plus 2 sauts.*

Preuve :
Avant le traitement d'une ligne, tous les pixels de la ligne et ceux de la ligne précédente ont un accès direct à leur région. Ceci parce que chacun des pixels de la ligne courante forme une région et que la proposition 5.2 nous assure que c'est également vrai pour la ligne précédente.

Quand une région est constituée d'un pixel unique et qu'une opération `Union` est nécessaire, il suffit de rajouter ce pixel à l'autre région. Les choses se compliquent lorsque l'on doit réaliser l'union de deux régions possédant chacune plusieurs pixels. En effet la profondeur de l'arbre va s'agrandir et on risque la prochaine fois de multiplier les sauts lors des opérations `Chercher` pour retrouver la racine de ces régions.

Montrons que nous ne rencontrerons jamais un pixel nécessitant plus de deux sauts pour trouver sa racine. Supposons que nous venons juste de fusionner deux régions, appelons les *Chef* et *Thésard*, et qu'une nouvelle région, appelons la *Directeur*, doit être regroupée avec l'union des deux autres, nous risquons de former une chaîne de longueur 3. Comme le montre la figure 5.6, il y a seulement deux possibilités[4] ; soit lier *Directeur* à *Thésard* ou lier *Chef* à *Directeur*. Le premier cas est impossible. En effet, avant de faire une `Union`, nous faisons toujours un `ChercherComprimer`, de plus l'`Union` est toujours réalisée à partir des racines des régions. Puisque *Chef* et *Directeur* sont les deux seules racines mises en cause dans cette exemple, *Thésard* ne sera jamais consulté pour une telle `Union`.

Dans le second cas, tous les pixels liés directement à *Thésard* vont avoir besoin de 3 sauts pour retrouver le nouveau représentant de leur région, c'est à dire *Directeur*.

[4]Il est à noter que l'on peut inverser *Chef* et *Thésard* sans pour autant changer le raisonnement !

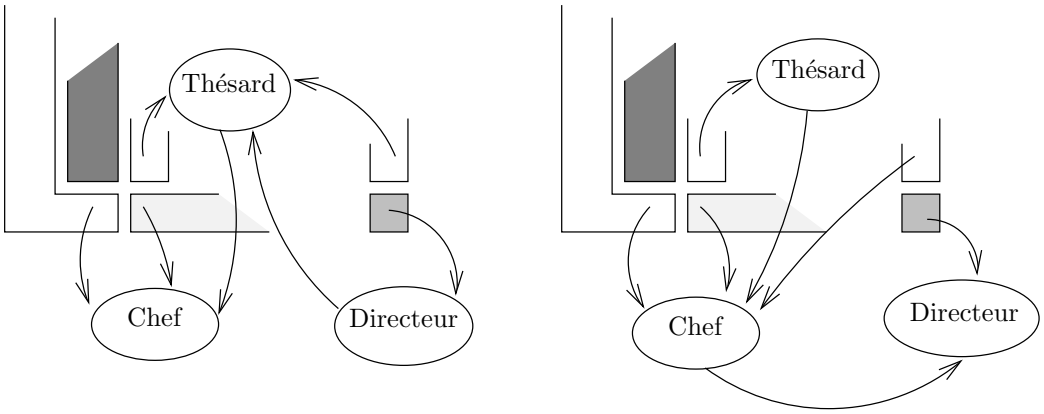


FIG. 5.6 – Les deux possibilités pour obtenir une chaîne de longueur 3.

Mais nous ne rencontrerons pas de tels pixels sur cette ligne. En effet puisque *Chef* a été lié à *Directeur*, c'est que ce dernier est plus « ancien » sur la ligne. Ceci est dû à notre opération `Union` particulière. Comme *Directeur* est une région, c'est un ensemble connexe de pixels, donc il encadre *Chef* et *Thésard*, comme le montre la figure 5.7. On retrouve ici la notion de parenthésage dont nous avions précédemment parlé.

Puisque nous ne rencontrerons pas de chaîne de longueur 3 ou plus, alors d'après la propriété (5.2.4), chaque `ChercherComprimer` réalisé nécessitera au plus 2 sauts. □

On peut maintenant prouver le théorème 5.1.

Preuve(théorème5.1) : Nous faisons l'hypothèse que le coût de l'algorithme est dû essentiellement au nombre de sauts à réaliser. Pour chaque ligne, nous la parcourons et effectuons 2 opérations `ChercherComprimer` pour chaque pixel de cette ligne : un pour le pixel au-dessus et un autre pour le pixel à gauche. De plus nous faisons au plus 2 opérations `Union` par pixel de la ligne, mais le lien d'une racine à une autre se fait en temps constant et n'effectue pas de sauts.

Calculons maintenant le nombre total de sauts effectués. L'opération `MiseÀplat` est répétée pour chaque pixel de chaque ligne, le coût total est calculé grâce à la propriété (5.2.6) : $2.w.l = 2n$, où n est le nombre total de pixels. Pour chaque pixel nous réalisons en outre 2 opérations `ChercherComprimer`, soit d'après la proposition 5.2 un coût total de : $4.w.l = 6n$. Donc le nombre de sauts total est de $8n$ soit une complexité en $O(n)$. □

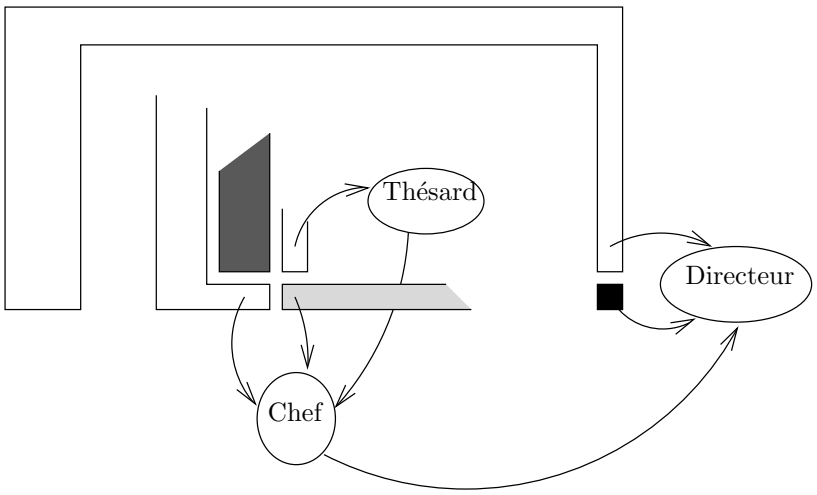


FIG. 5.7 – Effet du « parenthésage » des régions plus « anciennes » sur les plus « récentes ».

5.2.2.2 MergeSquare

Nous allons maintenant présenter un algorithme qui est également linéaire, mais qui possède la propriété supplémentaire de pouvoir être directement parallélisé comme on le verra à la section 5.4.1. Pour trouver la bonne borne supérieure de la complexité, il sera nécessaire de calculer le nombre de sauts effectués par l'opération `Chercher` sur toute l'exécution de l'algorithme. En réalité un effet d'amortissement se produit sur le nombre d'opérations `Chercher`. En conséquence si à un niveau de la récursivité on peut avoir un nombre logarithmique, et non pas constant, de sauts effectués, au total on obtient un nombre proportionnel au nombre de pixels. Pour garantir au plus un nombre logarithmique de sauts par opération `Chercher`, nous utilisons la variante classique de l'opération `Union` présentée à la section 5.2.1, c'est à dire celle consistant à lier la plus petite région à la plus grande afin de ne pas augmenter la hauteur de l'arbre. Grâce à ce choix, nous avons la propriété suivante, (voir par exemple [Meh84]), dont nous aurons besoin plus tard :

(5.2.7) Chaque opération `Chercher` peut être réalisée avec $\log s$ sauts, s étant le cardinal de l'ensemble en question.

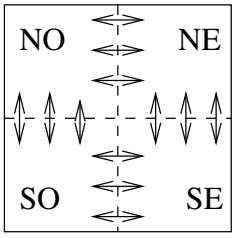


FIG. 5.8 –

Pour cette variante, nous faisons l'hypothèse que l'image est un carré de $\sqrt{n} \times \sqrt{n}$, $\sqrt{n}$ étant une puissance de 2. Nous procédons alors en divisant récursivement l'image en 4, en commençant donc par des carrés de $\frac{\sqrt{n}}{2} \times \frac{\sqrt{n}}{2}$, pour finir sur des carrés de la taille d'un pixel. À chaque retour d'un appel récursif, les régions des quatres carrés sont fusionnées le long des bords de ces carrés : pour chaque paire de pixels voisins appartenant à deux carrés différents et à deux régions différentes, on fusionne ces deux régions si l'`Oracle` est d'accord. Ce principe est illustré à la figure 5.8.

Pour faciliter la rédaction de l'algorithme nous supposerons avoir facilement accès aux quatres « sous-carrés » du bitmap à un niveau donné de la récursivité : le bitmap étant nommé bm, nous noterons par $bm[NO]$ la partie de la matrice couverte par le carré en haut à gauche, c'est à dire au **N**ord-**O**uest, ... La variante récursive de `MergeSquare` est donnée par l'algorithme 4.

Algorithme 4: – `MergeSquare(`*bm,k*`)`

Données : Un entier k et un bitmap bm de taille $2^k \times 2^k$.

si *k=0* **alors** fin;
$h^- \leftarrow 2^{k-1} - 1$;
$h^+ \leftarrow 2^{k-1}$;
pour *DIR=NO* ***à*** *SE* **faire** `MergeSquare(`*bm[DIR],k-1*`)`;
pour *i=0* ***à*** 2^k **faire**
  gauche ← `Chercher(`$bm[i,h^-]$`)`;
  droite ← `Chercher(`$bm[i,h^+]$`)`;
  si *Oracle(*gauche,droite *)* **alors** `Union(`*gauche,droite* `)`;
pour *i=0* ***à*** 2^k **faire**
  dessus ← `Chercher(`$bm[h^-,i]$`)`;
  dessous ← `Chercher(`$bm[h^+,i]$`)`;
  si *Oracle(*dessus,dessous *)* **alors** `Union(`*dessus,dessous* `)`;

Théorème 5.4 *L'algorithme* `MergeSquare` *examine toutes les paires de pixels voisins et s'exécute en temps linéaire en le nombre de pixels.*

Preuve :
Il est facile de vérifier que `MergeSquare` examine bien toutes les paires de pixels voisins. Pour prouver la complexité, plaçons nous à un niveau de la récursivité, c'est à dire pour une taille de carré de $2^k \times 2^k$. Nous avons 4 appels récursifs et $2^k \times 2^k = 2^{2k}$ opérations `Union` possibles. Nous supposerons que le coût d'une telle opération est borné par le nombre de sauts effectués par les 2 opérations `Chercher` lors de la recherche des racines correspondantes afin de lier l'une à l'autre.

Les régions à fusionner ont une taille bornée par $2^k \times 2^k$. Donc d'après la propriété 5.2.7 nous savons que chaque opération `Union` nécessite au plus $2 \log 2^{2k} = 4k$ sauts. Donc nous avons au plus $8k2^k$ sauts pour fusionner deux régions. A une étape k donnée, nous avons $n/(2^{2k})$ carrés de taille $2^k \times 2^k$, donc à une étape donnée nous avons au total

$$8k2^k n/(2^{2k}) = 8(k/2^k)n \tag{5.1}$$

sauts, d'où la borne supérieure suivante pour le nombre de sauts total pour tout l'algorithme :

$$\sum_{k=1}^{\log \sqrt{n}} 8(k/2^k)n < 8n \sum_{k=1}^{\infty} k\frac{1}{2}^k \tag{5.2}$$

Pour $-1 < x < 1$ nous avons l'identité bien connue :

$$\frac{x}{(1-x)^2} = \sum_{k=1}^{\infty} kx^k \tag{5.3}$$

Cela peut être facilement vérifié en développant la fonction $\frac{x}{(1-x)^2}$ par une série de Taylor en 0. La partie droite de l'équation 5.2 peut alors être évaluée à $16n$, on a donc bien une complexité linéaire en le nombre de pixels n de notre image. □

5.2.2.3 variante itérative

Si l'on examine de près un niveau l de la récursivité, on remarque que les paires de pixels concernées par les opérations `Union` se trouvent toutes sur des lignes et colonnes spécifiques de la matrice. On pourra vérifier que toutes ces paires sont de la forme :

– $\Big((i,j),(i+1,j)\Big)$ avec $i = r.2^l + 2^{l-1}$ pour $r = 0,\ldots,\sqrt{n}/2^l$ et $j = 1,\ldots,n$;
– $\Big((i,j),(i,j+1)\Big)$ avec $i = 1,\ldots,n$ et $j = r.2^l + 2^{l-1}$ pour $r = 0,\ldots,\sqrt{n}/2^l$.

On peut alors faire ces fusions dans n'importe quel ordre. En fait, en commençant avec $l = 1$, on peut procéder itérativement et obtenir un algorithme équivalent à la variante récursive mais pouvant travailler sur des images rectangulaires.

5.3 Critère unifié de décision

Nos algorithmes travaillent donc à partir des pixels pour faire de la segmentation en régions. Il est donc facile d'imaginer une coopération régions-contours. Plus précisément on peut envisager un `Oracle` travaillant à partir d'informations locales, comme la différence des niveaux de gris des pixels ou la présence d'un lignel de contour entre eux[5], et d'informations globales, c'est à dire ayant trait aux régions comme la variance des niveaux de gris ou la différence des moyennes. On se retrouve alors avec plusieurs paramètres inter-agissant. Comment combiner de manière correcte ces différentes informations et comment contrôler leurs interactions est un problème délicat, voir par exemple [SG93]. La façon habituelle de procéder est de comparer chaque critère à une valeur de seuil et de réaliser un *ou* logique entre les différentes réponses données par chacun d'eux.

Cette méthode de fonctionnement présente, à notre avis, certains inconvénients :
– il n'est pas facile de choisir un seuil pour chaque critère, surtout s'ils sont nombreux ;
– il est encore plus difficile d'appréhender la manière dont ils vont s'influencer au sein de l'oracle ;
– il n'est pas possible de donner plus d'importance à l'un ou l'autre des critères ;
– la méthode est très sensible au bruit et au légères variations se produisant autour des valeurs de chaque seuil.

Nous proposons une méthode générale permettant de combiner différents critères ayant les caractéristiques suivantes :

[5] On pourrait également travailler avec un détecteur de contours classiques, mais se pose alors un problème de localisation de la frontière entre les régions, en effet les contours sont localisés sur les pixels.

– un paramètre unique pour éviter le problème du choix de multiples seuils ;
– possibilité de donner à chaque critère une importance relative ;
– résistance au bruit ou aux petites variations autour des valeurs du seuil.

5.3.1 description de la méthode

Tout d'abord, nous faisons la distinction entre les *critères de regroupement* et les *oracles*. Ces deux types de fonctions prennent en paramètre deux pixels appartenant à deux régions différentes. Pour obtenir des informations sur les régions concernées elles n'ont qu'à faire appel à l'opération `Chercher` de la structure présentée à la section 5.2.1[6]. Mais elles diffèrent sur le type de résultat renvoyé :

– Les *critères* sont des fonctions normalisées, c'est à dire qui vont renvoyer une valeur dans l'intervalle $[0, 1]$. Un tel résultat sera interprété comme la probabilité pour que les deux régions unifiées correspondent bien à la définition d'une région ;
– Au contraire, un *oracle* est une fonction qui va finalement donner la décision à prendre. Le résultat renvoyé doit être 0 ou 1. Une fonction simple de décision sera un oracle utilisant un seul critère et muni d'un seuil. On retrouve alors le même type de fonction de décision que celles habituelles présentées plus haut.

Par contre pour combiner plusieurs critères on utilisera un *produit pondéré de critères* dont voici la définition :

Définition 5.1 (produit pondéré de critères) *Soient des critères normalisés* $C_1, \ldots, C_k$ *et les poids* $w_1, \ldots, w_k$ *tels que* $\sum_{i=1}^{k} w_i = 1$*, le* produit pondéré de critères $P_w(C_1, \ldots, C_k)$ *est donné par :*

$$P_w(C_1, \ldots, C_k) = \prod_{i=1}^{k} C_i^{w_i}$$

.

Cette formulation possède les avantages suivants :

1. Elle permet de passer outre l'indécision de l'un des critères.

[6]Si cette structure n'est pas utilisée, on suppose qu'il existe un mécanisme permettant de donner pour chaque pixel la région à laquelle il appartient.

2. Si les critères ne changent pas brusquement de valeurs autour d'un seuil donné, le produit pondéré des critères sera robuste au bruit et aux légères variations autour de ce seuil[7].
3. Les poids nous permettent de donner une importance différente à chaque critère.

On définira l'*oracle* comme une fonction $\overline{P}(s, C_1, \ldots, C_k)$ qui dépend, en plus des critères et des poids, d'un seuil s. Il ne fait que comparer ce seuil au produit pondéré des critères : $P_w(C_1, \ldots, C_k) > s$. Si les critères calculés sont exponentiels, comme on le propose à la section suivante, il sera plus facile de comparer les logarithmes de chacune des valeurs, on obtient alors :

$$\sum_{i=1}^{k} w_i \log C_i > \log s$$

5.3.2 quelques critères particuliers

Nous montrons ici comment mettre en œuvre la méthode que nous proposons avec des critères locaux et globaux classiques. Cette liste n'est pas exhaustive, ce ne sont que quelques exemples.

critères basés sur une distance Parmi les critères les plus souvent utilisés en traitement d'images, on trouve ceux basés sur une distance entre les niveaux de gris des pixels, pour un critère local, ou de distance entre les niveaux de gris moyens des régions pour un critère global. Un tel critère ne nous restreint pas seulement aux images en niveaux de gris, en effet des fonctions de distance sur les couleurs ont été étudiées, voir par exemple [SMT82].

La manière la plus facile d'utiliser une telle distance entre les niveaux de gris est la comparaison avec un seuil, avec tous les inconvénients cités plus haut.

[7] C'est le cas de la plupart des critères habituellement utilisés comme la différence des moyennes par exemple. La différence peut-être proche d'un seuil donné, mais la valeur normalisée retournée par un critère tel qu'on le propose ne variera pas beaucoup. Par contre habituellement une légère variation peut entraîner le passage à une valeur inférieure au seuil et donc passer d'une valeur 0 à une valeur 1 !

Avec un minimum d'efforts on peut définir un critère normalisé répondant à nos spécifications. Par exemple dans le cas d'images en 256 niveaux de gris la distance maximale entre deux niveaux de gris c_1 et c_2 est de 256. On pourra alors définir la distance normalisée $d(c_1, c_2) = |c_1 - c_2|/256$ qui nous permettra aisément de définir un critère normalisé. D'une manière générale pour une fonction de distance d' entre les niveaux de gris ayant une valeur maximale d'_∞, on obtient la distance normalisée $d(c_1, c_2) = d'(c_1, c_2)/d'_\infty$.

En fait pour obtenir un critère normalisé valide avec un telle distance il faudra prendre $1 - d(c_1, c_2)$. On appellera un critère de ce type, un *critère linéaire de distance*. Il a l'avantage de dépendre de manière continue des données, par contre il a le désavantage d'avoir un assez grand intervalle de valeurs pour lesquelles il est indécis.

opérateurs gradients D'autre critères habituellement utilisés sont ceux basés sur des valeurs de l'opérateur gradient ou Laplacien. Ils sont utilisés pour savoir si un pixel est un point de contour. Par exemple, pour calculer la norme du vecteur gradient, on approxime les dérivées par des différences (voir par exemple [GW87]), on obtient alors la relation suivante[8] :

$$G[f(x,y)] \cong |f(x,y) - f(x+1,y)| + |f(x,y) - f(x,y+1)|$$

Nous avons alors le *critère gradient normalisé* :

$$\Big(|f(x,y) - f(x+1,y)| + |f(x,y) - f(x,y+1)|\Big) / d'_\infty$$

critères exponentiels Pour éviter les inconvénients posés par les critères linéaires qui ont un grand intervalle de valeurs pour lesquelles ils sont indécis, nous proposons de passer le critère comme argument d'une fonction exponentielle de la manière suivante : $e^{-\alpha C}$ où C est le critère normalisé. Le choix du coefficient α influence le comportement de ce nouveau critère. Plus α est grand, plus la courbe sera incurvée et le comportement du critère se rapproche d'un critère avec seuil ; pour un α petit la courbe sera presque droite et on retrouve le comportement d'un critère linéaire. La figure 5.9 donne des exemples avec différentes valeurs de α pour

[8] Cette relation est équivalente à l'opérateur gradient défini par L.G. Roberts [Rob65] à l'aide de masques de convolution 2×2.

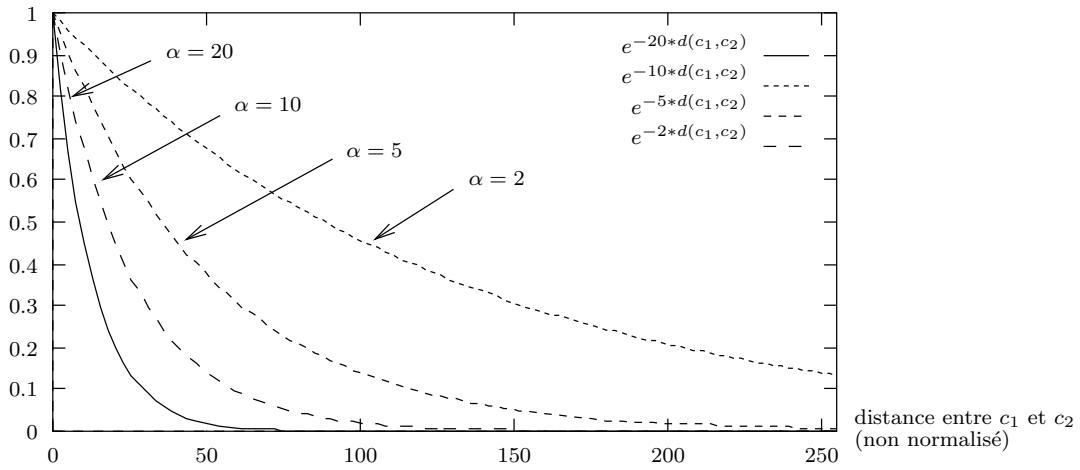


FIG. 5.9 – Influence du coefficient α sur le comportement d'un critère exponentiel.

un critère basé sur la distance normalisée entre deux niveaux de gris[9].

On remarquera qu'une tel critère ne peut jamais prendre la valeur 0. En particulier pour une distance $d(c_1, c_2) = 1$, le critère classique répondrait 0, ici nous obtenons la valeur $e^{-\alpha}$. Contrairement à la méthode précédente, un tel critère ne peut à lui seul interdire la fusion de deux régions. On appelle cette valeur la *capacité de veto* du critère exponentiel. Cette capacité de veto est la valeur que doivent contrecarrer les autres critères pour provoquer une fusion. Plus cette valeur est petite, plus c'est difficile. Ainsi un critère exponentiel ayant une capacité de veto de 0 ne pourra pas être contrecarré dans les situations où il renvoie cette valeur. La fusion des deux régions sera alors impossible.

5.4 Conclusion et perspectives

Nous avons donné dans ce chapitre deux algorithmes linéaires de segmentations en régions. Ces algorithmes ont la particularité de permettre une coopération région-contour aisée. De plus ils réalisent un étiquetage des composantes connexes.

[9]Rappel : la distance normalisé est alors : $|c_1 - c_2|/256$.

C'est à notre connaissance les seuls algorithmes ayant toutes ces fonctionnalités réunies et s'exécutant en une seule passe sur l'image. D'ailleurs la complexité théorique se traduit par une vitesse d'exécution relativement grande comme on peut le voir dans la table des temps donnée à la section 5.5. De plus les résultats obtenus en segmentation sont comparables à ceux que l'on peut voir habituellement dans les livres ou articles (voir par exemple [HM93, HS92, BB92, AB94, Vei94]). Le lecteur pourra juger de lui-même, nous donnons plusieurs exemples à la section 5.5.

L'étiquetage des composantes connexes, et la relation privilégiée qu'offre la structure d'ensembles disjoints entre les éléments (ici un pixel) et les ensembles auxquels ils appartiennent (les régions dans notre cas), peuvent être très utiles pour la réalisation d'autres algorithmes. Ainsi comme on l'a vu au chapitre 4, cela nous a permis de mettre au point un algorithme linéaire d'extraction du graphe des frontières (une vision différente du graphe d'adjacence). Ces algorithmes offrent de plus des perspectives intéressantes. Nous présentons par exemple la parallélisation de `MergeSquare` à la section 5.4.1 et la mise en correspondance de deux images segmentées à la section 5.4.2. On pourra également se reporter à [d'A94] pour une version parallèle de l'algorithme `ScanLine`.

5.4.1 parallélisation de MergeSquare

L'efficacité de l'algorithme `MergeSquare` pourrait être grandement augmentée si on le parallélisait. De la définition de l'algorithme on en déduit facilement la proposition suivante :

Proposition 5.5 *`MergeSquare` peut être implanté efficacement sur une PRAM*[10] *avec $p < 2^k$ processeurs avec une complexité de $O(2^{2k}/p)$ pour un bitmap de taille $2^k \times 2^k$.*

Pour implanter `MergeSquare` de manière efficace, il est important

1. de ne pas avoir à copier les données pour un appel récursif,
2. qu'une routine à un niveau inférieur de la récursivité n'ait pas à se préoccuper de la taille globale des données,

[10] ***machine parallèle à accès aléatoire*** *: c'est une machine composée de plusieurs processeurs séquentiels ordinaires disposant d'une mémoire globale partagée et pouvant y accéder en parallèle.*

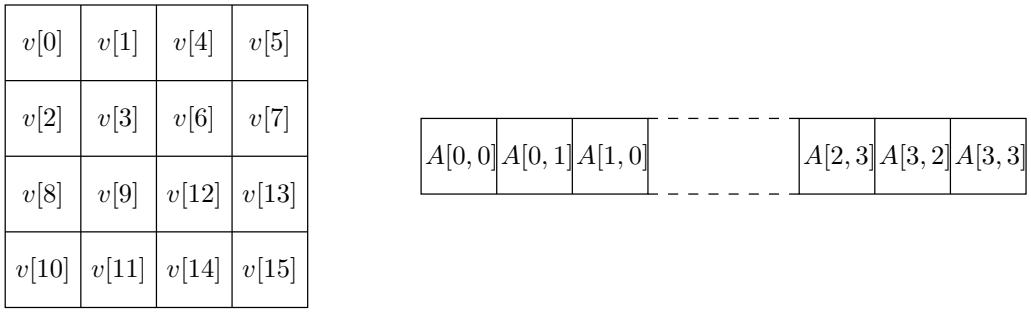


FIG. 5.10 – Une linéarisation adéquate de la mémoire.

3. que la matrice soit linéarisée en mémoire de telle sorte que chaque appel trouve ses données regroupées en une seule partie de la mémoire.

Alors tous les appels récursifs pourront s'exécuter de manière totalement indépendante les uns des autres. Cela peut être réalisé grâce à une définition récursive en arbre quaternaire de la matrice, c'est à dire en plaçant en mémoire d'abord la partie NE, puis NO, etc... (voir plus haut (section 5.2.2.2)la définition de NE, NO, ...). Un pixel (i, j), tel que les représentations binaires de i et j sont respectivement $i_{k-1} \dots i_0$ et $j_{k-1} \dots j_0$, sera alors placé à la position $i_{k-1}j_{k-1} \dots i_0j_0$. Autrement dit, l'index de la position dans la mémoire d'un élément particulier (i, j) de la matrice est donné en alternant les bits de i et j. Ainsi, une matrice 4×4 sera linéarisée en un vecteur de longueur 16 (voir figure 5.10) et on placera, par exemple, l'élément $(2, 2)$, soit $(10, 10)$ en binaire, de la matrice, à la place à la position 12, soit 1100 en binaire, du vecteur mémoire.

Cette linéarisation particulière de la matrice s'apparente aux arbres quaternaires linéaire [Sam84] et plus particulièrement à la construction d'un quad-tree par un parcours de Peano [CP95].

5.4.2 mise en correspondances de deux images segmentées

Ici nous supposons que nous avons réalisé deux segmentations différentes, par exemple avec nos deux algorithmes, d'une même image. Nous voulons maintenant faire une synthèse de ces deux segmentations. Pour cela nous allons mettre en cor-

respondance les régions de chaque segmentation ayant des pixels en commun. C'est aisément réalisable si les segmentations ont été réalisées avec nos algorithmes. En effet nous bénéficions de l'étiquetage des composantes connexes, et en parcourant chaque image en parallèle on peut construire une liste de paires de régions où chaque paire indique que les deux régions sont couvertes par un même pixel. Il faut maintenant collecter ces données, en regrouper toutes les paires identiques en un triplet contenant l'identification des deux régions, et le nombre d'occurrences de la paire correspondante. On pourra alors traiter ces données pour améliorer une des deux segmentations. Par exemple on pourra forcer le regroupement de deux régions de la première segmentation si les triplets nous indiquent que ces deux régions sont pratiquement entièrement couvertes par une seule région de la deuxième segmentation.

Qu'en est-il de la complexité d'un tel processus de traitement ? On sait déjà que les deux segmentations et l'étiquetage des composantes connexes peuvent être réalisés en temps linéaire. Reste la collecte des données. On l'a vu, l'obtention de toutes les paires peut se faire en un balayage de chaque image segmentée, donc toujours en temps linéaire. Reste à regrouper ces paires. Pour cela il suffit de les trier et de parcourir la liste triée afin de regrouper les différentes occurrences de chaque paire. On choisira ici d'effectuer un tri par base (voir par exemple [CLR90][11]). La complexité sera alors en $O(k+m)$ où k est le nombre de clefs (ici le nombre de régions) pour les tris par dénombrement nécessaires pour le tri par base, et m le nombre de paires à trier (ici le nombre n de pixels). Le nombre de régions étant au plus égal au nombre de pixels (bien que dans ce cas on ne puisse pas dire que la segmentation ait eu un quelconque effet), on obtient finalement une complexité en $O(n)$. La collecte s'effectue également en $O(n)$ puisqu'il faut parcourir la liste de toutes les paires. Au total tout ce processus s'exécutera donc en un temps linéaire.

5.5 Présentation de quelques résultats

La complexité théorique s'est vérifiée dans la pratique au niveau des temps d'exécution. En effet, nous avons implanté `ScanLine` et `MergeSquare` en C^{++}, sans

[11] Une traduction en français est disponible : [CLR94].

optimisations particulières autre que celles fournies par les options du compilateur, et nous obtenons, sur une Sun Sparc 5, les temps de calcul donnés par le tableau :

image	256×256	512×512	705×576
initialisation :	$0.12s$	$0.68s$	0.71
algorithme :	$0.56s$	$3.76s$	$4.58s$
algo/pixel :	$8.54\mu s$	$14.3\mu s$	$11.2\mu s$

Ce qui revient à un temps d'exécution de 10 à $15\mu sec$ par pixel, soit pratiquement du temps réel.

D'autre part nous avons mis en place la méthode de critère unifié à l'aide de ces mêmes algorithmes. Voici quelques résultats obtenus...

La figure 5.11 montre l'image « Lena » segmentée en régions par l'algorithme `ScanLine`, les critères utilisés sont des critères linéaires de distance sur les pixels et les régions.

Un exemple de résultat obtenu par l'algorithme `MergeSquare` est donné à la figure 5.12. On peut voir que le découpage en arbre quaternaire pour traiter l'image fait apparaitre des frontières artificielles rectilignes. Il faudrait appliquer un post-traitement cherchant à enlever ces lignes.

À la figure 5.13.c on peut voir un exemple de coopération régions-contours utilisant notre détecteur de contours interpixel. Ceci n'est qu'une première approche, en effet lors de l'algorithme, à chaque examen d'une paire de pixels, nous regardons le lignel séparant les deux pixels ; or il faudrait également tenir compte des autres lignels le long de la frontière. La prise en compte d'un tel critère ne peut pas se faire sans précaution, sinon on risque de perdre la complexité linéaire de nos algorithmes, et donc perdre en performance. Les critères utilisés dans cette figure sont :

1. Le critère linéaire de distance, c'est à dire $(1 - d(R_1, R_2)/255)$ où $d(R_1, R_2)$ est la valeur absolue de la différence des niveaux de gris moyens des deux régions concernées.
2. Un critère booléen égal à 0 si le lignel entre les pixels est un contour, égal à 1 sinon.

L'image 5.13.b a été segmentée en utilisant le même critère global et, comme critère local, le critère linéaire de distance, mais cette fois-ci sur le niveau de gris des pixels.

FIG. 5.11 – Lena segmentée en régions par `ScanLine`.

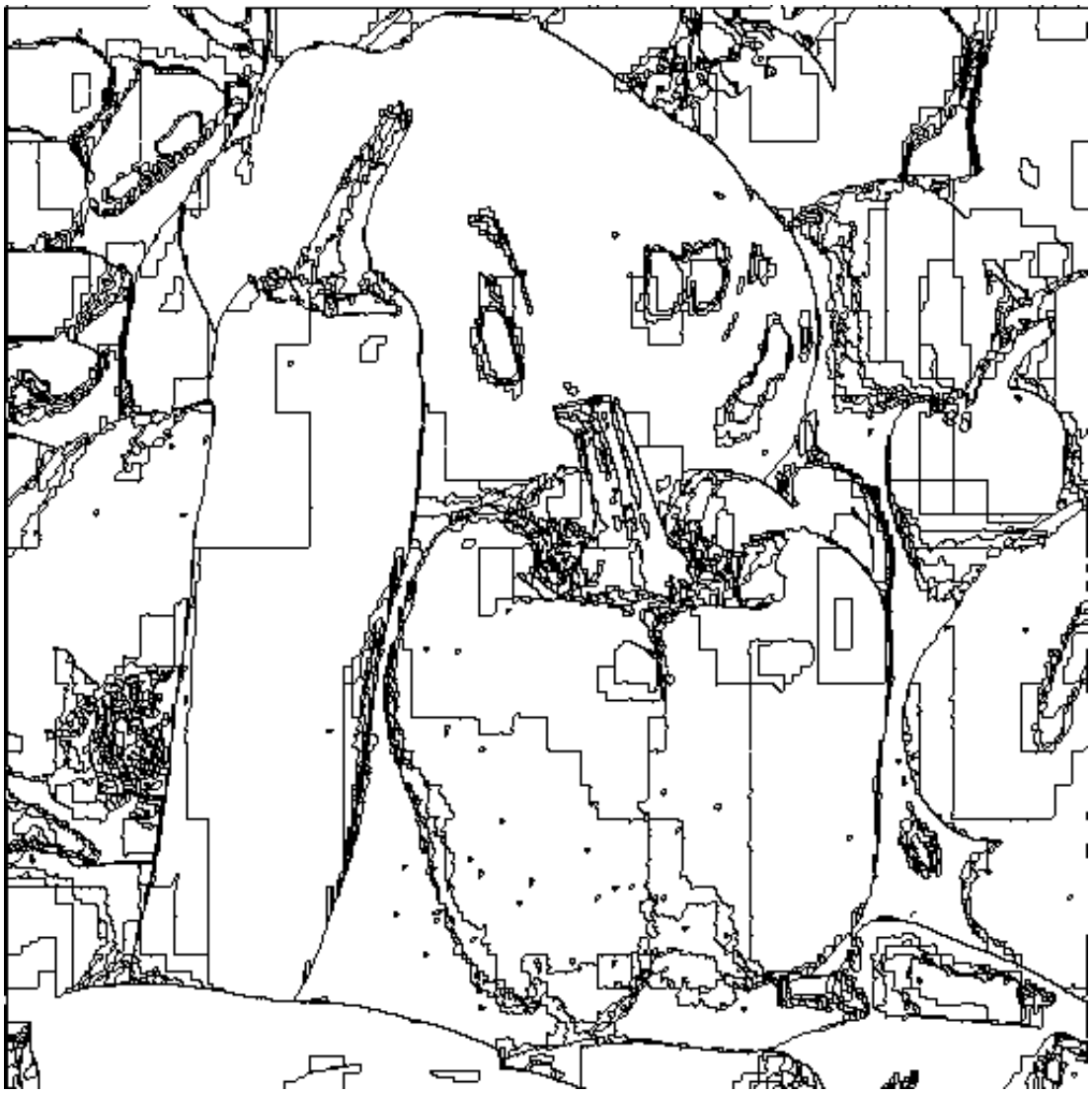
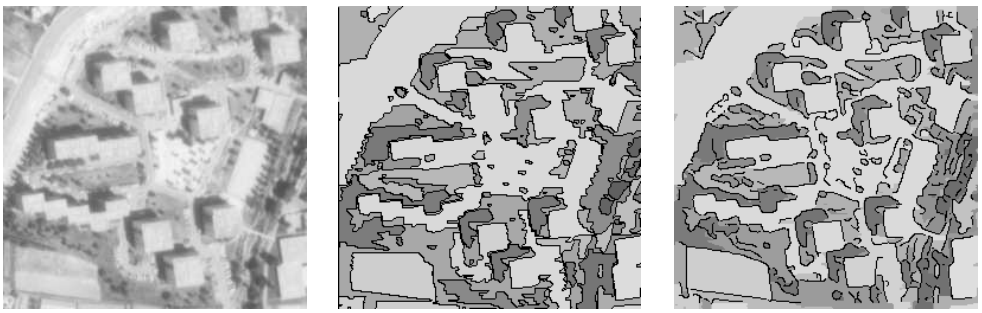


FIG. 5.12 – Image segmentée par `MergeSquare`.



a. image originale b. critère de distance c. contours et distance

FIG. 5.13 – Segmentation par coopération regions-contours.

Chapitre 6

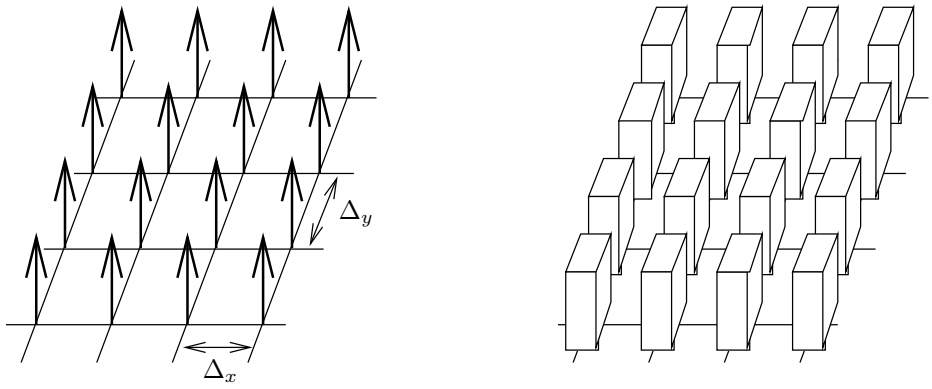
Détection de contours

Le but des détecteurs de contours est de détecter dans l'image les points de contour, les frontières entre les objets. Ce sont souvent des techniques issues du domaine du traitement du signal. L'hypothèse faite est que l'image est un signal échantillonné bruité. On calcule alors ces variations sur les points d'échantillonnage que sont les pixels. La démarche semble naturelle, mais qu'est-ce qu'un contour, une frontière dans l'image ? Intuitivement on pourrait dire qu'une frontière apparaît entre deux pixels quand la différence de leur intensité lumineuse est significative[1]. Elle devrait donc être localisée à l'interpixel et non pas sur les pixels. D'autre part si l'on considère qu'une ligne de pixels forme une arête, alors comment représenter un objet d'un pixel d'épaisseur ? C'est pourquoi il nous a semblé important de mettre au point un détecteur de contours en interpixel.

6.1 Modèle général des détecteurs de contours.

Après un bref rappel sur la formation d'une image discrète nous présenterons le principe de fonctionnement d'un détecteur de contour. Nous exposerons ensuite les propriétés principales d'une catégorie particulière de filtres : les filtres linéaires séparables. En effet nous nous intéresserons plus particulièrement aux détecteurs

[1]La notion de différence significative est dépendante de la distribution des niveaux de gris dans l'image. C'est un des problèmes des détecteurs qui finalement essayent de repérer toute différence maximale localement, puis laissent le soin de «nettoyer l'image» à une étape ultérieure de filtrage.



a. Fonction peigne de Dirac b. Fonction crête

FIG. 6.1 – Fonctions d'échantillonage

basés sur ces types de filtres. Nous ferons alors un bref rappel sur les plus classiques des détecteurs avant de présenter notre détecteur interpixel. Puis nous montrerons, toujours dans le cadre de la séparation entre information et affichage, une extension de nos résultats à la définition d'un détecteur subpixel. Nous en profiterons alors pour présenter cette nouvelle notion ainsi que les principaux travaux existant déjà.

6.1.1 naissance d'une image discrète

L'objectif ici est de transformer une image en une information exploitable par un système de vision par ordinateur. Cette information sera représentée sous la forme d'une image numérique. Pour cela on dispose d'un capteur capable de transformer l'intensité lumineuse qu'il reçoit en un signal analogique. Ce signal est ensuite échantillonné en temps et en fréquences afin de donner l'image numérique recherchée. Théoriquement l'échantillonnage consiste à multiplier le signal analogique par une fonction peigne de Dirac $\delta(x, y)$ (voir figure 6.1.a). En fait le principe de fonctionnement des capteurs CCD fait que le signal obtenu est déjà le résultat d'un premier échantillonnage où chaque valeur correspond à l'intégration de la quantité lumineuse reçue sur une surface de dimension $dx \times dy$. Cet échantillonnage est représenté par la fonction crête $\prod_{\frac{dx}{2},\frac{dy}{2}}(x, y)$ présentée à la figure 6.1.b. Une

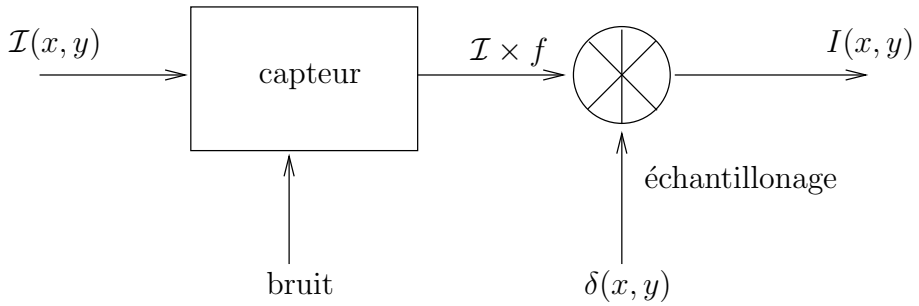


FIG. 6.2 – Un modèle d'acquisition d'images

fois échantillonné, le signal est quantifié afin par exemple de ramener les valeurs dans un intervalle $[0, 255]$.

Tout au long de ce processus de formation de l'image discrète se produisent des dégradations dues aux bruits du capteur (bruit intrinsèque), de la nature corpusculaire de la lumière (bruit extrinsèque), de la quantification et de l'optique dont nous n'avons pas parlé ici. Nous ne développerons pas ces aspects de l'acquisition, ni les aspects photométriques de la lumière. Le lecteur intéressé pourra se reporter par exemple à [Tab94, GW87] ou pour les aspects relatif aux filtres digitaux et au traitement du signal numérique à [Jac89, EV92].

Un système de vision et d'acquisition complet serait complexe à représenter car il met en jeu de nombreux procédés et fait intervenir de nombreux paramètres. Nous adopterons finalement le modèle simplifié d'acquisition présenté à la figure 6.2 où f symbolise la fonction d'acquisition du capteur. Elle peut d'ailleurs être vue comme le résultat de la convolution de la fonction peigne (voir figure 6.1) par la réponse impulsionnelle h du capteur : $f = h * \delta$.

6.1.2 schéma de fonctionnement d'un détecteur de contour

Dans la section 3.1 nous avons présenté un point de contour comme une différence significative de l'intensité lumineuse. Dans notre modèle où l'image numérique est le résultat de la discrétisation de la fonction $I_f = \mathcal{I} \times f$ cela revient à repérer les brusques variations de cette fonction. Il y a plusieurs types de variations auxquels correspondent plusieurs types de contours (voir figure 6.3). Nous nous

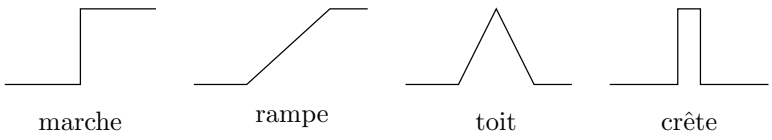


FIG. 6.3 – Différents types de contours

intéresserons principalement aux contours de type *marche*. Étant donné que nous avons une approche interpixel et que nous voulons également obtenir en définitive une description de l'image en terme de régions, les autres types de contours tels que les lignes (*crête* et *toit*) ne font pas partie de nos préoccupations. En effet une ligne sera pour nous une région fine avec deux frontières que nous pouvons alors considérer comme un contour de type marche[2].

Pour repérer les variations de la fonction I_f, il suffit de repérer les points d'inflexion, c'est à dire les maximaux de la dérivée première (approche du type gradient) ou les passages à zéro de la dérivée seconde (approche laplacien)[3]. On appelle *opérateur du premier ordre* les détecteurs travaillant à partir des dérivées partielles premières, et *opérateur du second ordre* ceux travaillant à partir des dérivées partielles secondes. Pratiquement ces deux types d'opérateurs fournissent des résultats proches. En fait, théoriquement en 1D, les contours obtenus seront les mêmes. Mais en 2D il n'en est pas de même. Ainsi on remarquera par exemple qu'un opérateur du second ordre fournira des contours fermés. Par contre il sera très sensible au bruit et fournira alors beaucoup de faux contours. Pour détecter les contours du type marche on utilisera essentiellement les deux types d'opérateur (directionnels ou non directionnels) suivants :

– l'opérateur directionnel du premier ordre, c'est à dire le vecteur gradient calculé en chaque point de l'image :

$$\vec{\nabla} I =^t (\frac{\partial I_f(x,y)}{\partial x}, \frac{\partial I_f(x,y)}{\partial y}) \text{ noté } \vec{\nabla} I =^t (I_x, I_y)$$

[2]L'application d'un filtre de lissage peut mettre en défaut cette hypothèse, la quantité de signal exprimant la valeur haute de la marche étant nettement moins importante que la quantité basse.

[3]On remarquera que l'approche gradient consiste en fait à déterminer les passages par zéro de la dérivée seconde *dans* la direction du gradient. Ce qui est différent des passages par zéro du laplacien.

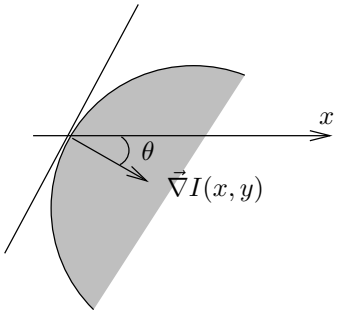


FIG. 6.4 – Direction θ du gradient en un point d'une image.

On calculera également le module $||\vec{\nabla} I||$ de ce vecteur afin de connaître «la force» du contour. Ce module pourra être calculé par exemple des deux manières suivantes :

$$\sqrt{{I_x}^2 + {I_y}^2} \quad et \quad |I_x| + |I_y|$$

En fait on utilisera généralement la première formulation car c'est la plus simple pour laquelle le module est invariant par rotation. La direction du vecteur gradient est normale par rapport au contour. On peut donc obtenir la direction θ du contour (voir figure 6.4 par le calcul suivant :

$$\theta = \arctan(\frac{\partial I_f(x,y)}{\partial y} / \frac{\partial I_f(x,y)}{\partial x})$$

– l'opérateur non-directionnel du second ordre. Par exemple le laplacien dont les valeurs à zéro donnent les points de contour. En chaque point de l'image on calculera la valeur suivante :

$$\bigtriangledown^2 I_f(x,y) = \frac{\partial I_f^2(x,y)}{\partial x^2} + \frac{\partial I_f^2(x,y)}{\partial y^2}$$

Cet opérateur est également invariant par rotation.

On appelle cette étape d'application d'un tel opérateur, la différentiation de l'image.

On a pu voir dans la section 6.1.1 qu'une image discrète contenait toujours du bruit inhérent au procédé d'acquisition. Afin d'en réduire la quantité présente dans l'image, même dans le cas de l'utilisation d'un opérateur du premier ordre, on va donc précéder la recherche des discontinuités par une étape de filtrage. Les procédés modernes se ramènent à la détermination d'un filtr de lissage ; l'étape de

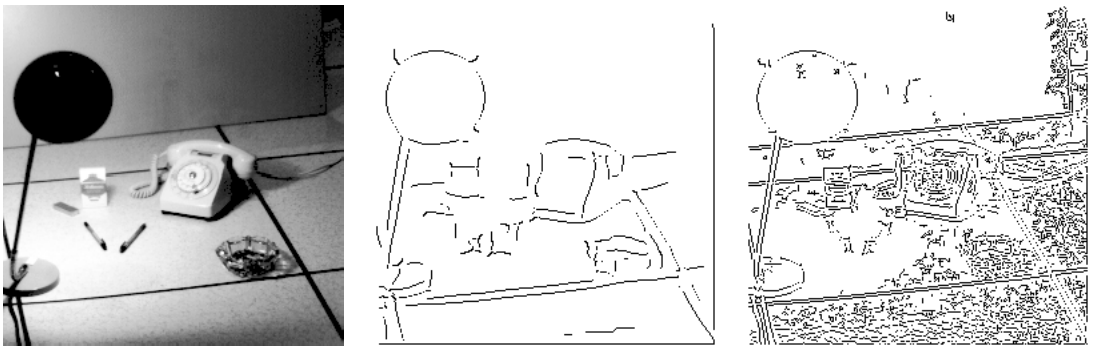


image originale | contours avec lissage à haute échelle | contours avec lissage à faible échelle

FIG. 6.5 – Effet du lissage à différentes échelles sur la détection de contours

différentiation consistant alors en l'application de la dérivée du filtre de lissage (voir plus loin à la section 6.2.2). Ces filtres seront construits sur l'hypothèse d'un bruit blanc gaussien[4]. On pourra voir un filtre comme une fonction f_α où α est l'échelle du filtre. À haute échelle, l'image sera fortement lissée et la détection des contours sera bonne (forte probabilité de marquer un vrai contour et faible probabilité de marquer un faux contour) mais la localisation sera mauvaise. Inversement à faible échelle la localisation sera bonne mais la détection mauvaise car l'image contiendra trop de détails. La figure 6.5 présente des contours détectés par le filtre de Deriche (voir section 6.2.2.3) avec seuillage par hystérésis (voir plus loin les explications concernant ce type de seuillage) à haute et basse échelle. L'inconvénient majeur du lissage est, comme on peut le voir sur les images de la figure 6.5, l'élimination et le déplacement[5] de certains contours, ainsi que la création de faux contours. Les variations d'un signal à différents lissages peuvent être représentées grâce à l'*espace échelle*. Dans le cas monodimensionnel, il décrit un plan $x - \alpha$ où α correspond à l'échelle de lissage et x décrit la valeur du signal correspondant. Le lecteur intéressé pourra se reporter par exemple à [Wit83, SMCM91]. Pour palier ces variations, A. Rosenfeld a le premier proposé d'utiliser différentes tailles de filtres et de combiner

[4]Ce qui est faux dans la plupart des cas réels.

[5]Déplacement des contours du bras de la lampe sur la figure 6.5, par exemple

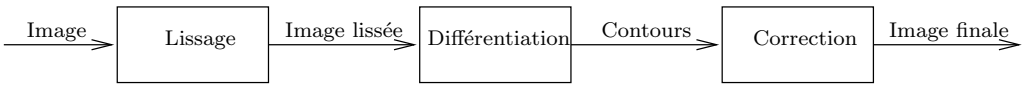


FIG. 6.6 – Schéma de fonctionnement d'un détecteur de contours.

les résultats obtenus [RT71]. Sur ce principe s'est développée une classe particulière de filtres appelés *détecteur multi-échelle*. On citera par exemple les travaux de [Zio91, LJ92, Tab94]. Des méthodes d'extraction multi-échelle de régions ont même été proposées (voir par exemple [BA89]). On voit donc à travers ces travaux que le choix de l'échelle est primordial.

Il existe deux sortes de filtres, les filtres linéaires et les non-linéaires. L'avantage d'un filtre linéaire est qu'il est caractérisé par sa réponse impulsionnelle. L'opération de filtrage se traduit alors de manière analytique par un produit de convolution, dans le cas discret :

$$I_f = I * f_\alpha = \sum_{t=-K/2}^{+K/2} I(x-t) f_\alpha(t)$$

où K est la largeur du filtre. Le schéma de fonctionnement d'un détecteur de contour peut donc être représenté par la figure 6.6. Le principal inconvénient des filtres linéaires est que la réduction du bruit entraîne un étalement des transitions du signal. Les filtres non-linéaires n'ont pas cet inconvénient, par contre ils introduisent généralement des modifications irréversibles de l'image. De plus pour la plupart d'entre eux, les pixels ne contribuent pas de la même façon au lissage. Cette contribution dépendra de sa position spatiale dans l'image.

On remarquera que sur le schéma de la figure 6.6 réside une étape supplémentaire dont nous n'avons pas encore parlé : la correction. En fait cette étape n'est pas toujours présente. En pratique elle s'exprimera par un dernier filtrage de l'image. En fait elle consiste à essayer d'augmenter une dernière fois le rapport signal/bruit en éliminant si possible les faux contours détectés. On pourra également essayer de supprimer les « trous » dans les contours par un algorithme de fermeture des contours. Ce dernier type de correction n'est pas couramment utilisé alors que le filtrage de l'image lui est systématiquement fait. Ainsi par exemple dans le cas d'un détecteur basé sur un opérateur du premier ordre, on peut le faire d'une

manière simple en supprimant tous les contours dont le module du gradient est inférieur à une certaine valeur. Un seuillage plus performant est également utilisé : le seuillage par hystérésis. Il consiste à garder tous les contours au-dessus d'un seuil haut, et tous ceux dont la valeur du module est supérieure à un seuil bas tout en étant connexes[6] à un contour dont la valeur du gradient est supérieure au seuil haut. C'est ce type de seuillage que nous utiliserons. Le sujet de cette thèse n'étant pas centré sur la détection de contours, nous avons choisi de travailler à partir d'un type particulier de détecteurs à cause des avantages qu'ils présentent (voir section 6.1.3) : ceux établis à partir d'un filtre linéaire séparable.

6.1.3 les filtres linéaires séparables

Nous avons déjà présenté les filtres linéaires ci-avant. Pour un filtre à deux variables x et y, la propriété de séparabilité indique que l'on peut décomposer le filtre en produit de deux filtres suivant chaque direction x et y, indépendantes l'une de l'autre. Ainsi le filtre f pourra s'écrire $f(x, y) = f_x(x)f_y(y)$. Les avantages de ce type de filtres sont multiples :

- réduction du temps de calcul : une convolution par un filtre de largeur K s'effectuait en $O(K^2)$ opérations, alors que pour un filtre séparable elle s'effectue en $O(K)$ opérations ;
- généralisation immédiate à une dimension quelconque ;

De plus, certains, comme les filtres exponentiels, permettent une implantation récursive de l'opération de filtrage[7].

Il faut faire attention avec les filtres séparables car ils sont souvent anisotropiques[8] selon les directions x et y. En fait le filtre gaussien est l'un des rares filtres linéaires séparables isotropes. C'est à partir de l'approximation proposée par R, Deriche (voir section 6.2.2.3) que nous avons mis au point notre détecteur inter-pixel.

[6] Ici connexe s'entend par suite de pixels deux à deux adjacents, c'est à dire ayant un coté ou un sommet en commun.

[7] Le filtrage récursif est une technique bien connue en traitement du signal. Pour plus d'information le lecteur intéressé pourra se référer à [HM93, Der87] par exemple.

[8] Anisotrope : se dit d'un corps, d'un objet dont les propriétés diffèrent suivant la direction considérées.

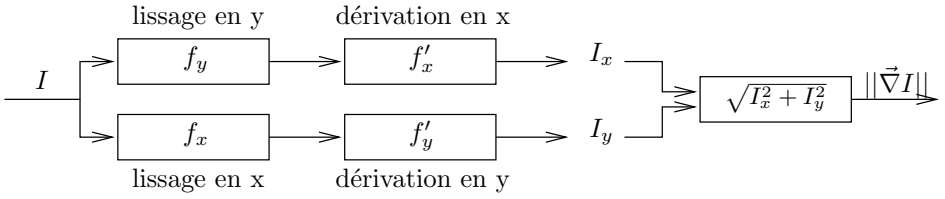


FIG. 6.7 – Schéma général du calcul des valeurs du gradient des points d'une image.

Regardons maintenant de manière analytique que devient notre schéma de fonctionnement des détecteurs de contours pour les détecteurs à base de filtre linéaire séparable, et plus particulièrement les étapes de filtrage et de différentiation. Nous rappelons que l'étape de filtrage consiste à convoluer l'image par notre fonction f de filtrage. Dans le cas qui nous intéresse nous obtenons :

$$I_f = I * f(x,y) = I * (f_x(x) f_y(y)) = I * f_x(x) * f_y(y)$$

I étant l'image initiale, I_f l'image filtrée et f_x, f_y les réponses impulsionnelles du filtre f selon les directions x et y. Pour réaliser l'étape de différentiation, il suffit de calculer les dérivées partielles selon x et y de l'expression précédente :

$$\frac{\partial I}{\partial x}(x,y) = \frac{\partial}{\partial x}\Big(I * f_x(x) * f_y(y)\Big) = I * f'_x(x) * f_y(y)$$
$$\frac{\partial I}{\partial y}(x,y) = \frac{\partial}{\partial y}\Big(I * f_x(x) * f_y(y)\Big) = I * f_x(x) * f'_y(y)$$

Dans le cas de filtres linéaires, l'opération de convolution étant commutative on pourra retenir comme schéma général du calcul des valeurs du gradient d'une image, celui présenté à la figure 6.7. Ce schéma de calcul nous permet donc d'obtenir la valeur du module du gradient en chaque point de l'image. Nous pouvons également déterminer en chaque point à l'aide des valeurs des dérivées partielles qui ont été calculées, la direction du vecteur gradient, c'est à dire la normale à la direction du contour. Il ne faut pas oublier que l'utilisation d'un opérateur du premier ordre nécessite une étape supplémentatire de suppression des non maxima locaux du gradient afin d'avoir des contours d'épaisseur unité. En chaque point

nous regardons les deux voisins les plus proches dans la direction du gradient[9], le point considéré est marqué comme point de contour potentiel si la valeur du module du gradient en ce point est supérieure à celle de ces deux voisins. Il ne reste alors plus qu'à appliquer un seuillage par hystérésis pour réaliser l'étape de correction. Les filtres présentés à la section 6.2.2 pourront être implantés suivant ce schéma.

Afin de familiariser le lecteur aux techniques de détection de contour, nous présentons maintenant un bref aperçu de certains détecteurs parmi les plus connus ou utilisés.

6.2 Exemples de détecteurs classiques.

Les travaux en détections de contours étant nombreux, il n'est pas possible dans le cadre de cette thèse de tous les recenser. Nous présenterons donc seulement, dans un premier temps les premiers détecteurs de contours apparus dans les années soixante, puis ensuite les détecteurs «modernes» fonctionnant sur le schéma précédent, notamment les plus célèbres, et les plus utilisés : le filtre de Shen-Castan, puis respectivement Canny et Deriche.

6.2.1 première approche

Les filtres présentés ici sont ce que l'on pourrait appeler des filtres informels. Ils ont tous pour but de détecter des contours de type marche et sont tous basés sur le même principe : une approximation des dérivées partielles. En effet considérons l'approximation suivante :

$$\frac{\partial I(x,y)}{\partial x} = I(x,y) - I(x-1,y) \quad et \quad \frac{\partial I(x,y)}{\partial y} = I(x,y) - I(x,y-1)$$

On obtient alors les vecteurs suivants :

$$G_x = [1\ \text{-}1] \quad et \quad G_y = \begin{bmatrix} 1 \\ \text{-}1 \end{bmatrix}$$

À partir de là L.G. Roberts [Rob65] fut le premier à proposer des masques de

[9]En pratique on approximera la direction à l'une des huit possibles dans un quadrillage carré, c'est à dire 0, 45, 90, 135, 180, 225, 270, 315 et 360 degrés.

convolution pour approximer le calcul du module du gradient par $G = \sqrt{G_x^2 + G_x^2}$ où G_x et G_y sont les valeurs calculées à l'aide des masques de convolution suivants :

$$G_x = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \quad et \quad G_y = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

Sur le même principe, J. Prewitt [Pre70] a proposé des masques de taille impaire afin de ramener les valeurs de gradients calculés sur les pixels, facilitant ainsi le traitement :

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad et \quad G_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

On citera également I.E. Sobel [Sob70] qui proposa les masques :

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad et \quad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

L'opérateur suivant [BB82] permet d'approximer le laplacien :

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Sur le même modèle il existe de nombreux autres détecteurs avec notamment des masques de convolution plus grands. En effet ce type de détecteur est très dépendant de la taille du masque ; l'augmenter signifie souvent augmenter la qualité des résultats obtenus car le résultat est alors moins dépendant du bruit. Mais cela revient aussi à augmenter le nombre d'opérations à effectuer et donc augmenter les temps de calcul. De plus tous ces détecteurs se limitent à une phase de différentiation et sont très sensibles au bruit. C'est pourquoi une étape de lissage a été introduite. Les filtres que nous présentons dans la section suivante procèdent tous suivant le schéma élaboré à la section 6.1.2.

6.2.2 filtres optimaux de lissage et de dérivation

Les détecteurs que nous présentons ici sont dits optimaux car s'appuyant sur le modèle de contour de type marche, ils cherchent à obtenir les meilleurs résultats

possibles : partant du principe que l'étape de lissage a entraîné une perte d'information (voir section 6.1.2) il faut essayer de minimiser cette perte. L'idée est donc de déterminer des critères de performances et essayer de définir le détecteur optimisant ces critères. Cette approche et cette modélisation est dûe à J. Canny [Can86].

6.2.2.1 Canny [Can86]

Pour J. Canny, les critères à optimiser sont l'exhaustivité de la détection et l'exactitude de la localisation. Il a montré [Can86] que le filtre 1D répondant à ces exigences est le résultat d'une combinaison linéaire de quatre exponentielles. Il a utilisé finalement la dérivée première d'une gaussienne qui est une bonne approximation de son filtre. L'implantation 2D est réalisée à l'aide d'un masque de convolution. L'intérêt de ces travaux réside dans le fait que Canny a proposé des critères d'évaluation des détecteurs à partir d'un modèle théorique de contour.

6.2.2.2 Shen-Castan [SC86]

Pour J. Shen et S. Castan minimiser ces pertes revient à minimiser l'énergie du bruit et maximiser l'énergie du signal. De plus comme on va procéder à une étape de différentiation, il faut minimiser le bruit dans la dérivée première du filtre. La différence de leur modèle par rapport à celui de J. canny, c'est qu'ils effectuent une combinaison différente de ses critèresde. Ils obtiennent le filtre :

$$s(x) = c.e^{-\alpha|x|} \tag{6.1}$$

où α est l'échelle du filtre[10] et c est une constante de normalisation calculée pour avoir un maximum de la réponse (soit une réponse égale à 1) en $x = 0$, on a alors (en discret)

$$\sum_{-\infty}^{+\infty} s(n) = 1$$

Ce qui nous donne :

$$c = \frac{1 - e^{-\alpha}}{1 + e^{-\alpha}}$$

[10]Plus alpha est petit plus l'échelle de filtrage est grande, c'est à dire plus le lissage est fort.

et pour le filtre de dérivation :

$$s'(x) = \begin{cases} d.e^{-\alpha|x|} & si\ x \geqslant 0 \\ -d.e^{-\alpha|x|} & si\ x \leqslant 0 \end{cases} \tag{6.2}$$

6.2.2.3 Deriche [Der87]

R. Deriche propose une solution exacte au modèle proposé par J. Canny ainsi qu'une implantation. En rajoutant des contraintes il a obtenu un filtre plus performant au sens des critères de J. Canny. Pour le calcul des critères de performance du filtre de Deriche le lecteur intéressé pourra se reporter à [HM93]. Nous nous contenterons ici de donner les équations du filtre de lissage et de dérivation :

$$d(x) = C.(\alpha|x|+1)e^{-\alpha|x|} \quad avec \quad C = \frac{(1-e^{-\alpha})^2}{1+2\alpha e^{-\alpha}-e^{-2\alpha}} \tag{6.3}$$

$$d'(x) = -C'.xe^{-\alpha|x|} \quad avec \quad C' = \frac{(1-exp{-}\alpha)^2}{e^{-\alpha}} \tag{6.4}$$

Le lecteur pourra également trouver à l'Annexe A les équations de récurrence ainsi que tous les coefficients.

C'est à partir de ce filtre, qui possède de bonnes performances, que nous avons mis au point notre détecteur de contour interpixel.

6.3 Un détecteur de contours interpixel.

Les détecteurs que nous avons présentés jusqu'à présent, excepté celui de L.G. Roberts, localisent tous les points de contour sur les pixels. Nous allons présenter dans cette section un détecteur interpixel, basé sur le filtre de Deriche, localisant les contours entre les pixels, c'est à dire sur les lignels et les pointels, à la différence de L.G. Roberts dont le détecteur localise les contours de manière implicite uniquement sur les pointels.

6.3.1 présentation pour un filtre 1D

Si l'on examine un contour de type marche, il est clair sur la figure 6.8 que la frontière se situe entre les pixels et non pas sur les pixels de l'une ou l'autre des régions. On remarquera par contre que les valeurs du signal discrétisé sont

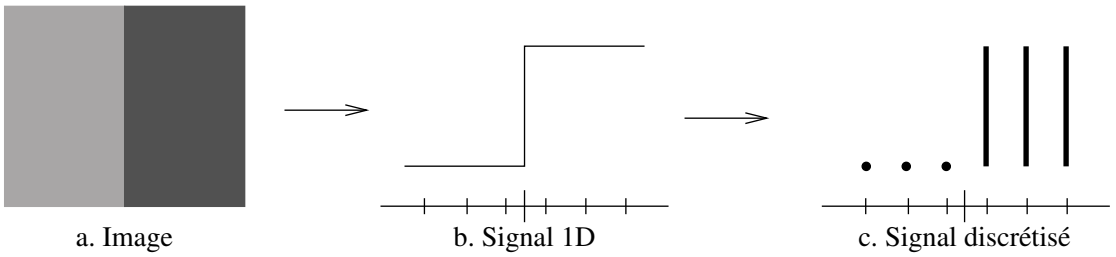


FIG. 6.8 – Visualisation d'un contour de type marche.

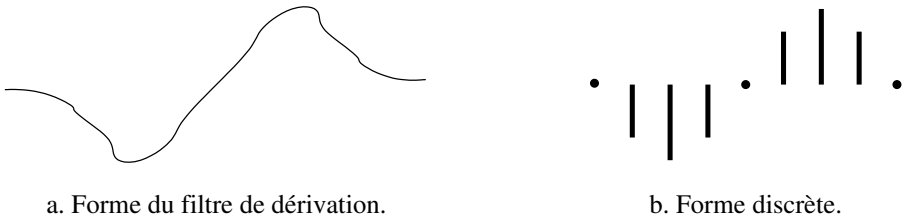
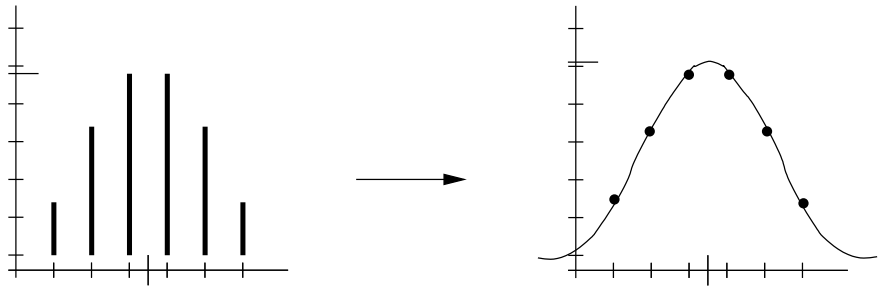


FIG. 6.9 – Schéma du filtre de dérivation de Deriche.

bien sûr elles-aussi localisées sur les pixels. Aucune d'elles n'est donc positionnée exactement sur le contour. Que donne alors un détecteur du type de celui de R. Deriche [Der87] ? On peut voir sur la figure 6.9 la forme du filtre de dérivation de R. Deriche. Si l'on applique ce filtre, ce qui revient dans le cas discret à faire une convolution de l'image par le filtre de dérivation, on obtient alors deux valeurs maximales pour la réponse discrète à ce filtre. (voir figure 6.10.a). La suppression des non maxima locaux va donc entrainer, soit la détection de deux contours, soit aucune détection[11]. De plus l'interpolation de cette réponse par une spline (figure 6.10.b) montre bien que le maximum peut se situer à l'interpixel.

La bonne localisation du contour est très importante pour de plus en plus d'applications. Ainsi, par exemple, pour des applications de reconstruction 3D, de

[11]Nous rapelons que la suppression des non maxima locaux consiste à ne garder comme point de contour que les valeurs supérieures à ses deux voisines dans la direction du gradient. Ici les deux valeurs voisines sont la précédente et la suivante. En fontion de l'opérateur de comparaison choisi (supérieur strictement — supérieur ou égal) on gardera soit les deux valeurs, soit aucune.



a. Deux valeurs maximales en discret b. Une seule valeur maximal sur la spline

FIG. 6.10 – Réponse discrète au filtre de dérivation de la figure 6.9.

mesures de paramètres sur les objets détectés, la précision de la localisation du contour va nettement influencer la qualité des résultats. En outre dans le cadre d'une coopération régions-contours, il devient indispensable de localiser les points de contour à l'interpixel afin de pouvoir les faire coïncider avec les frontières entre les régions. L'application que nous donnons dans le chapitre 5 en est un exemple.

L'idée naturelle de notre détecteur interpixel est de « décaler » le signal du filtre du détecteur, ce qui revient à modifier sa phase. En fait nous allons le décaler de la valeur d'un demi-pixel. On verra ensuite que cette démarche pose des problèmes en 2D. On peut voir sur la figure 6.11 que nous trouvons bien alors un seul contour et, de plus, localisé à la position de l'interpixel. C'est le problème de l'adéquation de la phase du filtre au signal.

Les équations du filtre de lissage et de dérivation du filtre de R. Deriche sont alors :

$$d_{\frac{1}{2}}(x) = C_{\frac{1}{2}}.(1+\alpha|x-\frac{1}{2}|)e^{-\alpha|x-\frac{1}{2}|} \tag{6.5}$$

$$d'_{\frac{1}{2}}(x) = -C'_{\frac{1}{2}}\alpha^2|x-\frac{1}{2}|e^{-\alpha(x-\frac{1}{2})} \tag{6.6}$$

où $C_{\frac{1}{2}}$ et $C'_{\frac{1}{2}}$ sont des coefficients de normalisation :

$$\int_{-\infty}^{+\infty} C_{\frac{1}{2}} d_{\frac{1}{2}}(x).dx = 1 \quad soit \quad \sum_{-\infty}^{+\infty} C'_{\frac{1}{2}} d'_{\frac{1}{2}}(n) = 1$$

$$\int_{-\infty}^{+\infty} C'_{\frac{1}{2}} x d'_{\frac{1}{2}}(x).dx = -1 \quad soit \quad \sum_{-\infty}^{+\infty} C'_{\frac{1}{2}}(n-\frac{1}{2}) d'_{\frac{1}{2}}(n) = -1$$

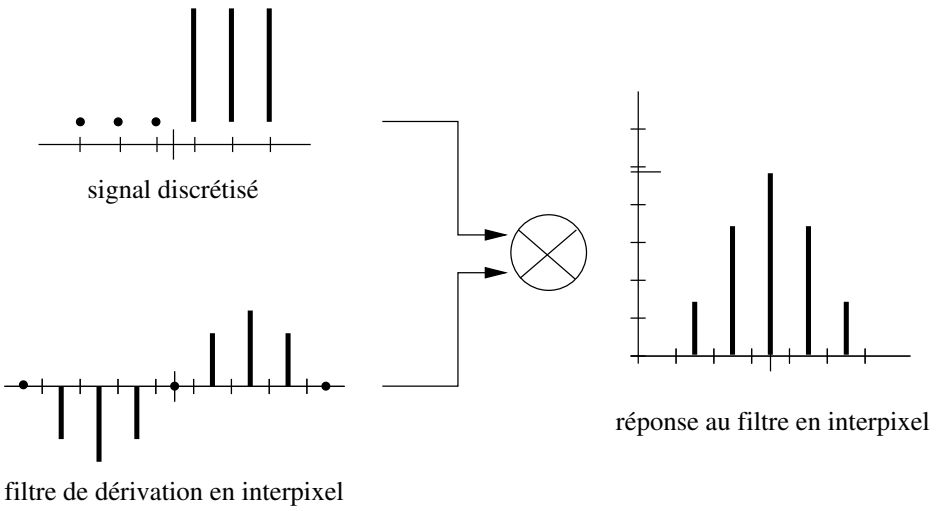


FIG. 6.11 – Réponse du filtre en interpixel.

Nous allons montrer maintenant comment mettre en œuvre un tel détecteur.

6.3.2 application à une image 2D

Nous avons présenté le fonctionnement du filtre interpixel sur un signal 1D. Or une image est un espace à deux dimensions, nous allons donc, dans un premier temps, montrer comment appliquer les résultats précédents en 2D. Puis nous donnerons les différentes équations nécessaires à l'implantation du filtre. Et enfin, nous montrerons quelques résultats en essayant de mettre en évidence sur des images réelles et synthétiques les différences entre la détection de contours interpixel et la méthode classique.

6.3.2.1 mise en œuvre

On se rappellera que la détection des points de contour nécessite de calculer une image gradient (voir schéma de fontionnement d'un détecteur à la figure 6.6). Cette image gradient nécessite de filtrer et dériver l'image suivant les directions x et y (voir figure 6.7). Or l'application d'un filtre (de lissage ou de dérivation) en interpixel sur une image représentée par une matrice à k colonnes et l lignes

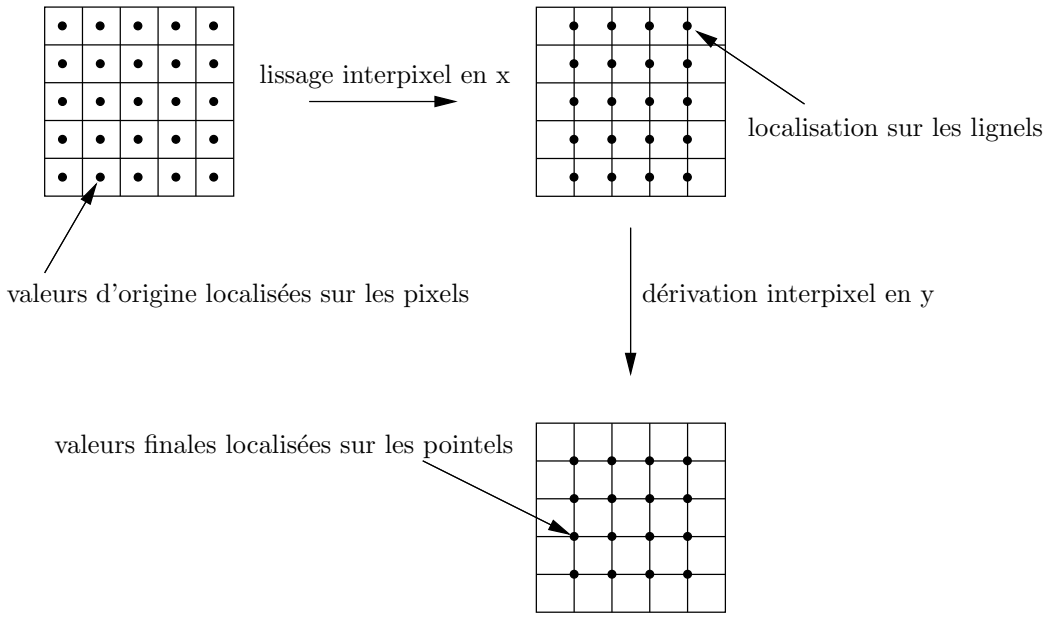


FIG. 6.12 – Décalage provoqué par un lissage et une dérivation en interpixel.

donne une image résultat de dimension $(k-1) \times l$ ou $k \times (l-1)$ selon la direction choisie[12]. Donc une fois calculée l'image lissée en x par exemple, l'application d'un filtre de dérivation interpixel donnerait des valeurs localisées sur les pointels et non pas sur les lignels (voir figure 6.12).

Il faut donc combiner les deux types de filtrage afin d'obtenir une image résultat où les valeurs sont bien des valeurs en interpixel localisées sur les lignels. On est de plus contraint à calculer séparément l'image contenant les valeurs des points de contour localisés sur les lignels verticaux et celle contenant les valeurs des points de contour localisés sur les lignels horizontaux.

Le schéma de calcul de l'image gradient pour les lignels verticaux devient donc celui présenté à la figure 6.13. On déduit aisément de la figure 6.13 le schéma de calcul de l'image gradient des points de contour localisés sur les lignels horizontaux. On notera que si l'on veut également les valeurs des points de contour localisés sur les pointels il suffit de faire un lissage et une dérivation en interpixel.

[12]Une ligne de n pixels possède $(n-1)$ interpixels (voir exemple de la figure 6.11)

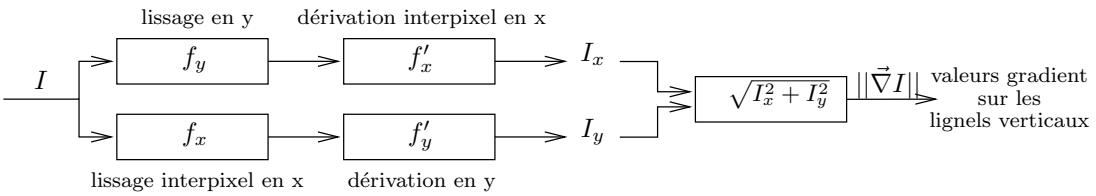


FIG. 6.13 – Schéma de calcul de l'image gradient pour une localisation des points de contours sur les lignels verticaux.

Dorénavant nous parlerons de détection *interpixel* lorsqu'il s'agira juste décaler la phase d'un filtre, obtenant ainsi les contours sur les lignels, et de subpixel lorsque nous intégrons toutes les informations, c'est à dire les lignels, pointels et également les pixels. Ainsi nous avons une généralisation immédiate de notre détecteur interpixel a un détecteur subpixel en précision demi-pixel.

6.3.2.2 résultats analytiques

Maintenant que nous avons présenté le schéma d'implantation du détecteur interpixel, nous allons donner les équations permettant de le mettre en œuvre. On trouvera dans l'annexe A les équations de récurrence du filtre de Deriche classique d'après [HM93].

Rappelons tout d'abord que le filtre de Deriche est un filtre séparable, récursif. Le résultat du filtrage sera donc la somme de deux équations de récurence. Soit la formule générale du filtre de lissage interpixel donnée par l'équation (6.5), on obtient les équations suivantes[13] :

$$\begin{aligned} y^+(n) &= by^+(n-1) + cy^+(n-2) + C_0' x(n-1) + a C_0' x(n-2) \quad (6.7) \\ &\quad pour\ n = 1, ..., N \\ y^-(n) &= by^-(n+1) + cy^-(n+2) + C_0' x(n) + a C_0' x(n+1) \quad (6.8) \\ &\quad pour\ n = N, ..., 1 \\ y(n) &= y^+(n) + y^-(n) \quad (6.9) \\ &\quad pour\ n = 2, ..., N-1 \end{aligned}$$

[13]Les valeurs en dehors des bornes sont mises à 0

avec

$$C_0 = \frac{(-1+e^{\alpha})^2}{e^{\frac{\alpha}{2}}\,(-2+\alpha+2\,e^{\alpha}+\alpha\,e^{\alpha})} \;\;;\;\; C_0' = C_0\,e^{-\alpha}(\delta+\gamma) \;\text{ et}$$

$$a = \frac{\alpha-2}{\alpha+2}e^{-\alpha} \;\;;\;\; b = 2e^{-\alpha} \;\;;\;\; c = -e^{-2\alpha} \;\;;\;\; \delta = \alpha e^{\frac{\alpha}{2}} \;\;;\;\; \gamma = (1-\frac{\alpha}{2})e^{\frac{\alpha}{2}}$$

Le filtre de dérivation en interpixel, dont l'équation 6.6 donne la formule générale, sera obtenu par les équations de récurence[13] :

$$y^+(n) = b\,y^+(n-1) + c\,y^+(n-2) + C_1'\,x(n-1) + a\,C_1'\,x(n-2) \quad (6.10)$$
$$\textit{pour } n = 2,...,N$$
$$y^-(n) = b\,y^-(n+1) + c\,y^-(n+2) - C_1'\,x(n) - a\,C_1'\,x(n+1) \quad (6.11)$$
$$\textit{pour } n = N-1,...,1$$
$$y(n) = y^+(n) + y^-(n) \quad (6.12)$$
$$\textit{pour } n = 2,...,N-1$$

avec

$$C_1 = \frac{2\,(-1+e^{\alpha})^3}{\alpha^2\,e^{\frac{\alpha}{2}}\,(1+6\,e^{\alpha}+e^{2\,\alpha})} \;\;;\;\; C_1' = C_1\,\frac{\alpha^2}{2}e^{-\frac{\alpha}{2}} \;\text{ et}$$

$$a = \frac{\alpha-2}{\alpha+2}e^{-\alpha} \;\;;\;\; b = 2e^{-\alpha} \;\;;\;\; c = -e^{-2\alpha}$$

Remarque : L'annexe B donne le détail des calculs pour l'obtention des équations de récurrence (6.7), (6.8), (6.10), (6.11).

6.3.2.3 présentation de quelques résultats

Nous présentons dans un premier temps une étude comparée de la précision de notre détecteur en précision demi-pixel sur un modèle de contour circulaire. La méthode de génération de l'image de synthèse du disque, ainsi que des exemples d'images obtenues par cette méthode sont données à l'annexe D. Dans un deuxième temps nous présentons une série d'images résultats.

Nous donnons ici une série de courbes présentant l'erreur, en pourcentage, trouvée sur la surface du cercle[14]. Chaque courbe est relative à un cercle d'un rayon donné, et donne les résultats pour le détecteur en précision pixel et interpixel en fonction du bruit blanc gaussien introduit dans l'image. L'amplitude du signal est de 72 (différence des niveaux de gris entre le fond et le cercle). La quantité de bruit est donnée en fonction de sa variance σ.

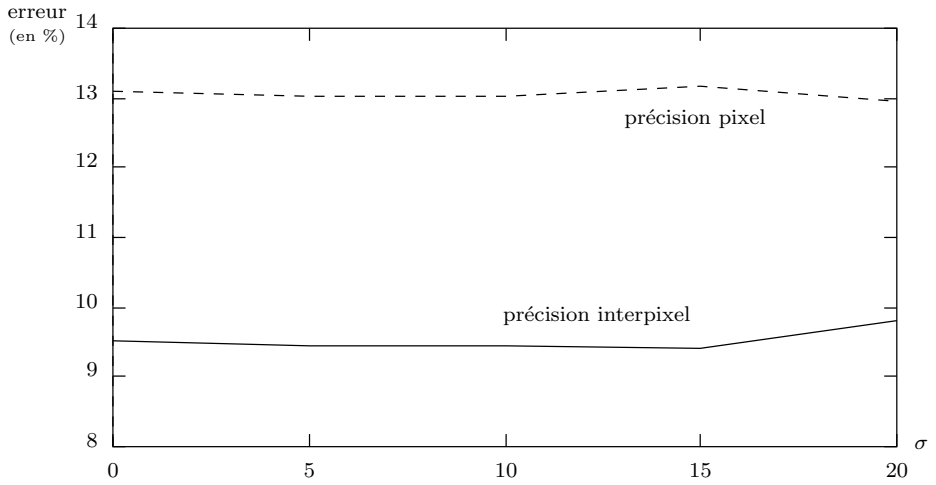


FIG. 6.14 – Cercle de rayon 20

On remarque que les deux détecteurs, en précision pixel et demi-pixel, réagissent de la même manière aux variations de la variance du bruit. Par contre on obtient systématiquement une meilleure précision en interpixel qu'en pixel. Le contour étant celui d'un modèle de type marche, cela conforte l'idée que pour un tel modèle de contour il faut travailler en interpixel. On remarquera également que plus le cercle est grand, plus l'écart entre les deux détecteurs se réduit. En effet la pente du contour a une amplitude de l'ordre de 1 à 3 pixels, donc plus le rayon est grand moins les variations de localisation du contour à l'intérieur de cette amplitude influencent le calcul de la surface.

Nous avons comparé notre détecteur de contour en précision demi-pixel avec le détecteur de contour de Deriche sur des images synthétiques contenant des bandes noires et blanches. On pourra remarquer sur la figure 6.18 que pour des bandes

[14]Le type de cercle utilisé correspond à celui de rayon 40 donné en exemple à l'annexe D.

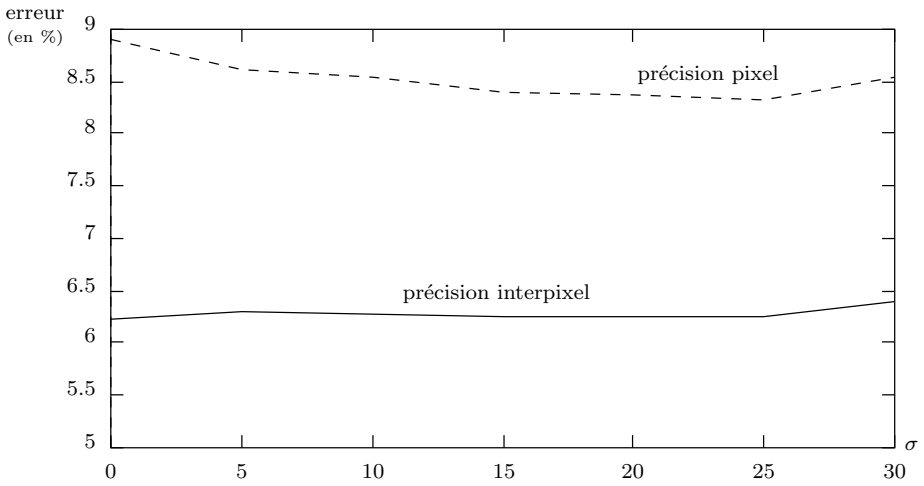


FIG. 6.15 – Cercle de rayon 30

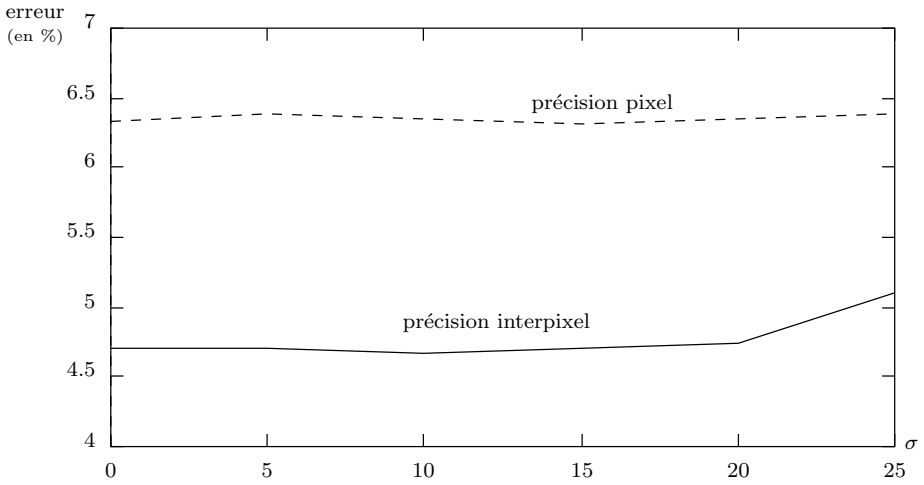


FIG. 6.16 – Cercle de rayon 40

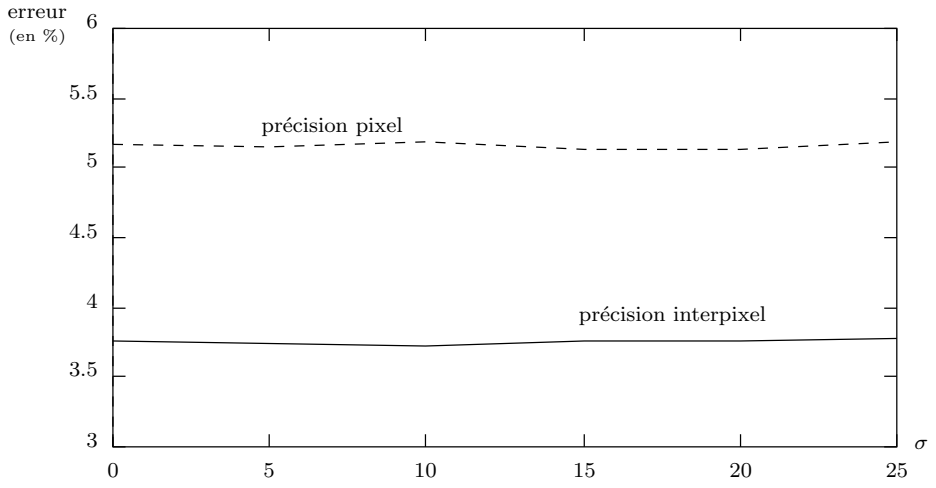


FIG. 6.17 – Cercle de rayon 50

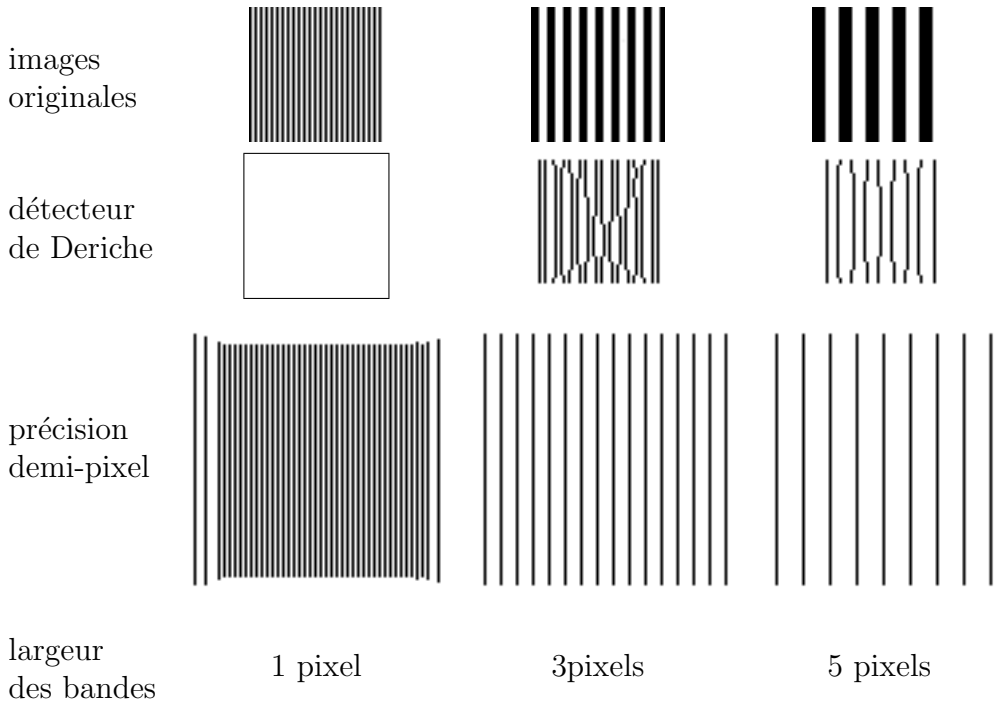
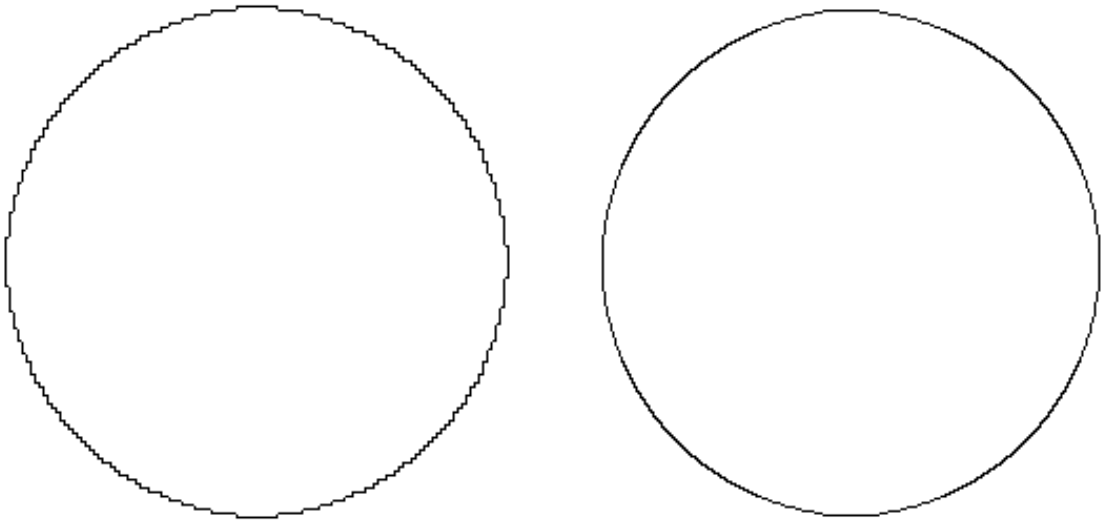


FIG. 6.18 – comparaison entre le détecteur de Deriche et le détecteur en précision demi-pixel.



précision pixel précision interpixel

FIG. 6.19 – Contours d'un cercle de rayon 60.

de 1 pixel de large le détecteur de Deriche ne détecte rien, par contre en précision demi-pixel les contours sont trouvés. Ceci confirme les hypothèses théoriques que nous avions faites lors de la présentation du filtre 1D (voir section 6.3.1). Sur les autres images où les bandes font trois et cinq pixels de large, le détecteur classique est fortement perturbé par la présence de contours proches, alors que le détecteur en précision demi-pixel ne semble pas affecté. On remarquera que les images en précision demi-pixel sont quatre fois plus grande, en effet le nombre de lignels, pointels et pixels de contour est quatre fois supérieur à celui des points de contour localisés sur les pixels.

Les images que nous présentons ci-dessous ont toutes été obtenues avec un coefficient de lissage $\alpha = 1$ et, sauf spécification contraire, un filtrage par hystérésis a été appliqué. Les images originales peuvent être vues à l'annexe E.

La figure 6.19 présente le contour d'un cercle en précision pixel et interpixel. On voit sur ces images, ramenées à la même échelle, que la détection interpixel est plus précise et donne un cercle mieux dessiné. Remarquons que ceci paraît normal puisque, rappelons-le, une image en précision demi-pixel est quatre fois plus grande.

avec filtrage par hystérésis sans filtrage par hystérésis

FIG. 6.20 – Carte des lignels de contour horizontaux et verticaux sur une image scanner d'un crâne.

La figure 6.20 montre deux cartes de lignels de contour. Ici les pointels de contour n'ont pas été représentés. On obtient une image déformée car à chaque ligne de pixels de l'image originale correspondent deux lignes de l'image des lignels, une pour les lignels verticaux, une autre pour les lignels horizontaux (ceux de dessous par exemple). La deuxième image est la même que la première mais sans filtrage par hystérésis.

Enfin sur la figure 6.21, on peut voir la même image (image test couramment utilisée en analyse d'images afin que le lecteur puisse faire des comparaisons avec

d'autres méthodes), à laquelle on a appliqué notre détecteur interpixel[15] et le détecteur « classique » de Deriche.

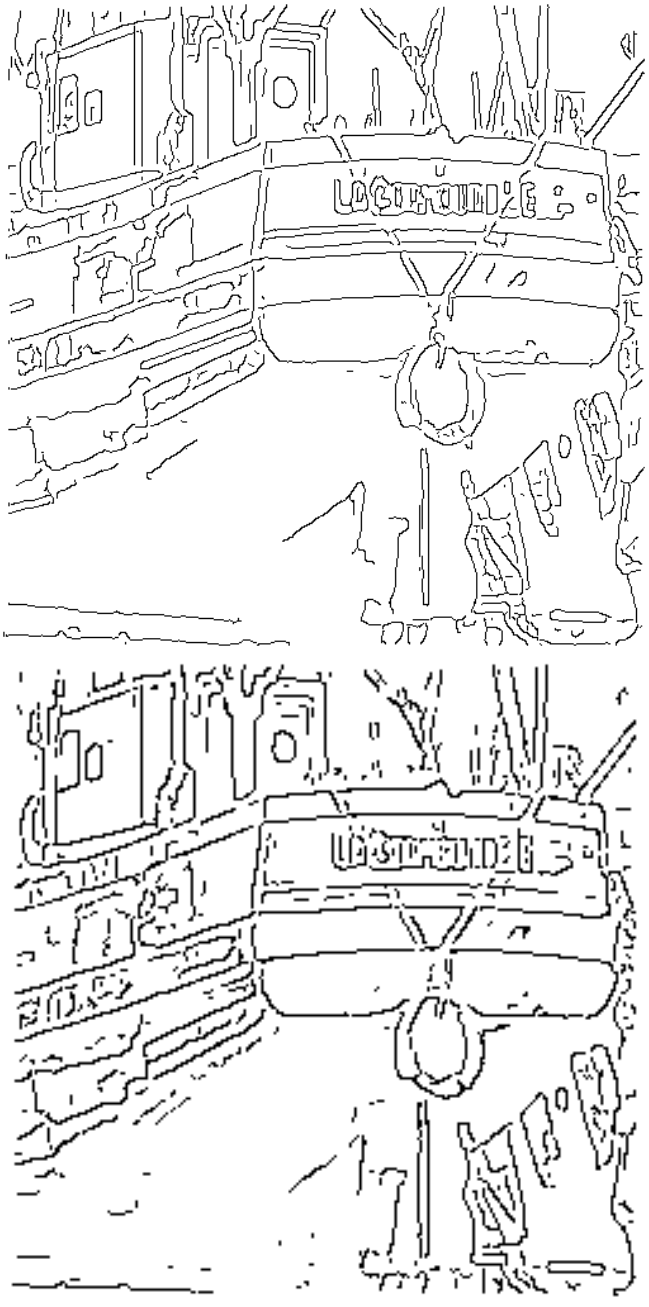
6.4 Conclusion et perspectives.

L'objectif cherché a été atteint : l'obtention d'un détecteur interpixel avec en plus celui en précision demi-pixel qui en a découlé. Nous montrons ainsi que l'approche interpixel ne se restreint pas à des algorithmes de segmentation en régions. De plus les résultats obtenus avec ce détecteur sont meilleurs du point de vue de la localisation. Quant à l'exactitude de la localisation, il faudrait faire une étude du point de vue des critères de J. Canny.

Sur une image donnée, les détecteurs interpixel et demi-pixel détectent mieux les détails d'une image. Globalement les résultats ne semble donc pas « à l'oeil » meilleurs. L'approche néanmoins très prometteuse. Nous poursuivons d'ailleurs avec P. Montesinos nos recherches en ce domaine, notamment dans la mise au point d'un détecteur de contour en précision subpixel. Contrairement aux autres méthodes travaillant en subpixel [Tab94, KVW94, LE95], nous comptons proposer un algorithme permettant de travailler à une précision fixée, voulue et choisie par l'utilisateur.

L'intérêt d'un détecteur interpixel, pour nous, réside également dans le fait que nous allons pouvoir l'intégrer dans un processus complet d'analyse en interpixel. Il faut notamment à court terme, utiliser ce détecteur avec nos algorithmes de segmentation présentés au chapitre 5, afin d'avoir une coopération régions-contours plus performante. Une première approche immédiate a été réalisée, elle consiste simplement, lors du test de validité de la fusion de deux régions, à vérifier s'il n'y a pas de lignel de contour entre les deux pixels considérés – pour une présentation plus détaillée de la méthode se reporter au chapitre en question –, un exemple de résultat sur une vue aérienne est donné à la section 5.5. Mais il faudrait améliorer cette méthode en prenant en compte tous les lignels de contours sur la frontière entre les deux régions. Une étude est en cours à ce sujet.

[15] Il ne faut pas se fier à la finesse des traits sur l'image des contours en interpixel. En effet une image en interpixel est normalement quatre fois plus grande (il faut afficher non seulement les pixels mais aussi les lignels et les pointels). La finesse des traits est due à la réduction de l'image à la même taille que l'originale.



interpixel

pixel

FIG. 6.21 – Détection des contours du bateau *Cornouaille* en précision interpixel et pixel.

Chapitre 7

Conclusion et perspectives

Nous disposons actuellement d'un modèle cohérent de représentation sur lequel nous avons bâti des algorithmes efficaces. Notre approche est originale à double titre. En effet aussi bien notre modèle formel d'image, *la topologie-étoile*, que nos algorithmes de segmentation en régions s'appuyant sur un problème classique en algorithmique combinatoire, ou que notre représentation, *le graphe des frontières*, tous ces résultats s'inscrivent dans le cadre d'une approche combinatoire de la problématique de l'analyse bas-niveau d'images. D'autre part, contrairement aux méthodes habituelles, nous ne considérons pas l'image comme seulement un ensemble de pixels. Nous pensons plutôt qu'il est important de tenir compte des éléments de bord des pixels, les *lignels*, *pointels* et autres *surfels*. Cette approche *interpixel* nous a permis de proposer une topologie dérivée de celle de $\mathbb{R}^n$ et adaptée à l'analyse d'images. Ce travail étant une thèse, nous espérons qu'il suscitera des discussions et débats qui nous permettront de valider notre approche. Les algorithmes de segmentation que nous présentons suivent cette démarche puisque les frontières des régions définies par de tels algorithmes sont naturellement situées à l'interpixel. Nous montrons également que l'on peut toujours appliquer les méthodes de filtrage et de détections de points de contour en adaptant le détecteur de R. Deriche [Der87] en précision interpixel.

Nous ne prétendons pas avoir trouvé une méthode ou des algorithmes universels de segmentation. Mais les différents résultats présentés dans cette thèse peuvent s'interconnecter facilement. Ainsi nos algorithmes de segmentations favorisent la coopération régions-contours, la structure de données utilisée est utile à l'algo-

rithme d'extraction du graphe, tout cet ensemble s'appuyant sur un même modèle formel. Les principaux résultats obtenus dans cette thèse sont donc :

- un modèle topologique formel de modélisation des images : la *topologie-étoile* ;
- deux algorithmes très efficaces réalisant à la fois une segmentation en régions et un étiquetage des composantes connexes, permettant une coopération régions-contours : `ScanLine` et `MergeSquare` ;
- un détecteur de contour interpixel et subpixel à la précision $\frac{1}{2}$;
- une représentation combinatoire : *le graphe des frontières*.

Ces résultats ne sont pas une fin en eux-mêmes, au contraire ils nous ouvrent des perspectives intéressantes, à la fois pratiques et théoriques. Ainsi l'algorithme `ScanLine` est en cours d'adaptation pour une segmentation « à la volée » dans le cadre d'une application de contrôle à la qualité de fabrication du coton. Il faut d'autre part passer à la réalisation de certains algorithmes présentés ici, comme celui de la mise en correspondance de deux segmentations et celui de l'extraction du graphe des frontières. Il faut également continuer l'étude sur le graphe des frontières afin de découvrir tout le potentiel d'une telle représentation, notamment les facilités de manipulation que devraient nous apporter les applications α et σ.

À court terme il faudra également développer la coopération régions-contours de nos algorithmes, ceci afin d'intégrer la connaissance totale des contours interpixels, notamment la prise en compte de la présence de lignels de contour sur toute la frontière séparant les deux régions à fusionner. Il serait également intéressant de modifier la définition du graphe des frontières afin de prendre en compte totalement la topologie de l'image, c'est à dire en modélisant l'adjacence stricte (voir définition 3.30) et pas seulement les frontières entre les régions.

À plus long terme nous comptons nous orienter sur le traitement des images 3D. Ceci nous invite à développer les bases formelles avec les outils des complexes cellulaires, et notre représentation en s'appuyant toujours sur les hypercartes. Un problème qui nous intéresse déjà est celui de la définition d'une surface 3D. Nous avons également commencé à aborder le problème de la segmentation 3D en mettant au point, avec A. Serrano [Ser95], un algorithme d'extraction des surfaces régulières [RT71]. Enfin nous espérons un jour aboutir à une application de reconnaissance des formes utilisant l'algorithmique des graphes.

Annexe A

Équation de récurrence du filtre de Deriche classique.

Les équations données ici sont extraites de [HM93].

Le filtre de lissage de Deriche est un filtre séparable, récursif du second ordre. Soit N la largeur de l'image à filtrer[1], les équations de récurrence sont[2] :

$$\begin{aligned} y^+(n) &= a_0x(n) + a_1x(n-1) - b_1y^+(n-1) - b_2y^+(n-2) \quad (A.1)\\ &\quad pour\ n = 1, ..., N \\ y^-(n) &= a_2x(n+1) + a_3x(n+2) - b_1y^-(n+1) - b_2y^-(n+2) \quad (A.2)\\ &\quad pour\ n = N, ..., 1 \\ y(n) &= y^-(n) + y^+(n) \quad (A.3)\\ &\quad pour\ n = 1, ..., N \end{aligned}$$

avec :

$$a_0 = h\ ;\ a_1 = k(\alpha-1)e^{-\alpha}\ ;\ a_2 = k(\alpha+1)e^{-\alpha}\ ;\ a_3 = -ke^{-2\alpha}\ ;$$
$$b_1 = -2e^{-\alpha}\ ;\ b_2 = e^{-2\alpha}\ ;\ k = \frac{(1-e^{-\alpha})}{1+2\alpha e^{-\alpha}-e^{-2\alpha}}$$

Le filtre de dérivation de Deriche est un filtre séparable, récursif du second

[1] Ceci pour un filtrage en x, pour un filtrage en y on appliquera les même équations mais N représente alors la hauteur de l'image.

[2] Rappel : la formule générale a été donnée à la section 6.2.2.3 par l'équation 6.3.

ordre. Les équations de récurrence sont[3] :

$$\begin{aligned} y^+(n) &= ax(n-1) - b_1 y^+(n-1) - b_2 y^+(n-2) \quad (A.4) \\ &\quad pour\ n = 1, ..., N \\ y^-(n) &= ax(n+1) - b_1 y^-(n+1) - b_2 y^-(n+2) \quad (A.5) \\ &\quad pour\ n = N, ..., 1 \\ y(n) &= y^-(n) + y^+(n) \quad (A.6) \\ &\quad pour\ n = 1, ..., N \end{aligned}$$

avec :

$$a = ce^{-\alpha}\ ;\ b_1 = -2e^{-\alpha}\ ;\ b_2 = e^{-2\alpha}\ ;\ c = \frac{(1-e^{-\alpha})}{e^{-\alpha}}$$

[3]Rappel : la formule générale a été donnée à la section 6.2.2.3 par l'équation 6.4.

Annexe B

Calculs pour le filtre de Deriche en précision demi-pixel.

B.1 Quelques calculs préliminaires.

Pour $X \in [0,1[$

$$\sum_{n=0}^{\infty} X^n = \frac{1}{1-X}$$

donc :

$$\sum_{n=1}^{\infty} X^n = \frac{1}{1-X} - 1 = \frac{X}{1-X}$$

$$\sum_{n=0}^{\infty} nX^n = \frac{X}{(1-X)^2}$$

$$\sum_{n=0}^{\infty} n^2 X^n = \frac{X(X+1)}{(1-X)^3}$$

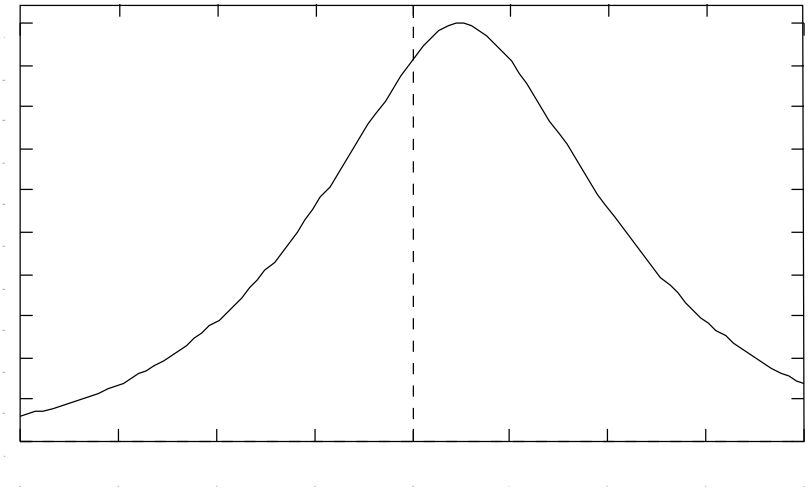


FIG. B.1 – Fonction de lissage $\frac{1}{2}$ pixel.

B.2 Définition des équations du filtres en précision interpixel

B.2.1 lissage

Définition continue de la fonction de lissage (voir fig B.1) :

$$f_{\frac{1}{2},0}(x) = \begin{cases} C_{\frac{1}{2},0}\,(1+\alpha(x-\frac{1}{2}))e^{-\alpha\left(x-\frac{1}{2}\right)} & \text{pour} \quad x > \frac{1}{2} \\ C_0\,(1-\alpha(x-\frac{1}{2}))e^{\alpha\left(x-\frac{1}{2}\right)} & \text{pour} \quad x \leqslant \frac{1}{2} \end{cases}$$

Dans le cas discret cette fonction s'ecrit :

$$f_0(n) = \begin{cases} C_0\,(1+\alpha(n-\frac{1}{2}))e^{-\alpha\left(n-\frac{1}{2}\right)} & \text{pour} \quad n > 0 \\ C_0\,(1-\alpha(n-\frac{1}{2}))e^{\alpha\left(n-\frac{1}{2}\right)} & \text{pour} \quad n \leqslant 0 \end{cases}$$

B.2.2 dérivation

Définition continue de la fonction de dérivation (voir fig B.2) :

$$f_1(x) = \begin{cases} -C_1\left(x-\frac{1}{2}\right)\,\alpha^2 e^{-\alpha(x-\frac{1}{2})} & \text{pour} \quad x > \frac{1}{2} \\ -C_1\left(x-\frac{1}{2}\right)\,\alpha^2 e^{\alpha(x-\frac{1}{2})} & \text{pour} \quad x \leqslant \frac{1}{2} \end{cases}$$

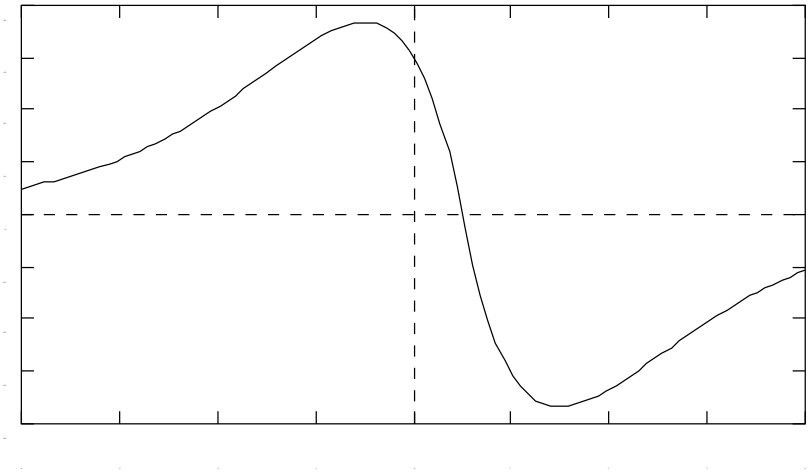


FIG. B.2 – Fonction de dérivation $\frac{1}{2}$ pixel.

Dans le cas discret cette fonction s'ecrit :

$$f_1(n) = \begin{cases} -C_1\left(n-\frac{1}{2}\right)\alpha^2 e^{-\alpha(n-\frac{1}{2})} & \text{pour} \quad n>0 \\ -C_1\left(n-\frac{1}{2}\right)\alpha^2 e^{\alpha(n-\frac{1}{2})} & \text{pour} \quad n\leqslant 0 \end{cases}$$

B.3 Coefficient de normalisation

B.3.1 lissage

Nous allons calculer ici le coefficient C_1

La fonction de lissage est normalisée de la manière suivante :

$$\sum_{n=-\infty}^{\infty} f_1(n) = 1$$

Soit :

$$C_0\left[\sum_{n=-\infty}^{0}(1-\alpha(n-\frac{1}{2}))e^{\alpha\left(n-\frac{1}{2}\right)}+\sum_{n=1}^{\infty}(1+\alpha(n-\frac{1}{2}))e^{-\alpha\left(n-\frac{1}{2}\right)}\right]=1$$

Pour des raisons de symétrie :

$$C_0 \sum_{n=1}^{\infty}(1+\alpha(n-\tfrac{1}{2}))e^{-\alpha\left(n-\frac{1}{2}\right)} = \tfrac{1}{2}$$

$$\Longleftrightarrow \quad C_0\, e^{\frac{\alpha}{2}} \left[(1-\tfrac{\alpha}{2})\sum_{n=1}^{\infty} e^{-\alpha n} + \alpha \sum_{n=1}^{\infty} n e^{-\alpha n}\right] = \tfrac{1}{2}$$

Posons : $X = e^{-\alpha}$, alors :

$$C_0\, e^{\frac{\alpha}{2}} \left[(1-\frac{\alpha}{2})\sum_{n=1}^{\infty} X^n + \alpha \sum_{n=1}^{\infty} n X^n\right] = \frac{1}{2}$$

Or :

$$\sum_{n=1}^{\infty} X^n = \frac{X}{(1-X)} = \frac{e^{-\alpha}}{(1-e^{-\alpha})} \quad \text{et} \quad \sum_{n=1}^{\infty} n X^n = \frac{X}{(1-X)^2} = \frac{e^{-\alpha}}{(1-e^{-\alpha})^2}$$

Donc :

$$C_0\, e^{\frac{\alpha}{2}} \left[(1-\frac{\alpha}{2})\frac{e^{-\alpha}}{(1-e^{-\alpha})} + \alpha \frac{e^{-\alpha}}{(1-e^{-\alpha})^2}\right] = \frac{1}{2}$$

Et finalement pour le coefficient de normalisation :

$$\boxed{C_0 = \frac{(-1+e^{\alpha})^2}{e^{\frac{\alpha}{2}}\,(-2+\alpha+2\,e^{\alpha}+\alpha\,e^{\alpha})}}$$

B.3.2 Dérivation.

Nous allons calculer ici le coefficient C_1

La fonction de dérivation est normalisée de manière à obtenir des des dérivées exactes pour des polynomes, soit

$$\sum_{n=-\infty}^{\infty}(k-(n+\tfrac{1}{2}))f_1(n)=1$$

$$\iff \sum_{n=-\infty}^{\infty}kf_1(n)-\sum_{n=-\infty}^{\infty}(n+\tfrac{1}{2})f_1(n)=1$$

$$\iff \sum_{n=-\infty}^{\infty}(n+\tfrac{1}{2})f_1(n)=-1$$

$$\iff C_1\,\alpha^2\sum_{n=-\infty}^{0}(n+\tfrac{1}{2})(n-\tfrac{1}{2})e^{\alpha(n-\frac{1}{2})} + C_1\,\alpha^2\sum_{n=1}^{\infty}(n+\tfrac{1}{2})(n-\tfrac{1}{2})e^{-\alpha(n-\frac{1}{2})}=1$$

$$\iff C_1\,\alpha^2e^{\frac{-\alpha}{2}}\sum_{n=-\infty}^{0}(n^2-\tfrac{1}{4})e^{\alpha n}+\ C_1\,\alpha^2e^{\frac{\alpha}{2}}\sum_{n=1}^{\infty}(n^2-\tfrac{1}{4})e^{-\alpha n}=1$$

$$\iff C_1\,\alpha^2e^{\frac{-\alpha}{2}}\sum_{n=0}^{\infty}(n^2-\tfrac{1}{4})e^{-\alpha n}+\ C_1\,\alpha^2e^{\frac{\alpha}{2}}\sum_{n=1}^{\infty}(n^2-\tfrac{1}{4})e^{-\alpha n}=1$$

$$\iff C_1\,\alpha^2e^{\frac{-\alpha}{2}}\left[\sum_{n=0}^{\infty}n^2e^{-\alpha n}-\tfrac{1}{4}\sum_{n=0}^{\infty}e^{-\alpha n}\right] + C_1\,\alpha^2e^{\frac{\alpha}{2}}\left[\sum_{n=1}^{\infty}n^2e^{-\alpha n}-\tfrac{1}{4}\sum_{n=1}^{\infty}e^{-\alpha n}\right]=1$$

Si on pose $X=e^{-\alpha}$ alors :

$$\iff C_1\,\alpha^2e^{\frac{-\alpha}{2}}\left[\tfrac{X(X+1)}{(1-X)^3}-\tfrac{1}{4}\tfrac{1}{1-X}\right]+\ C_1\,\alpha^2e^{\frac{\alpha}{2}}\left[\tfrac{X(X+1)}{(1-X)^3}-\tfrac{1}{4}\tfrac{X}{1-X}\right]=1$$

Nous trouvons alors pour C_1 :

$$\boxed{C_1\ =\ \frac{2\,(-1+e^{\alpha})^3}{\alpha^2\,e^{\frac{\alpha}{2}}\,(1+6\,e^{\alpha}+e^{2\,\alpha})}}$$

B.4 Équations de récurence

B.4.1 fonction de lissage

B.4.1.1 partie causale de la fonction

Soit $f_0^+(n)$ la partie causale de la fonction :

$$f_0^+(n) = \begin{cases} C_0\,(1+\alpha(n-\frac{1}{2}))e^{-\alpha\left(n-\frac{1}{2}\right)} & \text{pour} \quad n > 0 \\ 0 & \text{pour} \quad n \leqslant 0 \end{cases}$$

Calculons la transformée en **Z** $F_0^+(Z)$ de cette fonction :

$$F_0^+(Z) = \sum_{n=1}^{\infty} Z^{-n} f_0^+(n)$$

Soit :

$$F_0^+(Z) = C_0\,\gamma \sum_{n=1}^{\infty} X^n + C_0\,\delta \sum_{n=1}^{\infty} nX^n$$

avec :

$$X = e^{-\alpha}Z^{-1}\ ;\ \beta = (1-\frac{\alpha}{2})\ ;\ \gamma = \beta e^{\frac{\alpha}{2}}\ ;\ \delta = \alpha e^{\frac{\alpha}{2}}$$

Donc :

$$F_0^+(Z) = C_0\,\gamma\,\frac{X}{1-X} + C_0\,\delta\,\frac{X}{(1-X)^2}$$

$$\begin{aligned} \Longleftrightarrow \quad F_0^+(Z) &= C_0\,\tfrac{-\gamma+\delta\,e^{\alpha}Z+e^{\alpha}\,\gamma Z}{(-1+e^{\alpha}Z)^2} \\ &= C_0\,\tfrac{-\gamma+\delta\,e^{\alpha}Z+e^{\alpha}\,\gamma Z}{1-2e^{\alpha}Z+e^{2\alpha}Z^2} \\ &= C_0\,\tfrac{e^{\alpha}(\delta+\gamma)Z^{-1}-\gamma Z^{-2}}{e^{2\alpha}-2e^{\alpha}Z^{-1}+Z^{-2}} \\ &= C_0\,\tfrac{e^{-\alpha}(\delta+\gamma)Z^{-1}-\gamma\,e^{-2\alpha}Z^{-2}}{1-2e^{-\alpha}Z^{-1}+e^{-2\alpha}Z^{-2}} \end{aligned}$$

B.4.1.2 équation de récurence

$$\frac{Y^+(Z)}{X(Z)} \;=\; F^+(Z) \;=\; C_0\,\frac{e^{-\alpha}(\delta+\gamma)Z^{-1}-\gamma\,e^{-2\alpha}Z^{-2}}{1-2e^{-\alpha}Z^{-1}+e^{-2\alpha}Z^{-2}}$$

Alors :

$$\begin{aligned} & Y^+(Z) - 2e^{-\alpha} Z^{-1} Y^+(Z) + e^{-2\alpha} Z^{-2} Y^+(Z) \\ = \; & C_0\, e^{-\alpha}(\delta+\gamma) Z^{-1} X(Z) - C_0\, \gamma\, e^{-2\alpha} Z^{-2} X(Z) \end{aligned}$$

Et dans l'espace réel :

$$Y^+(n) = 2e^{-\alpha} Y^+(n-1) - e^{-2\alpha} Y^+(n-2) + C_0\, e^{-\alpha}(\delta+\gamma) X(n-1) - C_0\, \gamma\, e^{-2\alpha} X(n-2)$$

Si on pose :

$$C_0' = C_0\, e^{-\alpha}(\delta+\gamma) \;;\; a = \frac{\alpha-2}{\alpha+2} e^{-\alpha} \;;\; b = 2e^{-\alpha} \;;\; c = -e^{-2\alpha}$$

alors :

$$\boxed{Y^+(n) = bY^+(n-1) + cY^+(n-2) + C_0' X(n-1) + aC_0' X(n-2)}$$

B.4.1.3 partie anticausale de la fonction

Soit $f_0^-(n)$ la partie anticausale de la fonction :

$$f_0^-(n) = \begin{cases} 0 & \text{pour} \quad n > 0 \\ C_0\,(1 - \alpha(n - \frac{1}{2}))e^{\alpha\left(n-\frac{1}{2}\right)} & \text{pour} \quad n \leqslant 0 \end{cases}$$

Calculons la transformée en $\mathbf{Z}$ $F_0^-(Z)$ de cette fonction :

$$F_0^-(Z) = \sum_{n=-\infty}^{0} Z^{-n} f_0^-(n)$$

Soit :

$$F_0^-(Z) = C_0 \sum_{n=-\infty}^{0} \left(1 - \alpha(n - \frac{1}{2})\right) e^{\alpha\left(n-\frac{1}{2}\right)} Z^{-n}$$

$$
\begin{aligned}
\Longleftrightarrow \quad F_0^-(Z) &= C_0\,(1+\tfrac{\alpha}{2})e^{-\frac{\alpha}{2}} \sum_{n=0}^{\infty} Z^n e^{-\alpha\, n} + C_0\, e^{-\frac{\alpha}{2}} \alpha \sum_{n=0}^{\infty} n Z^n e^{-\alpha\, n} \\
&= C_0\, e^{\frac{\alpha}{2}}\, \tfrac{2e^{\alpha}+\alpha e^{\alpha}-2Z+\alpha Z}{2e^{2\alpha}-4e^{\alpha}\, Z+2Z^2} \\
&= C_0\, e^{\frac{\alpha}{2}}\, \tfrac{(1+\frac{\alpha}{2})e^{\alpha}-(1-\frac{\alpha}{2})Z}{e^{2\alpha}-2e^{\alpha}\, Z+Z^2} \\
&= C_0\, e^{\frac{\alpha}{2}}\, \tfrac{(1+\frac{\alpha}{2})e^{-\alpha}-(1-\frac{\alpha}{2})e^{-2\alpha}Z}{1-2e^{-\alpha}\, Z+e^{-2\alpha}Z^2} \\
&= C_0\, \tfrac{(1+\frac{\alpha}{2})e^{\frac{\alpha}{2}}e^{-\alpha}-(1-\frac{\alpha}{2})e^{\frac{\alpha}{2}}e^{-2\alpha}Z}{1-2e^{-\alpha}\, Z+e^{-2\alpha}Z^2}
\end{aligned}
$$

Soit :

$$F_0^-(Z) = C_0\, \frac{(\gamma\, +\, \delta)e^{-\alpha} -\, \gamma\, e^{-2\alpha} Z}{1 - 2e^{-\alpha} Z + e^{-2\alpha} Z^2}$$

Nous trouvons alors pour l'équation de récurence :

$$Y^-(n) \;=\; b\, Y^-(n+1) + c\, Y^-(n+2) + C_0^{'}\, X(n) + a\, C_0^{'}\, X(n+1)$$

B.4.1.4 expression complète du filtre

L'expression finale du filtre de lissage en précision demi pixel est la suivante :

$$
\begin{aligned}
(6.7) \quad Y^+(n) &= bY^+(n-1) + cY^+(n-2) + C_0^{'}\, X(n-1) + a\, C_0^{'}\, X(n-2) \\
(6.8) \quad Y^-(n) &= bY^-(n+1) + cY^-(n+2) + C_0^{'}\, X(n) + a\, C_0^{'}\, X(n+1) \\
(6.9) \quad Y(n) &= Y^+(n) + Y^-(n)
\end{aligned}
$$

B.4.2 filtre de dérivation

B.4.2.1 partie causale de la fonction

Soit $f_1^+(n)$ la partie causale de la fonction :

$$f_1^+(n) = \begin{cases} -C_1 \left(n - \frac{1}{2}\right) \alpha^2 e^{-\alpha(n-\frac{1}{2})} & \text{pour} \quad n > 0 \\ 0 & \text{pour} \quad n \leqslant 0 \end{cases}$$

Calculons la transformée en **Z** $F_1^+(Z)$ de cette fonction :

$$F_1^+(Z) \ = \ \sum_{n=1}^{\infty} Z^{-n} f_1^+(n)$$

Soit :

$$\begin{aligned} F_1^+(Z) &= -C_1\, \alpha^2 e^{\frac{\alpha}{2}} \left[\textstyle\sum_{n=1}^{\infty} n e^{-\alpha n} Z^{-n} - \frac{1}{2} \sum_{n=1}^{\infty} e^{-\alpha n} Z^{-n}\right] \\ \iff F_1^+(Z) &= -C_1\, \tfrac{\alpha^2}{2} e^{\frac{\alpha}{2}} \tfrac{e^{-\alpha} Z^{-1} + e^{-2\alpha} Z^{-2}}{1 - 2e^{-\alpha} Z^{-1} + e^{-2\alpha} Z^{-2}} \\ \iff Y^+(Z) &= 2e^{-\alpha} Z^{-1} Y^+(Z) - e^{-2\alpha} Z^{-2} Y^+(Z) \\ &\quad - C_1\, \tfrac{\alpha^2}{2} e^{\frac{-\alpha}{2}} Z^{-1} X(Z) \\ &\quad - C_1\, \tfrac{\alpha^2}{2} e^{\frac{\alpha}{2}} e^{-2\alpha} Z^{-2} X(Z) \end{aligned}$$

Posons :

$$b = 2e^{-\alpha} \ ; \ c = -e^{-2\alpha} \ ; \ d = e^{-\alpha} \ ; \ C_1' = C_1 \frac{\alpha^2}{2} e^{\frac{-\alpha}{2}}$$

L'équation de récurence pour la partie positive de la fonction de dérivation s'écrit alors :

$$Y^+(n) = b\, Y^+(n-1) + c\, Y^+(n-2) + C_1' X(n-1) + d\, C_1' X(n-2)$$

B.4.2.2 partie anticausale de la fonction

Soit $f_1^-(n)$ la partie anticausale de la fonction :

$$f_1^-(n) = \begin{cases} 0 & \text{pour} \quad n > 0 \\ -C_1 \left(n - \frac{1}{2}\right) \alpha^2 e^{\alpha(n-\frac{1}{2})} & \text{pour} \quad n \leqslant 0 \end{cases}$$

Calculons la transformée en **Z** $F_1^-(Z)$ de cette fonction :

$$F_1^-(Z) = \sum_{n=-\infty}^{0} Z^{-n} f_1^-(n)$$

Soit :

$$\begin{aligned} F_1^-(Z) &= C_1 \alpha^2 e^{-\frac{\alpha}{2}} \left[\sum_{n=0}^{\infty} n e^{-\alpha n} Z^n + \frac{1}{2} \sum_{n=0}^{\infty} e^{-\alpha n} Z^n\right] \\ &= -C_1 \frac{\alpha^2}{2} e^{-\frac{\alpha}{2}} \frac{1+e^{-\alpha} Z}{1-2e^{-\alpha} Z+e^{-2\alpha} Z^2} \\ &= -C_1' \frac{1+dZ}{1-bZ-cZ^2} \end{aligned}$$

L'équation de récurence pour la partie négative de la fonction de dérivation s'écrit alors :

$$Y^-(n) = b\,Y^-(n+1) + c\,Y^-(n+2) - C_1^{'}\,X(n) - d\,C_1^{'}\,X(n+1)$$

B.4.2.3 équations générales de récurence du filtre

$$(6.10) \quad Y^+(n) = b\,Y^+(n-1) + c\,Y^+(n-2) + C_1^{'}\,X(n-1) + d\,C_1^{'}\,X(n-2)$$

$$(6.11) \quad Y^-(n) = b\,Y^-(n+1) + c\,Y^-(n+2) - C_1^{'}\,X(n) - d\,C_1^{'}\,X(n+1)$$

$$(6.12) \quad Y(n) = Y^+(n) + Y^-(n)$$

Annexe C

Définition de la fonction d'Ackerman et de son inverse

C.1 fonction d'Ackerman $A(i,n)$

La fonction d'Ackerman $A(i,n)$ est définie de la manière suivante :

$$\begin{array}{lcll} A(0,n) & = & 2n & \text{pour} \quad n \geqslant 0 \\ A(i,0) & = & 1 & \text{pour} \quad i \geqslant 1 \\ A(i,n) & = & A(i-1, A(i,n-1)) & \text{pour} \quad i \geqslant 1, n \geqslant 1 \end{array}$$

C.2 inverse le la fonction d'Ackerman $\alpha(n,m)$

L'inverse[1] de la fonction d'Ackerman $\alpha(n,m)$ est définie pour $m, n \geqslant 1$ par :

$$\alpha(m,n) = min\left\{ i \geqslant 1 | \, A\left(i, 4\left\lfloor \frac{m}{n} \right\rfloor \geqslant \log n\right)\right\}$$

[1]Bien que la fonction présentée ici ne soit pas rigoureusement inverse de la fonction d'Ackerman au sens mathématique du terme, sa croissance, qui est aussi lente que la fonction d'Ackerman est rapide traduit bien la notion d'inverse.

Annexe D

Génération d'un cercle de synthèse

Nous présentons ici la méthode que nous avons employée pour générer les cercles qui nous ont servis à faire les tests présentés au chapitre 6.

D.1 Méthode employée

Le but de cette méthode est d'éviter d'avoir des effet d'aliasing (marches d'escalier) dans le dessin de notre disque. Pour cela la couleur affectée à un pixel sera proportionnelle à la surface du disque couvrant ce pixel.

La méthode est la suivante : soit $\mathcal{C}_R$ le disque de centre $O(0,0)$ et de rayon R (voir figure D.1), on calcule pour chaque pixel (i,j) l'aire du pixel recouverte par $\mathcal{C}_R$ (on suppose que la surface d'un pixel est d'une unité carré).

On note $P_A(x,y)$ la puissance analytique du point M de coordonnées (x,y) :

$$\begin{aligned} P_A(x,y) &= ||d(O,M)||^2 - R^2 \\ &= \sqrt{x^2+y^2} - R^2 \end{aligned}$$

et on a :

$$\begin{array}{lll} \text{si} & P_M(x,y) > 0 & , \quad M \notin \mathcal{C}_R \\ \text{si} & P_M(x,y); eqslant 0 & , \quad M \in \mathcal{C}_R \end{array}$$

Pour simplifier les calculs on raisonne sur le premier quart (c'est $x \geqslant 0$ et $y \geqslant 0$). On a alors si configurations possibles :

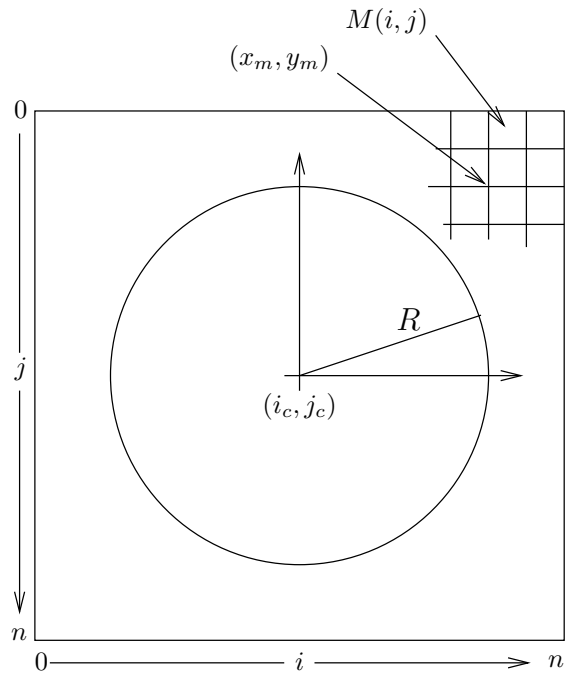
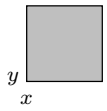
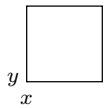
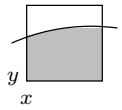
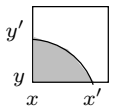


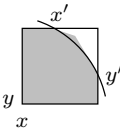
FIG. D.1 – Schéma de génération du disque de synthèse.

1.  : $P_A(x,y) \leqslant 0$ et $P_A(x+1,y) \leqslant 0$ et $P_A(x,y+1) \leqslant 0$ et $P_A(x+1,y+1) \leqslant 0$

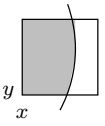
2.  : $P_A(x,y) > 0$ et $P_A(x+1,y) > 0$ et $P_A(x,y+1) > 0$ et $P_A(x+1,y+1) > 0$

3.  : $P_A(x,y) \leqslant 0$ et $P_A(x+1,y) \leqslant 0$ et $P_A(x,y+1) > 0$ et $P_A(x+1,y+1) > 0$

4.  : $P_A(x,y) \leqslant 0$ et $P_A(x+1,y) > 0$ et $P_A(x,y+1) > 0$ et $P_A(x+1,y+1) > 0$

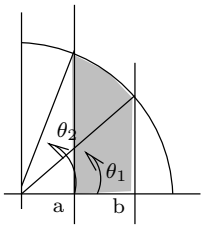
5. 

: $P_A(x,y) \leqslant 0$ et $P_A(x+1,y) \leqslant 0$ et $P_A(x,y+1) \leqslant 0$ et $P_A(x+1,y+1) > 0$

6. 

: $P_A(x,y) \leqslant 0$ et $P_A(x+1,y) > 0$ et $P_A(x,y+1) \leqslant 0$ et $P_A(x+1,y+1) > 0$

On note $\mathcal{A}_{\mathcal{C}_R}(a,b)$ l'aire du domaine limitée par le cercle $\mathcal{C}_R$, les droites d'équation $x = a$, $x = b$ et l'axe des abscisses :



$$\mathcal{A}_{\mathcal{C}_R}(a,b) = \int_a^b \sqrt{R^2 - x^2}dx$$

Grâce au changement de variable $x = R\cos\theta$ on calcule $\mathcal{A}_{\mathcal{C}_R}(a,b)$ et on trouve :

$$\mathcal{A}_{\mathcal{C}_R}(a,b) = \frac{1}{2}\left(b\sqrt{R^2-b^2} - R^2\arccos\frac{b}{R}\right) - \frac{1}{2}\left(a\sqrt{R^2-a^2} - R^2\arccos\frac{a}{R}\right)$$

D'où l'algorithme 5 de génération d'un disque de synthèse :

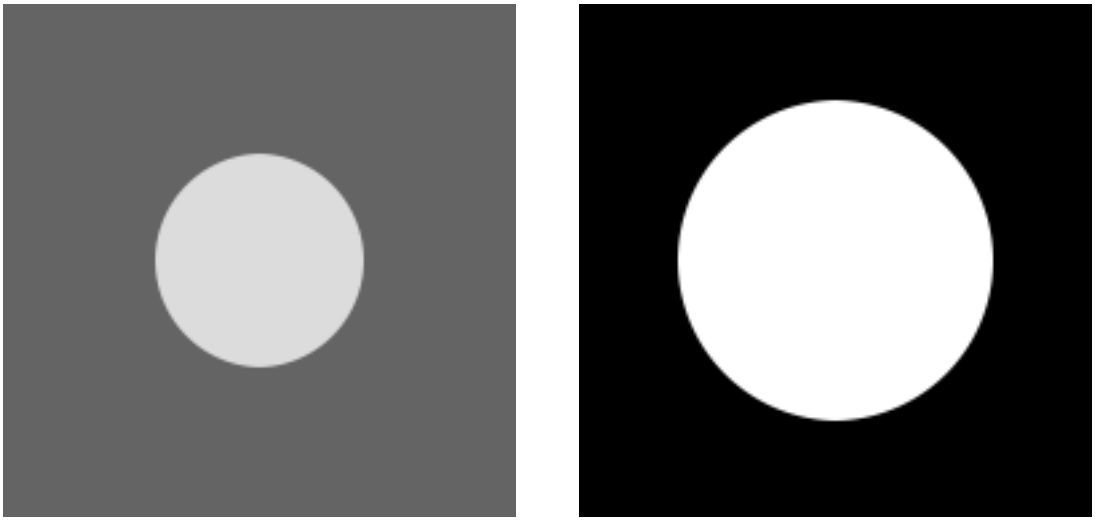
Algorithme 5: Génération d'un disque de synthèse de rayon R

% Cet algorithme génère un quart de disque de centre $(0,0)$ et de rayon R.
% Le quart généré correspond aux abscisses et ordonnées positives.
% $M(i,j)$ est la couleur (comprise entre 0 et 1).
% du pixel de coordonnées (i,j)

pour tout $(i,j) \in [i_c, n] \times [0, j_c]$ **faire**
  $x = i - i_c$
  $y = j_c - j$
  si $P_A(x,y) \leqslant 0$ **alors** $M(i,j) = 0$
  sinon
    si $P_A(x+1,y) > 0$ *et* $P_A(x,y+1) > 0$ *et* $P_A(x+1,y+1) > 0$ **alors**
      $M(i,j) = 1$
    sinon si $P_A(x+1,y) \leqslant 0$ *et* $P_A(x,y+1) > 0$ *et* $P_A(x+1,y+1) > 0$ **alors**
      $M(i,j) = \mathcal{A}_{\mathcal{C}_R}(x, x+1) - y$
    sinon si $P_A(x+1,y) > 0$ *et* $P_A(x,y+1) > 0$ *et* $P_A(x+1,y+1) > 0$ **alors**
      $y' = \sqrt{R^2 - x^2}$
      $M(i,j) = \mathcal{A}_{\mathcal{C}_R}(y, y;) - (y' - y).x$
    sinon si $P_A(x+1,y) \leqslant 0$ *et* $P_A(x,y+1) \leqslant 0$ *et* $P_A(x+1,y+1) > 0$ **alors**
      $x' = \sqrt{R^2 - (y+1)^2}$
      $M(i,j) = (x' - x) + \mathcal{A}_{\mathcal{C}_R}(x', x+1) - (x + 1 - x').y$
    sinon si $P_A(x+1,y) > 0$ *et* $P_A(x,y+1) \leqslant 0$ *et* $P_A(x+1,y+1) > 0$ **alors**
      $M(i,j) = \mathcal{A}_{\mathcal{C}_R}(y, y+1) - x$
    sinon Erreur !!!
fin

D.1.1 Résultats

Voici deux exemples de disques générés :



Cercle de rayon 40 Cercle de rayon 60

FIG. D.2 – Disques de synthèses générés grâce à l'algorithme 5

Annexe E

Images originales employées dans le document

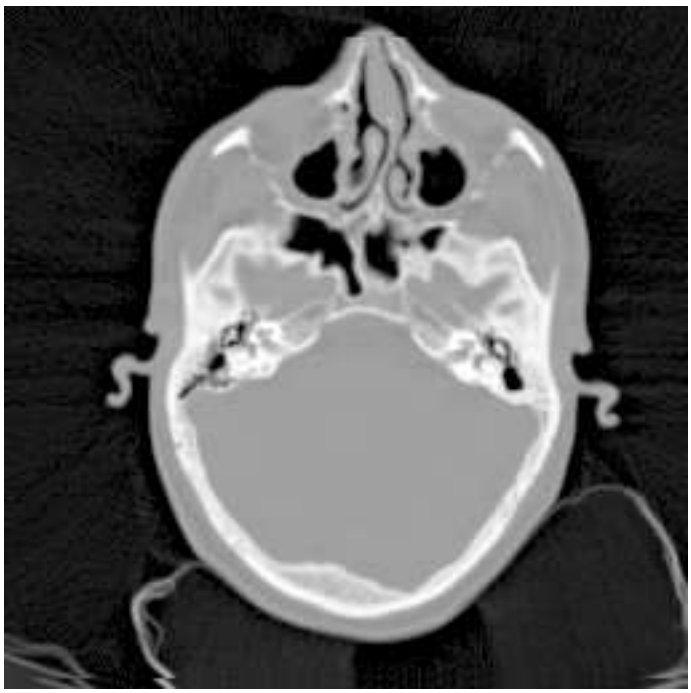


FIG. E.1 – Coupe scanner d'un crâne.

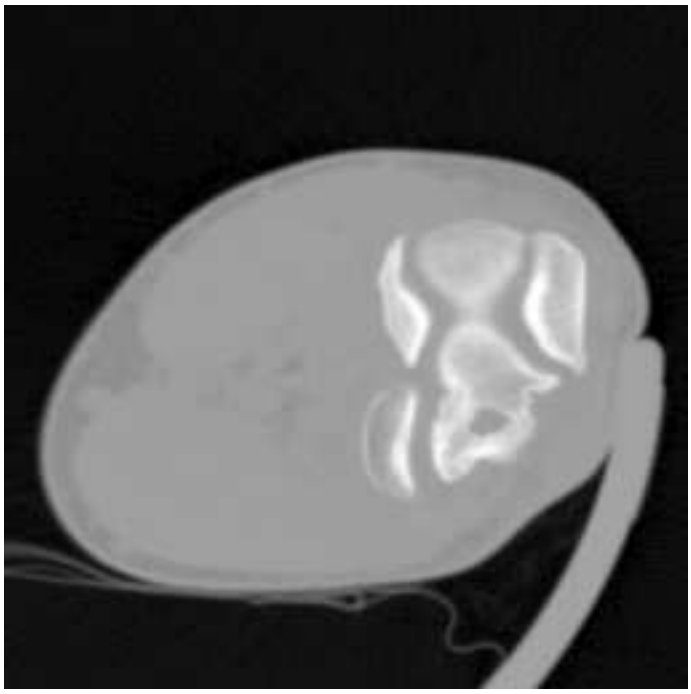


FIG. E.2 – Coupe scanner d'un coude.

FIG. E.3 – Peppers.

FIG. E.4 – Bateau *Cornouaille*

FIG. E.5 – Lena [Sjö72].

Bibliographie

[AAF95] E. Ahronovitz, J.-P. Aubert, and C Fiorio. The star-topology : a topology for image analysis. In *5th International Conference on Discrete Geometry for Computer Imagery (DGCI'05)*, pages 107–116. Groupe GDR PRC/AMI du CNRS, september 1995.

[AB94] R. Adams and L. Bischof. Seeded region growing. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 16(6) :641–647, june 1994.

[ACF94] E. Ahronovitz, P. Charnier, and C. Fiorio. A high level merging tool in image segmentation applications. In *Workshop on Machine Vision Applications (MVA'94)*, pages 218–221, Kawasaki, Japan, december 1994. IAPR.

[Ahr85] E. Ahronovitz. *Algorithmes de compression d'images et codes de contours*. Thèse de doctorat, École des Mines et Université de Saint Étienne, 1985.

[BA89] D. Blostein and N.A. Ahuja. A multiscale region detector. *Computer Vision, Graphics, and Image Processing.*, 45 :22–41, 1989.

[BB82] D.H. Ballard and C.M. Brown. *Computer Vision*. Prestige Hall, 1982.

[BB92] J. Benois and D. Barba. Image segmentation by region-contour cooperation for image coding. In *11th Int. Conf. on Pattern recognition (ICPR92)*, pages 331–334, 1992.

[BD94] Y. Bertrand and J.-F. Dufourd. Algebraic specification of a 3d-modeler based on hypermaps. *CVGIP : Graphical Models and Image Processing*, 56(1) :29–60, january 1994.

[BDFL92] Y. Bertrand, J.-F. Dufourd, J. Françon, and P. Lienhardt. Mod[è]lisation volumique [à] base topologique. Rapport de recherche 92/16, Universit[è] Louis Pasteur, 7, rue Ren[è] Descartes, 67094 Strasbourg, France, 1992.

[Ber70] Claude Berge. *Graphes et hypergraphes.* Dunod, Paris, 1970.

[Ber92] Y. Bertrand. *Spécification algébrique et réalisation d'un modeleur interactif d'objets géométriques volumiques.* Thèse de doctorat, Université Louis-Pasteur de Strasbourg, 1992.

[Bro68] R. Brown. *Elements of Modern Topology.* Mc Graw Hill, 1968.

[Brø83] A. Brøndstedt. *An Introduction to Convex Polytopes.* Pure and Applied Mathematics. Springer Verlag, 1983.

[BW91] A. Bhalerao and R. Wilson. Multiresolution image segmentation combining region and boundary information. In *7th Scandinavian Conference on Image Analysis*, pages 148–162, Aalborg, august 1991.

[Can86] J. Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 8(6) :679–698, 1986.

[Ced79] R. L. T. Cederberg. Chain-link coding and segmentation for raster scan devices. *Computer Graphics and Image Process.*, 10 :224–234, 1979.

[Cha79] J.-M. Chassery. Connectivity and consecutivity in digital pictures. *Computer Graphics and Image Process.*, 9 :294–300, 1979.

[Cha95] P. Charnier. *Outils algorithmiques pour le codage interpixel et ses applications.* Thèse de doctorat, Université Montpellier II, Janvier 1995.

[CLR90] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms.* MIT Press, Cambridge, Massachusets, USA, 1990.

[CLR94] T. Cormen, C. Leiserson, and R. Rivest. *Introduction à l'algorithmique.* Dunod, 1994.

[CM91] J.-M. Chassery and A. Montanvert. *Géométrie discrète en analyse d'images.* Hermès, 1991.

[Cor75] R. Cori. Un code pour les graphes planaires et ses applications. In *Astérisque*, volume 27. Soc. Math. de France, Paris, France, 1975.

[CP95] J.P. Cocquerez and S. Philipp. *Analyse d'images : filtrage et segmentation*. Enseignement de la Physique. Masson, 1995.

[d'A94] G. d'Andréa. Traitement parallèle des images. Mémoire de dea, Université Montpellier II, Juin 1994.

[Dan82] P.-E. Danielsson. An improved segmentation and coding algorithm for binary and nonbinary images. *IBM J. of Research and Development*, 26(6) :698–707, 1982.

[Der87] R. Deriche. Separable recursive filtering for efficient multi-scale edge detection. In *Int. Workshop on Machine Vision and Machine Intelligence*, Tokyo, december 1987.

[DG93] R. Deriche and G. Giraudon. A computational approach for corner and vertex detection. *Int. J. of Computer Vision*, 10(2) :101–124, 1993.

[DHM67] R. O. Duda, P. E. Hart, and J. H. Munson. Graphical data processing research study and experimental investigation. Technical report, AI Center, SRI International, 333 Ravenswood Ave, Menlo Park, CA 94025, march 1967. Report 26, Fourth Quarterly Report. Technical Report ECOM-01901-26. 1 November 1966 to 31 January 1967. SRI Project 5864. Prepared for US Army Electronics Command. Contract DA 28-043 AMC-01901(E).

[DST92] M.B. Dillencourt, H. Samet, and M. Tamminen. A general approach to connected-component labeling for arbitrary image representation. *J. of the Association for Computing Machinery*, 39(2) :253–280, april 1992. Corr. p. 985–986.

[Duf91] J.-F. Dufourd. Formal specification of topological subdivisions using hypermaps. *Computer-Aided Design*, 23(2) :99–116, 3 1991.

[Edm60] J. Edmonds. A combinatorial representation for polyhedral surfaces. *Notices Amer. Math. Soc.*, 7 :646, 1960.

[ES81] H. Elliot and L. Srinivasan. An application of dynamic programming to sequential boundary estimation. *Computer Graphics and Image Process.*, 17(4) :219–314, 1981.

[EV92] A.W.M. Van Den Enden and N.A.M. Verhoeckx. *Traitement numérique du signal, Une introduction*. Enseignement de la Physique. Masson, 1992.

[FG96] C. Fiorio and J. Gustedt. Two linear time Union-Find strategies for image processing. *Theoretical Computer Science*, 154 :165–181, 1996.

[Fio96] C. Fiorio. A topologically consistent representation for image analysis : the frontiers topological graph. In S. Miguet, A. Montanvert, and Ubeda S., editors, *6th International Conference on Discrete Geometry for Computer Imagery (DGCI'96)*, number 1176 in Lecture Notes in Computer Sciences, pages 151–162, Lyon, France, november 1996.

[Fra95] J. Françon. Discrete combinatorial surface. *Graphical Models and Image Processing*, 57(1) :20–26, january 1995.

[Fra96] J. Françon. On recent trends in discrete geometry in computer science. In S. Ubeda S. Miguet, A. Montanvert, editor, *6th International Conference on Discrete Geometry for Computer Imagery (DGCI'96)*, number 1176 in Lecture Notes in Computer Sciences, pages 1–16, Lyon, France, november 1996.

[Fre74] H. Freeman. Computer processing of line-drawing images. *ACM Computing Surveys*, 6(1) :57–97, march 1974.

[Fre91] W. Freeman. La physiologie de la perception. *Pour La Science*, 162 :70–78, Avril 1991.

[GHPT89] M. Gangnet, J.-C. Hervé, T. Pudet, and J.-M.V. Thong. Incremental computation of planar maps. report 1, Digital PRL, 5 1989.

[GI91] Z. Galil and G. F. Italiano. Data structures and algorithms for disjoint set union problems. *ACM Computing Surveys*, 23(3) :319–344, september 1991.

[GM95] M. Gondran and M. Minoux. *Graphes et algorithmes*. Collection de la Direction des Études et Recherches d'Électricité de France. Editions Eyrolles, 61 bd Saint Germain Paris 5^e, 3eme edition, 1995.

[GP74] V. Guillemin and A. Pollack. *Differential topology*. Prentice Hall, 1974.

[Gru67] Grunbaüm. *Convex polytopes*, volume 16 of *Pure and Applied Mathematics*. John Wiley, interscience edition, 1967.

[GT84] H. N. Gabow and R. E. Tarjan. A linear-time algorithm for a special case of disjoint set union. *Journal of Computer and System Sciences*, 30(2) :209–221, 1984.

[GW87] R.C. Gonzalez and P. Wintz. *Digital image processing.* Addison-Wesley, second edition, 1987.

[Her90] G.T. Herman. On topology as applied to image analysis. *Computer Vision, Graphics, and Image Processing.*, 52 :409–415, 1990.

[HM93] R. Horaud and O. Monga. *Vision par Ordinateur, outils fondamentaux.* Trait[é] des nouvelles technologie, s[é]rie informatique. Hermes, Paris, 1993.

[HP74] S. L. Horowitz and T. Pavlidis. Picture segmentation by a directed split-and-merge procedure. In *International Conference on Pattern Recognition*, pages 424–433, 1974.

[HS85] Robert M. Haralick and Linda G. Shapiro. Survey : Image segmentation techniques. *Computer Vision, Graphics, and Image Processing.*, 29 :100–132, 1985.

[HS92] R. M. Haralick and L. G. Shapiro. *Computer and Robot Vision*, volume 1-2. Addison-Wesley Publishing Company, Inc., june 1992.

[Jac70] A. Jacques. Constellations et graphes topologiques. In *Combinatorial Theory and Applications*, pages 657–673, Budapest, 1970.

[Jac89] L.B. Jackson. *Digital Filters and Signal Processing.* Kluwer Academic Publishers, 2 edition, 1989.

[Jac92] M. Jacquot. *Manipulation d'images binaires à l'aide du codage inter-pixel. Application à la reconnaissance de formules chimiques.* Thèse de doctorat, Université Montpellier II, 1992.

[Jol94] J.-M. Jolion. Computer vision methodologies. *CVGIP : Image Understanding*, 59(1) :53–71, january 1994.

[Kam95] H. Kamel. Analyse d'images à la volée. Mémoire de dea, Université de Montpellier II Sciences et Techniques du Languedoc, june 1995.

[KKM90a] E. Khalimsky, R. Kopperman, and P. R. Meyer. Boundaries in digital planes. *J. of Applied Mathematics and Stochastic Analysis*, 3(1) :27–55, 1990.

[KKM90b] Efim Khalimsky, Ralph Kopperman, and Paul R. Meyer. Computer graphics and connected topologies on finite ordered sets. *Topology and its Applications*, 36 :1–17, 1990.

[KM95] W. G. Kropatsch and H. Macho. Finding the structure of connected components using dual irregular pyramids. In *5th International Conference on Discrete Geometry for Computer Imagery (DGCI'05)*, pages 147–158, september 1995. invited lecture.

[Kov89] V.A. Kovalevsky. Finite topology as applied to image analysis. *Computer Vision, Graphics, and Image Processing.*, 46 :141–161, 1989.

[KP91] V. Koivunen and M. Pietäinen. Experiments with combined edge and region-based range image segmentation. In *7th Scandinavian Conference on Image Analysis*, pages 162–177, Aalborg, august 1991.

[KR89] T.Y. Kong and A. Rosenfeld. Digital topology : introduction and survey. *Computer Vision, Graphics, and Image Processing.*, 48(3) :357–393, december 1989.

[Kro94] W. G. Kropatsch. Building irregular pyramids by dual graph contraction. Technical Report PRIP-TR-35, Dept. for Pattern Recognition and Image Processing, Institute for Automation, Technical Univerity of Vienna, july 1994.

[KVW94] M. Kisworo, S. Venkatesh, and G. Wedst. Modeling edges at subpixel accuracy using the local energy approach. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 16(4) :405–410, april 1994.

[Lat93] L. Latecki. Topological connectedness and 8-connectedness in digital pictures. *CVGIP : Image Understanding*, 57(2) :261–262, march 1993.

[LE95] X. Liu and R. W. Ehrich. Subpixel edge location in binary images using dithering. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 17(6) :629–634, june 1995.

[Lef30] S. Lefschetz. *Topology.* Number 12. American Mathematical Society Colloquium Publications, New York, 1930.

[Lie90] Pascal Lienhardt. Topological models for boundary representation : A survey. research report R 90-02, Université Louis Pasteur, Computer

Science dpt, 7, rue Ren 'e Descartes, 67084 Strasbourg Cedex, february 1990.

[Lie91] P. Lienhardt. Topological models for boundary representations : a comparison with n-dimensional generalized maps. *Computer Aided Design*, 23(1) :59–82, 1991.

[Lie92] P. Lienhardt. Extensions de la notion de carte et modélisation géométrique 'a base topologique. Habilitation à diriger des recherches, janvier 1992.

[LJ92] Y. Lu and R.C. Jain. Reasonning about edges in scale space. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 14(4) :451–467, april 1992.

[MA68] J.L. Muerle and D.C. Allen. Experimental evaluation of techniques for automatic segmentation of objects in a complex scene. In G. C. Cheng et al., editors, *Pictorial Pattern Recognition*, pages 3–13. Thompson, Washington, 1968.

[Mal93] G. Malandain. On topology in multidimensional discrete spaces. Rapport de recherche 2098, INRIA, Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex (France), november 1993. Programme 4 — Robotique, image et vision — Projet Épidaure.

[Mar82a] D. Marr. *Vision*, page 31. Freeman, New York, 1982.

[Mar82b] D. Marr. *Vision.* Freeman press, San Francisco (CA), USA, 1982.

[Meh84] K. Mehlhorn. *Data Structures and Algorithms : Sorting and Searching.* Springer, 1984.

[MMR91] A. Montanvert, P. Meer, and A. Rosenfeld. Hierarchichal image analysis using irregular tessalations. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 13(4) :307–316, april 1991.

[Mon87] O. Monga. An optimal region growing algorithm for image segmentation. *International Journal of Pattern Recognition and Artificial Intelligence*, 1 :351–375, 1987.

[Mon90] P. Montesinos. *Une méthodologie en vue de l'interprétation automatique d'images aérienne : application à la détection des bâtiments.* Thèse de doctorat, Université de Nice Sophia-Antipolis, february 1990.

[MWD87] O. Monga and B. Wrobel-Dautcourt. Segmentation d'images : vers une méthodologie. *Traitement du Signal*, 4(3) :169–193, 1987.

[Pav77] T. Pavlidis. *Structural Pattern Recognition.* Springer-Verlag, New York, 1977.

[PCCB91] M. Popovic, F. Chantemargue, R. Canals, and P. Bonton. Several approaches to implement the merging step of the split and merge region segmentation. In F.H. Post and W. Barth, editors, *EUROGRAPHICS '91*, pages 399–412. Elsevier Science publishers B.V., 1991.

[PH91] C. G. Perrott and L. G. C. Hamey. Object recognition, a survey of the literature. Macquarie Computing Reports 91–0065C, Macquarie University, school of MPCE, Macquarie University, NSW 2109 Australia, january 1991.

[PP93] Nijhil R. Pal and Sankar K. Pal. A review on image segmentation techniques. *Pattern Recognition*, 26(9) :1277–1294, 1993.

[Pre70] J. Prewitt. Object enhancement and extraction. In B. Lipkin and A. Rosenfeld, editors, *Picture Processing and Psychopictorics*, pages 75–149. Academic Press, New York, 1970.

[Rob65] L.-G. Roberts. Machine perception of three dimensional solids. In J.-T. et al. Tippet, editor, *Optical and Electro-optical Information Processing*, pages 159–197. MIT Press, Cambridge, 1965.

[Ros74] A. Rosenfeld. Adjacency in digital pictures. *Inform. and Control*, 26(1) :24–33, 1974.

[RT71] A. Rosenfeld and M. Thurston. Edge and curve detection for visual scene analysis. *Computers, IEEE Transactions on*, C-20(5) :562–569, 1971.

[Sal94] M. Salotti. *Gestion des informations dans les premières étapes de la vision par ordinateur.* Thèse de doctorat, Institut National Polytechnique de Grenoble., 1994.

[Sam84] H. Samet. The quadtree and related hierarchical data structures. *Computing Surveys*, 16(2) :187–260, 1984.

[SB93] H. Sato and T. O. Binford. Finding and recovering shgc objects in an edge image. *CVGIP : Image Understanding*, 57(3) :346–358, 1993.

[SC86] J. Shen and S. Castan. An optimal linear operator for edge detection. In *Proc. Conference on Computer Vision and Pattern Recognition*, pages 109–114, Miami Beach, Florida, USA, june 1986.

[Ser95] A. Serrano. Extraction de surfaces dans une image trois dimension. Mémoire de dea, Universit 'e de Montpellier II Sciences et Techniques du Languedoc, juin 1995.

[SG93] M. Salotti and C. Garbay. Cooperation between edge detection and region growing : the problem of control. In *Image Processing : Theory and Applications*, pages 95–98. Elsevier Science Publishers, 1993.

[Sjö72] Lena Sjöblom. playmate. *Playboy magazine*, 11 :138, 1972.

[SMCM91] P. Saint-Marc, J.S. Chen, and G. Medioni. Adaptive smoothing : A general tool for early vision. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 13(6) :514–528, june 1991.

[SMT82] J. P. W. Stark, Y. Mahdavieh, and T. Tjahjardi. A fast algorithm for colour region segmentation. In D. S. Greenaway and E. A. Warman, editors, *EUROGRAPHICS 82*, pages 47–56. Eurographics Association, North-Holland Publishing Company, 1982.

[Sob70] I.E. Sobel. *Camera Models and Machine Perception*. PhD thesis, Electrical Engineering Department, Stanford University, Stanford, 1970.

[Tab94] S. Tabonne. *Détection multi-échelle de contours subpixel et de jonctions*. Thèse de doctorat, Institut National Polytechnique de Lorraine, 1994.

[Tar75] R. E. Tarjan. Efficiency of a good but not linear set union algorithm. *J. of the Association for Computing Machinery*, 22 :215–225, 1975.

[Tar79] R. E. Tarjan. A class of algorithms which require nonlinear time to maintain disjoint sets. *Journal of Computer and System Sciences*, 18(2) :110–127, 1979.

[TvL84] R. E. Tarjan and J. van Leeuwen. Worst-case analysis of set union algorithms. *J. of the Association for Computing Machinery*, 31(2) :245–281, 1984.

[VdGV91] G. Venturi, A. di Giuliani, and G. Vernazza. Image segmentation using edge and region information. In *International Conference on Image Analysis and Processing*, pages 65–73, Villa Olmo, Como, Italy, 4–6 September 1991. Progress in Image Analysis and Processing II.

[Vei94] A. Veijanen. Unsupervised image segmentation using an unlabeled region process. *Pattern Recognition*, 27(6) :841–852, 1994.

[vKO93] M. J. van Kreveld and M. H. Overmars. Union-copy structures and dynamic segment trees. *J. of the Association for Computing Machinery*, 40(3) :635–652, july 1993.

[Vos92] G. Vosselman. *Relational Matching.* Lecture Notes in Computer Science. Springer-Verlag, Berlin Heidelberg Germany, 1992.

[Wit83] A.P. Witkin. Scale-space filtering. In *8th International Conference on Artificial Intelligence*, pages 1019–1022, Karlsruhe (Germany), 1983.

[YF86] Tzay Y. Young and King-Sun Fu. *Handbook of Pattern Recognition and Image Processong.* Academic Press, Inc., 1986.

[Zio91] D. Ziou. *La détection de contours dans des images à niveaux de gris : mis en œuvre et sélection de détecteurs.* PhD thesis, Institut National Polytechnique de Lorraine, 1991.

[Zuc76] Steven W. Zucker. Survey : Region growing : Childhood and adolescence. *Computer Vision, Graphics, and Image Processing.*, 5 :382–399, 1976.