Contribution à l'interprétation d'images et vérification de la consistance d'un graphe

Yann Hodé

UNIVERSITÉ DE STRASBOURG

ÉCOLE DOCTORALE Mathématiques, Sciences de l'Information et de l'Ingénieur

Laboratoire ICUBE Equipe CSTB

THÈSE présentée par :

Yann HODE

soutenue le : **12 novembre 2016**

pour obtenir le grade de : **Docteur de l'université de Strasbourg**

Discipline/ Spécialité : Informatique

Contribution à l'interprétation d'images et vérification de la consistance d'un graphe

THÈSE dirigée par :
Mme DERUYVER Aline Maitre de Conférence HdR , Université de Strasbourg

RAPPORTEURS :
Mr BRUN Luc Professeur, Ensicaen
Mr CONTE Donatello Maitre de Conférence HdR, Université de tours

---

AUTRES MEMBRES DU JURY :
Mr COLLET Pierre Professeur, Université de Strasbourg

UNIVERSITE DE STRASBOURG

Résumé

Ecole doctorale mathématiques, sciences de l'information et de l'ingénieur, Laboratoire ICUBE - UMR 7357

Docteur de l'université de Strasbourg
Discipline/Spécialité: Informatique

Contribution à l'interprétation d'images et vérification de la consistance d'un graphe

par Yann Hodé

Dans cette thèse nous montrons que le raisonnement symbolique associé à la vérification de la consistance d'arc avec propagation de contraintes est un outil efficace pour interpréter les images.
Nous montrons dans un premier temps que ce cadre théorique permet de vérifier l'organisation spatiale de différentes composantes d'un objet complexe dans une image.
Nous proposons ensuite d'étendre l'utilisation de celui-ci à la reconnaissance sélective des formes décrites par des équations mathématiques, grâce à la notion de consistance d'hyper-arc à deux niveaux de contraintes.
La pertinence et la faisabilité de cette approche ont été validées par de multiples tests. En outre, les résultats obtenus sur des images sur-segmentées montrent que la méthode proposée est résistante au bruit, même dans des conditions où les humains (dans certains cas d'agnosie visuelle) peuvent échouer.
Ces résultats soutiennent l'intérêt du raisonnement symbolique dans la compréhension de l'image.

Remerciements

A Aline DERUYVER. Cette thèse est la concrétisation d'une passionnante aventure scientifique commune, débutée il y a déjà de nombreuses années.

A Luc BRUN. Merci d'avoir pris de ton temps pour relire cette thèse et pour tes remarques avisées. La rigueur de ton regard mathématique assortie d'une grande gentillesse ont été très aidants.

A Donatello CONTE. Merci d'avoir accepté d'être rapporteur de cette thèse, et merci également pour les retours très utiles.

A Pierre COLLET. Merci d'avoir accepté de participer à mon jury de Thèse. Ta curiosité et ton engouement pour les aspects novateurs de l'informatique sont des qualités que j'apprécie beaucoup.

A Alain DIETERLEN. Merci pour l'intérêt porté à ce travail et pour avoir accepté de participer au jury de cette Thèse.

Les mathématiques ou l'informatique ont l'avantage qu'il est possible de travailler seul, sans équipe et sans moyen. C'est une chance qui m'a permis de supporter l'ennui que généraient des conditions professionnelles avec peu d'opportunités proposées et ne favorisant pas la créativité. Merci à toute cette communauté de chercheurs et d'ingénieurs qui ont créés et développés des ordinateurs puissants. Grâce à eux, ces soirées, ces week-end et ces vacances passés à élaborer des constructions algorithmiques, à les programmer et à les tester ont été un échappatoire salutaire.

Merci enfin à toutes celles et ceux qui dans mon autre activité "professionnelle" qu'est la psychiatrie partagent ma passion et mes combats pour une médecine à la fois plus efficace, plus scientifique, plus humaine et plus aidante.

Table des matières

Chapitre 1

Introduction

1.1 Introduction

1.1.1 *La vision artificielle est dominée aujourd'hui par les modèles de discrimination statistique*

Brooks dans un célèbre article [1] disait "les éléphants ne jouent pas aux échecs" bien qu'ils sachent résoudre de nombreux problèmes complexes que nous n'avons pas encore pu faire résoudre par l'intelligence artificielle. Le relatif échec des approches basées sur des règles de raisonnement, le développement resté marginal des langages comme Prolog, Lisp ou leurs dérivés tend à montrer que dans les faits la communauté de l'intelligence artificielle adhère largement à ce point de vue.

Les bonnes performances des animaux dans des tâches de discrimination et de prise de décision visuelle semblent montrer que le traitement des données visuelles n'exige pas un degré élevé de raisonnement symbolique. Aujourd'hui les travaux de l'intelligence artificielle dans le domaine de l'analyse d'images sont majoritairement dominés par des approches de discrimination statistiques, basés parfois sur des modèles physiques ou inspirés de modèles cérébraux. Le succès actuel de l'apprentissage profond et en particulier des réseaux neuronaux convolutifs [2] qui ont permis ces dernières années d'analyser et d'interpréter des milliers d'images courantes avec un taux d'erreurs très honorable et surtout la croissance régulière des performances des logiciels basés sur ces principes, tend à reléguer toute autre approche au passé. Cependant l'analyse plus attentive des succès de l'apprentissage profond montre qu'on est encore très loin d'obtenir des performances humaines. Des cibles prédéfinies peuvent être retrouvées dans des images mais de façon globale avec une localisation approximative et les labélisations sémantiques des contenus d'images réservent encore un taux d'erreurs non négligeable et parfois des erreurs assez

FIGURE 1.1 – L'analyse avec un réseau de neurone convolutif de cette image répond qu'il s'agit d'une girafe qui se tient près d'une barrière dans un champs (tiré de Richard Zemel Learning to Generate Images and Test, support de cours à l'International Summer School on Deep Learning , Bilbao juillet 2017).

cocasses (Voir Figure 1.1). Les systèmes de reconnaissance ont des capacités de reconnaissance très liées à leur base d'apprentissage et l'extraction automatiques de concepts et leur manipulation, pour intéressante qu'elle soit, n'atteint pas les capacités humaines. On pourrait faire l'hypothèse que ce n'est qu'une question de temps : Les capacités de calcul des ordinateurs d'accroissent régulièrement, les expérimentations visant à trouver des architectures de réseaux neuronaux plus performantes se multiplient et une recherche théorique intense se développe pour prévoir de façon plus efficace quelles architectures et quels hyperparamètres [2] sont les plus appropriés pour tel ou tel problème de vision artificielle.

1.1.2 *Mais d'autres approches considérées comme anciennes pourraient avoir un potentiel d'intérêt*

Il est difficile d'affirmer que l'émergence des capacités de raisonnement symbolique chez l'homme lors de la sélection darwinienne ne présente aucun avantage évolutif. De la même façon, il est difficile de croire que l'interprétation d'images ne peut pas bénéficier du raisonnement symbolique, qui est un outil puissant de l'intelligence humaine. Notre but ici n'est pas de discuter des avantages et des inconvénients des différentes approches utilisées en analyse d'image (voir [3]), mais plutôt de montrer que le raisonnement symbolique dans le cadre des problèmes de satisfaction de contraintes, bien que majoritairement délaissé par la communauté de la vision artificielle, offre des potentialités intéressantes.

Nous allons montrer à travers les nouveaux outils conceptuels développés au cours de cette thèse que le raisonnement symbolique permet d'effectuer une reconnaissance des formes satisfaisante. En d'autres termes, nous chercherons à montrer qu'il est possible d'identifier des objets décrits sur le mode "langagier" par l'application de règles de raisonnement logiques visant à rechercher si les pixels de l'images forment des entités qui correspondent aux éléments descriptifs de l'objet à reconnaitre. Cette approche

se différencie des techniques de l'apprentissage profond qui recherchent, par apprentissage, des liens "statistiques" entre des éléments du langage et la présence de certaines formes dans une image. Dans notre approche, l'humain définit explicitement l'objet à reconnaître, alors que dans l'apprentissage profond, l'humain indique au logiciel s'il a réussi à reconnaitre l'objet cible à travers de multiples exemples, afin que celui-ci se construise par essai-erreurs une représentation implicite de l'objet.

En Intelligence Artificielle, une vaste gamme de problèmes (vision par ordinateur, planification, raisonnement temporel, problèmes de graphes, etc.) sont considérés comme des problèmes de satisfaction de contraintes (PSC) [4–9]. Les problèmes d'interprétation d'images peuvent être considérés comme des PSC sur un domaine fini. Ce cadre donne la possibilité de travailler avec une connaissance symbolique. Dans ce cas, les contraintes représentent la connaissance décrivant le contenu supposé de l'image. Le problème consiste à mettre en correspondance les régions segmentées d'une image avec cette connaissance.

Nous avons centré notre travail sur les PSC car ce cadre théorique important en intelligence artificielle a été insuffisamment exploité dans le domaine de l'interprétation d'images. Peu d'auteurs ont appliqué les PSC à l'interprétation des images [10–13] ce qui suggère certaines difficultés de l'approche. Cependant, ce cadre nous a semblé intéressant d'un point de vue pragmatique : il était adapté à la façon de procéder d'un expert anatomiste qui analyse une coupe cérébrale, ce qui a été notre cas, et il répondait à un objectif de recherche appliquée dans le domaine de l'imagerie cérébrale que nous avions au début de ce travail. Devant identifier des régions cérébrales sur des coupes, c'est par le positionnement des régions entre elles, tel que cela est décrit dans les livres d'anatomie, qu'il était possible de délimiter et d'identifier avec certitude les différentes sous structures anatomiques du cerveau. Cette approche se différencie des deux grands types de logiciels d'analyses d'images cérébrales qui en 2018 n'ont pas encore amené de résultats satisfaisants :

— Ceux du premier type utilisent des atlas et identifient des régions d'un cerveau donné en faisant une transformation géométrique pour apparier ce cerveau avec le modèle qui est donné par l'atlas [14]. Dans la pratique, ces transformations géométriques donnent des résultats en moyenne d'apparence correcte pour un non expert. Cependant, dès qu'on observe les résultats dans le détail, la labélisation des pixels est insatisfaisante pour faire des études volumétriques de qualité. Par ailleurs l'emploi d'un atlas de cerveau normal pose des problèmes dès qu'un cerveau est trop "anormal", ce qui est un problème en imagerie médicale, puisque si on réalise des images c'est souvent à la recherche d'anormalité parfois structurelle.

— Ceux du deuxième type utilisent de systèmes de règles, mélangées avec des approches ad hoc pour certaines régions [12]. Le caractère hybride de ces approches

nous a ennuyé car cela nuisait à leur généralisation et correspondait à un type d'images strictement lié au logiciel. Le fait que les connaissances doivent être explicitées sous forme de règles alors que ce n'est pas sous cette forme qu'est représentée la connaissance dans les livres d'anatomie nous a semblé une faiblesse potentielle. En effet, la traduction de la connaissance demande, dans ce cas, une double compétence d'informaticien et d'expert anatomiste. Il nous semblait préférable que les connaissances anatomiques puissent être décrites par un non informaticien, et que la façon d'utiliser ces connaissances soit gérée par un algorithme général indépendant du contenu de cette connaissance.

Dans le cadre de la recherche de satisfaction de contraintes appliquées aux images, David Waltz avait proposé dans les années 70-80 une façon très intéressante de labéliser des objets dans une image en décrivant des primitives, des relations spatiales entre ces primitives et des contraintes spatiales que doivent satisfaire ces primitives pour être labélisées comme partie d'un objet défini. L'aspect fortement combinatoire de la recherche de satisfaction de contrainte (problème NP complet) était résolu par la stratégie de recherche de consistance locale des solutions et propagation des solutions éliminées. Dans la pratique la recherche de la consistance locale peut en effet parfois suffire pour atteindre une consistance globale. Waltz avait défini un monde simplifié formé d'objets géométriques simples qu'il a appelé monde de blocs. L'approche de Walz est devenue dans de nombreux ouvrages d'intelligence artificielle un modèle paradigmatique de la résolution de problèmes de labélisation dans le cadre de la recherche de consistance locale et de propagation de contraintes. Par la suite, d'autres auteurs [4, 10–12, 15] ont proposé différentes méthodes pour appliquer les PSC à l'interprétation des images. Toutes ces approches fonctionnent de la même manière : faire correspondre de façon univoque les nœuds d'un RAG (graphe d'adjacence de régions), qui représente les relations spatiales entre les régions segmentées ou primitives extraites, avec les nœuds du réseau sémantique décrivant ce qui est recherché. Ce modèle correspondait bien à notre problème, et nous avons essayé de l'exploiter.

Cependant, les PSC classiques présentent un inconvénient majeur : une seule valeur doit être affectée à une variable discrète. Cette affectation bijective ne permet pas l'association de plusieurs régions segmentées à un concept (ou variable). Cet inconvénient était rédhibitoire dans notre cas en raison de la sursegmentation fréquente des images cérébrales. Les travaux que nous présentons montrent qu'il est possible de dépasser ces limites et d'utiliser les PSC pour interpréter des images. Ceci non seulement pour analyser l'organisation spatiale des différents composants de l'image mais aussi pour reconnaître les caractéristiques morphologiques des formes elles-mêmes.

Ce mémoire de thèse est organisé de la façon suivante :
Dans le chapitre 2 nous rappelons brièvement l'état de l'art dans le domaine de la reconnaissance des formes utilisant le formalisme des graphes, puis dans les chapitres suivants nous présenterons un historique de la progression de nos travaux :
Dans le chapitre 3 nous présenterons les notions d'arc-consistance faibles développées dans le cadre de la consistance d'arc à deux niveaux de contraintes pour traiter des cas d'objets non prévus dans le modèle et des cas d'occlusion.
Dans le chapitre 4 nous présenterons un formalisme permettant d'exprimer des relations spatiales complexes afin de pouvoir construire des modèles décrivant précisément les objets ou les scènes à reconnaître et ainsi assurer une meilleure interprétation.
Dans le chapitre 5 nous présenterons une façon de guider un processus de segmentation par la connaissance en utilisant notre système d'interprétation d'image.
Dans le chapitre 6 nous étendrons notre système d'interprétation d'images par contrainte à la reconnaissance des caractéristiques morphologiques des formes, et nous y introduirons la notion d'hyper-arc consistance à deux niveaux de contraintes.
Nous terminerons par une discussion résumant les contributions apportées par ce travail et les perspectives de développement.

Chapitre 2

Etat de l'art : Graphes et reconnaissance des formes

2.1 Introduction

Les graphes sont un mode de représentation très commun et pratique du contenu d'une image. Leur structure spatiale permet des isomorphismes naturels entre différents niveaux de structuration de l'image. La bicomposition des graphes (nœuds et arcs) permet de représenter des entités variées (pixels, ensemble de pixels, concepts visuels, ...) et des relations entre ces entités (relations spatiales, directionnelles, topologiques, métriques, relation sémantiques, ...). De très nombreux travaux utilisent cette représentation, comme l'illustre la synthèse réalisée dans "Image Processing and Analysis with Graphs : Theory and Practice" édité par Olivier Lezoray et Leo Grady [16].
Une des approches très féconde aujourd'hui dans le domaine des graphes appliqués à la reconnaissance des formes est d'essayer de traduire la représentation des graphes en une représentation vectorielle, plus rudimentaire, mais pour laquelle il existe des outils mathématiques très performants [17]. Cela suppose de trouver comment effectuer cette traduction. Lorsqu'on gère des relations spatiales simples ou des relations métriques, c'est relativement faisable. Cependant, certains types de relations complexes entre les nœuds des graphes que nous avons utilisés se prêtaient mal à trouver une traduction idoine dans le domaine vectoriel. Nous ne discuterons donc pas ici de ces approches, trop éloignées de la piste que nous avons poursuivie.
Il y a deux grand types d'approches complémentaires très classiques en intelligence artificielle lorsqu'il faut réaliser un isomorphisme entre deux ensembles structurés par des relations non métriques (Comme nous l'avons dit précédemment, nous excluons ici les approches dans lesquelles la structure de graphe peut être traduite sous forme matricielle

sur laquelle il est possible d'appliquer des outils appartenant au domaine des espaces vectoriels, par exemple les méthodes spectrales [18], etc.) :

— les approches recherchant la solution par exploration de l'arborescence des solutions hypothétiques. Des heuristiques appropriées peuvent optimiser cette exploration qui est souvent très gourmande en temps de calcul.

— les approches par propagation de contraintes visant à réduire préalablement l'espace d'exploration avec la possibilité que parfois cette réduction soit assez drastique pour donner la solution directement.

Après avoir rappelé quelques bases du formalisme des graphes, nous décrirons brièvement les travaux du premier type appliqués à la mise en correspondance de graphe puis nous présenterons les approches basées sur les graphes de contraintes et la propagation de contraintes.

2.2 Représentation des images par les graphes

Dans toute la suite, nous noterons un graphe G par le couple (X, E), où X est l'ensemble des sommets et E l'ensemble des arcs. L'ordre n du graphe est le nombre de nœuds et sa taille m le nombre d'arcs. On trouvera dans [19] les définitions et les algorithmes classiques en théorie des graphes. On pourra également se reporter au livre de Gondran et Minoux [20]. Les graphes seront généralement attribués aussi bien au niveau des nœuds que des arcs. On définit un graphe relationnel attribué par $\mathrm{G} = (X, E, \mu, \nu)$, avec :

— $\mu : X \rightarrow L_X$ fonction d'attribution des attributs aux sommets (interpréteur de sommets)

— $\nu : E \rightarrow L_E$ fonction d'attribution des attributs aux arcs (interpréteur d'arcs)

Le graphe le plus couramment utilisé est le graphe des pixels, où chaque pixel représente un nœud du graphe et chaque arc est défini par la 4- ou la 8-connexité. Il existe aussi des graphes dont les nœuds représentent non pas un pixel mais un ensemble de pixels permettant une représentation plus compacte de l'image. Le graphe d'adjacence (GRA) construit à partir d'une sur-segmentation en est un exemple. Dans ce cas, on associe souvent aux nœuds représentant les régions des attributs comme le niveau de gris moyen, la surface, des indices texturaux, etc. Quant aux arcs entre deux régions adjacentes, on leur associe par exemple la longueur du contour ou une mesure de contraste.

Des graphes également couramment utilisés sont les graphes de Voronoï construits à partir d'un nuage de points ou, de façon duale, la triangulation de Delaunay [21]. Il est important de noter que de nombreux graphes en traitement d'images sont des graphes

planaires (i.e pour lesquels il existe une représentation graphique dans laquelle les arcs ne se coupent pas [20]). Ceci présente l'intérêt de rendre certains algorithmes de complexité polynomiale plutôt qu'exponentielle.

Il existe encore beaucoup d'autres types de graphes comme par exemple les graphes aléatoires (typiquement utilisés dans les champs markoviens), les graphes flous et les graphes d'attributs flous [22], et il est difficile d'en faire une liste exhaustive, ce qui montre la capacité de représentation de données et de concepts offerte par ce formalisme.

2.2.1 La mise en correspondance de graphes

Les graphes sont très largement utilisés pour la reconnaissance des formes [23]. Le principe de base qui a souvent prévalu dans les débuts de la reconnaissance de formes structurelle est le suivant : l'objet est défini par un ensemble de primitives qui constituent les nœuds du graphe, et des relations binaires de compatibilité entre primitives constituent les arcs du graphe. Une clique du graphe représente alors un sous-ensemble de primitives compatibles 2 à 2 qui est une configuration possible de l'objet. La reconnaissance se fait alors par la détection des cliques maximales du graphe (i.e des cliques auxquelles on ne peut plus rajouter aucun sommet). Si plusieurs cliques maximales existent, on définit une fonction de coût pour effectuer le choix. La recherche des cliques maximales d'un graphe est un problème NP-complet. De nombreuses solutions ont été proposées pour cela, une stratégie simple pour y parvenir pouvant être de construire un arbre de décision (un nœud de l'arbre correspond à une clique du graphe) qui est élagué afin de ne pas réengendrer les mêmes cliques.

Cette façon de présenter le problème peut être plus généralement posée comme la recherche d'un isomorphisme entre deux graphes. Soit $G = (X, E, \mu, \nu)$ et $G' = (X', E', \mu', \nu')$ deux graphes. G est isomorphe à G' si et seulement si $\exists f : X \to X'$ tel que

- $\forall x \in X, \exists! x' = f(x) \in X'$ et $\forall x' \in X', \exists! x \in X, f(x) = x'$
- $e = (x_1, x_2) \in E \Leftrightarrow e' = ((f(x_1), f(x_2)) \in E'$
- $\forall x \in X, \mu(x) = \mu\prime(f(x))$ et $\forall e = (x_1, x_2) \in E, \nu(x_1, x_2) = \nu'(f(x_1), f(x_2))$

De nombreux problèmes de traitement d'images peuvent se poser en termes d'appariement de deux graphes : un graphe modèle de référence représentant l'objet cherché et un graphe de données déduit de l'image à analyser. C'est par exemple le cas en imagerie médicale où on dispose d'un atlas anatomique avec lequel on cherche à mettre l'image en correspondance ; ou en imagerie aérienne ou satellitaire avec une carte ou un système d'information géographique ; ou encore en reconnaissance d'objets (par exemple des caractères) où un modèle de l'objet à détecter va être construit.

Cependant, l'acquisition de l'image amène des problèmes qui rendent la recherche d'appariement de graphes plus complexe : par exemple des parties de l'objet peuvent être cachées amenant ainsi des parties manquantes dans le graphe des données par rapport au graphe modèle. Dans la réalité, le graphe des données est souvent bruité et le graphe modèle éventuellement incomplet. Cela amène a rechercher dans ce cas, non pas un isomorphisme de graphe, mais plutôt soit le plus grand sous graphe commun soit un isomorphisme de sous-graphes [24] [25], soit un isomorphisme de sous-graphes avec tolérance d'erreurs [26–31]. Dans ce dernier cas, on peut utiliser des algorithmes exacts (lorsque le nombre de nœuds est petit) ou des algorithmes approximatifs.
Nous allons aborder brièvement ces différentes approches dans les sections suivantes.

2.2.1.1 L'isomorphisme de sous graphe

L'isomorphisme de sous graphe se pose de façon légèrement différente. Il s'agit ici de trouver la fonction f tel que

- $\forall x \in X, \exists! x' = f(x) \in X'$
- $e = (x_1, x_2) \in E \Leftrightarrow e' = ((f(x_1), f(x_2)) \in E' \cap f(X) \times f(X)$
- $\forall x \in X, \mu(x) = \mu\prime(f(x))$ et $\forall e = (x_1, x_2) \in E, \nu(x_1, x_2) = \nu'(f(x_1), f(x_2))$

Le problème de la recherche d'un isomorphisme de sous-graphes est un problème NP-complet, excepté dans le cas des graphes planaires où la complexité devient polynomiale (cette situation est assez fréquente en traitement d'images). Il existe deux grands algorithmes de recherche. Le premier consiste à construire un graphe d'association G_a, où chaque sommet correspond à l'association entre deux sommets de même attribut et chaque arc correspond à des associations de sommets compatibles (i.e reliés dans les graphes d'origine G_1 et G_2 par des arcs de même attribut le cas échéant). Chaque clique du graphe d'association correspond alors à un isomorphisme de sous-graphe (au sens large). On recherche dans G_a la plus grande ou la "meilleure" clique maximale au sens d'un critère donné (un exemple de graphe d'association est illustré figure 2.1). Un exemple pour de la mise en correspondance en stéréo-vision est donné dans [32].

Un autre algorithme est l'algorithme de Ullman [33] qui consiste à construire un arbre de décision, chaque feuille correspondant à un ensemble d'associations compatibles. En cas d'échec (cas d'isomorphisme de sous-graphes strict ici), on remonte dans l'arbre pour essayer une nouvelle association. Pour accélérer la recherche, une matrice d'association permet de vérifier s'il existe des appariements possibles pour tous les sommets restants. La complexité d'appariement de 2 graphes G et G' est dans le pire cas en $O(m^n n^2)$ (n ordre de G et m de G' avec $n < m$).

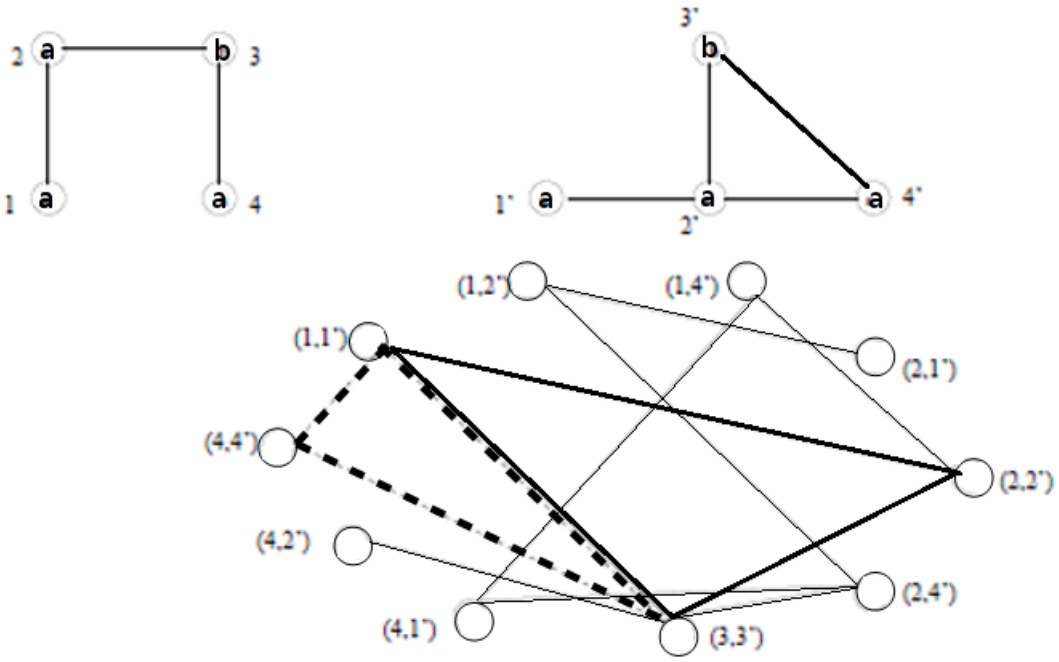


FIGURE 2.1 – En haut, deux graphes en bas : le graphe d'association correspondant. Les deux plus grandes cliques maximales sont indiquées en gras (trait plein et en pointillés).

2.2.1.2 La mise en correspondance inexacte

Dans le monde réel, les images et les graphes qui en sont déduits sont sujets à de multiples distorsions et donc bruités (au niveau des nœuds comme au niveau des arcs) et incomplets. Une idée est de définir une distance entre graphes et de rechercher le sous-graphe du graphe des données à distance minimale du graphe de référence G. On parle dans ce cas d'isomorphisme de sous-graphes avec tolérance (ou correction) d'erreurs ou d'isomorphisme inexact[26–31].

On distingue parfois plusieurs situations, les isomorphismes tolérants à la substitution de nœuds ou d'arcs, et les isomorphismes inexacts. Les isomorphismes tolérants à la substitution se définissent ainsi :

— $\forall x \in X, \exists! x' = f(x) \in X'$

— $e = (x_1, x_2) \in E \Leftrightarrow e' = ((f(x_1), f(x_2)) \in E' \cap f(X) \times f(X)$

— $\forall x \in X, \mu(x) \approx \mu\prime(f(x))$ et $\forall e = (x_1, x_2) \in E, \nu(x_1, x_2) \approx \nu'(f(x_1), f(x_2))$

Il s'agit en fait d'autoriser des différences dans les valeurs associées aux nœuds ou aux arcs. Les isomorphismes inexacts représentent un relâchement supplémentaire dans la vérification de l'isomorphisme entre G et G', à savoir que, tant au niveau du graphe G qu'au niveau du graphe G', on peut supposer des nœuds et arcs fictifs supplémentaires afin d'obtenir un isomorphisme : Soient X_+ et X'_+ deux ensembles de nœuds fictifs et

E_+ et E'_+ deux ensembles d'arcs fictifs appartenant respectivement à $X \bigcup X_+ \times X \bigcup X_+$ et $X' \bigcup X'_+ \times X' \bigcup X'_+$

- $\forall x \in X \bigcup X_+, \exists! x' = f(x) \in X' \bigcup X'_+$
- $e = (x_1, x_2) \in E \bigcup E_+ \Leftrightarrow e' = ((f(x_1), f(x_2)) \in E' \bigcup E'_+$
- $\forall x \in X \bigcup X_+, \mu(x) \approx \mu\prime(f(x))$
- $\forall e = (x_1, x_2) \in E \bigcup E_+, \nu(x_1, x_2) \approx \nu'(f(x_1), f(x_2))$

Il existe des algorithmes optimaux (l'algorithme A* et ses variantes [23]) qui assurent une solution exacte au prix d'une complexité exponentielle, et de nombreux algorithmes approximatifs (algorithmes génétiques, recuit simulé, réseaux de neurones, relaxation probabiliste,...). Ils minimisent itérativement une fonction de coût et sont mieux adaptés à de grands graphes car plus rapides, mais leur convergence vers une solution optimale n'est pas assurée. Une distance entre graphes souvent utilisée est la distance d'édition ([34]). Elle consiste à définir des opérations d'édition et leur coût (substitution de l'attribut d'un sommet, de l'attribut d'un arc, suppression d'un sommet, d'un arc, insertion d'un arc,....). Un graphe édité est un graphe qui a subi une séquence d'opérations d'édition dont le coût est la somme des coûts élémentaires. La distance d'édition est définie comme le coût minimal du graphe édité pour lequel on a un isomorphisme avec le graphe objectif (cet isomorphisme existe toujours pour la séquence d'édition triviale qui consiste à supprimer tous les sommets de G_0 et à les remplacer par des sommets de G).
Le principe de l'algorithme A* est la construction d'un arbre de recherche par appariement successif des sommets avec évaluation de la fonction de coût à chaque état (seuls les états de coûts inférieurs sont ensuite propagés). Une amélioration possible est d'estimer une borne inférieure des coûts futurs pour ne pas propager inutilement des branches (ce qui est fait en associant chaque nœud au nœud le plus proche indépendamment des arcs).

Dans le cas d'une grande base de données de graphes modèles qu'on recherche dans une image, il peut être très intéressant de préconditionner les graphes modèles en les décomposant en sous-graphes communs de sorte à ne pas faire plusieurs fois un même appariement [34] (voir Figure2.2). Cette méthode a en particulier été adaptée dans les travaux de F. Fuchs [36] pour la reconstruction 3D de bâtiments. Il dispose d'un ensemble de graphes 3D de référence (représentant différentes formes de toits possibles, à 2-pans, à 4-pans, etc... et généralisés à l'aide d'une grammaire), qui sont stockés sous forme décomposée en adaptant l'approche de Messmer. Un graphe de données est construit mélangeant des informations polymorphes 2D et 3D (linéaires, planes, ponctuelles). Ce graphe est ensuite associé à tous les graphes de la base avec correction d'erreurs et la meilleure solution est gardée. La figure 2.3 montre un exemple de reconstruction

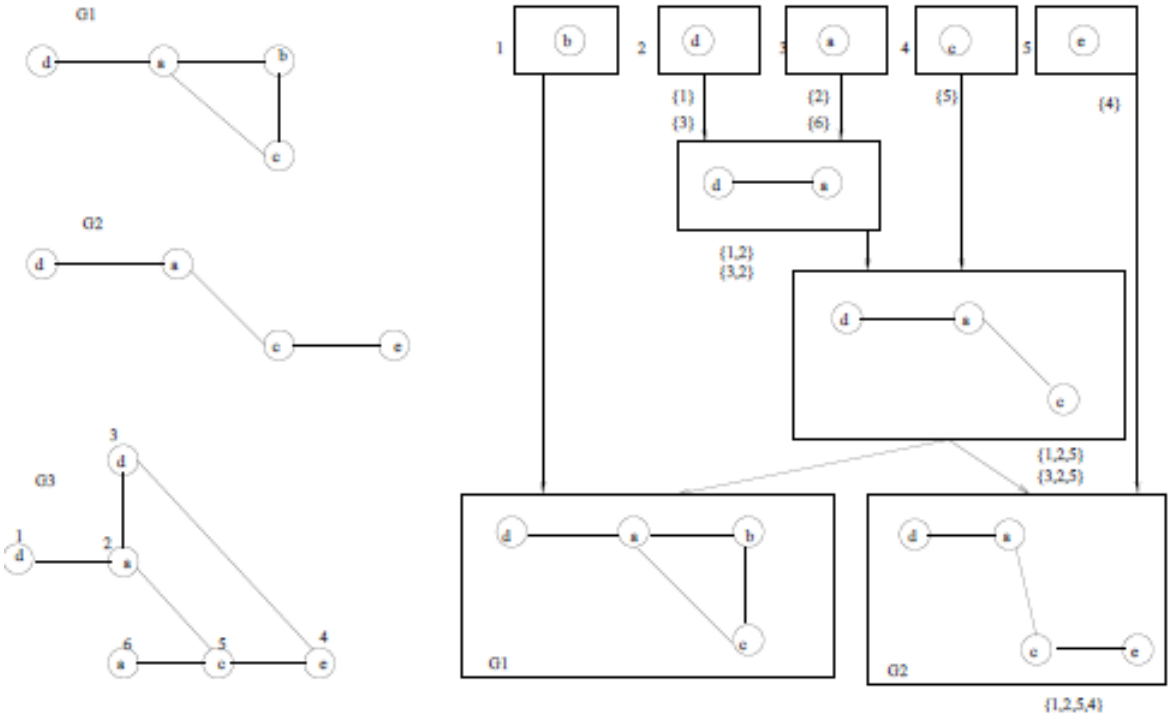


FIGURE 2.2 – Exemple de décomposition en éléments communs de deux graphes G1 et G2 et des étapes de mise en correspondance avec un graphe G3. Une grande partie des opérations d'appariement n'est faite qu'une seule fois puisque le sous-graphe d, a, c est commun à G1 et G2 (thèse de B. Messmer [35].)

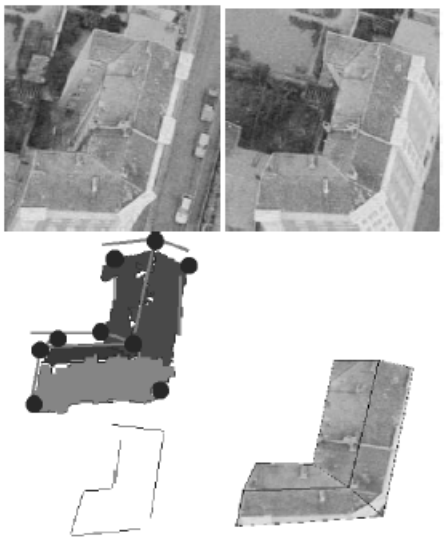


FIGURE 2.3 – Exemple de reconstruction 3D par appariement entre un ensemble de graphes modèles et un graphe de données [36].En haut les images de toit en stéréovision, en bas à gauche le modèle, en bas à droite la mise en correspondance sur l'image.

obtenue. L'une des difficultés majeure de la distance d'édition est la définition des coûts d'édition élémentaires. Ceux-ci sont généralement choisis empiriquement en fonction de l'application visée.

2.2.1.3 Appariement par algorithmes approximatifs

Dans ce cadre, la fonction de coût utilisée s'écarte d'une distance d'édition. Le problème se rapproche d'ailleurs plus d'un problème d'étiquetage puisque le graphe de référence représente des étiquettes possibles pour les données, entre lesquelles on veut respecter certaines relations. Dans ce cas, plusieurs nœuds du graphe de données peuvent être associés à un nœud du graphe modèle, et on introduit également le nœud vide pour prendre en compte le bruit des données ou l'incomplétude du modèle. Beaucoup de travaux se sont appuyés sur une approche probabiliste, notamment ceux de l'équipe de E. Hancock [37, 38], et de J. Kittler [39]. La solution cherchée maximise alors la probabilité a posteriori de f conditionnellement aux graphes modèle et de données. Les difficultés résident alors dans l'expression des probabilités (plusieurs formes ont été proposées faisant des hypothèses simplificatrices d'indépendance différentes), et dans le schéma d'optimisation (recuit simulé, recuit simulé par champ moyen, méthodes de relaxation avec règles de mise à jour, etc.). Néanmoins, il n'est pas nécessaire de passer par un formalisme probabiliste. La fonction de coût peut être définie de façon plus intuitive par une combinaison de fonctions de similarité entre les nœuds et les arcs associés. Les algorithmes d'optimisation peuvent être très divers, par exemple des algorithmes génétiques, des algorithmes gloutons, une méthode tabou, ou des algorithmes d'estimation de distributions (EDAs) [40].

2.3 Interprétation par vérification de contraintes : consistance d'un graphe

Toutes les approches présentées précédemment, basées sur l'appariement de graphes, sont à la base couteuses en temps, l'accélération des temps de calcul tirant souvent avantage d'une possibilité de représenter les deux graphes dans un même espace vectoriel et de bénéficier ainsi d'opérations de transformation rapides propres aux espaces vectoriels, permettant de vérifier des isomorphismes partiels et "approximatifs". La grande différence de nature entre un graphe d'adjacence de régions dans une image et un graphe sémantique décrivant un objet n'est pas toujours favorable à l'émergence de ce type de solution. Pour réduire les temps de calcul dans la recherche d'un appariement, une autre piste qui a été explorée est de réduire l'espace de recherche en éliminant les solutions

localement inconsistantes. Cette piste a été explorée par plusieurs auteurs qui ont encodé le problème comme un problème de satisfaction de contrainte. La vérification de contraintes a ainsi été utilisée pour reconnaître des formes comme par exemple des reconnaissances d'empreintes de pattes de rat, ou des reconnaissances de dessins). Nous présentons dans la suite la définition d'un problème de satisfactions de contraintes dans un domaine fini.

2.3.1 Définition

On utilise les conventions suivantes : Les variables sont représentées par les entiers naturels 1,. . . , n. Chaque variable i possède un domaine associé D_i. D est l'union de tous les domaines D_i et d la taille du plus grand domaine

> *Pour donner un caractère concret dans le cadre de l'analyse d'image pour la reconnaissance d'un visage par exemple, les variables pourraient être les différentes parties du visage (nez, bouche, œil droit, œil gauche, ...), chaque nom étant associé de façon biunivoque un entier naturel. Le domaine de chaque variable est un ensemble, et ce peut être l'ensemble des patchs de pixels (régions) issus d'une segmentation.*

Toutes les contraintes sont binaires et mettent en relation deux variables. Une contrainte entre deux variables i et j est notée C_{ij}.

> *Toujours dans le cas de l'analyse d'image, ces contraintes peuvent être par exemple spatiales (une région du nez doit être au dessus d'une région de la bouche) ou concerner des niveaux de gris (une région du sourcil doit être plus sombre qu'une région de la paupière).*

$C_{ij}(v, w)$ est une valeur booléenne obtenue lorsque les variables i et j sont remplacées par les valeurs v et w respectivement. $\neg C_{ij}(v, w)$ note la négation de la valeur booléenne $C_{ij}(v, w)$. On note R l'ensemble de ces relations de contrainte.

Un problème de satisfaction de contrainte dans un domaine fini consiste à trouver tous les ensembles de n-uplets $(a_1, ..., a_n) \in D_1 \times ... \times D_n$, pour (1,. . . ,n) satisfaisant toutes les relations appartenant à R.
On associe un graphe G à un problème de satisfaction de contrainte de la façon suivante :

1. G possède un nœud i pour chaque variable i.
2. Un arc orienté (i, j) est associé à chaque contrainte C_{ij}.
3. Arc(G) est l'ensemble des arcs de G et e est le nombre d'arc de G.
4. Node(G) est l'ensemble des nœuds de G et n est le nombre de nœuds de G.

Il s'agit donc de vérifier la consistance de ce graphe. La vérification de la consistance globale d'un graphe étant un problème NP-complet, on se contente souvent de vérifier la consistance d'arcs que nous décrirons plus loin, ou parfois la consistance de chemin.

Plusieurs algorithmes de vérification de la consistance d'arcs d'un graphe ont été proposés (AC3, AC4, AC5, AC6 ...)[7–10, 41–43], notamment pour optimiser le nombre de vérifications nécessaires.

2.3.2 Limite des approches classiques

Dans cette définition classique des Problèmes de Satisfaction de Contraintes dans un Domaine Fini, une variable est associée à une seule valeur. Cette hypothèse ne peut pas s'appliquer à certaines classes de problèmes où il est nécessaire d'associer à une variable plusieurs valeurs. Par exemple, dans une image, une variable représente le nom des régions segmentées et les valeurs sont les régions segmentées. En cas de sur-segmentation, plusieurs régions (valeurs) peuvent être associées au même nom (même variable). Les applications à la reconnaissance des formes sont donc limitées car cela suppose que la segmentation de l'image soit parfaite (pas d'objets sur-segmentés) ce qui est très rare dans la pratique.

2.4 Conclusion

Nous venons de voir que la représentation par graphe est un outil qui est utilisé de façon très diverse en traitement des images. Même si des applications de bas niveau comme la segmentation peuvent tirer avantageusement parti des techniques fournies par la théorie des graphes, leur utilisation est le plus souvent dédiée à l'interprétation d'images et à la reconnaissance des formes par appariement d'un graphe de données et d'un graphe modèle.

Chapitre 3

Interprétation d'images avec des objets non prévus dans le modèle et des occlusions : notion de consistance d'arcs faible

3.1 Introduction

L'appariement non-univoque entre des données et un graphe sémantique dans le cadre des PSC sur un domaine fini, n'a pas été possible jusqu'à ce que l'introduction de la consistance d'arcs avec deux niveaux de contraintes soit proposée ($FDCSP_{BC}$) [44, 45]. Cette extension a généré un nouvel intérêt pour l'application de ce type d'approche logique à l'interprétation des images. Grâce à cela il est en effet devenu possible d'associer une variable (nœud du graphe) à plusieurs valeurs (régions segmentées de l'image). Cependant, cette extension est seulement appropriée à une mise en correspondance surjective, c'est à dire lorsqu'il est possible d'avoir des données segmentées (régions) associées à chaque nœud, et que toutes les données peuvent toujours être associées à un nœud. En termes d'interprétation d'image, cela signifie qu'une région segmentée peut toujours être interprétée par rapport au graphe sémantique.
L'hypothèse de la surjectivité peut être mise en défaut dans deux situations :

— Dans le cas de la sous-segmentation d'une image. Il est parfois possible d'éviter ce problème avec un réglage ad hoc de l'algorithme de segmentation.

— La présence d'un objet inattendu dans une image est une autre situation plus difficile. Ce cas peut être rencontré dans les images médicales, par exemple lorsqu'il y a une tumeur qui ne fait pas partie du graphe sémantique décrivant l'anatomie

normale. Comme les tumeurs peuvent apparaître un peu partout, il est impossible de créer un graphe sémantique avec un nœud dont les liens sémantiques ne sont pas connus au préalable. Avec des objets inattendus, certaines données ne seront donc pas associées à un nœud et la stratégie d'appariement décrite précédemment échouera.

La question est donc de savoir comment, dans ces cas de figures, utiliser les contraintes du graphe sémantique afin de pouvoir faire correspondre les données avec la représentation de la connaissance. Nous proposons une extension de l'algorithme de vérification de l'arc-consistance à deux niveaux de contraintes (AC_{BC}) afin de résoudre le problème de la satisfaction de contraintes quand un objet manque ou lorsqu'un objet inattendu, comme une tumeur dans l'anatomie du cerveau, apparaît. A cet effet, nous introduirons deux nouvelles notions : l'arc-consistance indirecte et la quasi arc-consistance. Cet algorithme a été appliqué à des images cérébrales obtenues par résonance magnétique nucléaire. L'intégration de ces deux nouvelles notions de consistance d'arcs permet de détecter correctement une tumeur parmi les régions segmentées, sans interférer avec l'étiquetage des autres régions.

Ce chapitre est organisé comme suit : Dans la section 3.2, nous donnerons la définition de la $FDCSP_{BC}$ et la définition du problème de consistance d'arcs associé AC_{BC} [20] qui permet de faire face à une appariement non-injectif entre les données et un graphe sémantique. A partir de ces définitions, nous proposerons dans la section 3.3 deux nouvelles extensions de l'algorithme $AC4_{BC}$ permettant de résoudre le problème d'une mise en correspondance non-univoque (relations non fonctionnelles). Dans la section 3.4, nous décrirons une application de ces extensions concernant l'interprétation des images cérébrales par résonance magnétique nucléaire. Cette application ne serait pas possible sans la notion de consistance d'arcs faible. La Section 3.5 donne les conclusions de ce travail.

3.2 Notions de base et travaux antérieurs

3.2.1 Satisfaction de Contraintes à deux niveaux de contraintes

Dans le chapitre 2 nous avons vu que dans la définition classique des FDCSP, une variable est associée à une valeur. Cette hypothèse ne peut pas tenir pour certaines classes de problèmes où nous avons besoin d'associer une variable à un ensemble de valeurs liées, comme cela est décrit dans [45]. Nous appelons cette nouvelle classe de problèmes, les problèmes de satisfaction de contraintes dans un domaine fini à deux niveaux de contraintes ($FDCSP_{BC}$). Dans ce type de problèmes, nous définissons deux types de contraintes : les contraintes binaires inter-nœuds C_{ij} entre deux nœuds et les

contraintes binaires intra-nœuds Cmp_i entre deux valeurs qui pourraient être associées au nœud i. Le problème est défini comme suit (nous reprenons les notations définies au chapitre 2) :

Définition 1. Soit Cmp_i une relation de compatibilité telle que (a, b) $\in Cmp_i$ ssi a et b sont compatibles. Soit C_{ij} une contrainte entre i et j. Considérons un couple (S_i , S_j) tel que $S_i \subset D_i$ et $S_j \subset D_j$, $S_i, S_j \models C_{ij}$ signifie que (S_i, S_j) satisfait la contrainte orientée C_{ij} .

$$S_i, S_j \models C_{ij} \Leftrightarrow \begin{cases} \forall a_i \in S_i,\ \exists (a'_i, a_j) \in S_i \times S_j, \\ \textit{tel que } (a_i, a'_i) \in Cmp_i \textit{ et } (a'_i, a_j) \in C_{ij} \end{cases}$$

REMARQUES : 1) Dans le cadre de l'analyse d'images, S_i et S_j sont des ensembles de régions segmentées. 2) $\forall a_i \in S_i$, de façon évidente $(a_i, a_i) \in Cmp_i$

Les ensembles $S_1, ..., S_n$ satisfont $FDCSP_{BC}$ ssi $\forall C_{ij}$, on a S_i, $S_j \vDash C_{ij}$.

Comme dans le cas des $FDCSP$ classiques, on associe un graphe G à un problème de satisfaction de contrainte de la façon suivante : (1) G possède un nœud i pour chaque variable i. (2) Un arc orienté (i, j) est associé à chaque contrainte C_{ij}. (3) Arc(G) est l'ensemble des arcs de G et e est le nombre d'arcs de G. (4) Node(G) est l'ensemble des nœuds de G et n est le nombre de nœuds de G.

3.2.1.1 Le problème de consistance d'arcs à deux niveaux contraintes.

L'algorithme classique de vérification de la consistance d'arcs ne peut pas classer un ensemble de données dans un nœud du graphe comme nous aimerions le faire pour interpréter des images sur-segmentées.
Nous définissons donc une nouvelle classe de problèmes appelée problèmes de consistance d'arcs à deux niveaux de contraintes (AC_{BC}). Elle est associée au $FDCSP_{BC}$ (voir Définition 1) et est définie comme suit :

Définition 2. Soit $(i, j) \in arc(G)$. Soit $P(D_i)$ l'ensemble des parties de D_i. Arc(i,j) est arc-consistant étant donné $P(D_i)$ et $P(D_j)$ ssi $\forall S_i \in P(D_i)$, $\exists S_j \in P(D_j)$ tel que $\forall v \in S_i$ $\exists t \in S_i$, $\exists w \in S_j$: $Cmp_i(v, t)$ et $C_{ij}(t, w)$ (v et t peuvent être identiques).

La définition d'un graphe arc-consistant devient :

Définition 3. Soit $P(D_i)$ l'ensemble des parties du domaine D_i. Soit P = $P(D_1) \times ... \times P(D_n)$. Un graph G est arc-consistant étant donné P ssi $\forall (i, j) \in arc(G)$: (i, j) est arc-consistant étant donnés $P(D_i)$ et $P(D_j)$.

La but d'un algorithme de vérification de l'arc consistance à deux niveaux de contraintes est, étant donné G et un ensemble P, de calculer P' le plus grand domaine arc-consistant

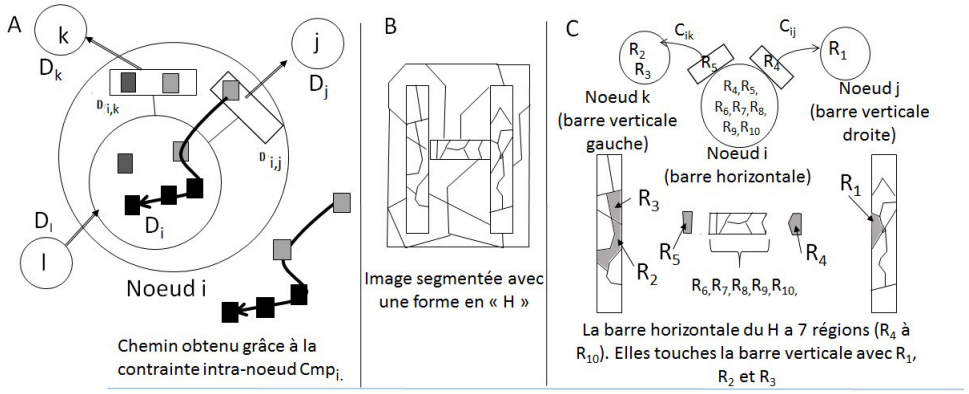


FIGURE 3.1 – A. Consistance d'arcs à deux niveaux de contraintes : Quatre nœuds i, j, k et l sont représentés et le nœud i est détaillé. Un nœud est constitué d'un ensemble d'interfaces (une par arc) et d'un noyau (correspondant à l'ensemble du domaine D_i). Les labels du nœud i sont contraints par deux arcs orientés (i, j) et (i, k) et les labels du nœud l sont contraints par l'arc orienté (l, i).
Les carrés gris et noirs sont des labels (régions segmentées) associés au nœud i.
Les labels noirs n'appartiennent pas à $D_{i,j}$. Cependant, grâce à la relation de compatibilité $\mathcal{C}mp_{\mathrm{i}}$, ces labels sont atteignables à partir du label gris, ils peuvent donc être gardés dans le nœud i. $D_{i,j}$ est appelé "Interface de i avec j" dans l'algorithme décrit plus loin). $D_{i,j}$ est l'ensemble à partir duquel les autres régions de i peuvent être atteintes étant donné $\mathcal{C}mp_{\mathrm{i}}$.
C. Détail du nœud i "barre horizontale du H". C_{ij} contraint une région R de i pour qu'elle touche à sa droite la région R' de j. C_{ik} contraint une région R de i pour qu'elle touche sur sa gauche une région R' de k. $\mathcal{C}mp_{\mathrm{i}}$ entre R de $D_{i,j}$ et R' de i est vraie (R et R' sont compatibles étant donné $\mathcal{C}mp_{\mathrm{i}}$) si il existe un chemin de régions connectées entre R et R' tel que toutes les régions de ce chemin soit "à gauche" de leur prédécesseur et "à gauche" de R.

à deux niveaux de contraintes pour G dans P. Dans notre contexte l'ensemble P contient les ensembles de régions segmentées qui satisfont les contraintes imposées par le graphe sémantique.

3.2.1.2 Algorithme de consistance d'arcs à deux niveaux de contraintes ($AC4_{BC}$)

Considérant les remarques précédentes, nous proposons un nouvel algorithme avec des contraintes à deux niveaux. Dans ce but, nous adaptons l'algorithme AC4 proposé par Mohr et Henderson en 1986 [42] [46] pour résoudre le problème AC_{BC}. On appelle cet algorithme $AC4_{BC}$ (voir [45] et Fig 3.3, 3.4 et 3.5 pour les détails de l'algorithme). Dans $AC4_{BC}$, on donne une nouvelle définition d'un nœud i appartenant à node(G). A présent, un nœud est constitué d'un noyau D_i et d'un ensemble d'interfaces D_{ij} associées aux arcs venant d'un autre nœud connecté au nœud i (voir Fig. 3.1). De plus, une relation de compatibilité intra-nœud Cmp_i est associée à chaque nœud du graphe. Elle décrit le lien sémantique entre les différentes sous-parties de l'objet qui peuvent être associées au nœud. La contrainte intra-nœud Cmp_i peut être spatiale ou morphologique comme on peut le voir Figure 3.2.

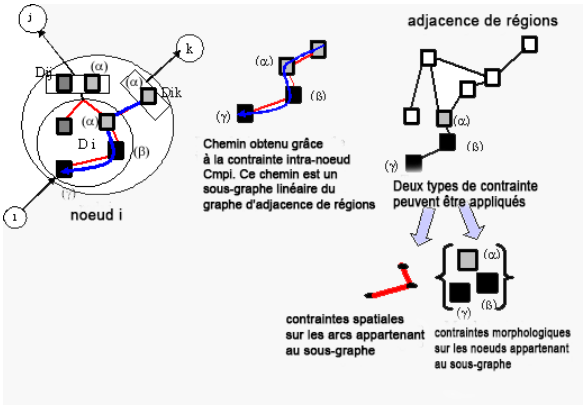


FIGURE 3.2 – Différents types de contraintes intra-nœud (Contraintes $\mathcal{C}$mpi). Les valeurs α, β et γ (régions segmentées) peuvent être associées au nœud i représentant un objet sémantique. Dans cet exemple $\alpha \in D_{ik}$, $\beta \notin D_{ik}$, $\gamma \notin D_{ik}$ and $\alpha \in D_{ij}$, $\beta \notin D_{ij}$, $\gamma \notin D_{ij}$. Dans un algorithme classique de contrôle de la consistance d'arcs, les valeurs β et γ auraient été retirées du nœud i car elles ne sont pas supportées par les autres régions. Grâce aux contraintes intra-nœud $\mathcal{C}$mpi, β et γ peuvent être gardées dans le nœud i car un chemin peut être trouvé entre la valeur α et les valeurs β et γ dans le graphe d'adjacence de régions représentant la segmentation de l'image.

Définition 4. Soit $i \in node(G)$, on note D_i est le domaine correspondant au noyau de i et on note $I_i = \{D_{ij} | (i,j) \in arc(G)\}$ l'ensemble des domaines des interfaces du nœud i.

L'algorithme est constitué de deux étapes principales : Une étape d'initialisation et une étape d'élagage. Cependant, alors que dans AC_4 une valeur était supprimée d'un nœud i si elle ne possédait pas de support direct, dans l'algorithme $AC4_{BC}$, une valeur est supprimée si elle n'a ni support direct ni support indirect obtenu en utilisant la relation de compatibilité Cmp_i.
Pour chaque $i \in node(G)$, la première étape consiste à initialiser les domaines D_i et D_{ij}. Cette étape se fait en deux temps :

— affecter au noyau de D_i toutes les valeurs b qui satisfont les contraintes de nœud unaires, comme dans l'algorithme AC_4 (par exemple, en analyse d'image on pourra considérer des critères d'intervalle des valeurs de couleur, de largeur, de hauteur et de taille des régions comme critères unaires) ;

— affecter aux interfaces D_{ij}, toutes les valeurs $b \in D_i$ telles que $\exists c \in D_j$ $C_{ij}(b,c)$ (en analyse d'image on affecte aux interfaces D_{ij} les régions qui satisfont les relations spatiales représentées par C_{ij}).

Nous avons ensuite une étape d'élagage qui supprime les valeurs qui ne satisfont pas les contraintes locales. L'étape d'élagage met à jour les nœuds en fonction des suppressions qui ont été réalisées à l'étape précédente afin de conserver la consistance d'arcs. Les propriétés de l'algorithme $AC4_{BC}$ sont prouvées dans [45].

```
begin AC_4BC
Etape 1 : Construction des structures de données.
1 InitQueue(Q) ;
2 for each i ∈ node(G) do
3  for each b ∈ D_i do
4     begin
5     S[i,b] := ensemble vide ;
6     end ;
7 for each (i, j) ∈ arc(G) do
8  for each b ∈ D_ij do
9  begin
10Total := 0 ;
11    for each c ∈ D_j do
12       if C_ij(b, c) then
13          begin
14          Total := Total +1
15          S[j,c] :=S[j,c] ∪ (i,b) ;
16          end
17    Counter[(i,j),b] := Total ;
18       if Total=0 then
19          D_ij := D_ij − {b} ;
20end ;
21 for each i ∈ node(G) do
22    for each D_ij ∈ I_i do
23    begin
24       CleanKernel(D_i, D_ij, I_i, Q) ;
25    end
```

FIGURE 3.3 – L'algorithme AC_{4BC} : Étape 1. La Figure 3.5 décrit la procédure CleanKernel appelée en ligne 24.

Théorème 1. L'algorithme $AC4_{BC}$ possède les propriétés suivantes :

1. $AC4_{BC}$ se termine toujours.
2. $AC4_{BC}$ est correct (cela signifie que l'algorithme renverra vrai et le plus grand domaine arc-consistant à deux niveaux de contraintes pour le graphe considéré si celui-ci est arc-consistant, faux sinon).

Théorème 2. La complexité de temps de l'étape d'élagage est en $O(n^2d)$ dans le pire des cas, où n est le nombre de nœuds et d est la taille du plus grand domaine D.

```
Etape 2 : Elagage des labels inconsistants
26 While not Emptyqueue(Q) do
27begin
28Dequeue(i,b,Q) ;
29   for each (j, c) ∈ S[i, b] do
30   begin
31   Counter[(j,i),c] := Counter[(j,i),c]-1 ;
32   if Counter[(j,i),c]=0 then
33      begin
34      D_ji := D_ji − {c} ;
35      CleanKernel(D_j, D_ji, I_j, Q) ;
36      end ;
37 end ;
38 end AC_4BC ;
```

FIGURE 3.4 – L'algorithme AC_{4BC} : Étape 2

```
begin CleanKernel(inD_i, D_ij, I_i, outQ)
1 begin
2       R := D_ij ;
3       while (Searchsucc(D_i, R, Cmpi, S)) do
4       begin
5          R := R ∪ S ;
6       end
7       for each b ∈ D_i - R do
8       begin
9             EnQueue(i,b,Q) ;
10               for each D_ij ∈ I_i do
11               D_ij = D_ij-{b} ;
12      end
13      D_i := R;
14 end ;
```

FIGURE 3.5 – La Procédure CleanKernel

3.3 Mise en correspondance avec des relations non fonctionnelles : le problème des objets manquants ou non prévus

Nous avons étudié (dans la section 3.2) le cas de la mise en correspondance entre plusieurs données et un nœud du graphe modèle. Deux autres cas, qui rendent cette mise en correspondance difficile, peuvent être rencontrés.

Le premier cas correspond à la présence de données supplémentaires qui ne peuvent être associées à aucun nœud. Le second cas correspond à des données manquantes qui

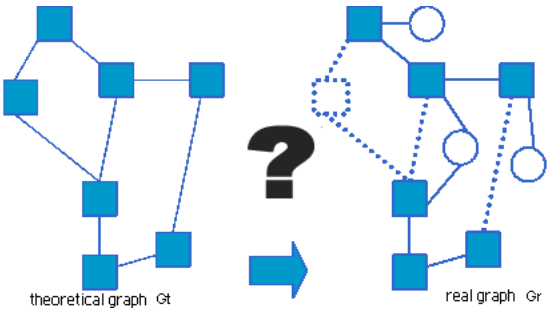


FIGURE 3.6 – Comment peut-on transformer le graphe théorique G_t (qui possède ici 7 nœuds) en un graphe réel G_r (qui possède ici 9 nœuds) représentant la véritable information ? Les carrés en pointillés et lignes en pointillés dans G_r représente les nœuds et les arcs supprimés de G_t, les cercles et leur lignes associés représentent les nœuds et leurs arcs associés de G_r qui ne sont pas dans G_t.

peuvent laisser des nœuds vides (si aucune donnée ne peut être associée à ces nœuds). Ces deux cas ne sont pas rares en analyse d'images où différents facteurs comme la présence de bruit ou une segmentation approximative peuvent conduire à des détections erronées. Dans ces cas, les hypothèses initiales associées aux $FDCSP$ et aux $FDCSP_{BC}$ ne peuvent être conservées. Il faut donc adapter le problème de satisfaction de contraintes à deux niveaux de contraintes à ces différents cas. La connaissance d'expert peut souvent être traduite en un graphe G_t décrivant une objet connu recherché dans une image donnée. Par exemple G_t peut être un graphe sémantique décrivant l'anatomie normale du cerveau : Ce graphe peut être tiré d'un livre d'anatomie [47, 48] où la description des différentes parties anatomiques est traduisible sous forme de graphe. En effet, il est possible de représenter les caractéristiques morphologiques des différentes parties par des contraintes de nœuds (forme, taille, orientation) et l'organisation spatiale de ces parties par des contraintes d'arc (relation spatiale, distance). Cependant ce graphe G_t ne peut pas être correctement mis en correspondance avec des données d'image si ces images contiennent une anatomie anormale ou si elles sont détériorées par un bruit important. Dans cette situation un graphe G_r, adapté à l'image concernée serait plus utile que G_t. Malheureusement, G_r n'est pas connu a priori et beaucoup de G_r sont possibles et non prédictibles. Par exemple il est difficile de créer tous les G_r possibles correspondant à toutes les positions et à toutes les formes de tumeurs dans le cerveau. Cependant, on sait que G_t et G_r partage des similarités car ils représentent des variations du même type d'objet (l'un est normal et l'autre est pathologique). Il est donc possible de définir des contraintes a priori sur G_r.

Le problème est le suivant : comment peut-on utiliser G_t et comment peut-on utiliser les contraintes qui doivent être satisfaites sur G_r pour réduire le nombre de solutions possibles ? Cette situation est illustrée sur la Figure 3.6.

Pour résoudre ce problème, nous proposons d'ajouter au graphe G_t un ensemble de nœuds N_s pour prendre en compte le fait que G_r puisse avoir des nœuds supplémentaires

(voir Figure 3.6). Le résultat de cette extension génère le graphe G' défini de la façon suivante :
Soit Arc(G')=Arc(G_t) $\cup\{(i,j) \mid i \in Node(G_t)$ *and* $j \in N_s$ *or* $j \in Node(G_t)$ *and* $i \in N_s\}$ et Nœud(G')= Nœud(G_t) $\cup N_s$. Trois cas peuvent être rencontrés :

1. *On sait que des données sont manquantes.*
   Dans ce cas, certains arcs peuvent être inconsistants et il est nécessaire de prendre cette possibilité en compte. Pour cela, on définit la notion de quasi-arc consistance et on associe à chaque nœud i un nombre maximum d'arcs inconsistants qui lui sont connectés. Si ce nombre vaut 0 on se retrouve dans la consistance d'arcs classique. Plus ce nombre est élevé plus on autorise un relâchement de l'exigence de consistance. Ce type d'approche a déjà été proposée par Freuder et Wallace [9] pour définir la satisfaction partielle de contraintes dans le cas classique. Nous allons donc adapter cette approche à la satisfaction de contrainte à deux niveaux de contraintes dont la définition a été donnée dans la section 3.1.

2. *On sait qu'il y a des données supplémentaires.*
   Il est nécessaire de définir un nouvel ensemble de nœuds N_s pour ces données supplémentaires non décrites dans le graphe initial G_t. Les connaissances d'experts concernant ces données supplémentaires associées à l'image donnée aident à définir N_s.
   Si les nouvelles relations entre les nœuds de G_t et N_s étaient connues, on obtiendrait un nouveau graphe G_r. Nous limitions notre étude aux situations où l'ajout d'un nouveau nœud génère un effet minimum sur le graphe G_t. Beaucoup de G_r différents peuvent malgré tout être rencontrés avec ces limites. Pour ces G_r, on suppose que les éléments de $Arc(G_r)$ possèdent les propriétés suivantes relativement à G_t et N_s :

$$(i,j) \in Arc(G_r) \Rightarrow \begin{cases} (i,j) \in Arc(G_t), \\ ou\ i \in N_s\ et\ j \in Node(G_t) \\ ou\ j \in N_s\ et\ i \in Node(G_t) \end{cases}$$

$$(i,j) \in Arc(G_t) \Rightarrow \begin{cases} (i,j) \in Arc(G_r), \\ ou\ (i,k) \in Arc(G_r)\ et\ k \in N_s\ et\ (k,j) \in Arc(G_r) \end{cases}$$

   Étant données ces propriétés, les arcs du graphe G' défini précédemment, doivent avoir la propriété suivante :

$$(i,j) \in Arc(G') \Leftrightarrow (i,j) \in Arc(G_t)\ ou\ (i,j) \in Arc(G_r)$$

Pour gérer le fait que le nœud i peut être connecté directement à un nœud j ($(i,j) \in G_t$) ou indirectement, à travers un nœud $k \in N_s$, à un nœud j, on introduit la notion de consistance indirecte. La définition formelle est donnée dans la section 3.3.2.

3. *On ne sait pas si des données sont manquantes ou s'il peut en apparaître de nouvelles.*
   Dans ce cas, le nombre d'arcs inconsistants pour chaque nœud ne peut être posé définitivement. Il est nécessaire d'augmenter ce nombre pendant le processus de relaxation jusqu'à ce que l'on trouve une solution.

Ces deux nouvelles notions de quasi arc-consistance et d'arc consistance indirecte sont appelées des arc-consistance faibles.

3.3.1 Quasi arc-consistance

De façon à mettre en correspondance des données avec un graphe de sémantique dans le cas d'une relation non fonctionnelle, nous proposons de permettre un relâchement de contraintes. Ce processus est utile lorsque les objets décrits dans le modèle ne sont pas présents dans l'image (par exemple lorsqu'il y a une occlusion).
Ce relâchement est introduit dans l'algorithme $AC4_{BC}$ de façon à ce que chaque nœud du graphe possède une information supplémentaire concernant le nombre de relâchements de contraintes autorisé. Cette information est notée $Relax(i)$ pour le nœud i.

Soit Γ_i l'ensemble des nœuds connectés au nœud i dans le graphe G tel que :
$\Gamma_i = \{j \in node(G), (i,j) \in arc(G)\}$

La fonction $Relax(i)$ associe à un nœud i le nombre nb de relâchements de contraintes dans le graphe G et est telle que :

$$\begin{array}{rccl} Relax: & node(G) & \rightarrow & \mathbb{N} \\ & i & \mapsto & Relax(i) = \quad nb \, tel \, que \, nb \geq 0 \, et \, nb \leq Card(\Gamma_i) \end{array}$$

Cette fonction permet que nb contraintes binaires associées au nœud i puissent être satisfaites.
Un exemple décrivant cette situation est illustré par la Figure 3.7. Cette notion donne la possibilité d'exprimer un OU logique entre deux contraintes. Soit $\Gamma^{ac}_{i,l}$ l'ensemble des nœuds $j \in \Gamma_i$ rendant arc-consistant un arc avec un nœud donné i et une valeur donnée $l \in D_i$ tel que :
$\Gamma^{ac}_{i,l} = \{j \in \Gamma_i \mid (i,j)$ est arc consistant en fonction de $l \in D_i \, et \, \mathcal{P}(D_j)\}$

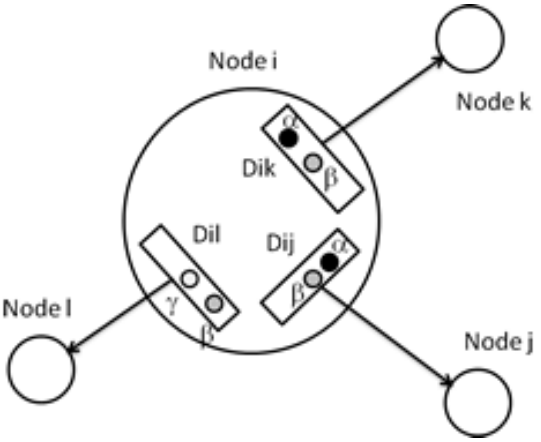


FIGURE 3.7 – Dans cet exemple, on suppose que la valeur γ ne peut pas être atteinte à partir des valeurs α ou β en utilisant la contrainte intra-nœud $\mathcal{C}$mpi. Si on suppose que le nombre de relâchements associé au nœud i est affecté à 1, l'algorithme de quasi arc-consistance ne retirera que la valeur γ. La valeur β est gardée car elle satisfait toutes les contraintes intra-nœud. La valeur α est gardée également, puisqu'elle ne satisfait pas qu'une seule contrainte (C_{il}) et ce nombre de contraintes non satisfaites est autorisé par le nombre de relâchements.

Grâce à la fonction Relax, une nouvelle classe de problèmes appelée problème de quasi-arc consistance peut être définie. La définition d'un graphe quasi arc-consistent est :

Définition 5. Soit $\mathcal{P}(D_i)$ l'ensemble des sous-parties du domaine D_i. Soit P=$\mathcal{P}(D_1)\times$ .... $\times\mathcal{P}(D_n)$. Un graphe G est quasi arc-consistent en fonction de P si et seulement si $\forall i \in node(G)$ on a

$$Card(\Gamma_i) - \min_{\forall l \in D_i}(Card(\Gamma^{ac}_{i,l})) \leq Relax(i)$$

3.3.1.1 Terminaison de AC_{4BC} avec Quasi arc-consistance

Comme l'algorithme AC_{4BC}, l'algorithme AC_{4BC} avec quasi arc-consistance fonctionne avec une queue Q contenant les éléments supprimés des domaines. Elle contient les couples (i, v), où $i \in node(G)$ et $v \in D_i$. Ces éléments sont reconsidérés par l'algorithme car ils peuvent supporter d'autres couples (j, w). En effet, si un élément supprimé était l'unique support de (j, w) alors (j, w) doit être également supprimé. Pour gérer cette queue, nous avons besoin de plusieurs opérations :

- La procédure InitQueue qui initialise la queue à l'ensemble vide.
- La fonction QueueVide qui teste si la queue est vide.
- La procédure EnQueue(i, v, Q) est utilisée à chaque fois qu'une valeur v est supprimée de D_i. Elle ajoute les éléments (i, v) dans la queue Q, où i est un nœud et $v \in D$.
- La procédure DeQueue qui supprime un élément de la queue.

L'algorithme AC_{4BC} avec quasi arc-consistance doit alors préserver l'invariant suivant :

$$\begin{aligned}\text{Status(i,b)} = \text{présent ssi } & b \in D_i \\ & \text{rejeté ssi } b \notin D_i \text{ et } (i,b) \notin Q \\ & \text{suspendu ssi } b \notin D_i \text{ et } (i,b) \in Q \\ & \text{rejeté d'une interface ssi } \exists j \in node(G),\ b \notin D_{ij} \\ & \text{et } b \in D_i.\end{aligned}$$

Cet invariant décrit les différents états de la valeur b étant donné un nœud i.
Plus intuitivement, les différents états ont les significations suivantes :

- l'état "présent" signifie que le label b satisfait toutes les contraintes imposées sur le nœud i et appartient donc au domaine de ce nœud.
- l'état "rejeté" signifie que le label b ne satisfait pas toutes les contraintes imposées sur le nœud i mais n'est pas encore déposé dans la queue des suppressions Q.
- l'état "suspendu" signifie que le label b ne satisfait pas toutes les contraintes imposées sur le nœud i et est déposé dans la queue des suppressions Q.
- l'état "rejeté d'une interface" signifie que le label b est supprimé du domaine de l'interface D_{ij} associée à la contrainte C_{ij} car il ne satisfait pas cette contrainte, mais n'est pas encore supprimé du domaine du nœud i.

Dans cet algorithme, la procédure CleanKernel définie dans [45] est modifiée comme indiqué dans la Figure 3.10. Dans cette procédure, la suppression d'un label b de D_i est faite en fonction du nombre de relâchements autorisés par la fonction $Relax(i)$. La post-condition de CleanKernel devient :
procédure CleanKernel (in D_i, I_i, inout Q)
Pre : $i \in node(G)$, $D_i \neq \{\}$, $\forall D_{ij} \in I_i$, $D_{ij} \neq \{\}$
Post :$\Delta_i = \{b \in D_i \mid \exists Relax(i)+1$ interfaces $D_{ij_n;n=0...Relax(i)} \in I_i$,
$\neg Path_i(b, D_{ij_n;n=0...Relax(i)})\}$ et $\forall b \in \Delta_i$ Status(i,b)= suspendu et $Q = Qprev \cup \Delta_i$.
On peut donc prouver le théorème suivant :

Théorème 1. L' algorithme AC_{4BC} avec Quasi arc-consistance (Cf. Figure 3.8, Figure 3.9 et Figure 3.10) possède les propriétés suivantes :
(1) L'invariant sur le statut de la structure de données est vrai en ligne 2 et 23.
(2) AC_{4BC} avec Quasi arc-consistance enfile et défile au plus en O(nd) éléments, et donc la taille de la queue est au plus en O(nd), où n est le nombre de nœuds et d est le nombre de données (la taille du plus grand domaine).
(3) AC_{4BC} avec Quasi arc-consistance se termine toujours.

Preuve :
La preuve est similaire à celle prouvant la terminaison de l'algorithme classique AC_{4BC}. Pour plus de détails voir [45].

3.3.1.2 AC_{4BC} avec Quasi arc-consistance est correct

Théorème 2. G est quasi arc-consistant quand AC_{4BC} avec Quasi arc-consistance se termine.

Preuve :
Hypothese initiale : soit $i \in node(G)$ et $Relax(i)$ le nombre de relâchement autorisés associé à ce nœud. Alors i est tel que $\exists Relax(i)+1$ nœuds $\in \Gamma_i$ et $D_i \neq \emptyset$. Considérons $c \in D_i$. On suppose que c n'est pas supporté directement ni via une contrainte intra-nœud (indirectement) par les nœuds $j_{n;n=0...Relax(i)}$, $j_n \in \Gamma_i$ lorsque AC_{4BC} avec Quasi arc-consistance se termine.
Autrement dit : $\forall b \in D_{j_n;n=0...Relax(i)}, (\forall a \in D_{ij_n}, \neg\mathcal{C}\text{mpi}(c,a)$ ou $\neg C_{ij_n}(a,b))$.
Si $c \in D_i$ n'est pas supporté :

— soit (1) il n'a jamais été supporté.

— soit (2) il a été supporté précédemment.

Cas(1) :
Dans ce cas, il y a une contradiction avec $c \in D_i$ car l'étape d'initialisation n'aurait pas mis c dans D_i (Ligne 11 de AC_{4BC} Figure 3.8).

Cas(2) :

— Pour n=0 ... Relax(i), soient $\{b_n^1, ...b_n^m\}$ $(m > 0)$ les ensembles d'éléments de $D_{j_n;n=0...Relax(i)}$ supportant c à une étape précédente. Puisqu'à la fin c n'est pas supporté, cela signifie que les $b_n^1, ...b_n^m; n = 0...Relax(i)$ sont supprimés pendant l'exécution de AC_{4BC} avec quasi arc-consistance, de $D_{j_n;n=0...Relax(i)}$.

— La suppression des $b_n^1, ...b_n^m; n = 0...Relax(i)$ les insère dans la queue Q.

— Tous ces éléments $b_n^1, ...b_n^m; n = 0...Relax(i)$ sont nécessairement retirés de Q lorsque AC_{4BC} termine et les compteurs Counter$[(j_n, i), c]$; n=0 ... Relax(i) deviennent nécessairement égaux à zéro lorsque AC_{4BC} termine.

— A ce moment-là c est supprimé des interfaces $D_{ij_n;n=0...Relax(i)}$.

— D'après l'hypothèse initiale, c n'est pas supporté indirectement par un chemin intra-nodal.
Cela signifie que $\forall a \in D_{ij_n;n=0...Relax(i)}$, $\neg\mathcal{C}\text{mpi}(c,a)$ ou $\neg C_{ij_n}(a,b)$. Nous avons donc deux cas à étudier :

— Dans un premier temps, si nous avons $\forall a \in D_{ij_n;n=0...Relax(i)}$,$\neg\mathcal{C}\text{mpi}(c,a)$, alors nous avons $\neg Path_i(c, D_{ij_n;n=0...Relax(i)})$.
Dans ce cas CleanKernel supprime c de D_i (lignes 4-11 de la Figure 3.10) ce qui est le contraire de l'hypothèse initiale.

```
begin AC_4BC
Etape1 : Construction des structures de données.
1 InitQueue(Q) ;
2 for each i ∈ node(G) do
3  for each b ∈ D_i do
4     begin
5     S[i,b] := ensemble vide ;
6     end ;
7 for each (i, j) ∈ arc(G) do
8  for each b ∈ D_ij do
9  begin
10Total := 0 ;
11   for each c ∈ D_j do
12      begin
13      if C_ij(b, c) then
14         begin
15         Total := Total +1
16         S[j,c] :=S[j,c] ∪ (i,b) ;
17         end
18         end
19   Counter[(i,j),b] := Total ;
20      if Total=0 then
21         D_ij := D_ij − {b} ;
22end ;
23 for each i ∈ node(G) do
24   CleanKernel(D_i, I_i, Q) ;
```

FIGURE 3.8 – L'algorithme AC_{4BC} avec quasi arc-consistance : Étape 1. Figure 3.10 décrit la procédure CleanKernel.

— Le deuxième cas est $\forall b \in D_{j_n;n=0,\ldots,Relax(i)}$, $\exists a \in D_{ij_n}$, $\mathcal{C}\text{mpi}(c,a)$ et $\neg C_{ij_n}(a,b)$. Ce cas ne peut pas arriver. En effet $\forall b \in D_{j_n;n=0\ldots Relax(i)}\ \neg C_{ij_n}(a,b) \Rightarrow a \notin D_{ij_n}$. L'affirmation contra positive est obtenue
$a \in D_{ij_n;n=0\ldots Relax(i)} \Rightarrow \exists b \in D_{j_n}\ C_{ij_n}(a,b)$. Donc $\forall a \in D_{ij_n;n=0\ldots Relax(i)}$, $\mathcal{C}\text{mpi}(c,a) \Rightarrow \exists b \in D_{j_n}$ telle que $C_{ij_n}(a,b)$

L'hypothèse initiale mène donc à une contradiction. Donc G est quasi arc-consistant lorsque AC_{4BC} avec Quasi arc-consistance se termine.

3.3.2 La trans-consistance

On peut imaginer que deux nœuds d'un graphe peuvent être reliés directement ou indirectement via un autre nœud. Ceci peut se produire en interprétation d'images cérébrales lorsqu'une tumeur apparaît entre deux structures anatomiques cérébrales. Si la tumeur n'est pas trop grosse, elle préserve l'architecture cérébrale. Dans ce cas, les relations entre

```
Etape2 : Elagage des labels inconsistants
23 While not Emptyqueue(Q) do
24begin
25Dequeue(i,b,Q) ;
26   for each (j, c) ∈ S[i, b] do
27   begin
28   Counter[(j,i),c] := Counter[(j,i),c]-1 ;
29   if Counter[(j,i),c]=0 then
30      begin
31      D_ji := D_ji − {c} ;
32      CleanKernel(D_j, I_j, Q) ;
33      end ;
34 end ;
35 end AC_4BC ;
```

FIGURE 3.9 – L'algorithme AC_{4BC} avec quasi arc-consistance : étape 2

```
begin CleanKernel(inD_i, I_i, outQ)
1 begin
2  for each b ∈ D_i do
3  nbrelach=0 ;
4     for each D_ij ∈ I_i do
5     if ¬Path_i(b, D_ij) then
6        begin
7        nbrelach=nbrelach+1 ;
8           if nbrelach > Relax(i) then
9           begin
10             EnQueue(i,b,Q) ;
11             D_i = D_i-{b} ;
12                for each D_ij ∈ I_i do
13                begin
14                D_ij = D_ij-{b} ;
15                end
16          end
17       end
18 end ;
```

FIGURE 3.10 – La Procédure CleanKernel avec quasi-arc consistance

la tumeur et chaque structure sont les mêmes que les relations définies entre les deux structures. Pour résoudre ce type de situation, on définit la notion d'arc-consistance indirecte (trans-consistance). Par simplification, on utilise C^a_{ij} pour noter une contrainte reliant deux nœuds i et j en fonction de la relation élémentaire $a \in \Sigma_T$ où $\Sigma_T = \{a_1, ..., a_n\}$ avec les relations élémentaires $a_1...a_n$ (c'est à dire, par exemple "sur la droite", "sur la gauche", "au dessus", "en dessous", ...). C^a_{ik} et C^a_{kj} signifient que ces contraintes sont de même type que la contrainte C^a_{ij}. Soit la contrainte intra-nœud $\mathcal{C}_{mpk}$ du nœud k (Par exemple, pour la détection de tumeurs, la caractéristique d'une telle structure peut être une forme ovoïde).

Définition 6. Soit $(i,j) \in arc(G)$. Soit $\mathcal{P}(D_i)$ l'ensemble des sous-parties du domaine D_i. L'arc (i,j) est indirectement consistant via un nœud k en fonction de $\mathcal{P}(D_i)$ et $\mathcal{P}(D_j)$:

- si $\forall S_i \in \mathcal{P}(D_i)\ \exists S_j \in \mathcal{P}(D_j)$ tel que $\forall v \in S_i\ \exists (t,w) \in (S_i \times S_j) : \mathcal{C}_{mpi}(v,t)$ et $C^a_{ij}(t,w)$, (Ce cas correspond à l'arc-consistance à deux niveaux de contraintes)
- ou si $\forall S_i \in \mathcal{P}(D_i), \forall v \in S_i, \exists S_k \in \mathcal{P}(D_k), \exists S_j \in \mathcal{P}(D_j), \exists (t,m,n,w) \in (S_i \times S_k \times S_k \times S_j)$
  tel que $\mathcal{C}_{mpi}(v,t)$ *et* $C^a_{ik}(t,m)$ *et* $\mathcal{C}_{mpk}(m,n)$ *et* $C^a_{kj}(n,w)$.

On définit la fonction d'indirection appelée $Indirect$. Elle associe à chaque arc un nœud à travers lequel l'arc peut être indirectement arc-consistant.

$$\begin{array}{rccc} Indirect: & E \subset Arc(G) & \rightarrow & Node(G) \\ & (i,j) & \mapsto & k \end{array}$$

La définition d'un graphe indirectement arc-consistant est :

Définition 7. Soit $\mathcal{P}(D_i)$ l'ensemble des sous-parties du domaine D_i. Soit P=$\mathcal{P}(D_1)\times$ .... $\times\mathcal{P}(D_n)$. Un graphe G associé à une fonction d'indirection $Indirect$ est indirectement arc-consistant en fonction de P ssi $\forall (i,j) \in Arc(G)$
(1) $Indirect(i,j)$ non définie $\Rightarrow (i,j)$ est arc-consistant, ou
(2) $Indirect(i,j) = k \Rightarrow (i,j)$ est indirectement arc-consistant via k.

L'algorithme calculant l'arc-consistance indirecte est décrit dans les Figures 3.13 et 3.14. Comme pour l'algorithme classique $AC4_{BC}$, l'algorithme est constitué de deux étapes : une étape d'initialisation (Figure 3.13) et une étape d'élagage (Figure 3.14). Il utilise une queue Q pour gérer les données supprimées du domaine du nœud, parce qu'elles ne satisfont par certaines contraintes. Ces suppressions doivent être prises en compte dans l'étape d'élagage. Deux compteurs, counter[(i,j),c] et counter2[(i,j),c] sont définis de façon à mettre à jour le nombre de supports directs et indirects du label c avec l'arc (i,j).

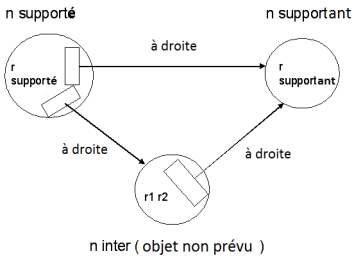


FIGURE 3.11 – Illustration d'un sept-uplet $(n_{supporté}, reg_{supporté}, n_{inter}, r1_{inter}, r2_{inter}, n_{supportant}, reg_{supportant}) \in S_2$. Les relations entre les nœuds $n_{supporté}$ et n_{inter} et les nœuds n_{inter} et $n_{supportant}$ doivent être du même type que la relation initiale entre les nœuds $n_{supporté}$ et $n_{supportant}$. Dans cet exemple, la relation est : "à la droite de". Si $reg_{supporté}$ est à la droite de $reg_{supportant}$, soit directement, soit indirectement via $r1_{inter}$ et $r2_{inter}$ du nœud n_{inter} alors $reg_{supporté}$ est gardée dans le nœud $n_{supporté}$.

Afin de calculer la consistance indirecte, l'ensemble $S2$ de sept-uplets est défini de la façon suivante :

Soit $(n_{supporté}, reg_{supporté}, n_{inter}, r1_{inter}, r2_{inter}, n_{supportant}, reg_{supportant}) \in S_2$ tel que

> $n_{supporté}, n_{inter}, n_{supportant} \in node(G)$
> et $reg_{supporté} \in D_{n_{supporté}, n_{init}}$
> et $reg_{supportant} \in D_{n_{supportant}}$ et $r1_{inter} \in D_{n_{inter}}$
> et $r2_{inter} \in D_{n_{inter}, n_{supportant}}$.

Les composants de ce sept-uplet ont la signification suivante :

Le nœud $n_{supporté}$ est contraint par le nœud $n_{supportant}$ et/ou par le nœud n_{inter}.

La valeur $reg_{supporté}$ du nœud $n_{supporté}$ est supportée par

la valeur $reg_{supportant}$: $(C_{n_{supporté}, n_{supportant}}(reg_{supporté}, reg_{supportant})$ est vrai)

et/ou par la valeur $r1_{inter}$: $(C_{n_{supporté}, n_{inter}}(reg_{supporté}, r1_{inter})$ est vrai)

et la valeur $r2_{inter}$ est supportée par $reg_{supportant}$: $(C_{n_{inter}, n_{supportant}}(r2_{inter}, reg_{support})$ est vrai).

La Figure 3.11 illustre un tel sept-uplet. La Figure 3.12 montre un exemple d'arc consistance indirecte.

3.3.2.1 Terminaison de AC_{4BC} avec arc-consistance indirecte

L'algorithme AC_{4BC} avec arc-consistance indirecte doit préserver l'invariant défini dans la section 3.2.3.1.

La nouvelle fonction Indirect n'a aucun effet sur l'invariant Status.

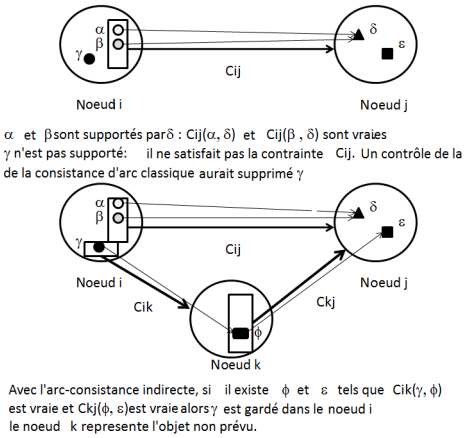


FIGURE 3.12 – Un exemple d'arc consistance indirecte

Théorème 3. L'algorithme AC_{4BC} avec arc-consistance indirecte (Cf. Figures 3.13 et 3.14) possède les propriétés suivantes :
(1) L'invariant sur la structure de données Status est préservé aux lignes 2 et 24.
(2) AC_{4BC} avec arc-consistance indirecte enfile et défile au plus O(nd) éléments, et la taille de la queue est donc au plus O(nd), où n est le nombre de nœuds et d est le nombre de données (la taille du plus grand domaine).
(3) AC_{4BC} avec arc-consistance indirecte se termine toujours.

Preuve :
Pour prouver ce théorème on considère l'algorithme décrit dans les Figures 3.13 et 3.14. La propriété (1) est vraie à l'initialisation. Supposons que cela est vrai en ligne 2, cela reste vrai après l'itération 7-21. La ligne 19 s'assure que (i, b) est retiré de l'interface pour tout b tel que :

— $\exists j \in node(G)$ avec Counter[(i,j),b]=0.

— et $\exists j \in node(G)$ avec Counter2[(i,j),b]=0.

Le second cas est vérifié par la fonction $FindIndirect$((i,j),k,C^a_{ij},b) à la ligne 18.

L'invariant est donc vrai en ligne 24. En effet la post-condition de CleanKernel s'assure que $\forall b \in D_i$, $\forall i \in node(G)$, (i, b) est suspendu si $\exists j \in node(G)$ tel que $\neg Path_i(b, D_{ij})$. L'exécution des lignes 25-41 préserve l'invariant. Les lignes 38-39 s'assurent que (j, c) est rejeté de l'interface pour tout c tel qu'il existe $i \in node(G)$ où Counter[(j,i),c]=0 et Counter2[(j,i),c]=0. La ligne 39 correspondant à l'appel de CleanKernel préserve l'invariant comme vu précédemment. Donc l'invariant est vrai aux lignes 2 et 24.

La propriété (2) est vraie car chaque élément de Status peut effectuer uniquement trois transitions :

— une de l'état "présent" à l'état "rejeté de l'interface" à travers les lignes 20 et 38.

— une de l'état "rejeté de l'interface" à l'état "suspendu" à travers la procédure CleanKernel.

— une de l'état "suspendu" à l'état "rejeté" à travers la procédure Dequeue.
Il ne peut donc y avoir que $O(nd)$ dequeues et enqueues.

La propriété (3) est une conséquence directe des propriétés (1) et (2) et des pré-conditions de la procédure Enqueue sur la structure de données Status.

3.3.2.2 AC_{4BC} avec arc-consistance indirect est correct

Théorème 4. G est indirectement arc-consistant lorsque AC_{4BC} avec arc-consistance indirecte se termine.

Preuve :
Comme pour la preuve précédente, on considère l'algorithme décrit dans les Figures 3.13 et 3.14.
Hypothèse initiale : soit $i \in node(G)$ tel que $\exists (i,j) \in arc(G)$ et $\exists c \in D_i$. Supposons que c n'est pas supporté directement par un nœud j, ni indirectement par un chemin intra-nodal, ni indirectement par un chemin inter-nodal avec deux arcs sur le nœud j via le nœud k lorsque AC_{4BC} avec arc-consistance indirecte se termine.
Autrement dit, $\forall b \in D_j, \forall a \in D_{ij}, \neg \mathcal{C}\text{mpi}(c,a)$ ou $\neg C_{ij}(a,b)$
ou $\forall t \in D_{ik}, (i,k) \in arc(G)\ et\ (k,j) \in arc(G)\ \forall (t,m,n,b) \in (D_{ik} \times D_k \times D_{kj} \times D_j)$ $\neg \mathcal{C}\text{mpi}(c,t)$ ou $\neg C^a_{ik}(t,m)$ ou $\neg \mathcal{C}\text{mpk}(m,n)$ ou $\neg C^a_{k,j}(n,b)$.
Si $c \in D_i$ n'est pas supporté :

— Soit (1) il n'a jamais été supporté.

— Soit (2) il a été supporté à une étape précédente.

Cas(1) :
Dans ce cas, il y a une contradiction avec $c \in D_i$ car l'étape d'initialisation n'aurait pas mis c dans D_i (Ligne 7 de AC_{4BC} Figure 3.13).
Cas(2) :

— Soit $b_1 ... b_m$ $(m > 0)$ l'ensemble des éléments de D_j supportant c à cette étape précédente. Puisqu'à la fin c n'est pas supporté, cela signifie que $b_1 ... b_m$ sont supprimés de D_j pendant l'exécution de AC_{4BC}.

— La suppression de $b_1 ... b_m$ provoque leur insertion dans la queue Q.

— Tous les éléments $b_1 ... b_m$ sont nécessairement retirés de Q lorsque AC_{4BC} se termine et les compteurs counter[(j,i),c] et counter2[(j,i),c] deviennent nécessairement égaux à zéro lorsque AC_{4BC} se termine (Ligne 26-33 de la Figure 3.14).

— A ce moment-là, c est retiré de l'interface D_{ij} (ligne 38 de la Figure 3.14).

— D'après l'hypothèse initiale c n'est pas supporté indirectement par un chemin intra-nodal.
Cela signifie que $\forall a \in D_{ij}$,$\neg\mathcal{C}\text{mpi}(c,a)$ ou $\neg C_{ij}(a,b)$.
Il y a donc deux cas à étudier :

1. Tout d'abord, si $\forall a \in D_{ij}$,$\neg\mathcal{C}\text{mpi}(c,a)$, alors $\neg Path_i(c, D_{ij})$. Dans ce cas CleanKernel supprime c de D_i (lignes 4-7 de la Figure 3.15) ce qui est le contraire de l'hypothèse initiale.
2. Le second cas est $\forall b \in D_j$, $\exists a \in D_{ij}$, $\mathcal{C}\text{mpi}(c,a)$ et $\neg C_{ij}(a,b)$. Ce cas ne peut pas arriver. En effet si on considère la ligne 19 de AC_{4BC},
$\forall b \in D_j$ $\neg C_{ij}(a,b) \Rightarrow a \notin D_{ij}$. L'affirmation contra-positive devient vraie.
$a \in D_{ij} \Rightarrow \exists b \in D_j$ $C_{ij}(a,b)$. Donc $\forall a \in D_{ij}$, $\mathcal{C}\text{mpi}(c,a) \Rightarrow \exists b \in D_j$ tel que $C_{ij}(a,b)$.

— D'après l'hypothèse, c n'est pas supporté indirectement par un chemin avec deux arcs.
Cela signifie que $\forall k \in node(G)$ tel que (i,k) et $(k,j) \in arc(G)$, $\forall(t,m,n,b) \in (D_{ik} \times D_k \times D_{kj} \times D_j)$, $\neg\mathcal{C}\text{mpi}(c,t)$ ou $\neg C^a_{ik}(t,m)$ ou $\neg\mathcal{C}\text{mpk}(m,n)$ ou $\neg C^a_{kj}(n,b)$.
Il y a donc 4 cas à étudier :

1. Tout d'abord, si on a $\forall t \in D_{ik}$,$\neg\mathcal{C}\text{mpi}(c,t)$, alors on a $\neg Path_i(c, D_{ik})$. Dans ce cas CleanKernel supprime c de D_i (lignes 4-7 de la Figure 3.15) ce qui est le contraire de l'hypothèse initiale.
2. Le deuxième cas est $\forall m \in D_k$, $\exists t \in D_{ik}$, $\mathcal{C}\text{mpi}(c,t)$ et $\neg C^a_{ik}(t,m)$. Ce cas ne peut pas arriver. En effet si on considère la ligne 19 de AC_{4BC},
$\forall m \in D_k$ $\neg C^a_{ik}(t,m) \Rightarrow t \notin D_{ik}$. L'affirmation contra-positive est vraie
$t \in D_{ik} \Rightarrow \exists m \in D_k$ $C_{ik}(t,m)$. Donc $\forall t \in D_{ik}$, $\mathcal{C}\text{mpi}(c,t) \Rightarrow \exists m \in D_k$ tel que $C_{ik}(t,m)$.
3. Le troisième cas est $\exists(t,m,b) \in (D_{ik} \times D_k \times D_j) \forall n \in D_{kj}$, $\mathcal{C}\text{mpi}(c,t)$ et $C^a_{ik}(t,m)$ et $\neg\mathcal{C}\text{mpk}(m,n)$ et $C^a_{kj}(n,b)$.
$\forall n \in D_{kj}$ $\neg\mathcal{C}\text{mpk}(m,n) \Rightarrow \neg Path_k(m, D_{kj})$. Dans ce cas CleanKernel supprime m de D_k (lignes 4-7 de la Figure 3.15) ce qui est le contraire de l'hypothèse initiale.
4. Le quatrième cas est $\exists(t,m,n) \in (D_{ik} \times D_k \times D_{kj}) \forall b \in D_j$ $\mathcal{C}\text{mpi}(c,t)$ et $C^a_{ik}(t,m)$ et $\mathcal{C}\text{mpk}(m,n)$ et $\neg C^a_{kj}(n,b)$. Ce cas ne peut pas arriver. En effet si on considère la ligne 19 de AC_{4BC}, $\forall b \in D_j$ $\neg C^a_{kj}(n,b) \Rightarrow n \notin D_{kj}$. L'affirmation contra-positive est vraie $n \in D_{kj} \Rightarrow \exists b \in D_j$ $C^a_{kj}(n,b)$. Donc $\forall n \in D_{kj}$, $\mathcal{C}\text{mpk}(m,n) \Rightarrow \exists b \in D_j$ tel que $C^a_{kj}(n,b)$.

En conclusion, l'hypothèse initiale conduit à une contradiction. Donc G est indirectement arc-consistant lorsque AC_{4BC} avec arc-consistance indirecte se termine.

begin AC_{4BC}
Etape 1 : Construction des structures de données.
1 **InitQueue**(Q) ;
2 **for** each $i \in node(G)$ **do**
3 **for** each $b \in D_i$ **do**
4 **begin**
5 S[i,b] := ensemble vide ;
6 **end** ;
7 **for** each $(i,j) \in arc(G)$ **do**
8 **for** each $b \in D_{ij}$ **do**
9 **begin**
10 Total := 0 ;
11 **for** each $c \in D_j$ **do**
12 **if** $C_{ij}(b,c)$ **then**
13 **begin**
14 Total := Total +1
15 S[j,c] :=S[j,c] $\cup$ (i,b) ;
16 **end**
17 Counter[(i,j),b] := Total ;
18 res :=FindIndirect($((i,j), Indirect(i), C_{ij}^a, b)$;
19 **if** Total=0 and not res **then**
20 $D_{ij} := D_{ij} - \{b\}$;
21 **end** ;
22 **for** each $i \in node(G)$ **do**
23 CleanKernel(D_i, I_i, Q) ;

FIGURE 3.13 – L'algorithme AC_{4BC} avec arc-consistance indirecte : étape 1. La Figure 3.15 décrit la procédure CleanKernel et la Figure 3.16 décrit la fonction Indirect. $S[i,b]$ dénote l'ensemble de couples (nœud, label) supportant le label b dans le nœud i. C_{ij}^a denote une contrainte reliant deux nœuds i et j en fonction de la relation élémentaire $a \in \sigma_T$.

3.4 Expérimentations : Application à la détection de tumeurs

Dans l'application suivante, les contraintes de nœuds sur les valeurs de couleurs sont très faibles car il y a une importante superposition entre les intervalles des valeurs de couleurs des différentes sous-parties des objets. Pour certains types d'images, ces contraintes ne sont pas spécifiées car les différentes sous-parties partagent le même intervalle de valeurs. Dans ces cas, l'étiquetage est uniquement le résultat du contrôle de la consistance d'arcs. Dans un ensemble d'images cérébrales, une tumeur cérébrale est représentée par un ensemble de régions qui va détériorer, d'une façon *apriori* inconnue, le plan relationnel du graphe sémantique. Pour étiqueter correctement les régions de l'image, la notion d'arc-consistance faible est utilisée. Dans ce but, on ajoute un nouveau nœud t correspondant

```
Etape 2 : Elagage des labels inconsistants
24 While not Emptyqueue(Q) do
25begin
26Dequeue(i,b,Q) ;
27   for each (j, c) ∈ S[i, b] do
28   begin
29   Counter[(j,i),c] := Counter[(j,i),c]-1 ;
30   for each (α, β, k, d, t, i, b) ∈ S2 do
31      if (α = j and β = c) or (k = j and (d = c or t = c))then
32      begin
33         Counter2[(j,i),c] := Counter2[(j,i),c]-1 ;
34         S2 := S2 − {((α, β, k, d, t, i, b)};
35      end
36   if Counter[(j,i),c]=0 et Counter2[(j,i),c]=0then
37      begin
38      D_ji := D_ji − {c} ;
39      CleanKernel(D_j, I_j, Q) ;
40      end ;
41 end ;
42 end AC_4BC ;
```

FIGURE 3.14 – L'algorithme AC_{4BC} avec arc-consistance indirecte : étape 2

```
begin CleanKernel(inD_i, I_i, outQ)
1 begin
2  for each b ∈ D_i do
3     for each D_ij ∈ I_i do
4     if ¬Path_i(b, D_ij) then
5        begin
6           EnQueue(i,b,Q) ;
7           D_i = D_i-{b} ;
8           for each D_ij ∈ I_i do
9              D_ij = D_ij-{b} ;
10       end
11 end ;
```

FIGURE 3.15 – La Procédure CleanKernel

```
begin FindIndirect((i, j), k, C^a_ij, b)
1 begin
2  res :=faux ;
3  for each d ∈ D_k do
4     if C^a_ik(b, d) then
5        begin
6        for each t ∈ D_kj do
7           if Cmp_k(d,t) then
8           for each c ∈ D_j do
9              if C^a_kj(t,c) then
10             begin
11                Total :=Total+1 ;
12                S2 :=S2 ∪(i, b, k, d, t, j, c) ;
13             end
14       end
15       if Total != ∅ then
16       begin
17          Counter2[(i,j),b] :=Total ;
18          res :=vrai ;
19       end ;
20return res ;
21 end ;
```

FIGURE 3.16 – La fonction FindIndirect

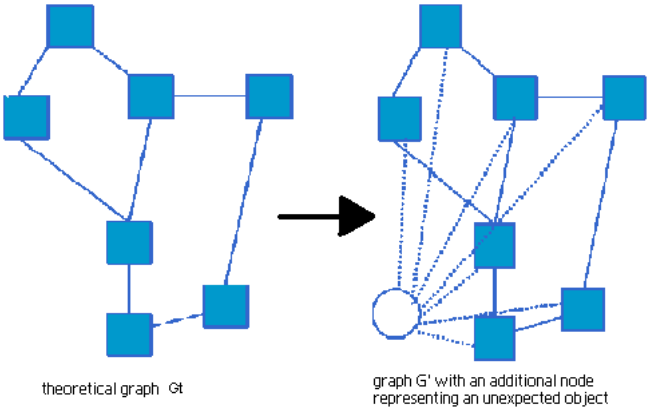
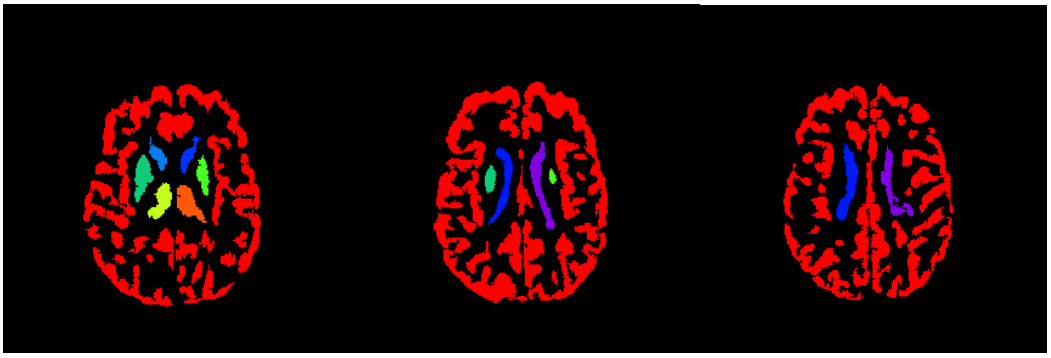


FIGURE 3.17 – A gauche : graphe théorique G_t. A droite : graphe G'.

à l'étiquetage d'un objet non prévu tel qu'une tumeur. De plus, chaque nœud du graphe possède un arc relié à ce nouveau nœud. Le graphe G' est donc défini de la façon suivante (voir Figure 3.17) : $Node(G') = Node(G) \cup \{t\}$ et
$Arc(G') = \{(i,j) \mid (i,j) \in Arc(G)\ ou\ i \in Node(G)\ \text{et}\ j = t\}$

a-



b-

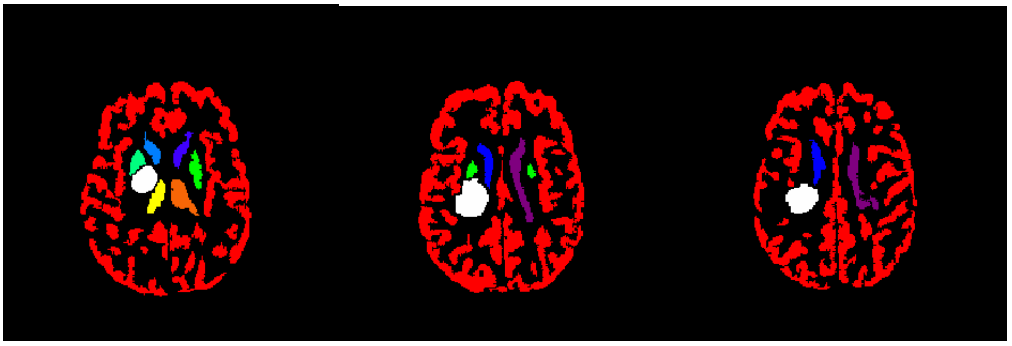


FIGURE 3.18 – Interprétation de la matière grise d'images anatomiques cérébrales :
a- Interprétation d'un cerveau normal. b- Interprétation avec l'introduction d'une tumeur (région blanche).
Chaque couleur est associée à une structure anatomique spécifique qui a été identifiée de façon unique (un nœud du graphe sémantique).

La fonction $Relax(i)$ utilisée par l'algorithme de satisfaction de la quasi arc-consistance et associant un nombre de relâchements de contraintes à chaque nœud du graphe G' est définie de la façon suivante :

$$\begin{aligned} &Relax : Node(G') \rightarrow \mathbb{N} \\ &i \mapsto Relax(i) = 1 \text{ si } i \neq t \\ &ou \;\; Relax(i) = Card(Node(G)) \; si \; i = t \end{aligned}$$

L'algorithme combine les deux notions de quasi arc-consistance et d'indirecte arc-consistance.
Il procède en deux étapes.
Il contrôle tout d'abord l'arc-consistance indirecte.
Si elle est satisfaite la deuxième étape peut s'exécuter :

— Au départ, un nombre de relâchements est associé à chaque nœud (ce nombre est fourni par la fonction $Relax$)

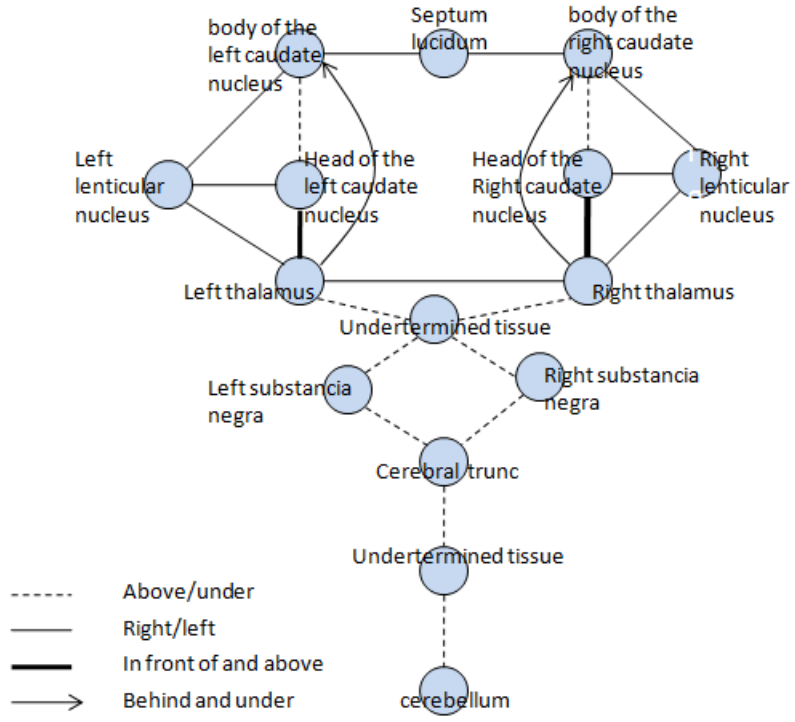


FIGURE 3.19 – Graphe sémantique décrivant l'organisation spatiale des structures anatomiques appartenant à la matière grise. Dans ce cas l'anatomie est décrite en trois dimensions.

— La quasi arc-consistance est ensuite contrôlée en fonction de ces nombres de relâchements. Si elle n'est pas satisfaite, le nombre de relâchements de contraintes est augmenté et cette deuxième étape et relancée.

Ceci est répété jusqu'à ce que la quasi-arc-consistance soit satisfaite.

Pour tous les arcs (i, j) de G', l'arc-consistance indirecte via le nœud t est contrôlée.
Pour tester cet algorithme, un ensemble de 20 images de 256x256 pixels a été utilisé. Cet ensemble correspond aux vingt coupes contigües du cerveau. Après avoir segmenté les images, on obtient 200 régions à étiqueter. Un graphe sémantique décrivant l'anatomie cérébrale dans les trois dimensions est construit. (Figure 3.19). Dans cette application, même si la segmentation semble correcte, les structures cérébrales sont sur-segmentées car on travaille en 3 dimensions (chaque structure cérébrale est présente dans plusieurs coupes). Plusieurs régions sont donc associées à un seul concept (nœud) du graphe sémantique. A l'intérieur de cet ensemble d'images, une tumeur 3D a été générée synthétiquement (voir [45] pour voir les détails de l'application de l'algorithme AC_{4BC} à l'interprétation d'images cérébrales obtenues par résonance magnétique (IRM)). Dans ce cas, l'algorithme AC_{4BC} tombe en échec si on n'introduit pas le relâchement de contraintes et le contrôle de l'arc-consistance indirecte. L'intégration de ce relâchement

de contraintes permet d'identifier correctement la tumeur parmi les régions segmentées et n'empêche pas l'étiquetage univoque des autres régions (Voir la Figure 3.18).

3.5 Conclusion

Dans ce chapitre, nous avons proposé des extensions aux notions de problèmes de satisfaction de contrainte et de consistance d'arcs. Ces extensions permettent de gérer des mises en correspondance non injectives et même des cas de mises en correspondance non fonctionnelles.

Malgré ces extensions, l'interprétation d'images peut rester difficile. Les échecs rencontrés peuvent être expliqués par trois différentes causes :

— Le modèle encodé dans le graphe n'est pas adapté. Pour réduire cette difficulté, nous étudierons dans le chapitre suivant comment construire un modèle qui décrive au mieux les informations recherchées dans l'image.

— L'objet recherché n'est pas dans l'image ou des objets non prévus apparaissent. Grâce aux extensions proposées, il devient possible de réduire une partie de ces difficultés, en ajustant les paramètres le relâchement des contraintes. Cet ajustement pourrait être automatisé par un système de feedback, en augmentant les relâchements si l'interprétation échoue.

— Le résultat de la segmentation n'est pas bon et des régions sous-segmentées font disparaitre certaines parties de l'objet recherchées dans l'image. Pour éviter cela, il pourrait être également possible d'utiliser l'interprétation de l'image comme critère de feedback pour choisir les paramètres de segmentation optimaux. Cet aspect fera l'objet du travail présenté dans le chapitre 5.

Chapitre 4

Un système de relations spatiales qualitatives

4.1 Introduction

Dans le chapitre précédent, nous avons développé un algorithme d'interprétation d'images dans le cadre des PSC. Nous avons vu que celui-ci est capable de gérer des cas d'objets non prévus dans le modèle et des cas d'occlusion. Cependant pour obtenir un résultat satisfaisant, il est nécessaire de fournir à cet algorithme un modèle sémantique suffisamment précis, pour décrire de façon non ambigüe la complexité de la description des objets. Notre modèle sémantique doit intégrer à la fois des contraintes spatiales et morphologiques. Une première approche consisterait à considérer que les relations spatiales peuvent être décrites par de simples relations d'adjacence comme on peut en trouver dans les graphes d'adjacence de régions (RAG). Ce formalisme a des propriétés intéressantes qui le rend très pratique pour décrire une image et a donc été choisi par de nombreux auteurs [49–54].

Cependant, la seule notion d'adjacence est trop pauvre pour décrire l'organisation spatiale complexe des différentes parties d'un objet. Pour gérer ces difficultés, certains auteurs proposent de décrire des relations spatiales en utilisant des outils numériques et probabilistes. Dans ce contexte, on peut mentionner les travaux de Bloch et al. dans la communauté des ensemble flous [50, 55]. Dans le domaine de la topologie, Cohn et al. [56] proposent de décrire des relations spatiales complexe entre les régions à l'aide d'un ensemble de relations de base et ils ont créé pour cela le formalisme RCC8. RCC8 fonctionne avec un ensemble de 8 relations "Jointly Exhaustive et Pairwise Disjoint" (JEPD) appelées relations de base : DéConnecté (DC), Connecté de manière Externe (EC), Partialement superposé (PO), Egal (EQ), Tangential Proper Part (TPP), Non

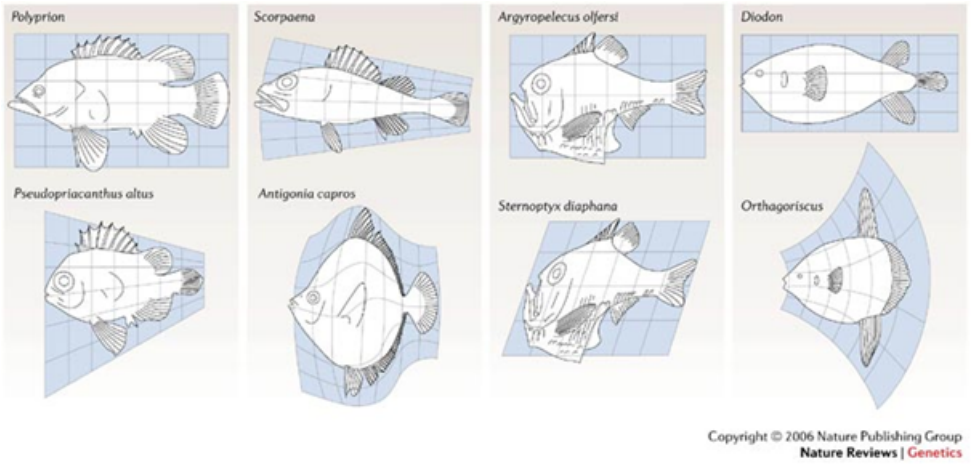


FIGURE 4.1 – Les transformations morphologiques de D'Arcy Thompson montrent combien les métriques sont importantes.

Tangential Proper Part (NTPP) et leur inverse (cf. Fig. 4.2). Cependant, ce formalisme ne prend pas en compte la forme des régions et les relations directionnelles. Skiadopoulos et Koubarakis [57] contournent cet inconvénient en définissant formellement les relations cardinales directionnelles (CDF). Ces relations utilisent la notion de boîte englobante minimale (mbb) et plusieurs auteurs ont proposé des façons de combiner les notions topologiques avec les notions de relations directionnelles [58]. Cette approche a plusieurs avantages : elle possède de bonnes propriétés de calcul (calculer la boite englobante minimale d'une région est rapide), il est possible d'hériter des propriétés des boîtes englobantes minimales à l'intérieur d'une pyramide de graphes d'adjacence, il est possible d'introduire une notion de métrique absolue ou relative et RCC8 peut être retrouvé à partir de ces relations[58].

Cependant, les notions topologiques et directionnelles ne prennent pas en compte la notion de distance entre deux régions alors que c'est une caractéristique importante [59] (Voir Figure 4.1). Par exemple, la différence entre les faces d'animaux d'espèces ou de races différentes réside principalement dans les différentes distances entre chaque partie de leur face (écart entre les yeux, écart entre les yeux et la gueule de l'animal).

Par ailleurs, travailler uniquement à l'aide des boîtes englobantes minimales a quelques inconvénients : lorsque deux boîtes englobantes minimales de deux régions sont superposées (see Fig. 4.3a), il n'est pas possible de calculer des distances significatives.

Dans ce chapitre, nous proposons un nouveau système de relations topologiques et directionnelles (CDMF), dérivé du système CDF. Ce système prend mieux en compte la notion de distance grâce à la notion de boîte englobante minimale des frontières interfaces (mbbbi). On l'utilise dans le cadre d'une représentation sous forme de graphe sémantique, et nous montrons comment interpréter des images sur-segmentées grâce à

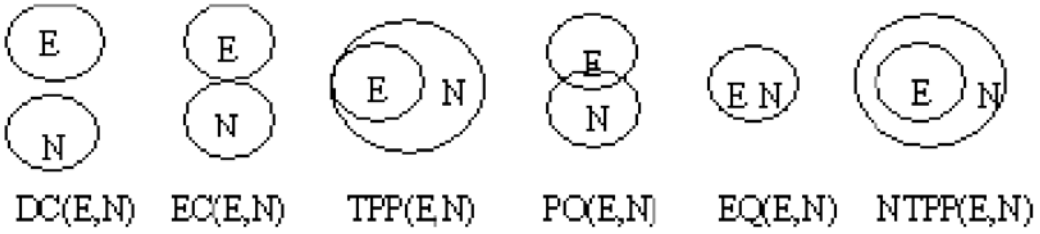


FIGURE 4.2 – Illustration des relations JEPD

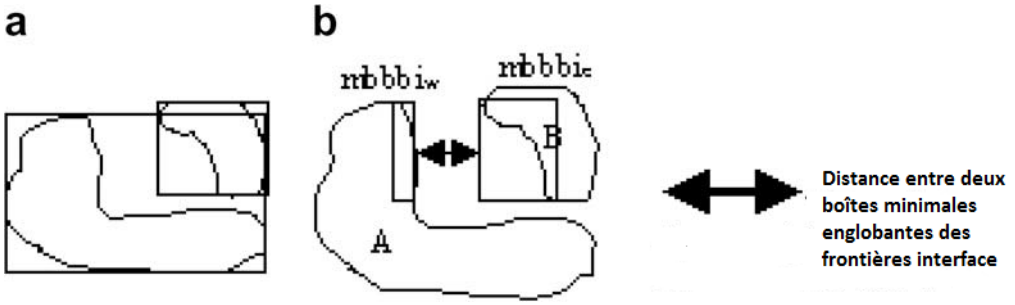


FIGURE 4.3 – (a) Dans ce cas, bien que les deux régions ne soient pas superposées, les deux boîtes englobantes minimales sont superposées. L'analyse des relations spatiales entre ces deux régions n'est donc pas possible en utilisant des boîtes englobantes minimales. (b) Avec la notion de boîte englobante de la frontière interface, définie dans la Section 4.2.1, il est possible de calculer une distance entre A et B. $mbbbi_w$ est la boîte englobante minimale de la frontière interface qui est à gauche de A. $mbbbi_e$ est la boîte englobante minimale de la frontière interface qui est à droite de la région B.

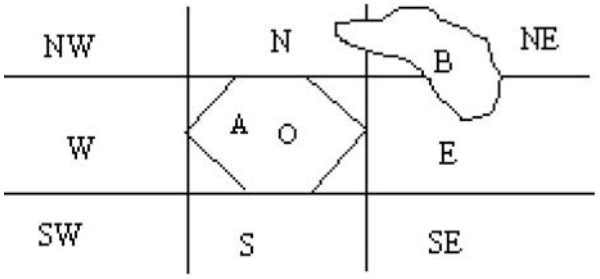


FIGURE 4.4 – Relations directionnelles cardinales entre deux régions A et B

un tel graphe. Dans la partie 4.2, nous décrirons un nouveau système de relations spatiales et dans la partie 4.3, nous définirons une nouvelle structure de graphe, dérivée de la notion de graphe sémantique [43, 45, 60–62], permettant d'introduire des relations spatiales complexes décrite dans le cadre du système CDMF. Cette structure permet d'exprimer toutes les combinaisons logiques de ces relations, contenant des opérateurs ET et OU, associées à un nœud. Nous préciserons comment cette structure de graphe peut être gérée dans le cadre de la satisfaction de la consistance d'arcs à deux niveaux de contrainte. Dans la partie 4.4, des expérimentations sont présentées avec différents types de modèles appliqués à des images du monde réel et à des images cérébrales anatomiques.

4.2 Relations spatiales complexes entre deux objets composites

4.2.1 Formalisme directionnel et cardinal

Dans le cadre du Formalisme Directionnel et Cardinal (FDC), Skiadopoulos et Koubarakis [57] ont défini formellement 9 relations directionnelles cardinales (Voir Fig.4.4). Soit A une région, la plus grande borne inférieure de la projection de A sur l'axe des x (respectivement sur l'axe des y) est notée par infx(A) (respectivement infy(A)). La plus petite borne supérieure de la projection de A sur l'axe des x (respectivement sur l'axe des y) est notée par supx(A) (respectivement supy(A)). La plus petite boîte englobante de A est notée par $mbb(A)$. Cette boîte est un rectangle dont les coordonnées du coin inférieur gauche sont x1=$infx(A)$, y1=$infy(A)$ et les coordonnées du coin supérieur droit sont x2=$supx(A)$, y2=$supy(A)$. Les relations de direction cardinale unique peuvent être définies de la façon suivante :

- A O B ssi infx(B) $\leq$ infx(A), supx(A) $\leq$ supx(B), infy(B) $\leq$ infy(A) et supy(A) $\leq$ supy(B)
- A S B ssi supy(A) $\leq$ infy(B), infx(B) $\leq$ infx(A) and supx(A) $\leq$ supx(B)
- A SW B ssi supx(A) $\leq$ infx(B) et supy(A) $\leq$ infy(B)
- A W B ssi supx(A) $\leq$ infx(B), infy(B) $\leq$ infy(A) et supy(A) $\leq$ supy(B)
- A NW B ssi supx(A) $\leq$ infx(B) et supy(A) $\leq$ supy(B)
- A N B ssi supy(B) $\leq$ infy(A), infx(B) $\leq$ infx(A) et supx(A) $\leq$ supx(B)
- A NE B ssi supx(B) $\leq$ infx(A) et supy(B) $\leq$ infy(A)
- A E B ssi supx(B) $\leq$ infx(A), infy(B) $\leq$ infy(A) et supy(A) $\leq$ supy(B)
- A SE B ssi supx(B) $\leq$ infx(A) et supy(A) $\leq$ supy(B)

4.2.2 Le formalisme Connectivité-Direction-Métrique ($CDMF$)

Les boîtes englobantes minimales de deux régions fournissent des informations sur leur relations spatiales mais cette information est parfois très pauvre, par exemple lorsque deux boîtes se superposent (Figure 4.3a). Pour décrire des organisations spatiales plus complexes, nous utilisons trois types d'information de base :

1. La notion de connectivité exprimée dans le cadre topologique de RCC8 par la relation primitive dyadique C(x,y) signifiant "x est connecté avec y".
2. La notion de boîte englobante minimale introduite dans le cadre du formalisme directionnel cardinal. Plusieurs propriétés peuvent être estimées par cette notion :

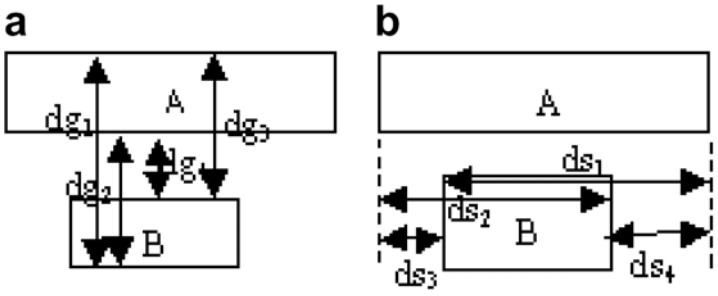


FIGURE 4.5 – : Les 8 métriques entre deux boîtes englobantes minimales : distances entre A et B et décalages latéraux entre A et B.

— La surface, la hauteur et la largeur d'une région peuvent être calculées.

— Les relations directionnelles entre deux régions. Dans notre contexte on définit quatre relations directionnelles : N (Nord), S (Sud), W (ouest) et E (Est).

$$\text{a N b ssi } \text{supy}(b) \leq \text{infy}(a),$$
$$\text{a S b ssi } \text{supy}(a) \leq \text{infy}(b),$$
$$\text{a W b ssi } \text{supx}(a) \leq \text{infx}(b),$$
$$\text{a E b ssi } \text{supx}(b) \leq \text{infx}(a)$$

— Plusieurs métriques entre deux régions peuvent être calculées à condition que les boîtes englobantes minimales ne se superposent pas. Huit métriques entre deux boîtes englobantes minimales peuvent être définies (Voir Fig. 4.5). Pour l'orientation nord/sud, les définitions sont :
$dg_1(A,B)$ = supy(A)-infy(B) , $dg_2(A,B)$ = infy(A)-infy(B), $dg_3(A,B)$ = supy(A)-supy(B), $dg_4(A,B)$= infy(A)-supy(B), $ds_1(A,B)$= supx(A)-infx(B), $ds_2(A,B)$= supx(B)-infx(A), $ds_3(A,B)$= infx(B)-infx(A),
$ds_4(A,B)$= supx(A)-supx(B).
Les définitions pour l'orientation est/ouest sont similaires. Les huit relations définies dans le formalisme CDF peuvent être facilement retrouvées à partir de nos relations avec les métriques appropriées.

3. Nous avons créé une nouvelle notion de boîte englobante minimale des interfaces frontières (*mbbbi*) entre deux régions pour chaque direction cardinale (N, S, E, W). Cette notion définie dans la section suivante, apporte des informations supplémentaires permettant d'enrichir la façon de décrire les organisations spatiales.

4.2.2.1 Boite englobante minimale d'une interface frontière entre deux régions

Pour rendre plus précise l'analyse spatiale, nous avons créé la notion de boîte englobante minimale d'une "interface frontière". On exprime par "interface frontière" la frontière

d'une région qui, étant donné une direction cardinale, est en face d'une autre région (Voir Fig. 4.3b).

Définition 8. Soit R une région (un ensemble de pixels connectés), et soit p(x,y) un pixel de R. E(R)=$\{p(x,y) \in R \mid p(x',y')$ un des 8 voisins de p(x,y), $p(x',y') \notin R\}$. Soit A et B deux régions :

- L'interface frontière $Cw(A,B)$ est définie par $Cw(A,B) = \{p(x,y) \in E(A)$ tel que $\exists p(x',y) \in E(B)$ et $\forall x''$, $x < x'' < x' \Rightarrow (p(x'',y) \notin A$ et $p(x'',y) \notin B)\}$
- L'interface frontière $Ce(A,B)$ est définie par $Ce(A,B) = \{p(x,y) \in E(A)$ tel que $\exists p(x',y) \in E(B)$ et $\forall x''$, $x > x'' > x' \Rightarrow (p(x'',y) \notin A$ et $p(x'',y) \notin B)\}$
- L'interface frontière $Cn(A,B)$ est définie par $Cn(A,B) = \{p(x,y) \in E(A)$ tel que $\exists p(x,y') \in E(B)$ et$\forall y''$, $y < y'' < y' \Rightarrow (p(x,y'') \notin A$et $p(x,y'') \notin B)\}$
- L'interface frontière $Cs(A,B)$ est définie par $Cs(A,B) = \{p(x,y) \in E(A)$ tel que $\exists p(x,y') \in E(B)$ et $\forall y''$, $y > y'' > y' \Rightarrow (p(x,y'') \notin A$ et $p(x,y'') \notin B)\}$

Définition 9. La boîte englobante minimale d'une interface frontière d'une région A avec une région B dans la direction d ($mbbbi_d$) est définie par (infx($Cd(A,B)$), infy($Cd(A,B)$)), (supx($Cd(A,B)$), supy($Cd(A,B)$))

Dans l'exemple de la Figure 4.3, les deux *mbb* des régions A et B sont superposées. Au contrainte la $mbbbi_w$ et la $mbbbi_e$ ne sont pas superposées. Donc, si on travaille avec les *mbbbi*, on peut déduire que la région A est à gauche de la région B avec une distance donnée. Cette déduction ne peut pas être obtenue si on travaille avec les *mbb*.
Des cas complexes peuvent être rencontrés lorsque l'utilisation des *mbbbi* ne peut pas fournir de distance utilisable entre deux objets très imbriqués. Cependant, cette notion que nous avons créée permet de gérer de nombreuses situations où les boîtes englobantes minimales sont insuffisamment informatives.

4.2.2.2 Relations supplémentaires entre deux régions.

Les boîtes englobantes minimales d'une interface frontière ($mbbbi_w$, $mbbbi_e$, $mbbbi_n$, $mbbbi_s$) permettent de décrire de nouvelles relations. Quatre relations spatiales entre A et B en lien avec la boîte englobante minimale d'une interface frontière correspondante $mbbbi_d$ peuvent être définies de la façon suivante :

$$\begin{aligned}
&\text{A Ei B ssi supx}(Cw(B,A)) \leq \text{infx}(Ce(A,B)),\\
&\text{A Wi B ssi supx}(Cw(A,B)) \leq \text{infx }(Ce(B,A)),\\
&\text{A Ni B ssi supy}(Cn(A,B)) \leq \text{infy}(Cs(B,A)),\\
&\text{A Si B ssi supy}(Cn(B,A)) \leq \text{infy}(Cs(A,B)),
\end{aligned}$$

Toutes ces relations peuvent être associées à une métrique d définie de la façon suivante : d(A,B)= infz(A) - supz(B) où z = y pour les relations Ni ou Si, et z = x pour les relations Ei ou Wi.

4.2.2.3 Relations élémentaires de CDMF.

Le formalisme CDMF permet de définir des relations très complexes en combinant les relations élémentaires. Une relation élémentaire $\Re e$ est une relation :

- (1) de connectivité ou de non connectivité
- (2) directionnelle entre *mbb* avec ou sans métrique choisie parmi les métriques *dsi* et *dgi* (i=1 ... 4) définies précédemment (avec des limites inférieures ou supérieures). Dans ce cas, les quatre relations directionnelles possibles sont : N (Nord), S (Sud), W (Ouest) et E (Est).
- (3) directionnelle entre *mbbbi* avec la métrique d définie précédemment (avec des limites inférieures et supérieures). Les quatre relations directionnelles sont : Ni, Si, Wi et Ei.

Propriété 1. Pour chaque relation élémentaire $\Re e$ entre A et B, $A \ \Re e \ B \Rightarrow \exists$ un pixel $a \in A$ et $\exists$ un pixel $b \in B$, $a \ \Re e \ b$.

Preuve 1. $\Re e$ peut être de trois types :

- Si $\Re e$ de type (1). Il est immédiat que :
  A est connecté à B $\Rightarrow \exists a \in A$ et $\exists b \in B$, a est connecté à b.
  A n'est pas connecté à B $\Rightarrow \exists a \in A$ et $\exists b \in B$, a n'est pas connecté à b.
- Si $\Re e$ de type (2). Soit $\Re e$ l'une des quatre relations N, S, E, W et d_i l'une des huit métriques définies entre deux *mbb* (i=1 ... 8). Soit min et max les limites inférieures et supérieures (en nombre de pixels) de la distance d_i. Il est immédiat que (A $\Re e$ B et $min \leq d_i(mbb(A), mbb(B)) \leq max$) $\Rightarrow$ ($\exists a \in A$ et $\exists b \in B$, a $\Re e$ b et $min \leq d_i(a, b) \leq max$).
- Si $\Re e$ de type (3). Soit $\Re e$ l'une des quatre relations Ni, Si, Wi, Ei définies en utilisant les *mbbbi* de la section 4.2.2. Soit d la métrique associée aux deux *mbbbi*. Soient min et max les limites inférieures et supérieures (en nombre de pixels) de la distance d et soit $\Re e^{-1}$ la direction opposée de $\Re e$. Il est immédiat que (A $\Re e$ B et $min \leq d(mbbbi_{\Re e}(A), mbbbi_{\Re e^{-1}}(B)) \leq max$) $\Rightarrow$ ($\exists a \in A$ et $\exists b \in B$, a $\Re e$ b et $min \leq d(mbbbi_{\Re e}(a), mbbbi_{\Re e^{-1}}(b)) \leq max$).

4.3 Application des relations de $CDMF$ à des objets sur-segmentés : L'intégration du formalisme $CDMF$ dans un CSP à deux niveaux de contrainte

L'interprétation d'images de haut niveau consiste habituellement à mettre en correspondance chaque partie de l'image avec une représentation qui a un sens. Le formalisme des graphes est une façon très naturelle et très pratique de représenter le contenu sémantique d'une image et le CDMF peut être utilisé pour définir les contraintes de nœuds et d'arcs. La mise en correspondance peut être réalisée en résolvant un problème de satisfaction de contraintes (CSP). Cet aspect a été étudié dans [62]. Pour réduire la complexité de temps de la mise en correspondance du graphe avec les différentes sous-parties d'une forme, il est possible, comme nous l'avons vu précédemment, de ne prendre en compte que les contraintes locales. En pratique, comme les problèmes sont souvent sur-contraints, la satisfaction de la consistance d'arcs est suffisante. Dans notre contexte, les données ne sont pas idéalement segmentées et habituellement les objets représentés dans une image sont sur-segmentés de façon arbitraire, dépendant de la distribution des niveaux de gris dans l'image.
Le problème est le suivant : si A et B sont deux objets (régions) dans une image, tel que $A\ \Re\ B$ avec $\Re$ une combinaison de relations $\Re e$ du CDMF, comment définir la relation $\Re$' entre chacune des sous-parties $a \in A$ et $b \in B$ tel que $a\ \Re'\ b \Rightarrow A\ \Re\ B$?
Les relations élémentaires de CDMF possèdent une propriété intéressante étudiée précédemment :

> Pour chaque relation élémentaire $\Re e$ entre A et B,
>
> $A\ \Re e\ B \Rightarrow \exists a \in A$ et $b \in B$, $a\ \Re e\ b$.

Donc les relations élémentaires $\Re e$ représentant des contraintes d'arcs dans le graphe peuvent représenter des contraintes sur les sous-parties des objets pouvant être associées à un nœud. Cependant, à cause de la sur-segmentation, une sous-partie d'un objet peut ne pas satisfaire toutes les contraintes. Dans ce cas le CSP échoue. Une solution a été décrite dans [45] et dans le chapitre 3 de ce document, en introduisant deux niveaux de contraintes dans le CSP classique.
Dans le paragraphe suivant nous étudierons la combinaison des relations spatiales du CDMF dans le contexte de la satisfaction de la consistance d'arcs à deux niveaux de contrainte. La notion de quasi arc-consistance décrite au chapitre précédent sera ici nécessaire. Ensuite, un exemple utilisant le CDMF dans ce contexte est décrit. En particulier, nous verrons comment exprimer les relations complexes "est entouré par" et "est partiellement entouré par".

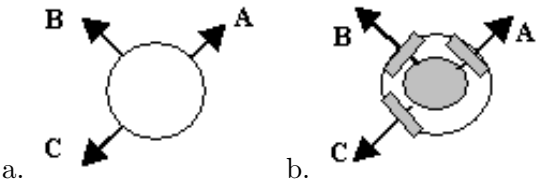


FIGURE 4.6 – a. Structure d'un nœud dans le cadre de la satisfaction de la consistance d'arcs classique. Trois contraintes A, B et C sont appliquées à ce nœud. Implicitement, ce nœud est contraint par A et B et C. b. Structure d'un nœud dans le cadre de la satisfaction de la consistance d'arcs à deux niveaux de contrainte.

4.3.1 Comment combiner les relations de $CDMF$ et comment les introduire dans un graphe sémantique.

Avec l'algorithme de satisfaction de la consistance d'arcs classique (AC4), les contraintes sur les arcs associés au même nœud doivent être satisfaites pour tous les labels classés à l'intérieur du nœud. Implicitement cela correspond à appliquer un ET logique entre chaque contrainte. Par exemple, dans la Figure 4.6a, la contrainte sur le nœud i est définie par l'expression logique : A ET B ET C. Elle doit être satisfaite pour tous les labels.

Dans le cadre de la satisfaction de la consistance d'arcs à deux niveaux de contrainte ($AC4_{BC}$), les contraintes sur les arcs doivent être satisfaites de la même manière pour tous les labels classés dans le nœud considéré. Cependant, chaque label n'a pas besoin de satisfaire directement chaque contrainte s'il peut satisfaire cette contrainte indirectement grâce aux relations Cmp_i définies au Chapitre 3 (Voir Figure 4.6b, les rectangles représentent les interfaces du nœud).

Dans certains cas, relier les contraintes avec un OU logique peut être utile pour décrire correctement l'objet recherché. Par exemple, dans un graphe sémantique décrivant un visage, au dessus des yeux on peut trouver soit des cheveux, soit l'extérieur de la tête. Le OU logique permet de garder la consistance dans le cas où il n'y a pas de cheveux. Cela peut être obtenu grâce à la notion de satisfaction de la quasi arc-consistance définie dans le chapitre précédent. Cette notion autorise certaines contraintes à ne pas être nécessairement satisfaites. Dans l'exemple de la Figure 4.5b, si on introduit un nombre de relâchement pour ce nœud, on obtient l'expression logique suivante :

- si le nombre de relâchement est égal à 0 on a : A ET B ET C
- si le nombre de relâchement est égal à 1 on a : (A ET B) OU (A ET C) OU (B ET C)
- si le nombre de relâchement est égal à 2 on a : A OU B OU C

Cependant certaines expressions logiques telles que (A OU B) ET C ne peuvent pas être exprimées. Pour gérer ce problème, nous avons créé au niveau de chaque nœud des

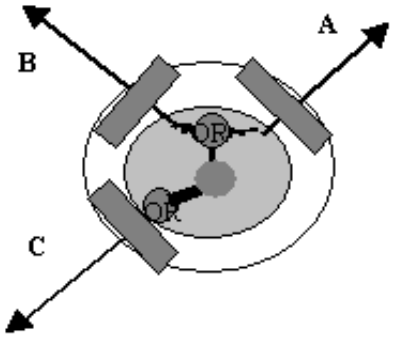


FIGURE 4.7 – Structure d'un nœud décrivant l'expression logique : (A OU B) ET C

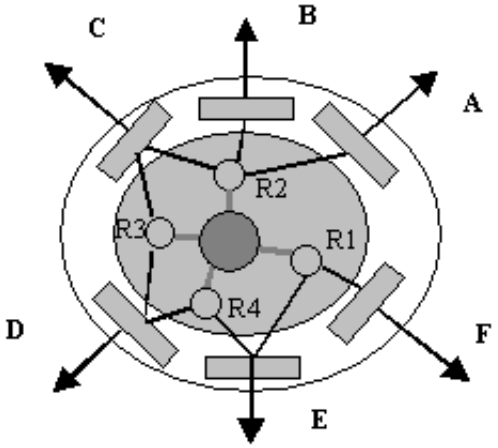


FIGURE 4.8 – Si les nombres de relâchement de chaque ensemble d'arcs R1, R3 and R4 sont égaux à 1 et si celui de R2 est égal à 2, on obtient l'expression logique suivante : (A OU B OU C) ET (C OU D) ET (D OU E) ET (E OU F)

"hubs" qui peuvent réunir plusieurs arcs (ensemble d'arcs) contraignant ce nœud et qui sont associés à un nombre autorisé de relâchements. Le nœud décrivant l'expression (A OU B) ET C a la structure décrite dans la Figure 4.7. Dans ce cas A et B sont réunis dans un même hub avec un nombre de relâchement égal à 1, ce qui revient à réaliser une opération logique OU entre les deux contraintes. C'est la seule contrainte de son hub et un nombre de relâchements égal à 0 lui est associé. Cette structure de hub rajoutée à l'intérieur du noyau du nœud, permet de décrire toutes les expressions logiques possibles avec des opérateurs ET et OU. La Figure 4.8 montre un exemple plus complexe.

4.3.2 Définir les relations "est entouré par" et "partiellement entouré par" en introduisant le CDMF dans un PSC_{BC}

En utilisant le formalisme $CDMF$, il est possible de définir la notion de "est entouré par" sur des régions sur-segmentées. Considérons le cas du graphe sémantique décrivant une

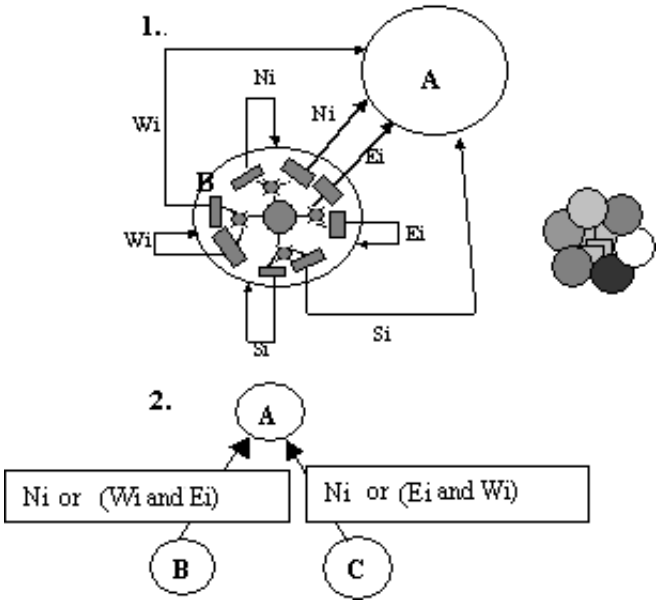


FIGURE 4.9 – 1. Graphe décrivant la relation "est entouré par" (par exemple, le centre d'une fleur (A) est entouré par les pétales (B)) 2. Graphe décrivant la relation "est partiellement entouré par" (par exemple les yeux sont partiellement entouré par les cheveux)

fleur. Supposons que l'un des nœuds décrit le centre de la fleur et que l'autre représente les pétales. Une région appartenant au centre d'une fleur a la propriété récursive suivante : dans chaque direction (Est, Ouest, Nord et Sud) de son voisinage, une autre région appartient au centre ou aux pétales. Dans ce cas, il est nécessaire d'imposer, pour chaque direction, un OU entre la contrainte reliant le nœud du centre avec lui-même (cette réflexivité vient de la propriété récursive définie précédemment) et avec le nœud des pétales.

Il n'est pas possible de gérer cette situation avec les contraintes intra-nœud Cmp_i. Ces contraintes ne contrôlent que les relations spatiales entre des régions appartenant à un même nœud i sans vérifier ce qu'il se passe à l'extérieur du nœud i. Dans ce cas il est nécessaire de contrôler récursivement les contraintes conjonctives imposées à l'intérieur du nœud et à l'extérieur du nœud (relations avec les pétales) en même temps. La quasi arc-consistance permet de résoudre ce problème.

Donc, "A est entouré par B" est défini de la façon suivante : $\forall a \in A, \forall \Re e \in \{N, S, W, E\}$ ($\Re e$ étant une relation spatiale dans l'une des quatre directions cardinales Nord, Sud, Ouest et Est), $\exists c \in A$ ou $\exists c \in B$, a connecté à c et a $\Re e$ c. Le graphe décrivant cette relation est présenté dans la Fig. 4.9.1.

Pour appliquer cette relation, les étapes principales de l'algorithme sont les suivantes :

— le nœud est initialisé avec l'ensemble des régions satisfaisant les contraintes unaires (contraintes morphologiques associées au nœud)

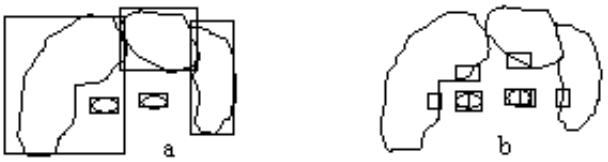


FIGURE 4.10 – a) Avec les boîtes englobantes minimales il n'est pas possible de calculer la distance entre l'œil gauche et les cheveux. b) Avec les *mbbbi* du CDMF, il est possible de calculer une distance.

— chaque région est placée dans les interfaces associées aux arcs si elle satisfait la relation spatiale associée à l'arc (les quatre directions cardinales Nord, Sud, Ouest et Est).

— la quasi arc consistance est alors appliquée (Voir la procédure CleanKernel décrite au chapitre 3) et les régions sont gardées ou supprimées du nœud en fonction du nombre de relâchements associé à chaque ensemble d'arcs réunis par un hub.

Un autre type de relation est "partiellement entouré par" avec une distance donnée. Ce cas peut être rencontré lorsque l'on souhaite identifier les yeux et les cheveux d'un visage humain (Voir la Fig. 4.10). Dans ce cas, les nœuds représentant les yeux et le nœud représentant les cheveux doivent être reliés par trois contraintes directionnelles Ni, Ei et Wi (Nord, Est et Ouest). La distance d est associée avec les trois relations (le graphe est décrit dans la Fig. 4.9.2).

4.4 Expérimentations

4.4.1 Application à l'interprétation d'images naturelles

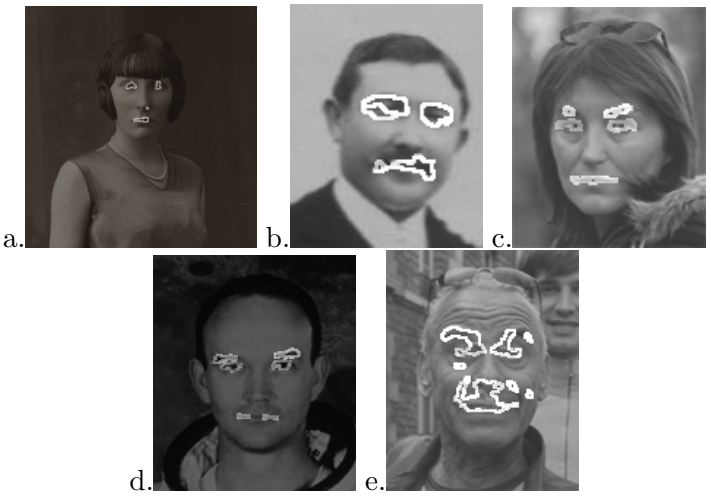
Cette analyse a été appliquée à des résultats de segmentation de façon à retrouver des objets spécifiques (visages, fleurs, voitures). Plusieurs types d'images test représentant différents objets ont été choisis :

— un ensemble d'images représentant des visages humains,

— un ensemble d'images représentant des voitures,

— et enfin un ensemble d'images représentant des fleurs.

Pour chaque type d'objet, un graphe sémantique a été construit. La Figure 4.11 montre l'interprétation réalisée sur différents visages. Les régions avec des contours blanc sont les régions correctement interprétées par l'analyse sémantique. Dans ce cas, il a été nécessaire d'utiliser la vérification de la quasi arc-consistance pour interpréter les images puisque le nœud "cheveux" peut être vide (voir image 'e' de la Figure 4.11). La Figure



Taille des images en nombre de pixels : a. 288x301 b. 98x124 c. 104x130 d. 158x184 e. 132x165

Nombre de labels à classifier dans chaque image : a. 99 b. 127 c. 216 d. 297 e. 26

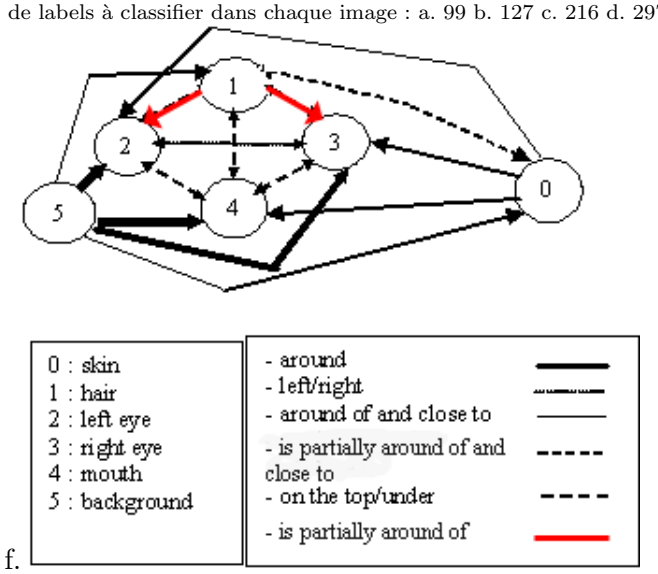


FIGURE 4.11 – Exemples d'interpretation de résultats de segmentation en fonction d'un graphe sémantique décrivant un visage. Les régions étiquetées sont les yeux et la bouche. Leur contour est superposé à l'image d'origine. f. Graphe sémantique décrivant un visage.

4.12 montre l'interprétation sur des images représentant des voitures et la Figure 4.13 montre l'interprétation sur des images représentant des fleurs.

4.4.2 Interprétation d'images cérébrales RMN

Un graphe sémantique décrivant une partie de l'anatomie cérébrale a été développé. Il contient 14 nœuds et 101 arcs (voir Figure 4.14). L'expérimentation a été fait sur l'image de la Figure 4.15a segmentée au préalable. La segmentation a trouvé 300 régions. Après avoir contrôlé la consistance d'arcs, chaque nœud contient toutes les régions correspondant à la structure anatomique décrite par le nœud et uniquement ces régions (Voir

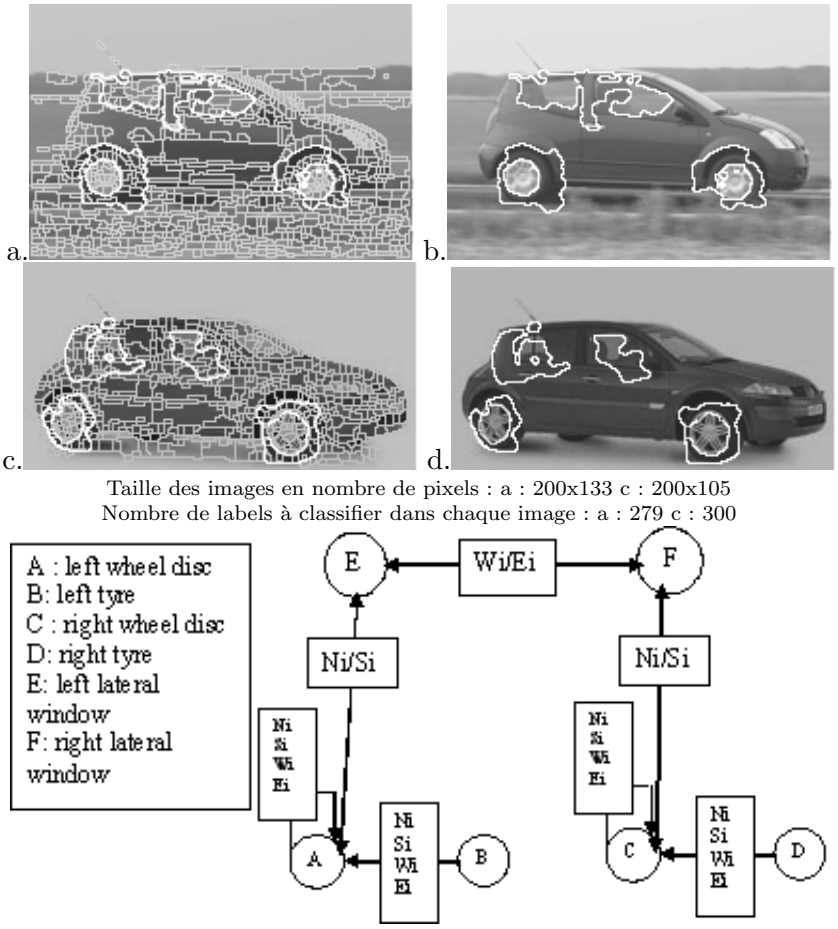
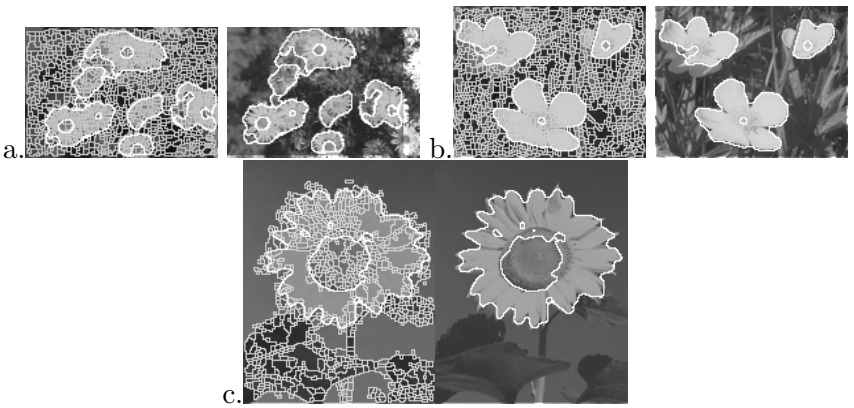


FIGURE 4.12 – Exemples d'interprétation d'une segmentation en fonction d'un graphe sémantique décrivant une voiture. A gauche (a,c) : régions détectées sur l'image d'origine. A droite (b,d) : régions détectées sur l'image segmentée par un algorithme de détection des lignes de partage des eaux. Les pneus ne sont pas parfaitement circulaires à cause des limites de l'algorithme de segmentation appliqué dans ces expérimentations.

Figure 4.15d). Vingt-cinq régions n'ont été associées à aucun nœud et correspondent à des régions bruitées ou à des vaisseaux sanguins non décrits dans le graphe. La table 4.1 montre le nombre de régions classées dans chaque nœud.

D'autres expérimentations ont été faites sur un ensemble d'images IRM obtenues sur le site internet "BrainWeb"(http ://www.bic.mni.mcgill.ca/brainweb/). Ce site internet contient une base de données d'images. Dans ces expérimentations, on s'intéresse à six structures cérébrales internes appelées "noyaux gris internes" et comprenant les thalamus, les noyaux lenticulaires et les noyaux caudés. L'interprétation est faite sur 10 coupes consécutives contenant ces noyaux (La Figure 4.16 montre 5 coupes extraites de cet ensemble d'images). L'analyse sémantique a été faite directement sur une segmentation fournie par un algorithme de détection des lignes de partage des eaux. L'algorithme de vérification de la consistance d'arcs doit gérer un grand nombre de régions (entre 500 et plus de 1000 régions). Sur chaque couche les 6 noyaux gris sont correctement



Taille des images en nombre de pixels : a. 142x95 b. 146x115 c. 148x187
Nombre de labels à classifier dans chaque image : a. 383 b. 460 c. 259

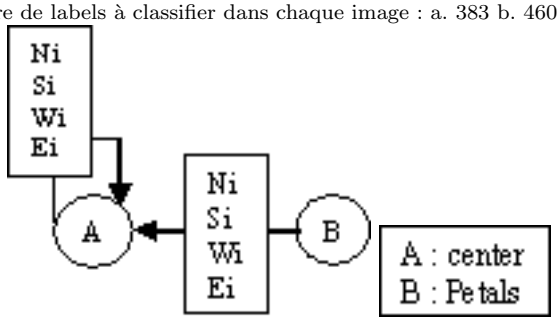


FIGURE 4.13 – Examples d'interpretation de segmentations en fonction d'un graphe sémantique décrivant une fleur (pétales et centre). A gauche : régions détectées sur l'image d'origine. A droite : régions détectées sur l'image segmentées par un algorithme de détection des lignes de partage des eaux.

identifiés. Les résultats sur le cortex pourraient être meilleurs si l'analyse avait été faire en 3 dimensions. L'extension de notre méthode à une analyse en 3 dimensions est un projet en cours.

4.5 Commentaires et conclusion

Dans ce chapitre, nous avons proposé une nouvelle façon d'exprimer des relations spatiales complexes, en particulier par l'apport d'une nouvelle notion que nous avons créée, celle de boîte minimale englobante des frontières interfaces.
Nous avons aussi introduit la notion de hub d'arcs dans la structure des nœuds pour pouvoir établir des relations logiques complexes entre les contraintes d'arcs.
Grâce à cela nous avons montré qu'il est possible d'exprimer des relations directionnelles ainsi que des relations topologiques telles que "est entouré par". Ces relations permettent de construire un graphe sémantique décrivant plus précisément un objet constitué de plusieurs sous-parties. Avec l'algorithme $AC4_{BC}$, ce graphe sémantique peut être utilisé pour retrouver des objets dans une image. Des expérimentations ont été faites sur des images réelles et nous avons montré qu'il est possible de retrouver différents type

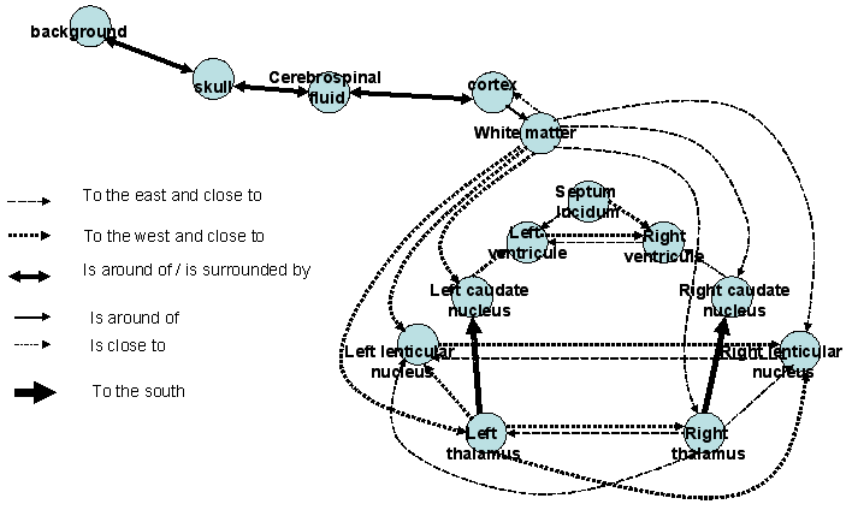


FIGURE 4.14 – Structure du graphe sémantique représentant l'anatomie du cerveau

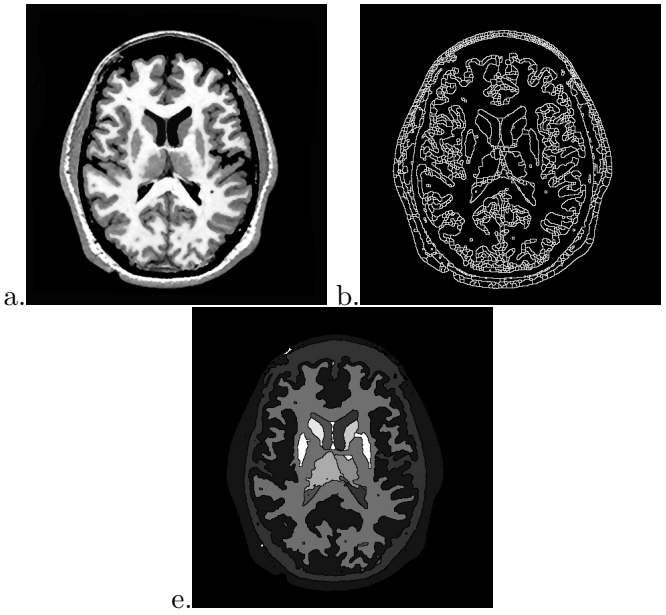


FIGURE 4.15 – a. Image d'origine. b. Image sur-segmentée par un algorithme de détection des lignes de partage des eaux. c. Image interprétée : Les noyaux gris internes gauches et droits ont été correctement différenciés.

Nom du nœud	Nb de régions à l'étape d'initialisation	Nb de régions après l'étape d'élagage
fond	5	1
Scalp	294	193
Cortex	196	51
Matière blanche	57	8
Ventricule gauche	17	2
Ventricule droit	17	2
Thalamus gauche	189	2
Thalamus droit	189	5
Noyau caudé gauche	155	1
Noyau caudé droit	155	3
Noyau lenticulaire gauche	155	1
Noyau lenticulaire droit	155	1
Septum lucidum	167	1
Liquide céphalo-rachidien	18	8

TABLE 4.1 – Nombre de régions dans chaque nœud, sélectionnées initialement sur un critère unaire, puis après les éliminations consécutives à la vérification de la consistance d'arcs.

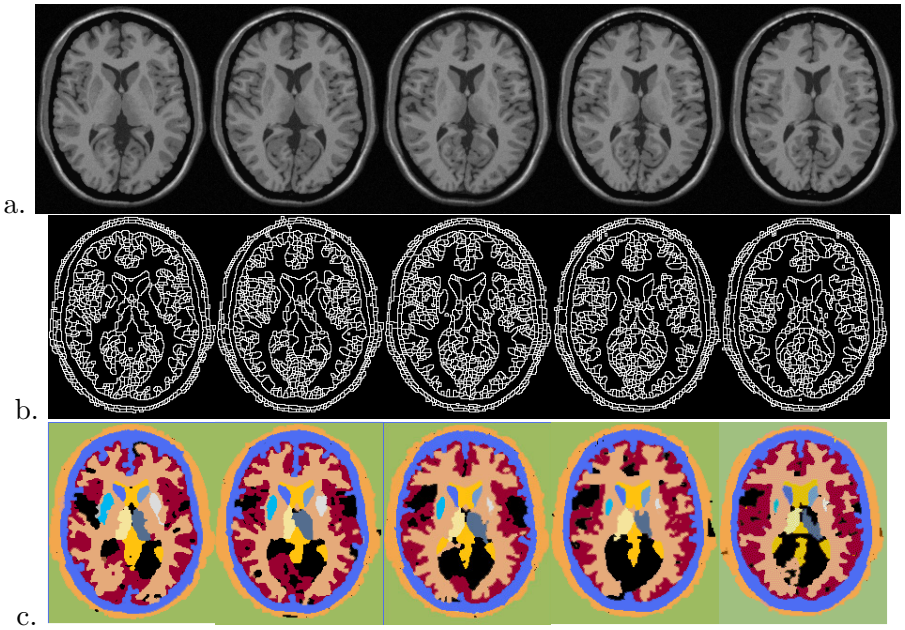


FIGURE 4.16 – Expérimentations sur un ensemble d'IRM cérébrales. a. Images d'origine, b. Segmentation obtenue avec un algorithme de détection des lignes de partage des eaux (nombre de régions dans chaque image avant d'appliquer l'analyse sémantique : 625, 552, 819, 746, 1102.)
c. Images interprétées : toutes les structures anatomiques sont identifiées par une couleur. Les six noyaux gris (Noyau caudé, Noyau lenticulaire et Thalamus droits et gauches) visibles sur chaque coupe sont correctement identifiés.

d'objets très différents tels que des visages, des voitures ou des fleurs et d'étiqueter des images anatomiques cérébrales.

Chapitre 5

Intégration d'un système d'interprétation d'images dans un processus de segmentation

5.1 Introduction

La segmentation d'images a souvent besoin d'un jugement humain pour ajuster les paramètres de segmentation afin d'avoir de bons résultats. Ce jugement de l'adéquation d'une segmentation peut être vu comme un processus sémantique. Introduire une analyse sémantique dans un processus de segmentation pourra donc améliorer le résultat de celle-ci. Juger de la qualité d'une segmentation n'est pas un problème trivial. Il dépend souvent de ce que l'on recherche dans l'image et de l'objectif de l'application (Voir Figure 5.1). Certains travaux proposent d'introduire une telle analyse sémantique dans un processus de segmentation, mais ils utilisent des représentations hétérogènes pour la connaissance de haut-niveau (les objets) et pour la connaissance de bas-niveau (intensités de niveau de gris) [63, 64]. Cette hétérogénéité entre les différents niveaux de représentation n'est pas satisfaisante. En effet, l'hétérogénéité est synonyme de complexité et l'homogénéité est synonyme de simplicité. La recherche de la simplicité est une heuristique utile en science. Pour cette raison, trouver une représentation homogène, adaptée pour permettre l'intégration naturelle d'une analyse sémantique dans un processus de segmentation, est plus simple.

Le formalisme des graphes est un outil qui peut permettre d'atteindre cet objectif. En effet, ce formalisme peut être appliqué à tous les niveaux de représentation, du plus bas niveau au plus haut.

Beaucoup de travaux utilisent les graphes sémantiques [64] pour décrire les objets

FIGURE 5.1 – Ces images ont des interprétations variées selon ce que l'on cherche dans l'image

présents dans une image. Dans ce type de graphe, les nœuds représentent les sous-parties des objets considérés et les arcs représentent les relations spatiales entre les sous-parties [8, 11, 42, 45, 60, 65]. Jusqu'à maintenant, la limite de beaucoup de ces travaux sur les graphes sémantiques réside dans la nécessité de travailler avec une image correctement segmentée, de façon à ce qu'elle puisse être étiquetée comme dans Bauckage et al. [60]. Malheureusement, obtenir une segmentation correcte reste un problème non résolu pour beaucoup d'images de la vie réelle.

La notion de graphe d'adjacence de régions est une représentation de bas niveau souvent utilisée pour segmenter une image. Beaucoup d'auteurs ont implanté cette approche avec succès. Par exemple, il est possible d'obtenir une segmentation efficace en construisant une pyramide de graphes d'adjacence [49, 52, 54, 66–70]. Dans ce type de processus les nœuds représentant les pixels ou des ensembles de pixels sont fusionnés dans une succession d'étapes hiérarchiques pour produire une pyramide de graphes où les nœuds sont associés à des régions ayant une signification. Ce processus de fusion peut être dirigé par plusieurs critères dans l'espoir d'obtenir des régions finales qui ont une signification.

Cela nous conduit donc à nous poser la question suivante : Comment est-il possible d'introduire une paramétrisation automatique dans le processus de fusion de façon à ce qu'il contrôle automatiquement la pertinence sémantique du résultat, en d'autres mots, la

compatibilité sémantique ? Même si les graphes d'adjacence et les graphes sémantiques sont utilisés pour différents niveaux de représentation, leur formalisme sont proches et peuvent être considérés comme homogènes. La principale différence réside dans le fait que les nœuds des graphes d'adjacence sont des régions de pixels et dans les graphes sémantiques les nœuds sont des composants du contenu sémantique de l'image. L'association de ces deux types de représentation permet de combiner, d'une façon très naturelle, les différents niveaux du traitement de l'image. Une idée simple consiste à appliquer un jugement sémantique sur le dernier graphe d'adjacence obtenu après une processus de segmentation de bas-niveau. Ceci peut être effectué en vérifiant si le graphe d'adjacence peut être mis en correspondance ou non avec le graphe sémantique. Le but de ce chapitre est de montrer comment le formalisme des graphes peut être utilisé pour représenter de façon homogène le contenu de bas-niveau (graphe d'adjacence) et le contenu sémantique (graphe sémantique) des images et de montrer comment il est possible dans ce contexte de vérifier la consistance sémantique d'une segmentation. Pour atteindre cet objectif :

— Nous montrerons comment introduire, dans un processus de segmentation, un processus de mise en correspondance de graphes à l'aide du principe de la vérification de la consistance d'arcs à deux niveaux de contraintes présentée dans les chapitres précédents (Partie 5.2). Cette analyse sémantique peut être faite indépendamment de l'algorithme de segmentation utilisé pour produire l'image segmentée. Cependant, de façon à montrer comment utiliser une telle analyse sémantique plus concrètement, nous proposons dans ce chapitre, son intégration dans une méthode de segmentation basée sur un processus de fusion pyramidale. Le critère de contrôle de ce processus pyramidale est automatiquement ajusté par un retour de la vérification sémantique finale.

— Nous montrerons sur quelques types d'images dont le contenu sémantique peut être décrit par un graphe sémantique, la capacité de notre approche à fournir une segmentation consistante grâce à cette connaissance (Partie 5.3).

— Enfin, ce type d'application nécessitant une exécution fréquente de l'algorithme de vérification de la consistance d'arcs à deux niveaux de contraintes, nous proposerons une optimisation de ce dernier.

5.2 Algorithme de segmentation guidé par la connaissance

Dans cette partie nous présentons une façon d'utiliser le contrôle sémantique dans le contexte du processus de segmentation. Le contrôle de la consistance sémantique peut être appliqué sur n'importe quel résultat de segmentation, quelle que soit la méthode de segmentation choisie. Il fournit une réponse oui ou non à la question de la consistance sémantique. Par elle-même, cette réponse ne donne pas la segmentation optimale,

qui dépend d'un ajustement adapté des paramètres de segmentation. Nous proposons d'ajuster automatiquement les paramètres d'un processus de segmentation en profitant de l'analyse sémantique.
Beaucoup de techniques de segmentation, fournissant une succession de résultats ordonnés et imbriqués en fonction des valeurs ordonnées de leurs paramètres, peuvent être concernées par cette approche. Parmi ces techniques, on peut citer les processus de segmentation basés sur un niveau d'échelle [71, 72], la décimation de graphes [68, 73] etc... Nous avons choisi le processus de segmentation basé sur la décimation de graphe pour illustrer la façon dont nous allons appliquer notre analyse sémantique à ce processus. La segmentation basée sur la décimation de graphe fusionne les régions adjacentes qui sont considérées comme similaires. Cette similarité est mesurée en utilisant des critères statistiques (écart-type, moyenne, etc.) calculés sur les niveaux de gris des régions [67]. Dans ce travail, nous considérons que deux régions sont similaires si leur différence de moyenne d'intensité des niveaux de gris est plus petite qu'un seuil donné. Le choix de ce seuil est souvent en lien avec la connaissance d'expert, qui évalue quel seuil conserve la consistance sémantique de la segmentation. Cette connaissance d'expert contient habituellement des informations sur l'organisation spatiale des régions décrivant le contenu de l'image. Cette connaissance peut être représentée de façon naturelle par un graphe sémantique. Dans la section 5.2.1, nous décrirons la construction d'une pyramide hiérarchique basé sur un processus de décimation de graphe et dans la section suivante comment introduire une analyse sémantique dans un tel processus.

5.2.1 Graphe hiérarchique et segmentation d'images

Un graphe d'adjacence de régions est défini de la façon suivante : Soit $G = (N, E)$ un graphe où N est l'ensemble des nœuds et E est l'ensemble des arcs. Soit $\delta_{ij} \stackrel{\text{def}}{=} (i, j) \in E$. Soit $V(i)$ le voisinage du nœud i défini par $\{j \in N : \delta_{ij}\}$. Nous supposons qu'un nœud donné i n'est pas un membre de son voisinage et que chaque nœud i est associé à deux valeurs :

- x_i : une information telle que le niveau de gris ou la couleur ;
- υ_i : la *qualité* relative d'un nœud, *i.e.* sa capacité *a priori* de survivre dans un processus de décimation. Nous reviendrons sur le sens de ce terme ci-dessous.

Dans la suite, nous considérons que les arc ne sont pas valués. Un processus de décimation transforme une graphe $G^{(k)}$ en $G^{(k+1)}$ tel que $|N^{(k+1)}| < |N^{(k)}|$. Meer a proposé dans [74] de contraindre ce processus avec deux règles.

Règle 1 : La décimation doit être maximale, *i.e.* $\forall (i, j) \in E^{(k)}$, i et j ne peuvent pas appartenir tout les deux à $N^{(k+1)}$.

C'est important si on souhaite que le processus de décimation converge rapidement vers un petit graphe.

Règle 2 : Tout nœud de $N^{(k)}$ doit être relié à un nœud de $N^{(k+1)}$, *i.e.* $\forall i \in N^{(k)}, (i \in N^{(k+1)} \vee V^{(k)}(i) \bigcap N^{(k+1)} \neq \emptyset)$.
La seconde règle assure qu'aucune information n'est perdue entre deux niveau successifs. Meer a alors introduit un algorithme dont le principal avantage est de pouvoir être exécuté de manière parallèle et de manière locale. Des détails concernant de processus de décimation peuvent être trouvés dans Lallich et al. [67] et Jolion [52]. La Figure 5.2 montre un exemple de ce processus de décimation. Le nouveau graphe est dans ce cas obtenu en deux itérations. Deux nœuds survivants sont extraits pendant la première itération mais la deuxième règle n'est pas satisfaite. La seconde itération extrait un troisième nœud et ainsi complète le nouveau graphe.

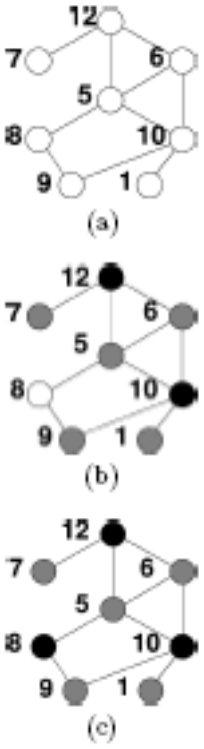


FIGURE 5.2 – Réduction de graphe par un processus itératif locale. (a) Le graphe valué. (b) Extraction des maxima locaux (en noir) et leurs voisins (en gris). Une cellule ne satisfait pas la deuxième règle. (c) le niveau suivant est complétement specifié en deux itérations.

Les nœuds de deux graphes consécutifs $G^{(k)}$ et $G^{(k+1)}$ sont reliés de la façon suivante : pour tout nœud $i \in N^{(k)}$, si $i \in N^{(k+1)}$ alors i est appelé un *nœud survivant*; on appelle donc les *pères* de i l'ensemble des nœuds défini par $\wp_i = V^{(k)}(i) \bigcap N^{(k+1)}$. L'ensemble inverse d'un nœud survivant i, appelé *enfants* de i, est défini par $C_i = \{j \in N^{(k)} : i \in \wp_j\}$.

Le processus de décimation peut être itéré jusqu'à ce que le graphe soit réduit à un unique nœud ($|N^{(apex)}| = 1$). Dans [66], ce processus a été appliqué à la segmentation d'images. Dans ce cas, la construction bottom-up d'une pyramide irrégulière peut être interprétée comme un algorithme de croissance de région. La valeur x_i est la moyenne des niveaux de gris du champ de réception, défini comme l'ensemble des nœuds dans $G^{(0)}$ relié au nœud survivant i par fermeture transitive sur la relation père. La qualité du nœud est en relation avec l'uniformité de son champ de réception, c'est à dire que υ_i est approximé par l'inverse de la variance de la distribution de x_j où j appartient au champ de réception de i. Les valeurs de qualité υ_i sont mises à jour pour chaque nouveau graphe. Ce processus est appelé *pyramide adaptative*. La principale différence avec le processus initial (i.e. la pyramide stochastique) est qu'un nœud non survivant i peut décider qu'il ne sera pas fusionné à un nœud survivant j si la différence entre leurs valeurs respectives, c'est à dire $|x_i - x_j|$, (le contraste local), est au dessus d'un seuil donné, appelé le *seuil de contraste*. Si le nœud satisfait au seuil, il survit mais ne peut pas être fusionné à aucun autre nœud. Le processus de décimation s'arrête lorsqu'aucun nœud ne peut fusionner. Augmenter le seuil fournit une segmentation plus grossière.

5.2.2 Introduction d'une analyse sémantique

Les relations spatiales peuvent être décrites dans un graphe sémantique. Nous intégrons alors un tel graphe sémantique dans un processus de décimation d'une pyramide adaptative [52]. Le point clé de ce processus de décimation est le choix d'un seuil de contraste décidant quelles régions sont similaires et quelles régions ne le sont pas. Dans cette étude nous considérons que deux régions ne peuvent pas être fusionnées si la différence de leurs moyennes d'intensité est plus grande qu'un seuil de contraste donné. Le meilleur contraste est celui fournissant une segmentation avec le plus petit nombre de régions compatibles avec la connaissance décrite dans le graphe sémantique. Puisque la segmentation obtenue peut être sur-segmentée, il est nécessaire de contrôler si elle satisfait la consistance d'arcs à deux niveaux de contrainte du graphe sémantique. L'algorithme suit un processus itératif (Figs. 5.3 et 5.4). Tant que la segmentation obtenue fournit un graphe arc-consistant, on recommence le processus avec un seuil de contraste augmenté. Dans les expérimentations présentées, nous avons simplement un pas d'incrémentation égal à 1. Cependant, de façon à aller plus vite, il est possible de modifier ce pas d'incrémentation en utilisant une stratégie dichotomique. Si le graphe devient inconsistant, on considère que la segmentation correcte est obtenue avec le seuil de contraste précédent.
Pour que cette approche d'exploration des seuils selon une stratégie dichotomique soit valide, le problème de l'unicité de la solution doit être considéré. De façon à assurer l'unicité, les contraintes imposées dans le graphe sémantique sont définies de façon à ce

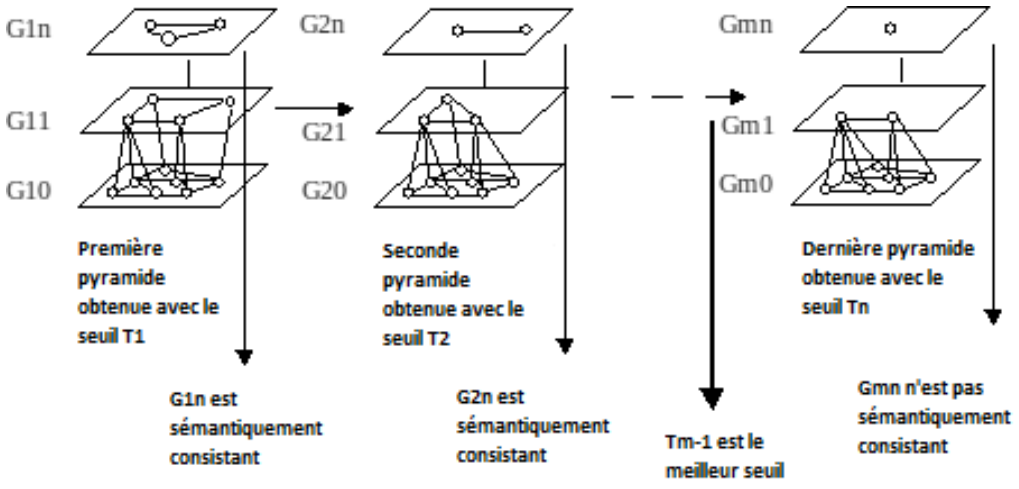


FIGURE 5.3 – Principe de l'algorithme. Une pyramide de graphes d'adjacence de régions (RAG) est construite pour chaque seuil T_i. Plus le seuil est haut, plus la fusion entre les régions du graphe d'adjacence est forte. Si la fusion est trop forte, le graphe d'adjacence du dernier niveau de la pyramide correspondra à une sous-segmentation. Dans ce cas, deux régions sémantiquement différentes seront fusionnées et le graphe d'adjacence ne sera pas sémantiquement consistant.

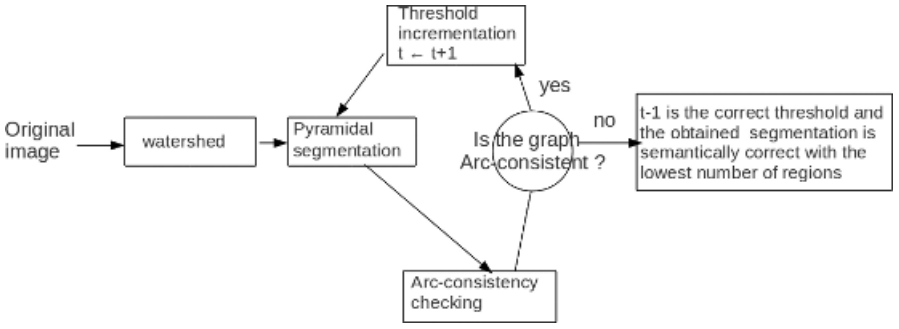


FIGURE 5.4 – Les differentes étapes du processus utilisé dans les expérimentations.

qu'elles soient vraies sur n'importe quelle sous-partie d'un objet ainsi que sur l'objet tout entier. Si le graphe est consistant à un niveau donné de segmentation, il est consistant à n'importe quel sous-niveau de la sur-segmentation.

5.3 Optimisation de l'algorithme $AC4_{BC}$: une solution systolique

L'introduction d'une analyse sémantique dans un processus de segmentation nécessite que l'analyse sémantique soit capable de gérer un grand nombre de régions segmentées le plus rapidement possible. Le point clé de la complexité de temps dans l'algorithme $AC4_{BC}$ réside dans l'appel de la procédure CleanKernel. Réduire le nombre d'appels

réduira la complexité de temps. $AC4_{BC}$ est dérivé de $AC4$. Dans l'étape d'élagage, à chaque fois un élément est retiré de la Queue, l'algorithme remplit à nouveau la Queue après l'avoir vidée. Cette stratégie est couteuse car elle implique de nombreux appels inutiles à la procédure CleanKernel qui produisent très peu d'effets. En effet une suppression dans une interface a très peu de chances de produire un changement dans le domaine Di du noyau du nœud. On démontre que la complexité de temps de la procédure CleanKernel est en $O(ed)$.

Théorème 5. La complexité de temps de l'étape de nettoyage (CleanKernel) est en $O(ed)$ dans le pire des cas, où e est le nombre d'arcs et d est la taille de D.

Preuve : On introduit la fonction SearchSucc(in Di, R, Cmpi, out S) (voir Figure 3.5 du chapitre 3) qui recherche les éléments successeurs de Di dans l'ensemble R en utilisant la relation $Cmpi$. Chaque nouveau successeur est marqué de façon à ce qu'un successeur déjà rencontré ne sera pas de nouveau considéré. Cette fonction est répétée jusqu'à ce que l'on ne rencontre plus de nouveau successeur. Puisque la taille de R est bornée par d, La complexité de temps des lignes 3-6 est au plus d fois. Le nombre d'interfaces D_{ij} à contrôler est donc au plus égal à e. Donc, la complexité de temps des lignes 7-12 est en $O(ed)$. Finalement, la complexité de temps de CleanKernel est en $O(ed)$.

En fait cette complexité de temps peut être définie de façon plus précise par ed_{it} où d_{it} est la taille du domaine Di au temps t de l'algorithme. Moins vite la taille de Di diminue, plus l'algorithme est lent.

Pour éliminer des appels inutiles de la procédure CleanKernel nous proposons de gérer la structure Queue d'une façon différente. La procédure CleanKernel n'est pas appelée tant qu'il est possible de retirer des labels des interfaces. Le processus implanté est systolique : La Queue est complètement vidée avant d'être à nouveau remplie. L'algorithme de la nouvelle étape d'élagage peut être décrite de la façon suivante :

1. Premièrement, la Queue est remplie avec les labels supprimés dans l'étape d'initialisation comme dans la version précédente de $AC4_{BC}$,
2. Deuxièmement, la Queue est vidée.
3. Troisièmement, la procédure CleanKernel est appelée pour chaque nœud ayant au moins un label supprimé. Cette étape remplit à nouveau la Queue. Ensuite les étapes 2 et 3 sont répétées jusqu'à ce qu'aucune suppression ne soit possible (voir la Figure 5.5).

Pour réaliser cela, un tableau Tabnode de booléens, ayant une taille égale au nombre de nœuds, est mis à jour à chaque fois qu'au moins une suppression a été faite dans un nœud. Donc, Tabnode[i] est égal à vrai si au moins un label a été retiré du nœud i. Ce tableau est initialisé à faux avant le début de l'étape d'élagage. Ce tableau permet de

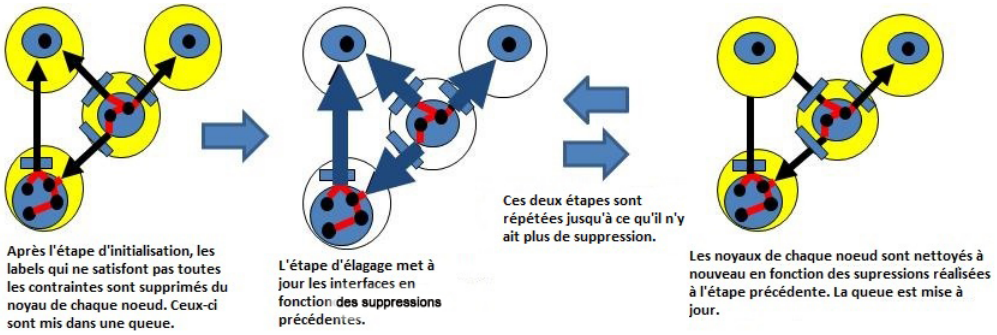


FIGURE 5.5 – Le processus systolique.

savoir quels sont les nœuds qui doivent être mis à jour par la procédure CleanKernel. Cette procédure n'est appelée que si c'est nécessaire, après avoir vidé la Queue, et étudié toutes les interfaces de tous les nœuds. Le pseudo code de l'étape d'élagage de la version optimisée de $AC4_{BC}$ appelée $OAC4_{BC}$ est donnés dans la Figure 5.6.

Réduire le nombre d'appels à CleanKernel réduira le temps de calcul du contrôle de la consistance d'arcs. Cependant, on peut imaginer que dans certains cas la structure Queue peut n'être remplie que par peu d'éléments. Donc le gain peut être faible et le changement de l'ordre de balayage des labels peut en fait conduire à un temps globalement plus long. En effet, cela peut se produire si on travaille en premier avec les labels dont la suppression a peu d'effet sur les autres labels. La complexité de temps dans le pire des cas de $AC4_{BC}$ et sa version optimisée $OAC4_{BC}$ sont les mêmes. Cependant, il est intéressant d'étudier expérimentalement la complexité de temps de cette nouvelle version. Nous avons réalisé cette étude sur un jeu de 26 images de compteur d'eau (voir Figure 5.7). L'objectif était ici de localiser le cadre et le centre du compteur d'eau. Dans ce cadre pendant le contrôle de la consistance d'arcs de chaque image, le nombre de labels supprimés des interfaces lorsque la Queue est vide et le nombre d'appels de la procédure CleanKernel pour obtenir le plus grand domaine arc consistant est enregistré à chaque cycle systolique. On compare

— le nombre de suppressions des interfaces x qui donne une idée du nombre d'appels à CleanKernel dans la version non optimisée de AC_{BC}.

— le nombre d'appels de CleanKernel y dans la version optimisée $OAC4_{BC}$.

Le ratio x/y est d'autant plus fort que l'optimisation est importante. Si l'optimisation change le coût en temps par un facteur d'échelle constant, x/y doit être constant pour tout x. Figure 5.8 montre que ce n'est pas vrai. La corrélation entre x et x/y (coefficient de Spearman $r = 0.93$, $p < 0.0001$) est très forte. Cela signifie que plus x est grand, plus le gain x/y est grand. Ce résultat suggère que l'ordre de la complexité de temps de $OAC4_{BC}$ est meilleur en moyenne que la complexité de temps de $AC4_{BC}$, au moins avec notre ensemble d'images test.

Step2 : Elagage des labels inconsistants
01 **for** each $i \in Arc(G)$ **do**
02 Tabnode[i] := false ;
03 remove :=true ;
04 **While** remove = true **do**
05 **begin**
06 remove :=false ;
07**While not** Emptyqueue(Q) **do**
08 **begin**
09 Dequeue(i,b,Q) ;
10 **for** each $(j, c) \in S[i, b]$ **do**
11 **begin**
12 Counter[(j,i),c] := Counter[(j,i),c]-1 ;
13 **if** Counter[(j,i),c]=0 **then**
14 **begin**
15 $D_{ji} := D_{ji} - \{c\}$;
16 Tabnode[j] := true ;
17 **end** ;
18 **end** ;
19**for** each $i \in Node(G)$ **do**
20 **begin**
21 **if**Tabnode[i]=true **then**
22 **begin**
23 Tabnode[i] :=false ;
24 **for** each $D_{ij} \in I_i$ **do**
25 **begin**
26 remove := CleanKernel(D_i, D_{ij}, I_i, Q) ;
27 **if** remove=true **then** Tabnode[i] :=true
28 **end**
29 **end**
30 **end**
31 **end**
32 **end** OAC_{4BC} ;

FIGURE 5.6 – Algorithme $OAC4_{BC}$: étape 2

5.4 Illustration

Dans la section précédente, il a été montré qu'il est possible d'introduire une connaissance sémantique dans un processus de segmentation.

Le but des expérimentations suivantes n'est pas de comparer notre méthode à d'autres techniques. En effet, notre analyse sémantique peut être appliquée à différentes techniques de segmentation fournissant une succession de résultats imbriqués en fonction des valeurs de ses paramètres et le résultat final dépend du choix d'une de ces techniques de segmentation.

Le but de ces expérimentations est de montrer que notre méthode peut améliorer un

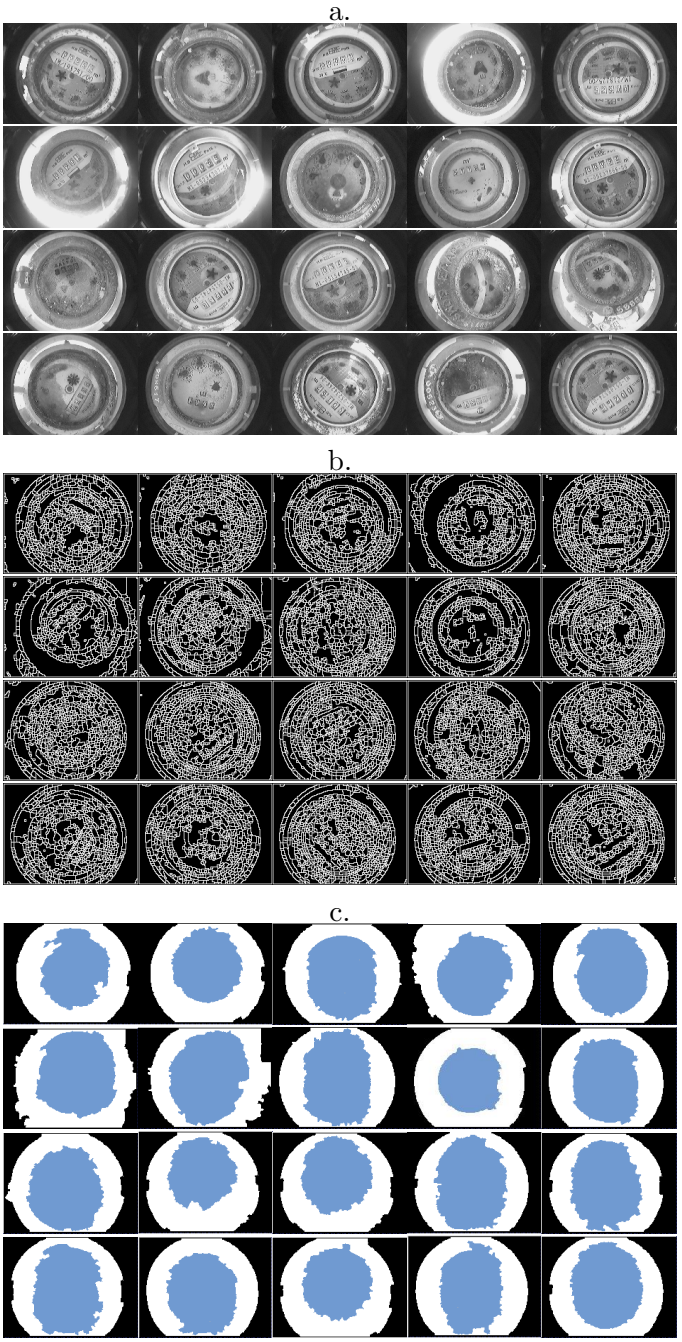


FIGURE 5.7 – Interprétation des images des compteurs d'eau . a : images d'origine, b : images segmentées avec un algorithme de détection des lignes de partage des eaux. c : détection du cadre et du centre du compteur d'eau. (Les images d'origine ont été fournies par la compagnie "Véolia")

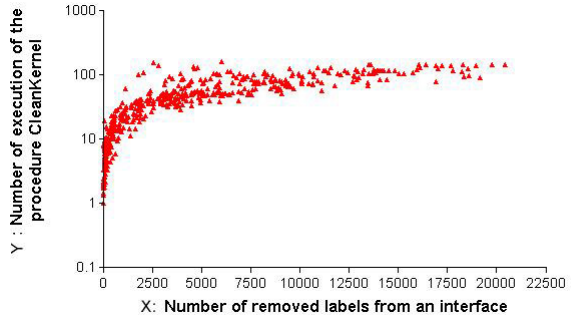


FIGURE 5.8 – Les expérimentations montrent que l'ordre de la complexité de temps de $OAC4_{BC}$ est meilleure en moyenne que la complexité de temps de $AC4_{BC}$

processus de segmentation initial en fournissant la possibilité d'ajuster automatiquement ses paramètres. Notre méthode est utile à chaque fois que le contenu de l'image concerne des objets spécifique constitués de plusieurs composants (par exemple, le cerveau est constitué du cortex, des noyaux gris, de la matière blanche etc...), qui peuvent être décrits par un graphe sémantique. Il est possible de construire un tel graphe pour des applications spécifiques telles que des applications médicales où il faut reconnaitre des structures anatomiques (par exemple l'anatomie du cerveau). Nous proposons ci-dessous trois types d'expérimentations.

La première est faite sur une image synthétique, la seconde est faite sur des images naturelles représentant des visages et la troisième est une application sur une image anatomique cérébrale. Le diagramme de la Fig. 5.4 montre les différentes étapes du processus utilisé dans les expérimentations. Dans toutes ces expérimentations, l'algorithme $OAC4_{BC}$ est utilisé.

5.4.1 Illustration sur une image synthétisée

5.4.1.1 Protocole

L'algorithme a été testé sur une image synthétisée représentant une fleur (voir la Fig. 5.9(a)). Cette fleur peut être décrite par un graphe sémantique très simple (voir la Fig. 5.9(b))qui représente les relations spatiales (à gauche/à droite, au dessus/en dessous) des différentes sous-parties (tige, feuille, pétales, centre) de la fleur. Les contraintes intra-nœud spatiales sont de deux types : "au dessus/en dessous" pour la tige et "autour de" pour les autres objets. La première contrainte est décrite par l'expression régulière $a^* + b^*$ où a dénote la relation "au dessus" et b dénote la relation "en dessous". La seconde contrainte est décrite par l'expression régulière $a^* + b^* + c^* + d^*$ où a, b, c et d dénotent les 4 directions cardinales. Une contrainte morphologique intra-nœud a été

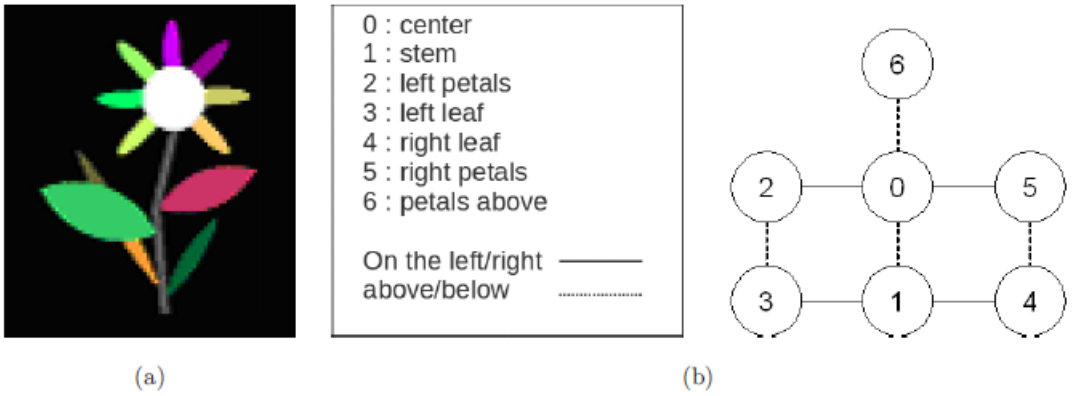


FIGURE 5.9 – Un graphe sémantique (b) décrivant une fleur et une image segmentée(a). Dans ce cas, le domaine D est constitué de l'ensemble des régions de l'image.

imposée sur la tige, qui doit être un objet vertical.
L'image synthétisée a été intentionnellement sur-segmentée (voir la Fig. 5.11(a)) et elle est constituée de 32 régions. Le graphe d'adjacence représentant cette segmentation est construit en associant une région segmentée à chacun des nœuds du graphe. Un arc est construit entre deux nœuds si les deux régions associées aux nœuds sont adjacentes. Ce graphe est le premier niveau de la pyramide. Le but est de retrouver la fleur initiale en fusionnant toutes les régions appartenant à la même partie de la fleur. Pour chaque valeur possible du seuil de fusion (dans cet exemple, la différence maximum entre la moyenne d'intensité de niveaux de gris de deux régions est choisie), la consistance sémantique du résultat est contrôlée avec notre analyse sémantique. S'il est consistant avec le graphe sémantique, le seuil est augmenté et le processus de fusion est poursuivie de façon à obtenir les plus grandes régions homogènes consistantes avec le graphe sémantique.

5.4.1.2 Résultats

Grâce au contrôle de la consistance d'arcs, la tige a été complètement identifiée et les feuilles sont correctement séparées des pétales (voir la Fig. 5.11(c)). Les régions appartenant au cœur de la fleur sont également bien identifiées, même si deux pétales sont fusionnés dans le cœur de la fleur. Les courbes de la Fig. 5.10 montrent que la valeur de seuil égale à 32 donne la segmentation optimale (segmentation avec le nombre de régions le plus petit permettant de rester sémantiquement consistant). Si cette valeur est augmentée, les feuilles fusionnent avec la tige et le graphe sémantique devient inconsistant. Dans cette expérimentation, notre méthode choisit le seuil qui aurait été manuellement sélectionné pour produire le meilleur résultat en fonction du critère choisi (ici c'est la moyenne en niveaux de gris de chaque région)

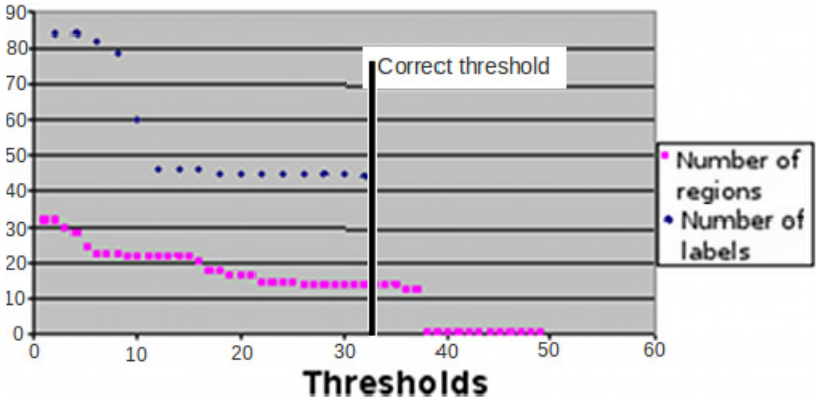


FIGURE 5.10 – Experimentations sur l'image synthétique : Nombre de régions et nombre de labels en fonction du seuil appliqué. Dans ce cas, le seuil correct est égal à 32, juste avant que le nombre de labels tombe à 0.

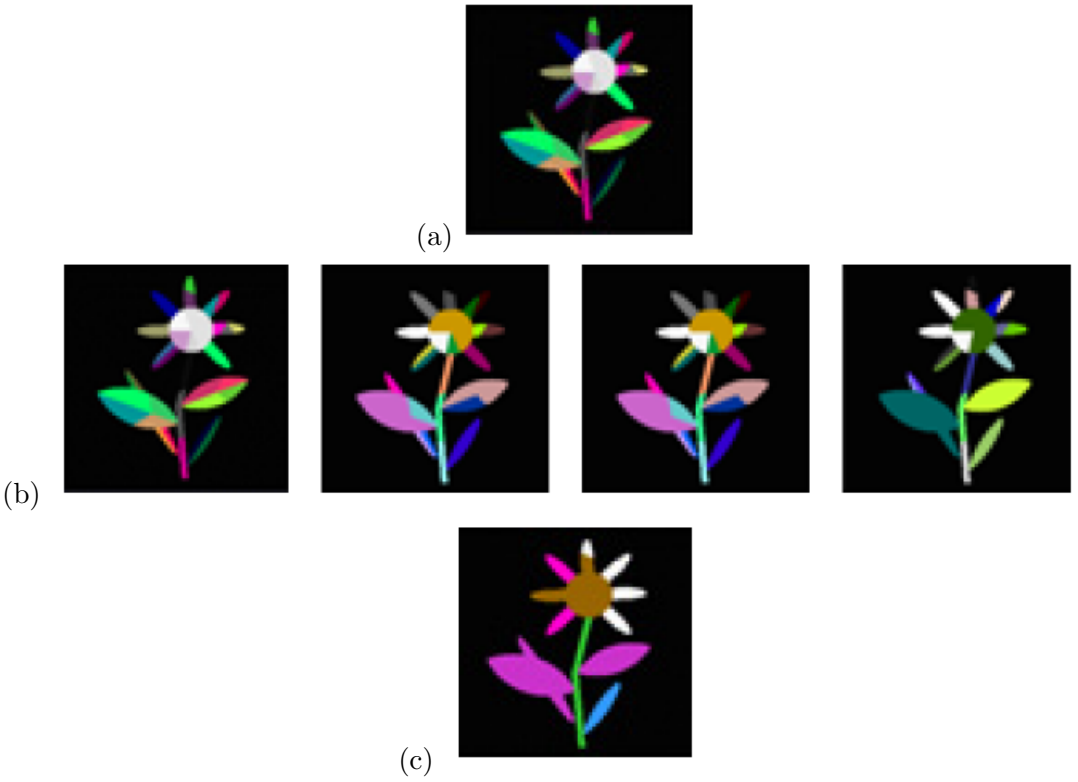


FIGURE 5.11 – (a) Image sur-segmentée ; (b) Les quatre étages de la pyramide avec un seuil égal à 32 ; (c) L'image étiquetée après le contrôle de la consistance d'arcs.

5.4.2 Illustration sur des images du monde réel

5.4.2.1 Protocole

Les expérimentations ont été réalisées sur des images de visages humains. Le but est de reconnaître les sous-parties correspondant aux cheveux, aux yeux, à la bouche, à la peau et au fond. Les relations spatiales entre ces différentes parties et leurs caractéristiques morphologiques peuvent être représentées par un graphe sémantique. Pour cet exemple, on utilise une description simple avec une graphe sémantique constitué de 6 nœuds, 63 arcs et 5 différents types de relations spatiales (Voir la Fig. 5.12). L'algorithme de segmentation est constitué de 4 étapes :

— Une segmentation initiale grossière : Bien que la base de la pyramide puisse être composée de régions d'un seul pixel, il est plus pratique de commencer avec des régions de plusieurs pixels, pourvu qu'elles soient suffisamment petites pour éviter des sous-segmentations non souhaitées. Le calcul des lignes de partage des eaux est un bon choix pour obtenir ces petites régions mais beaucoup d'autres algorithmes de segmentation sont possibles.

— Choix d'un critère capable de contrôler le processus de fusion. Beaucoup de critères peuvent être choisis (intensité moyenne des niveaux de gris des régions, variance, surface ...). Dans notre expérimentation, pour des raisons de simplicité, nous utilisons la moyenne d'intensité des niveaux de gris des régions.

— Construction automatique du graphe d'adjacence à partir des régions segmentées.

— Pour chaque valeur possible du seuil de fusion (différence maximale entre les moyennes des intensités des niveaux de gris entre les régions), exécution d'un processus de décimation pour fusionner successivement ces régions jusqu'à l'arrêt du processus de fusion lorsque le critère de fusion est supérieur au seuil. Comme pour l'exemple de l'image synthétisée, la meilleure valeur de seuil de fusion est automatiquement détectée grâce à notre analyse sémantique. Choisir un autre critère de fusion modifierait le résultat de la segmentation mais pas le fait que la valeur optimale de ce critère soit automatiquement détectée, quel que soit ce critère.

5.4.2.2 Résultats

Pour le premier exemple (voir la Fig. 5.13, 1ère ligne) le graphe est consistant jusqu'à ce que la valeur du seuil soit égale à 53. Avec cette valeur on obtient une segmentation faite de 192 régions et chaque région est correctement étiquetée. Avec le seuil suivant, l'œil gauche est fusionné avec la peau et le graphe devient inconsistant.
Pour le second exemple (voir la Fig. 5.13, deuxième ligne), le seuil qui donne le meilleur

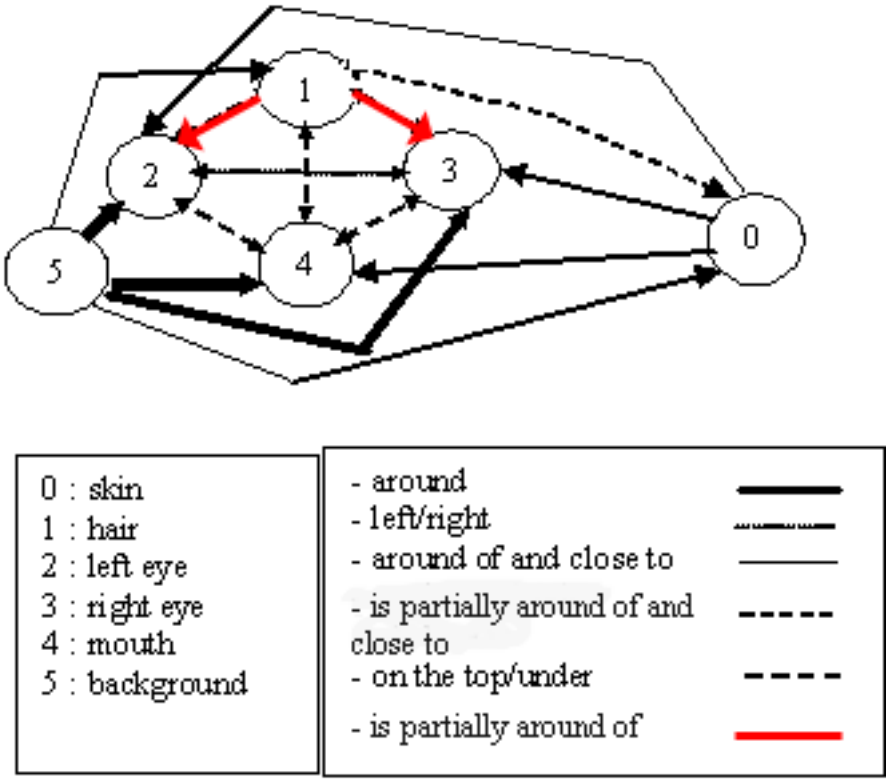


FIGURE 5.12 – Graphe sémantique décrivant un visage.

résultat de segmentation (arc-consistant) est 71. De même avec le seuil supérieur, l'œil gauche est fusionné avec la peau et le graphe devient inconsistant (voir la Fig. 5.13(d), deuxième ligne).

5.4.3 Illustration sur des images médicales

5.4.3.1 Protocole

Le protocole est similaire au précédent. Un graphe sémantique de l'anatomie du cerveau a été construit. Ce graphe contient 14 nœuds et 101 arcs. La structure générale de ce graphe est montrée dans la Fig. 5.14. Pour simplifier la figure, les relations "est autour de" et "est entouré par" sont représentées par un seul arc. Dans la pratique ces relations sont constituées de 4 arcs correspondants aux 4 directions cardinales. L'expérimentation a été faite sur l'image montrée sur la Fig. 5.15(a).

5.4.3.2 Résultats

Pour cette image (voir la Fig. 5.15) le graphe est consistant jusqu'à ce que le seuil soit égal à 34. Avec cette valeur on obtient une segmentation faite de 300 régions et chaque

FIGURE 5.13 – (a) Image d'origine ; (b) segmentation obtenue avec un algorithme de détection des lignes de partage des eaux ; (c) segmentation finale avec un seuil de fusion optimal ; (d) image obtenue avec le seuil de fusion situé juste au dessus :le graphe sémantique devient inconsistant.

région est correctement étiquetée. Avec le seuil du dessus, les thalamus gauche et droit sont fusionnés et le noyau caudé droit est fusionné avec le noyau lenticulaire. Le graphe devient alors inconsistant. Après la vérification de la consistance d'arcs, chaque nœud contient toutes les régions qu'un expert aurait mis dans ceux-ci et uniquement ces régions (voir la Fig. 5.15(e)). Vingt cinq régions ne sont associées à aucun nœud car ce sont des artéfacts ou des vaisseaux sanguins qui ne sont pas décrits dans le graphe sémantique.

5.5 Conclusion

Pour améliorer le processus de segmentation automatique d'une image, problème qui est loin d'être résolu, l'idée présentée ici est de diriger le processus de segmentation grâce à un contrôle sémantique.

Les travaux basés sur cette stratégie utilisent des représentations très différentes entre la bas niveau (pixels) et le haut niveau (concepts sémantiques) [63, 64, 75]. Des règles sémantiques simples sont parfois utilisées pour diriger unesegmentation [67, 69, 76]. Intégrer des heuristique simples pour améliorer une segmentation est une stratégies bien connues. Cependant, à notre connaissance, c'est la première fois qu'un processus de décimation dans une pyramide adaptative est dirigé par un critère sémantique décrit dans un graphe sémantique. Cette façon de faire dans le cadre du formalisme des graphes

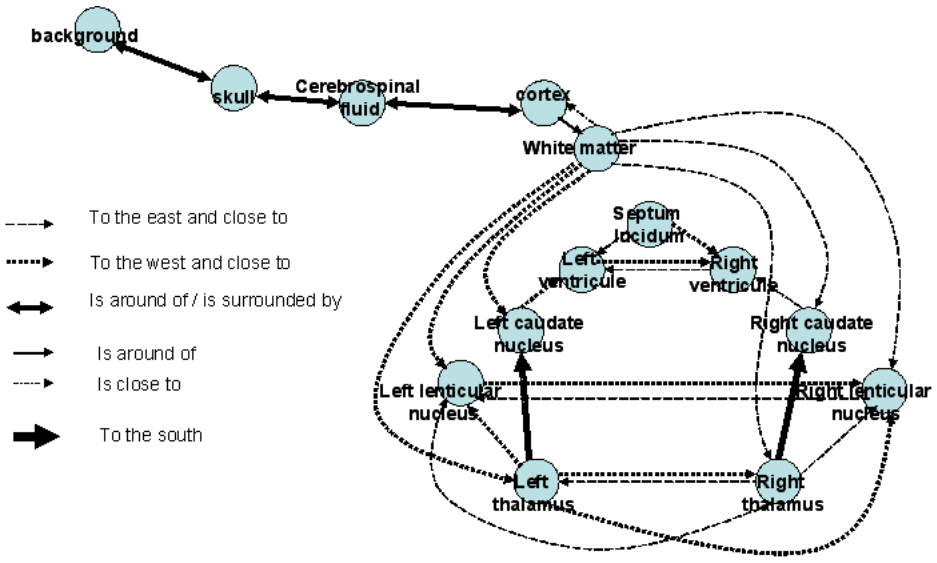


FIGURE 5.14 – Structure du graphe sémantique graph représentant l'anatomie du cerveau.

permet d'avoir une homogénéité de représentation de l'information de bas-niveau et de haut-niveau. Par ailleurs, l'optimisation de l'algorithme $AC4_{BC}$ permet d'utiliser cette analyse sémantique efficacement pour ajuster automatiquement les paramètres d'un processus de segmentation pyramidale. La version optimisée de l'algorithme $AC4_{BC}$ appelée $OAC4_{BC}$ a deux avantages :

— Elle donne la possibilité d'appliquer notre approche sur des images ayant plus de 800 régions segmentées et avec un graphe sémantique contenant 142 arcs. Cette expérimentation n'aurait pas été possible sans l'optimisation. Cela permet d'appliquer notre approche sur de vrais problèmes complexes.

— Elle donne la possibilité d'envisager la parallélisation de notre algorithme. Dans ce cas, la mise à jour de chaque nœud peut être considérée comme un processus individuel. Chaque nœud est mis à jour séparément (Voir les lignes 45-55 de la Figure 5.6). Les nœuds peuvent être mis à jour en une seule étape en parallèle. Les conséquences de cette mise à jour peuvent être envoyées aux autres nœuds dans une deuxième étape (Voir les lignes 32-43 de la Figure 5.6). Une telle implantation parallèle pourrait être faite dans le contexte de la programmation GPU.

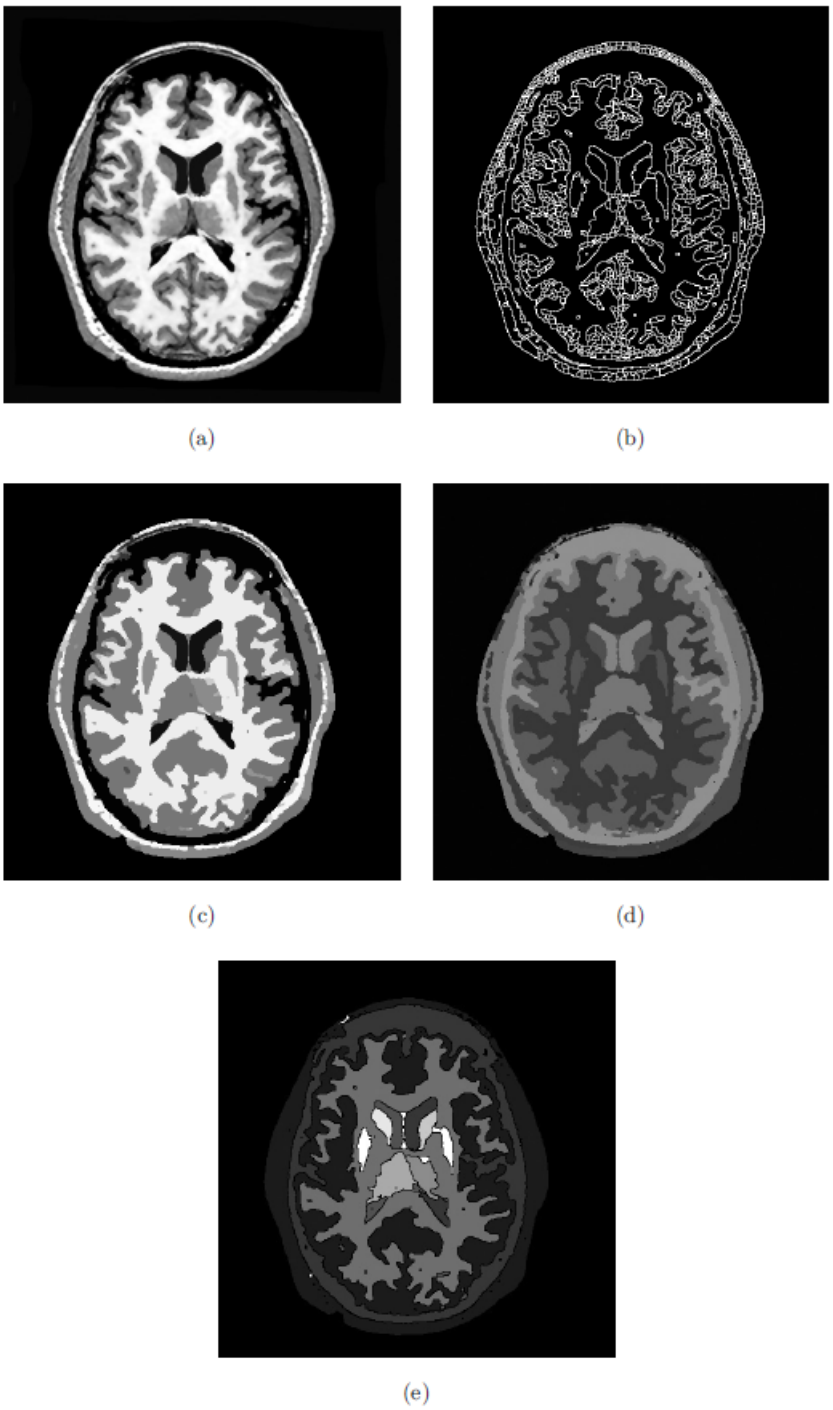


FIGURE 5.15 – (a) Image d'origine. (b) Image sur-segmentée par un algorithme de détection des lignes de partage des eaux. (c) Image segmentée sémantiquement consistante. (d) Image sous-segmentée : les thalamus gauche et droit sont fusionnés, le noyau lenticulaire droit est fusionné avec le noyau caudé. (e) Image interprétée : les noyaux gris internes gauches et droits sont correctement identifiés.

Chapitre 6

Extension de l'application de la satisfaction de contrainte à l'interprétation d'images : détection de formes pouvant être décrites mathématiquement

6.1 Introduction

Nous avons vu dans les chapitres précédents que les problèmes d'interprétation d'images peuvent être considérés comme des PSC sur un domaine fini. Ce cadre donne la possibilité de travailler avec une connaissance symbolique, les contraintes représentant la connaissance décrivant le contenu supposé de l'image. Le problème consiste à mettre en correspondance les régions segmentées d'une image avec cette connaissance.
Dans ces précédents travaux, l'objectif était de retrouver une organisation spatiale donnée entre les différentes parties d'un objet (généralement entre des éléments saillants de ces parties) et non de retrouver un objet avec un contour ayant une forme spécifique. Nous désirons maintenant retrouver plus précisément une forme donnée, ce qui implique une façon de décrire des contraintes applicables au contour des formes. Le travail présenté dans ce chapitre met l'accent sur la reconnaissance de formes géométriques qui peuvent être décrites par des équations mathématiques (lignes, cercles, courbes monotones, etc.). Comme il est généralement possible de subdiviser le contour d'une forme quelconque en un ensemble de segments ou lignes connectées (par exemple, une flèche épaisse peut être décrite par un ensemble de sept lignes, voir Figure 6.1), les possibilités de descriptions

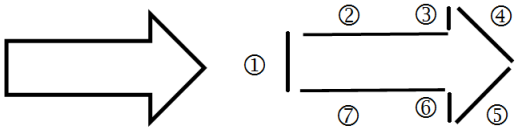


FIGURE 6.1 – Exemple de décomposition d'une forme (ici une flèche) en un ensemble de 7 lignes connectées

des objets deviennent très larges. Nous allons montrer dans ce chapitre qu'il est possible de reconnaître ces formes en les décrivant avec un graphe de contraintes. Toutefois, pour vérifier si une forme respecte les propriétés morphologiques décrites par des équations mathématiques, il est parfois nécessaire d'imposer des contraintes de type n-aires. Par exemple, pour vérifier si trois points sont sur un arc de cercle donné, nous devons construire une contrainte entre ces trois points. Dans ce cas, il est nécessaire de construire un hyper-arc et de vérifier la consistance de celui-ci. Cette vérification de la consistance d'hyper-arc à deux niveaux de contraintes constitue un nouveau développement et fournit de nouvelles possibilités pour vérifier des contraintes complexes. Dans un premier temps, nous exposerons les définitions de la consistance d'arcs et d'hyper-arcs à deux niveaux de contraintes (et les algorithmes associés vérifiant la satisfaction de contraintes). Ensuite nous présenterons les règles permettant de reconnaitre des formes géométriques, après avoir introduit au préalable la notion de points caractéristiques d'une région afin de définir des contraintes précises sur les régions segmentées. Enfin nous montrerons des résultats sur des images avec des formes géométriques variées de tailles et d'orientations différentes. Nous montrerons également qu'il est possible d'étendre notre approche à des formes non géométriques, bien que dans ce cas, la reconnaissance de la forme puisse ne pas être garantie.

6.2 Consistance d'arcs et consistance d'hypers-arc à 2 niveaux de contraintes

6.2.1 Problème de satisfaction de contraintes à 2 niveaux de contraintes ($FDPSC_{BC}$)

Dans le contexte du travail décrit ici, il est parfois nécessaire pour bien décrire les formes, de pouvoir définir plusieurs arcs entre deux nœuds donnés i et j. Nous donnons alors une nouvelle formulation de la définition 1 du Chapitre 3 (définition d'un $FDPSC_{BC}$) tenant compte de ce nouveau paramètre. Les arcs sont indicés par un nombre α.
On note $I_{i,\alpha}$ l'ensemble interface du nœud i associé à l'arc α. Comme nous l'avons vu dans le chapitre 3, dans un $FDPSC_{BC}$, on définit deux types de contraintes :

— les contraintes binaires inter-nœud $C_{ij,\alpha}$ représentées par un arc α entre deux nœuds i et j,

— les contraintes binaires intra-nœud $\mathcal{C}mp_{\mathrm{i},\alpha}$ entre deux valeurs du nœud i et associées à l'arc C_α. $\mathcal{C}mp_{\mathrm{i},\alpha}(a,b)$ signifie que a est compatible avec b étant donnée la contrainte $\mathcal{C}mp_{\mathrm{i},\alpha}$.

La définition de la satisfaction d'un $FDPSC_{BC}$ devient :

Définition 10. Soit $C_{ij,\alpha}$ une contrainte orientée entre i et j et soit $\mathcal{C}mp_{\mathrm{i},\alpha}$ une relation de compatibilité, telle que $(a,b) \in \mathcal{C}mp_{\mathrm{i},\alpha}$ ssi a et b sont compatibles. $\mathcal{C}mp_{\mathrm{i},\alpha}$ est associée à la contrainte $C_{ij,\alpha}$. Considérons $I_{i,\alpha} \subset D_i$ (Les éléments de $I_{i,\alpha}$ sont à "l'Interface" entre i et le nœud lié à i par l'arc α). $I_{i,\alpha}, D_j \models C_{ij,\alpha}$ signifie que $(I_{i,\alpha}, D_j)$ satisfait la contrainte orientée $C_{ij,\alpha}$ et $D_i, D_j \models (C_{ij,\alpha}, \mathcal{C}mp_{\mathrm{i},\alpha})$ signifie que (D_i, D_j) satisfait la contrainte orientée $C_{ij,\alpha}$. Dans le cadre de l'analyse d'image, les domaines D_i et D_j représentent les ensembles des régions segmentées ou des primitives extraites.

$$I_{i,\alpha}, D_j \models C_{ij,\alpha} \Leftrightarrow \begin{cases} \forall a_i \in I_{i,\alpha}, \exists a_j \in D_j, \\ tel\ que\ (a_i, a_j) \in C_{ij,\alpha} \end{cases}$$

$$D_i, D_j \models (C_{ij,\alpha}, \mathcal{C}mp_{\mathrm{i},\alpha}) \Leftrightarrow \begin{cases} \forall a_i \in D_i, \\ \exists (a'_i, a_j) \in I_{i,\alpha} \times D_j, \\ tel\ que \\ (a_i, a'_i) \in \mathcal{C}mp_{\mathrm{i},\alpha} \\ et\ (a'_i, a_j) \in C_{ij,\alpha} \end{cases}$$

L'ensemble $\{D_1, ..., D_n\}$ satisfait $FDPSC_{BC}$ ssi $\forall (C_{ij,\alpha}, \mathcal{C}mp_{\mathrm{i},\alpha}), D_i, D_j \models (C_{ij,\alpha}, \mathcal{C}mp_{\mathrm{i},\alpha})$. (Note : quand $I_{i,\alpha} = D_i$, nous sommes dans le cas de la consistance d'arcs classique. Donc, lorsque $\forall C_{ij},\ D_i, D_j \models (C_{ij})$, l'ensemble $\{D_1, ..., D_n\}$ satisfait le $FDPSC$ classique).

6.2.2 Problème d'Hyper-Arc consistance à 2 niveaux de contraintes (HAC_{BC}) associé à un $FDPSC_{BC}$

Afin de construire des contraintes plus précises, la gestion de contraintes n-aires est parfois nécessaire. Il est facile de généraliser la notion de contraintes à deux niveaux lorsque les contraintes orientées inter-nœuds deviennent n-aires. Dans ce cas, nous avons un hyper-graphe. Un PSC est hyper-arc consistant si toutes ses contraintes sont hyper-arc consistantes.

Les Figures 6.2 et 6.3 décrivent l'algorithme résolvant le problème HAC_{BC}. Cet algorithme comporte deux étapes : une étape d'initialisation et une étape d'élagage. La

begin HAC_{BC}
Step1 : Construction des structures de données.
InitQueue(Q) ;
for each $i \in node(G)$ **do**
  for each $b \in D_i$ **do**
    begin
      S[i,b] := empty set ; *//(S[i,b] est l'ensemble des*
      //(label,node,arc) supportés par b dans le nœud i
    end ;
for each $a(i, j_1, ..., j_n, C_\alpha) \in arc(G)$ **do**
  for each $b \in D_i$ **do**
    begin
    cpt :=0 ;
    Counter[a,b] :=0 ;
    for each $(c_1, c_2, ..., c_n) \in D_{j_1} \times D_{j_2} \times ... \times D_{j_n}$ **do**
      if $C_\alpha(b, c_1, ..., c_n)$ **then**
        begin
          cpt :=cpt+1 ;
          Counter[a,b] := Counter[a,b]+1
            // nombre de supports de 'b' par l'arc 'a'
          for each $(j_k, c_k) \in ((j_1, c_1), ..., (j_n, c_n))$ **do**
          S[j_k, c_k] :=S[j_k, c_k] $\cup$ ((i,b,a),cpt) ;
        end if
      if Counter[$a(i, j_1, ..., j_n, C_\alpha)$,b]$> 0$
      then $I_{i,\alpha} := I_{i,\alpha} \cup \{b\}$;
    end ;
for each $i \in node(G)$ **do**
    CleanKernel(D_i, I_i, Q) ;
      //supprime les labels non supportés dans nœud i
      //(non atteints par $\mathcal{C}mp_{i,a}$) et les met dans Q

Step2 : Elagage des labels inconsistants
  While not Emptyqueue(Q) **do**
    begin
      Dequeue(i,b,Q) ;
      for each $(j, c, a(j, i_1, ..., i_n, C_\alpha, cpt)) \in S[i, b]$ **do**
      begin
        for (k from 1 to n) **do** *// supprime les supports*
        //de c reliés au même hyper-arc et créés
        //avec la même valeur cpt
          S[i_k, b] :=S[i_k, b]- ((j,b,a),cpt)
        end for k
        Counter[a(...),c] := Counter[a(...),c]-1 ;
        if Counter[a(...),c]=0 **then**
          begin
          $I_{j,a} := I_{j,a} - \{c\}$;
          CleanKernel(D_j, I_j, Q) ;
          end ;
  end ;
  end HAC_{BC} ;

FIGURE 6.2 – L'algorithme HAC_{BC} avec quasi arc-consistance : étape 1 et étape 2.

```
CleanKernel(inD_i, I_i, outQ)
    for each b ∈ D_i do nbnotsatisfied(b)=0
    for each I_{i,α} ∈ I_i do //Pour chaque interface I_{i,α} de i
      for each r ∈ I_{i,α} do
      begin
        for each b ∈ D_i do markseen(r, b) := false;
        E := {r} //Ensemble initial
        E' := ∅ //Ensemble final
        while E ≠ ∅ do
          begin
          for each e ∈ E do
            for each v ∈ neighbourg(e) do
            begin
            if (CmpiChecking(r, e, v, equation)) then
                begin
                E' := E' ∪ v
                markseen(r, v) := true
                end if
            end for
        E := E'
        end while
      end
      for each b ∈ D_i do
        begin
        for each h ∈ Hub_i do //Ensemble des arcs associés à un nombre de relâchement
          for each I_{i,α} ∈ I_{i,h} do
              if ∀r ∈ I_{i,α}, ¬markseen(r, b) then
              nbnotsatisfied(b)=nbnotsatisfied(b)+1
          if nbnotsatisfied(b) > Relax(i, h, α) then
              begin
                  EnQueue(i,b,Q)
                  for each I_{i,α} ∈ I_i do I_{i,α} := I_{i,α}-{b}
                  D_i := D_i -{b}
              end if
          end for
        end for
        end
      end for
```

FIGURE 6.3 – La procédure CleanKernel

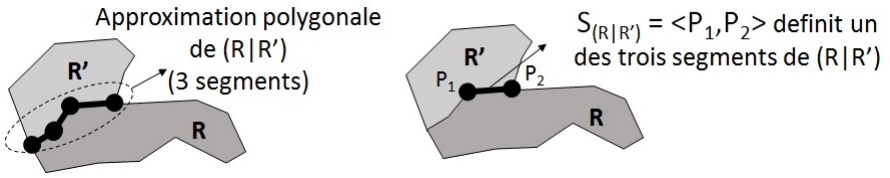


FIGURE 6.4 – Illustration de l'approximation polygonale.

procédure *CleanKernel* permet de gérer les relations intra-nœud. Elle appelle la fonction *CmpiChecking* qui vérifie si les régions affectées à un nœud donné satisfont une propriété morphologique donnée grâce à des équations mathématique décrivant la forme recherchée.
Dans la Figure 6.2, on note $a(i, j_1, ..., j_n, C_{i,\alpha})$ l'hyper-arc a associé au nœud i et à l'ensemble des nœuds $j_1, ..., j_n$. $C_{i,\alpha}$ est la contrainte appliquée à cet arc.

Pour savoir si une région segmentée est compatible avec les contraintes équationnelles, il faut vérifier si les pixels de bord de cette région sont compatibles avec les contraintes de l'équation. Stocker les coordonnées de tous les pixels de bord d'une région segmentée n'est pas très pratique et quelques pixels caractéristiques de la région segmentée peuvent résumer l'ensemble du contour de cette région.

6.3 Points caractéristiques

Pour définir des points d'intérêt décrivant les relations entre les deux régions dans une direction donnée, on définit la frontière qui se situe à l'interface entre une région R_1 et une région R_2 comme l'ensemble des pixels de la région R_1 qui sont connectés à des pixels de la région R_2. Parmi les différents points caractéristiques de cette frontière, on choisit les points suivants :

— L'extrémité convexe de la frontière.

— Les points correspondant aux extremums de la frontière.

En outre, une approximation polygonale est appliquée à la frontière dans chaque direction. Chaque segment de l'approximation est au plus à une distance de σ pixels de la frontière (Voir Fig. 6.4). Les sommets sont stockés afin de vérifier, dans le processus de satisfaction de contraintes, si ils satisfont à la contrainte imposée par une équation mathématique donnée d'une courbe théorique. Cette équation mathématique décrit une partie du bord de l'objet que nous recherchons.

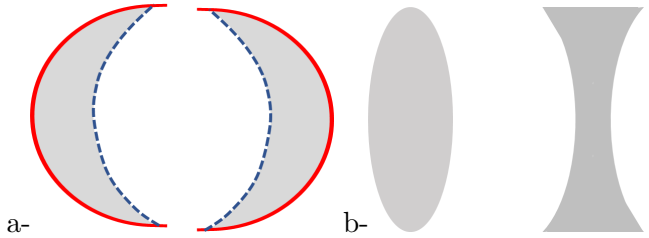


FIGURE 6.5 – La notion concavité et de convexité du bord d'une région, utilisée dans ce chapitre, est en référence à l'intérieur de la région.
A gauche, les courbes en traits pleins rouges sont convexes par rapport à la région et les courbes en traits pointillés sont concaves.
A droite la première forme est biconvexe et la deuxième est biconcave.

6.4 Règles pour retrouver une courbe avec des contraintes locales

Le problème est de savoir comment, à partir des contraintes locales (entre les petites régions segmentées), nous pouvons assurer une contrainte globale (la forme globale à retrouver dans l'image). Nous proposons trois règles locales qui garantissent que la forme globale suivra une courbe théorique.
Avant de présenter le formalisme qui nous permettra de démontrer la propriété que nous venons d'énoncer, nous allons présenter l'idée intuitive de ces 3 règles :

— Règle 1 : Une zone de bord doit être "bord interne compatible" (voir la figure 6.6). Si une zone de bord est une partie d'une courbe définissant une partie de la forme de l'objet, elle doit avoir au moins un segment de bord compatible avec la courbe et tous les autres bords de cette région doivent être à l'intérieur de la forme.

— Règle 2 : Aucune des régions connectées à une région constituant une partie du bord de la forme à détecter ne peut avoir des bords qui sont, à la fois, à l'intérieur et à l'extérieur de la forme.

— Règle 3 : La courbe de bord définissant une partie de la forme de l'objet doit avoir une dérivée monotone (la courbe est simple) et ne doit pas être concave (la concavité ou convexité de la courbe se défini par rapport à l'intérieur de la région comme la concavité ou la convexité d'un verre optique voir la figure 6.5).

Nous allons montrer que ces règles garantissent qu'en travaillant sur une courbe simple (il est généralement possible de segmenter une forme complexe en un ensemble de courbes simples), nous pouvons contraindre une chaîne de régions connectées à suivre la courbe du début à la fin sans interruption.

Notation :

— On note R une région d'une image I.

— On note $\hat{R}$ les contours de R et $\dot{R}$ l'intérieur de R tel que : $\dot{R}$ soit le complémentaire de $\hat{R}$ dans R.

— Soient deux régions $\{R, R'\} \subset I$ avec $R \bigcap R' = \emptyset$ et R connectée à R', on note $(R \mid R')$ les pixels de bord de R connectés aux pixels de R' dans une direction donnée (voir Fig. 6.4).

— On note $\mathcal{C}_f$ une courbe définie par une équation, non concave et ayant une dérivée monotone entre x_{min} et x_{max} telle que $\mathcal{C}_f$=$\{(x, y) \in I, y' = f(x)$, y est la partie entière de $y'\}$ (x est fixé si on travaille dans la direction nord ou dans la direction sud). $\mathcal{C}_f$ est une partie d'un contour d'une forme plus complexe.

— On note d une fonction de distance orientée (c'est à dire peut être négative selon le repère choisi) euclidienne telle que :

  — $d(P, P')$ est la distance entre un point P et un point P'.

  — $d(P, D)$ est la distance entre un point P et une ligne D.

  — $d(P, \mathcal{C}_f)$ est la distance entre un point P et une courbe $\mathcal{C}_f$. $d(P, \mathcal{C}_f) < -\epsilon$ si P est "interne" à la courbe définissant un bord d'une région, et $d(P, \mathcal{C}_f) > \epsilon$ si P est "externe" à la courbe définissant un bord d'une région.

— On note $S_{(R|R')} = \langle P_1^{S_{(R|R')}}, P_2^{S_{(R|R')}} \rangle$ une partie contigüe de $(\hat{R} \mid R')$ telle que $P_1^{S_{(R|R')}}$ et $P_2^{S_{(R|R')}}$ sont le début et la fin de $S_{(R|R')}$ et $[P_1^{S_{(R|R')}}, P_2^{S_{(R|R')}}]$ est un segment obtenu par approximation polygonale de $(\hat{R} \mid R')$. Pour simplifier, on notera par commodité S_R à la place de $S_{(R|R')}$ (Voir Fig. 6.4).

— On note CP_R l'ensemble des points caractéristiques de R. CP_R =$\{P_c \in \hat{R}$, $\exists S_R \subset \hat{R}$, $P_c = P_1^{S_R}$ ou $P_c = P_2^{S_R}\}$.

— Soient deux régions R_{head} et R_{tail} telles que $(x_{min}, f(x_{min}))$ est un point caractéristique de R_{head}, et $(x_{max}, f(x_{max}))$ est un point caractéristique de R_{tail}.

— Soit E un ensemble de régions connectées formant une chaîne entre R_{head} et R_{tail}.

— Soit Sh_{in} la partie interne d'une forme et Sh_{out} la partie externe d'une forme.

Pour tout R appartenant à E, on peut montrer que si R respecte les 3 règles énoncées précédemment et associées à une courbe théorique $\mathcal{C}_f$, l'ensemble E des régions connectées décrivent $\mathcal{C}_f$.
Présentons maintenant comment les 3 règles sont mises en œuvre.

Application de la règle 1 : R doit être "bord interne compatible" avec la forme définie par $\mathcal{C}_f$: Si D_i est le domaine du nœud i correspondant au bord de la forme, alors $R \in D_i$ reste dans le nœud si les propriétés suivantes sont satisfaites :

1. Tous les points caractéristiques P de R doivent être tels que $d(P, \mathcal{C}_f) \leq \epsilon$ ce qui signifie que P doit être soit sur la courbe à ϵ près (on autorise une petite marge

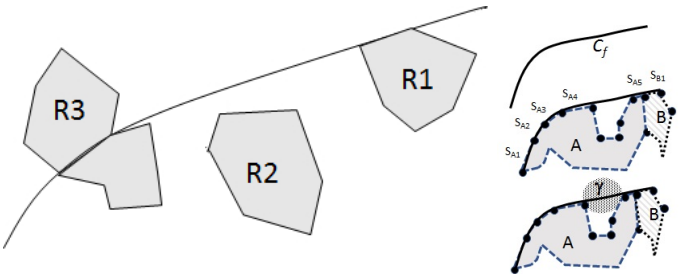


FIGURE 6.6 – A gauche : R1 suit la règle 1, mais ni R2 (qui ne forme pas un bord avec la courbe C tracée sur cette figure), ni R3 (qui a des pixels à la fois au dessus et en dessous la courbe C) suivent cette règle.
A droite : Détail de comment les régions A et B ont des bords qui suivent la courbe $\mathcal{C}_{\{}$, la région A a 5 segments qui suivent la courbe et la région B a 1 seul segment. La zone grisée γ en bas à droite illustre la question de la non rupture de la courbe $\mathcal{C}_{\{}$ par une région voisine de A. Comment garantir cette propriété ?

d'erreur pour tenir compte de l'irrégularité des bords des régions segmentées), soit dans la partie de la courbe définie comme étant l'intérieur de l'objet. $\forall P \in S_{cp_R}$, $d(P, \mathcal{C}_f) \leq \epsilon$ (Régions R1 et R2 de la Figure 6.6).

2. Au moins deux points définissant un segment du bord de R doivent être sur la courbe et le milieu de ce segment doit être cohérent avec l'équation de la courbe : $\exists S_R \in R$, $\{P_1^{S_R}, P_2^{S_R}\} \subset \mathcal{C}_f$ et soit P_3 le milieu de $[P_1^{S_R}, P_2^{S_R}]$, $P_3 \in \mathcal{C}_f$ à ϵ près c'est à dire $|d(P_3, \mathcal{C}_f)| \leq \epsilon$ (Régions R3 et R1 de la Figure 6.6).

Pour comprendre comment ces contraintes peuvent être appliquées, supposons que le nœud i est un bord d'un triangle. Nous créons la contrainte ternaire entre ce nœud et un nœud j définissant le "coin gauche d'un triangle" et un nœud k définissant le "coin droit d'un triangle" . La contrainte $Arc(i, j, k)$ appliquée au nœud i vérifie que le triplet $(a, b, c) \in D_i \times D_j \times D_k$, correspond à 3 points alignés.

Application de la règle 2 : Toute région R' connectée à R doit être compatible avec la forme définie par $\mathcal{C}_f$ et doit appartenir au nœud correspondant à la forme ou au nœud correspondant aux régions adjacentes à la forme. Cela signifie que, pour tout p_α et p_β points caractéristiques de R' :

— $p_\alpha \in Sh_{in} \Rightarrow p_\beta \in Sh_{in}$

— $p_\alpha \in Sh_{out} \Rightarrow p_\beta \in Sh_{out}$

Pour comprendre comment cette contrainte est appliquée dans l'exemple précédent, le nœud i étant le bord d'un triangle, dès qu'une région r_1 du nœud i satisfait la contrainte de la Règle 1 pour le triplet (r_1, r_2, r_3), on vérifie que pour toute les régions r_a connectées à r_1, on a pour tous les points caractéristiques p de r_a $d(p, \mathcal{C}_f) \leq \epsilon$ ou (exclusif) $d(p, \mathcal{C}_f) \geq 0$. Sur la Figure 6.6, la région R3 a des points caractéristiques tels que $d(p, \mathcal{C}_f) \leq \epsilon$ et d'autres tels que $d(p, \mathcal{C}_f) \geq \epsilon$, ce qui ne satisfait pas la Règle 2.

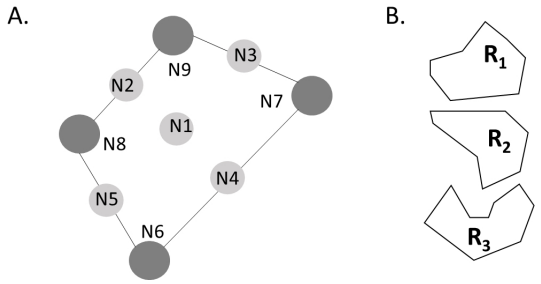


FIGURE 6.7 – A : Graphe avec les nœuds N1 à N9 représentant un trapèze, N6, N7, N8 et N9 étant les sommets du trapèze.
B : Seule la région R1 a un point saillant dans la direction nord et est compatible avec la contrainte de nœud N9.

Application de la règle 3 : Nous travaillons uniquement avec des parties de la courbe $\mathcal{C}_f$ non concaves qui ont une dérivée monotone. Dans ce but, les bords de la forme d'un objet sont divisés en différentes parties monotones. Chaque jonction entre les deux parties de courbes est associée à un nœud du graphe. Chaque partie est définie par trois nœuds : un pour chaque extrémité et l'autre pour la partie de courbe elle-même.

Voyons maintenant comment l'application de ces 3 règles permettent d'obtenir la propriété que nous avons énoncée :

Théorème 6. Soit une région R_{head} dans un nœud N_i et une région R_{tail} dans un nœud N_j. Soit une région R du nœud N_k telle que, il existe une contrainte équationnelle liée à la courbe $\mathcal{C}_f$ entre (N_i, N_j, N_k), le domaine de définition des valeurs de $\mathcal{C}_f$ étant donné par des points caractéristiques de R_{head} et R_{tail}. Si toutes les régions R satisfont les trois règles définies précédemment alors $\forall P \in \mathcal{C}_f$, $\exists R' \in E$ ($R' \in N_k$) tel que $P \in \hat{R}'$ (à un ϵ près c'est à dire que P est proche d'un pixel de $\hat{R}'$ à une distance qui n'excède pas une valeur λ très petite).

Ce théorème établit que travailler avec les points caractéristiques des régions de E pour vérifier qu'ils suivent $\mathcal{C}_f$, suffit pour vérifier que tous les pixels de bord des régions appartenant à E respectent $\mathcal{C}_f$ (c'est à dire qu'aucune région de E ne coupe la courbe) et que $\mathcal{C}_f$ est entièrement définie par des bords des régions de E, le décalage entre les pixels des bords des régions de E et les points de $\mathcal{C}_f$ ne dépassant pas une distance de λ choisie petite (2 à 3 pixels).

Preuve : Etant données les 3 règles précédentes, nous allons montrer que tous les pixels de la courbe $\mathcal{C}_f$ orientée dans une direction donnée (par exemple la direction nord, c'est à dire que les couples (x, y) de la courbe sont tels que les x sont sur l'axe est/ouest et les y ont une valeur croissante en direction du nord) appartiennent aux bords des régions segmentées de l'image. Au préalable nous allons démontrer les deux lemmes suivants :

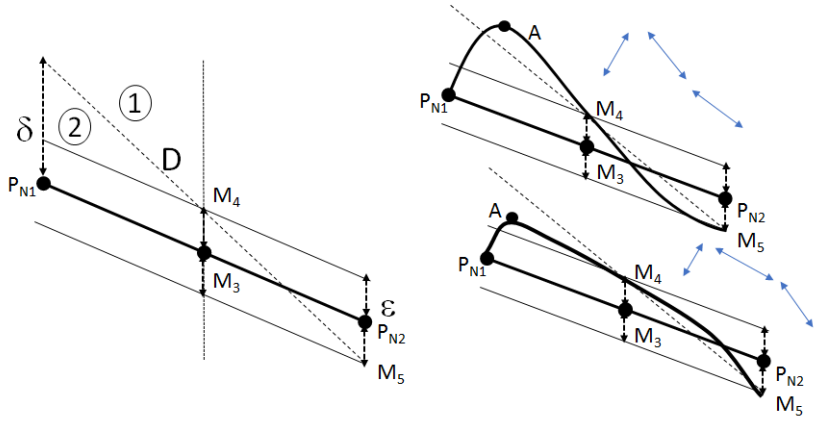


FIGURE 6.8 – A gauche : Segment formé par P_{N1} et P_{N2} et M_3 le milieu de ce segment. M_4 est le point distance de $+\epsilon$ de M_3 à la verticale de M_3. Le point M_5 est le point à distance $-\epsilon$ à la verticale de P_{N2} (c'est le point le plus éloigné pouvant appartenir à la courbe au point d'abscisse $x_{P_{N2}}$. La droite D est définie par M_5 et M_4. D'après le théorème de Thales $\delta = 3 \times \epsilon$.
A droite : La courbe du haut passe par un point A dans la zone 1 n'a pas une dérivée monotone. La courbe du bas dont aucun point n'est dans la zone 1 a une dérivée monotone.

Lemme 1 de la continuité intra-segments : $\{P_1^{S_R}, P_2^{S_R}\} \subset \mathcal{C}_f \Rightarrow \forall\ P \in \mathcal{C}_f$ entre $P_1^{S_R}$ et $P_2^{S_R}$, $P \in \hat{R}$ à un λ près. Cela veut dire que si on vérifie que le segment défini par $P_1^{S_R}$ et $P_2^{S_R}$ est sur la courbe (à un ϵ près, soit $|d(P_1^{S_R}, \mathcal{C}_f)| \leq \epsilon$ et $|d(P_2^{S_R}, \mathcal{C}_f)| \leq \epsilon$), alors tous les points de la courbe en regard du segment sont confondables avec les points du segments (à un λ près, lié à la valeur de ϵ).

Lemme 2 de la continuité inter-segments : $\forall\ R, \forall\ S_R, \{P_1^{S_R}, P_2^{S_R}\} \subset \mathcal{C}_f \Rightarrow S_R$ est connecté à des segments voisins qui définissent la courbe (les extrémités coïncident).

Si ces deux Lemmes sont vrais, comme au moins un point caractéristique de chaque région $R \in E$ est sur la courbe $\mathcal{C}_f$ à un ϵ près (imposé par la **Règle 1**) et comme une continuité de bord pour tous les points de la courbe en regard de chaque segment (Lemme 1) et comme les segments suivent de façon contigüe la courbe, c'est à dire qu'une extrémité de l'un est connexe à une extrémité du suivant (Lemme 2), alors $\forall\ P \in \mathcal{C}_f$, $\exists$ $R \in E$ tel que $P \in \hat{R}$ à un ϵ près. Autrement dit, comme les Lemmes 1 et 2 sont vrais, la courbe $\mathcal{C}_f$ est entièrement décrite par une sous-partie de l'ensemble des pixels de bord d'un ensemble de régions.

Preuve du Lemme 1 :

— Prenons un pixel P tel que $P(x, y) \in \mathcal{C}_f$ et $P(x, y) \in R$.

— Soit $P_N \in \hat{R}$ la projection de P sur le bord nord de R. $\exists S_R = \langle P_{N1}^{S_R}, P_{N2}^{S_R} \rangle$ tel que $P_N \in S_R$.

— Soit M_3 le milieu de $[P_{N1}^{S_R}, P_{N2}^{S_R}]$.

— Soit σ la tolérance acceptée par un algorithme de segmentation polygonale du bord d'une région. Cela signifie que $\forall\ p \in \langle P_{N1}^{S_R}, P_{N2}^{S_R} \rangle$, d($p$,$[P_{N1},\ P_{N2}]$)$< \sigma$.

On suppose que $P_{N1}^{S_R} \in \mathcal{C}_f$ et $P_{N2}^{S_R} \in \mathcal{C}_f$ à ϵ près : $|d(P_{N1}^{S_R}, (x_{P_{N1}^{S_R}}, f(x_{P_{N1}^{S_R}}))| \leq \epsilon$ et $|d(P_{N2}^{S_R}, (x_{P_{N2}^{S_R}}, f(x_{P_{N2}^{S_R}}))| \leq \epsilon$. Donc :

— $M_3 \in \mathcal{C}_f$ à ϵ près c'est à dire $|d(M3, (x_{M3}, f(x_{M3}))| \leq \epsilon$ (**Règle 1**).

— Comme $\mathcal{C}_f$ est non concave et que sa dérivée est monotone entre $P_{N1}^{S_R}$ et $P_{N2}^{S_R}$(**Règle 3**), $\forall P \in \mathcal{C}_f$ avec P_N entre $P_{N1}^{S_R}$ et $P_{N2}^{S_R}$, P est au dessus du segment $[P_{N1}^{S_R}, P_{N2}^{S_R}]$.

Soit un point A $\in \mathcal{C}_f$ compris entre $P_{N1}^{S_R}$ et $M3$ (On peut faire ensuite un raisonnement similaire pour A compris entre $M3$ et $P_{N2}^{S_R}$). Nous allons montrer que ce point ne peut pas être trop distant du segment défini par $P_{N1}^{S_R}$ et $P_{N2}^{S_R}$. Plaçons-nous dans le cas d'un segment descendant (un raisonnement similaire peut être fait avec un segment montant). Traçons la droite D définie par le point $M5$ qui est le point le plus bas que peut prendre $\mathcal{C}_f$ en regard de $P_{N2}^{S_R}]$ et $M4$ qui est le point le plus haut que peut prendre $\mathcal{C}_f$ en regard de $M3$. Deux cas peuvent être étudiés selon la localisation du point A :

— Si A est situé dans la zone 1 (Voir la Figure 6.8) : Au dessus de la ligne D et entre les deux lignes verticales passant respectivement par P_{N1} et par M_3. Supposons dans un premier temps que $\mathcal{C}_f$ passe par $M4$ et $M5$.

Le théorème de la valeur moyenne dit que, étant donné un arc planaire entre deux points terminaux, il existe au moins un point où la tangente à l'arc est parallèle à la sécante passant par ses points.

Donc il est possible de trouver un triplet de points de $\mathcal{C}_f$ (p, p', p''), p étant situé entre P_{N1} et A, p' entre A et M_3 et p'' entre M_3 et P_{N2}) tel que les tangentes à ces trois points soient égales aux pentes $s1$ de (P_{N1}, A), $s2$ de (A, M_4) et $s3$ de (M_4, $M5$).

Par construction, on voit bien sur la Figure 6.8 que lorsque A est dans la zone 1, (P_{N1}, A) a une pente supérieure à la pente (A, M_4) et (M_4, $M5$) et la pente (M_4, $M5$) est supérieure à (A, M_4). Donc $s1 > s3 > s2$.

Ceci rentre en contradiction avec l'hypothèse que la dérivée de $\mathcal{C}_f$ entre P_{N1} et P_{N2} est monotone. Donc le point A ne peut pas exister. Si $\mathcal{C}_f$ ne passe par $M4$ mais passe plus bas, $s2$ est encore plus fortement négatif. De même si $\mathcal{C}_f$ ne passe par $M5$ mais passe plus haut, $s3$ a une pente encore plus petite. Dans ces deux cas, A ne peut pas exister non plus.

— Si A est située dans la zone 2 (Voir la Figure 6.8) : en dessous de de la ligne D et entre les deux lignes verticales passant respectivement par P_{N1} et par M_3.

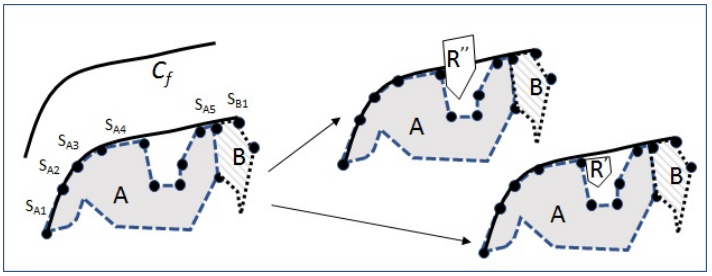


FIGURE 6.9 – En haut à gauche : Exemple de courbe $\mathcal{C}_f$) ; En bas à gauche : la région A a 5 segments qui s'ajustent avec la courbe $\mathcal{C}_f$) et la région B a un segment qui s'ajuste. Entre le segment S_{A4} et le segment S_{A5} existe-t-il une suite ininterrompue de segments qui suivent la courbe $\mathcal{C}_f$)
A droite : 2 cas de figure pour la suite de S_{A5}, soit il exite une région qui suit la courbe (en bas), soit la courbe passe à travers la région voisine et dans ce cas il n'existe pas de segment de cette région qui est connexe à S_{A5} et qui suit la courbe $\mathcal{C}_f$) (en haut). L'application de la règle 2 n'autorise pas ce dernier cas de figure

Il est possible dans ce cas d'avoir $s1 \geq s2 \geq s3$ mais A sera au maximum distant de $\delta = 3\epsilon$ du point du segment qui lui est en regard.

Comme par ailleurs chaque point du segment issu de la segmentation polygonale des bords d'une région est au maximum distant de σ du point du contour réel de la région, la distance maximale possible entre $\mathcal{C}_f$ et le point du morceau de contour de la région R qui dessine $\mathcal{C}_f$ est de $3\epsilon + \sigma$.

Donc $\mathcal{C}_f$ s'ajuste au bord des régions au pire à $3\epsilon + \sigma$ près. Si σ et ϵ sont pris suffisamment petits, et les points ne pouvant atteindre cette valeur que dans un espace restreint (zone 1 sur la figure 6.8), on peut considérer que le morceau de courbe $\mathcal{C}_f$ en regard de deux points d'un segment de bord épouse pour tous ses points une partie du bord de la région à une court écart près.

Les mêmes raisonnements peuvent être faits sur la partie droite pour les zones symétriques aux zones 1 et 2.

Preuve du Lemme 2 : Sur la Figure 6.9, le segment S_{A4} de la région A est-il connecté à droite avec un segment qui suit la courbe $\mathcal{C}_f$? En d'autres termes est-il possible que $\mathcal{C}_f$ traverse une région à une distance supérieure à un ϵ de ses bords et ne soit pas à proximité de ses bords ? Soit A une région dont un segment est sur la courbe $\mathcal{C}_f$. Si ce segment est connexe à une autre région, deux cas de figure peuvent se produire. Soit la région connexe suit le contour(cas de R' sur Figure 6.9 en bas à droite), soit elle ne le suit pas et interrompt la continuité du contour (cas de R'' sur Figure 6.9 en haut à droite). En fait la règle 2 n'autorise pas le cas R' et la continuité du suivi de $\mathcal{C}_f$ d'un segment à l'autre est assuré. En effet supposons qu'il existe $P(x,y), P(x,y) \in \mathcal{C}_f$ et $P(x,y) \in \dot{R}$, le complémentaire de $\hat{R}$ dans R (point blanc dans la Figure 6.10). Dans ce cas, la région R coupe la continuité de $\mathcal{C}_f$.

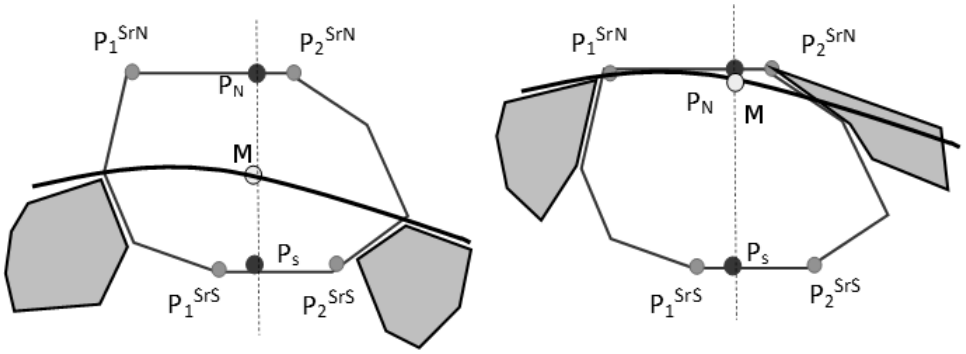


FIGURE 6.10 – Dans le dessin de gauche, on suppose que le point blanc est à l'intérieur de la région. Cependant, dans ce cas, il y a nécessairement un segment contenant la projection du point M (nord ou sud) qui est à l'extérieur de la forme délimitée par la courbe théorique et un autre à l'intérieur de la forme. Cette configuration n'est pas possible à cause de la Règle 2 qui impose que toutes les régions $R \in E$ suivent la courbe théorique (A droite de la Figure).

— Soit P_N la projection nord et P_S la projection sud de $P(x, y)$ sur la partie nord et la partie sud de $\hat{R}$ (points noirs dans la Figure 6.10).

— Soit $S_{R_N} = \langle P_1^{S_{R_N}}, P_2^{S_{R_N}} \rangle$, tel que $P_N \in S_{R_N}$, (respectivement $S_{R_S} = \langle P_1^{S_{R_S}}, P_2^{S_{R_S}} \rangle$ tel que $P_S \in S_{R_S}$.

$P \in \dot{R}$, $\Rightarrow [P_1^{S_{R_N}}, P_2^{S_{R_N}}]$ ou $[P_1^{S_{R_S}}, P_2^{S_{R_S}}]$ est nécessairement à l'extérieur de la forme délimitée par la courbe $\mathcal{C}_f$ et l'autre segment est à l'intérieur de la forme.
Cependant, comme $P \in \mathcal{C}_f$, il n'est pas possible d'avoir un segment à l'extérieur de la forme (contrainte imposée par la **Règle 2**). Donc, les extrémités des segments doivent être soit sur la courbe soit au dessous de la courbe.
Donc, le point $P \notin \dot{R}$ ce qui est le contraire de notre hypothèse initiale de continuité coupée, cette hypothèse amène donc a une contradiction et ne peut être retenue. La continuité de la courbe ne peut donc pas être coupée.

Ce théorème est important car il permet d'appliquer des contraintes très sélectives sur les formes. Toute forme dans une image qui n'est pas exactement comme la forme définie sera rejetée. Bien sûr, le créateur du graphe peut toujours adoucir ces contraintes si c'est nécessaire, mais au moins il a une nouvelle possibilité pour mieux contraindre la forme.
□

6.5 Comment construire un graphe : Application à la recherche d'un trapèze

Nous allons décrire comment représenter un trapèze avec un graphe. Cet exemple donne une idée sur la façon de représenter n'importe quelle forme dans notre cadre formel. Un trapèze est une forme moins contrainte qu'un triangle ou qu'un parallélogramme, on peut donc avoir un risque de sur-détections. Certains objets industriels ont une forme proche de celle d'un trapèze (par exemple, les ailes d'un avion) mais les forme trapézoïdales sont également rencontrées fréquemment dans la vie réelle car les formes rectangulaires ou les parallélogrammes peuvent devenir des trapèzes à cause de la perspective sur les vues 3D (bâtiments, routes, objets industriels,...).

La définition d'un trapèze est facilement traduite en un graphe de contraintes. Les pixels des segments de bord d'un trapèze doivent être compatible avec une équation de droite, deux droites et seulement deux doivent être parallèles, et les angles avec la plus grande des deux lignes parallèles doivent être aigus si on veut travailler avec des trapèzes aigus (et égaux si on souhaite travailler avec des trapèzes isocèles, ce qui est notre choix dans la suite). Les extrémités de chaque segment sont les points saillants de la forme (c.à.d. les coins) La compatibilité des points avec l'équation est vérifiée grâce à la propriété définie par le théorème précédent.
La construction du graphe de contraintes décrivant un trapèze est être faite en deux étapes, chaque étape ayant 4 sous étapes :

— Etape 1 : Création des nœuds du graphe. Les contraintes de nœud peuvent concerner beaucoup de propriétés de régions telles que leur forme, l'angle du secteur angulaire qui borne cette forme dans une direction donnée, le minimum de la taille maximum (hauteur, largeur, surface), la moyenne de leur niveau de gris, le contraste avec les voisins. Pour un graphe représentant un trapèze avec des coins aigus orientés vers le nord, on crée les 11 nœuds suivants (seuls 9 sont représentés dans la Figure 6.7) :

1. Les nœuds correspondant aux points convexes saillants du trapèze (coins) qui définissent les extrémités d'une courbe monotone (N6, N7, N8, et N9 dans la Figure 6.7). Chacun de ces nœuds contient au début toutes les régions possédant un point saillant convexe dans la bonne direction (voir la Figure 6.7).
2. Les nœuds correspondant aux bords de l'objet (segments) entre chaque point saillant (N2, N3, N4 et N5 dans la Figure 6.7).
3. Le nœud correspondant à la partie non bord de l'objet (l'intérieur).

4. Le nœud de l'objet tout entier (union des nœuds bords et des nœuds non-bords du trapèze)

— Etape 2 : Création des contraintes entre les nœuds. Trente-deux arcs sont créés appartenant à 4 types.

1. Les contraintes 4-aire (hyper-arcs) reliant les quatre nœuds des points saillant pour vérifier que les régions sont compatibles avec les contraintes équationnelles. Dans la Figure 6.7, cette contrainte 4-aire impose par exemple que :
   — Les points saillants dans la direction donnée par chaque nœud correspondants au couple de régions (r_1, r_2) classé dans (N8, N9) définissent une parallèle à la ligne définie par les points saillants d'un couple de régions (r_3, r_4) classé dans (N6, N7),
   — et les deux angles $(r_3\widehat{r_1, r_2})$ et $(r_4\widehat{r_2, r_1})$ définis par les points saillants correspondants des régions sont aigus.
2. Les contraintes binaires qui sont fonction des contraintes équationnelles, reliant le nœud du point saillant avec le nœud de bord de l'objet. Dans la Figure 6.7, toutes les régions de N2 doivent avoir leurs points caractéristiques compatibles avec l'équation de droite définie par une région de N8 et une région de N9.
3. Les contraintes binaires reliant les nœuds de points saillant avec les nœuds bord de l'objet qui s'assure que les trois règles définies dans la section 6.4 sont satisfaites. Le chemin entre les nœuds de points saillant et une région du bord n'est constitué que de régions appartenant au bord. La notion de contraintes intra-nœud définie dans la section 6.2.1 ($\mathcal{C}mp_{\mathrm{i},\alpha}$) est utilisée pour cela.
4. L'union des contraintes entre les nœuds de bord et de non bord et le nœud les réunissant.

L'extraction d'un trapèze avec un tel graphe peut être résumée de la façon suivante :

— premièrement, toutes les régions des images sont mises dans chaque nœud et les régions qui ne satisfont pas les contraintes de nœuds sont éliminées. Par exemple dans la Figure 6.11, R8 est éliminée du nœud N9 du graphe de la Figure 6.7 car elle n'a pas de coin saillant pointant dans la direction nord.

— deuxièmement, pour chaque arc, (pour plus de détails, voir le code de la Figure 6.2) si une région d'un nœud contrainte par cet arc n'a aucun support, cette région est supprimée de ce nœud. Par exemple dans la Figure 6.11 la contrainte 4-aire imposée sur les quatre coins d'un trapèze n'est pas satisfaite avec R5 dans le nœud N9 car il n'est pas possible de trouver trois autres régions qui pourront former les quatre coins d'un trapèze. R6 peut être compatible avec le nœud N7 et R7 avec le nœud N8, mais aucune région dans le nœud N6 ne peut former un trapèze avec (R5, R6, R7). Donc R5 est éliminée de N9.

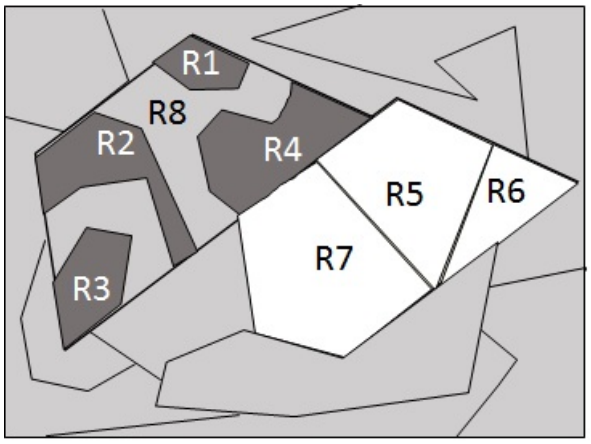


FIGURE 6.11 – Exemple illustrant l'application d'une contrainte 4-aire. La région R1 dans N9 comme le coin supérieur d'un trapèze est supportée par les régions R2, R3 et R4 dans les nœuds N8, N6 et N7. La région R5 dans le nœud N9 n'a aucune région qui la supporte (on peut trouver R6 dans N7 et R7 dans N8 mais aucune région ne peut être trouvée dans N6 pour obtenir un trapèze)

— troisièmement, la suppression de toutes les régions qui supportaient d'autres régions provoque la suppression d'autres régions dans d'autres nœuds etc ... (propagation de contraintes).

6.6 Expérimentations

Nous avons testé notre méthode sur des images synthétiques et sur des images de la vie réelle. Le premier type de tests concerne des images contenant des trapèzes et des leurres pour illustrer la robustesse et la sélectivité de la détection. Le deuxième type de tests concerne un ensemble varié de formes pour illustrer que de nombreuses formes géométriques, et même certaines formes plus complexes, peuvent être décrites et correctement détectées avec nos contraintes. Comme notre formalisme permet la description de la forme des objets, mais aussi des relations entre les objets dans une scène, ce qui est utile pour la reconnaissance de scènes, cette propriété a été illustrée par la détection de formes qui ont dans une image des relations spécifiques avec d'autres formes. Le troisième type de tests porte sur des images réelles contenant des formes non géométriques dans des environnements différents pour montrer que nos contraintes peuvent également être appliquées avec succès à ce type de forme.

6.6.1 Tests sur différents trapèzes

Le but de ces expériences est d'illustrer la capacité de notre méthode à détecter sélectivement des trapèzes ayant des tailles, des orientations et des hauteurs et/ou largeurs différentes.

Degrés de rotation	nb trapèzes dans l'image	nb trapèzes bien détectés	nb leurres	nb leurres faussement détectés
0	9	9	7	0
10,20	8	8	7	0
30,40	7	7	7	0
50,60	7	7	5	0
70,80	7	7	7	0
90	8	8	6	0
100,110, 120,130, 140,150	7	7	6	0
160-170	8	8	5	0
180	9	9	7	0

TABLE 6.1 – Table résumant les resultats obtenus sur des images tournées.

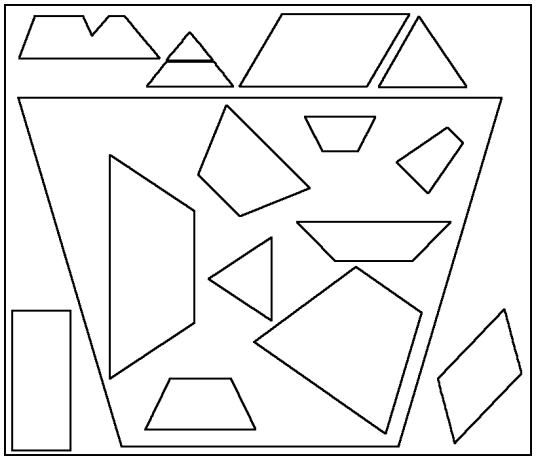
Des tests ont été effectués sur une image contenant neuf trapèzes avec différentes tailles et différentes orientations (voir Fig. 6.12). Cette image a été tournée et bruitée. Les coins des trapèzes peuvent être classés en plusieurs types : aiguë, obtus, avec un côté horizontal ou vertical, ou aucun côté de ce type. Pour réduire l'aspect combinatoire de l'algorithme, il se révèle avantageux de distinguer plusieurs types de trapèzes en fonction de leurs types de coins. Nous avons trois types de trapèzes qui couvrent toutes les orientations possibles des trapèzes (bases horizontales ou verticales, ou dans toute autre orientation). Chaque type de trapèze dispose d'un graphe correspondant (Voir un exemple dans la Fig. 6.7).

Des formes non trapézoïdales (leurres) présentant des similitudes avec des trapèzes (des triangles, des trapèzes tronqués et des parallélogrammes) ont été intentionnellement ajoutées à l'image afin de tester la spécificité de l'algorithme. Les tests ont montré qu'aucun leurre n'a été considéré comme un trapèze par l'algorithme et que les trapèzes ont tous été détectés (Voir Fig. 6.13). Nous avons également appliqué à l'image test des rotations allant 0 à 180 degrés par échelon de 10 degrés et les trapèzes sont toujours correctement détectés et aucun leurre n'est faussement détecté.

Nous avons aussi arbitrairement sur-segmenté les différentes régions de l'arrière-plan et des formes. Les mêmes graphes ont été appliquées. Tous les trapèzes continuent à être correctement détectés, et les leurres ne sont toujours pas détectés comme étant des trapèzes.

Ces différentes expériences montrent que quelles que soient les conditions (rotation, taille et bruit), tous les trapèzes sont correctement détectés de manière sélective.

a.



b.

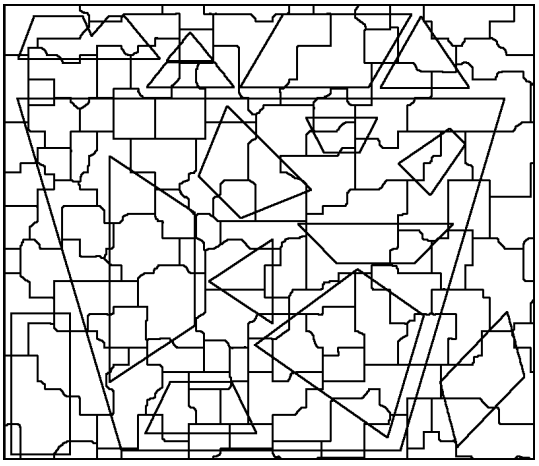


FIGURE 6.12 – a. Image de trapèzes sans bruit b.Image de trapèzes avec bruit (image sur-segmentée).

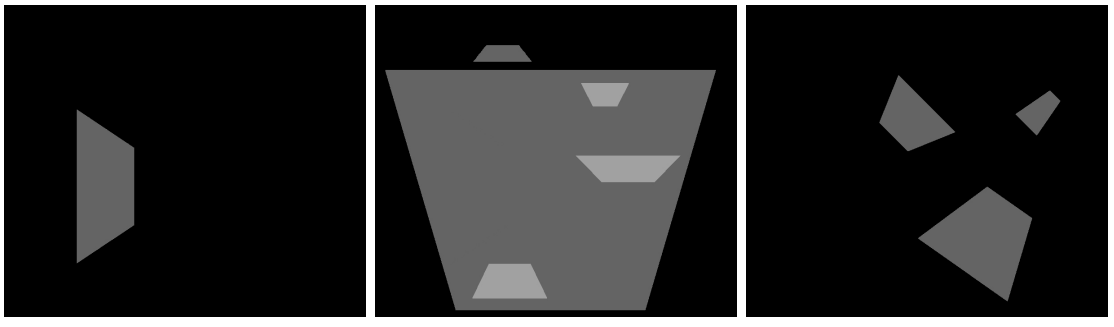


FIGURE 6.13 – Extraction des trapèzes de l'image 6.12.a. Trois graphes correspondant aux trois familles de trapèzes en fonction de leur orientation ont été utilisés. Chaque image résultat correspond au résultat d'extraction obtenu avec un seul graphe. On obtient le même résultat sur l'image bruitée et l'image non bruitée

6.6.2 Tests avec différentes formes constituées de parties décrites par une équation

Des graphes ont été construits pour modéliser des triangles, des rectangles, des parallélogrammes, des cercles, des anneaux, des losanges, des étoiles, des hexagones et des flèches. Beaucoup de types de contrainte peuvent être créés avec le formalisme que nous proposons, mais habituellement il est possible de définir précisément beaucoup de formes différentes avec seulement un petit ensemble de contraintes. Pour toutes les expérimentations et les différentes formes présentée dans ce chapitre, nous utilisons un ensemble de 11 contraintes d'arc inter-nœud :

— Trois contraintes binaires. La première vérifie si deux nœuds contiennent les mêmes régions. La seconde vérifie si les régions de deux nœuds satisfont une métrique donnée, et la troisième vérifie si les régions de deux nœuds se touchent.

— Une contrainte trinaire qui vérifie que les régions des trois nœuds satisfont les caractéristiques d'un triangle (métriques, angles)

— Deux contraintes 4-aire qui vérifient que les régions de quatre nœuds satisfont soit les caractéristiques d'un trapèze, soit les caractéristiques d'un parallélogramme (métriques, angles)

— Deux contraintes 5-aire. La première vérifie si les 5 extremums satisfont les caractéristiques d'un cœur et la seconde vérifie si les régions des 5 nœuds satisfont les caractéristiques d'une flèche.

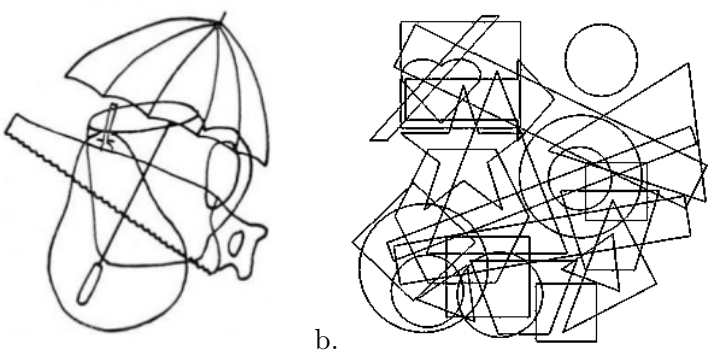
— Une contrainte 6-aire vérifie si les régions des 6 nœuds satisfont les caractéristiques d'un hexagone.

— Une contrainte 10-aire vérifie si les régions des 10 nœuds satisfont les caractéristiques d'une étoile à 5 branches.

De plus, on utilise trois contraintes intra-nœud $\mathcal{C}mp_{\mathrm{i},\alpha}$. Une pour vérifier si une ligne droite relie les régions d'un nœud d'une interface à une autre, une pour vérifier si un arc de cercle relie les régions d'un nœud d'une interface à une autre et une autre pour vérifier si une courbe convexe monotone relie deux régions d'un nœud d'une interface à une autre.
Une image contenant ces formes avec des orientations et des tailles différentes a été utilisée. Cette image "tout-en-un" (Fig. 6.14b) présente plusieurs avantages :

— Pour une forme donnée, les autres formes servent de leurre pour tester la sélectivité de la détection.

— Le chevauchement des formes crée du bruit, ce qui rend la détection des formes plus difficile. Cette difficulté est d'ailleurs utilisée en neuropsychologie, ce type d'image servant à détecter des agnosies visuelles (Fig. 6.14a).



a. b.

FIGURE 6.14 – a. Un exemple d'image utilisée en neuropsychologie pour tester l'agnosie visuelle. b. L'image test d'origine contenant des formes géométriques superposées.

Les résultats expérimentaux montrent que notre approche permet la détection correcte et sélective de toutes les différentes formes, en dépit du chevauchement de ces formes entre elles (Fig. 6.15). Pour chaque famille de formes, le graphe correspondant a permis la détection de toutes les formes sans changer les valeurs des paramètres du graphe pour tenir compte du changement de taille ou d'orientation des formes d'une même famille. Ainsi la méthode que nous proposons reconnait les formes indépendamment de leur orientation et de leur taille.
Les graphes représentant les triangles, les cercles et les losanges ont aussi été utilisés sur des images de la vie réelle présentant des formes géométriques (voir Fig. 6.16). Ils ne contenaient aucune information sur la taille, la localisation et le niveau de gris des objets dans les images. Les étiquettes initiales des nœuds des graphes sont les régions obtenues à l'aide d'un algorithme de détection des lignes de partage des eaux [77]. Là encore les formes géométriques cibles ont été détectées.

Comme notre formalisme de graphe peut également décrire des scènes (relations entre les objets), nous avons illustré cette possibilité par la modélisation des scènes simples :

— Un carré avec un triangle en dessous (relation spatiale directionnelle) (Voir Fig. 6.17a). Deux carrés de la figure 6.14b satisfont cette propriété.

— Un cercle dans un autre cercle, tous deux possédant le même point central (objets partageant une propriété de localisation relative non-directionnelle) (Voir Fig. 6.17b). Un seul cercle de la figure 6.14 satisfait cette propriété.

— Deux carrés de la même taille (objets qui partagent une propriété de taille relative) (voir Fig. 6.17c). Un couple de carrés de la figure 6.14 satisfait cette propriété

On peut noter que des relations spatiales, mais aussi des relations conceptuelles plus complexes (taille relative et position relative), peuvent être exprimées pour retrouver un objet.

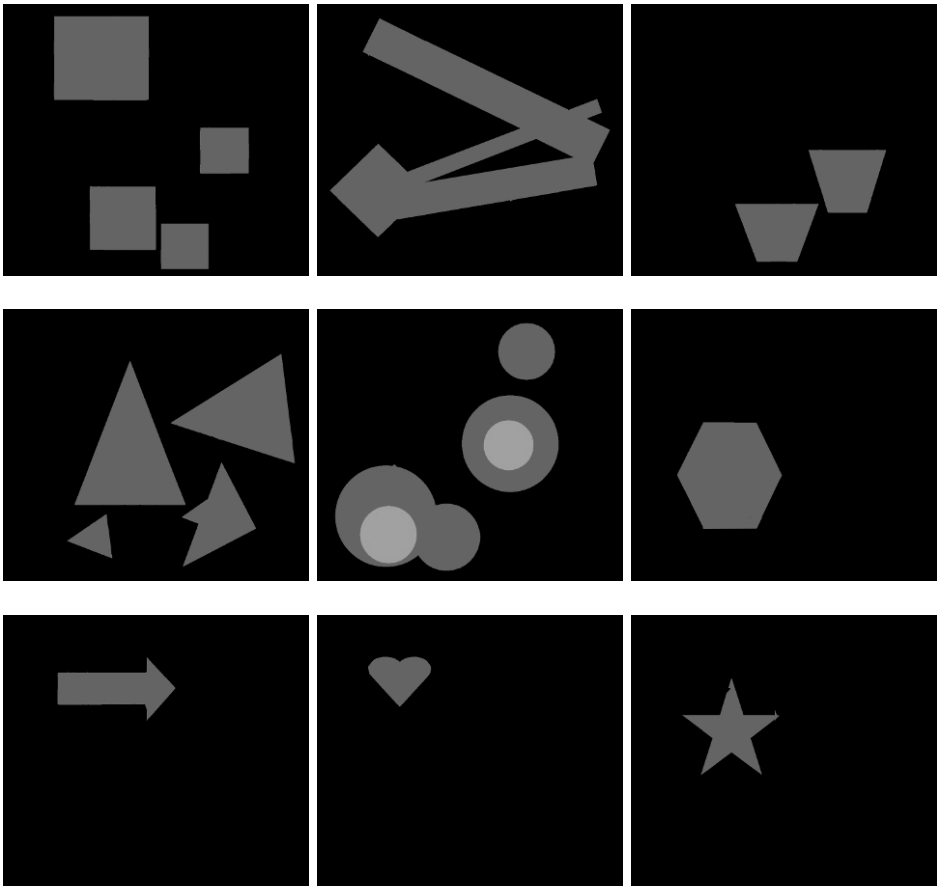


FIGURE 6.15 – Extraction des formes géométriques de la figure 6.14b. Chaque image correspond au resultat obtenu avec un graphe spécifique d'une forme donnée.

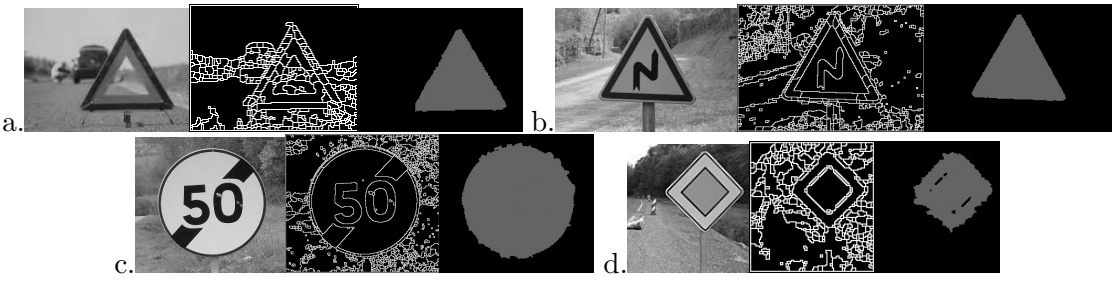


FIGURE 6.16 – Formes géométriques en environnement naturel. Pour chacun des 4 tests, présentation de l'image d'origine, de l'image segmentée, et de l'image contenant la forme reconnue. La forme superposée cache d'autres formes plus petites.

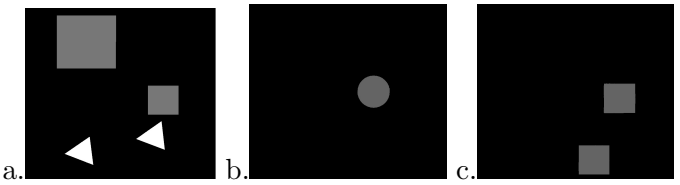


FIGURE 6.17 – Analyse de scènes de la figure 6.14.a. Carrés de l'image ayant un triangle en-dessous (les 2 carrés correspondant sont effectivement trouvés). b. Cercles de l'image qui se trouvent l'intérieur d'un autre cercle et qui partagent le même centre (le seul cas correspondant de l'image est trouvé). c. Carrés de l'image ayant la même taille (les 2 carrés de même taille sont effectivement trouvés)

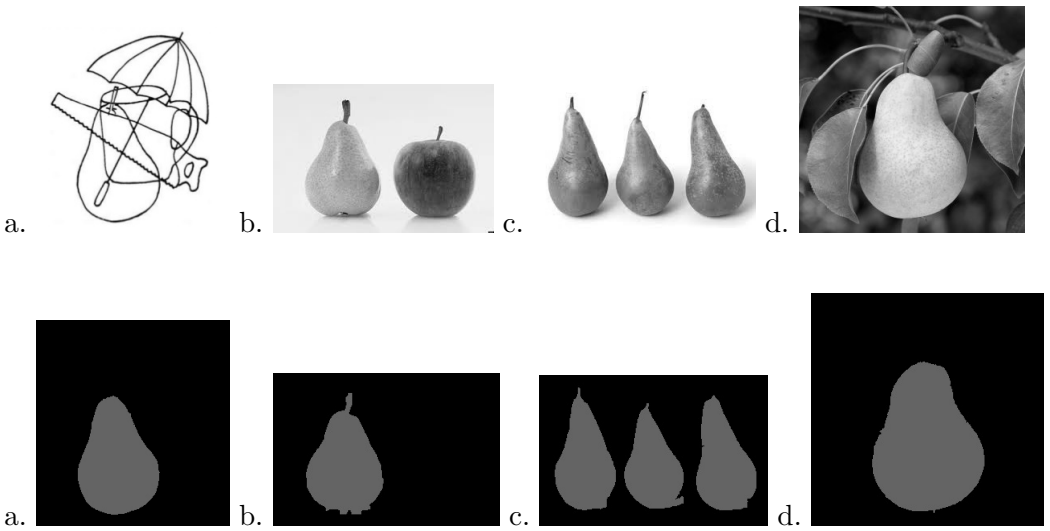


FIGURE 6.18 – En haut :Image originale de poires. En bas : Images interprétées

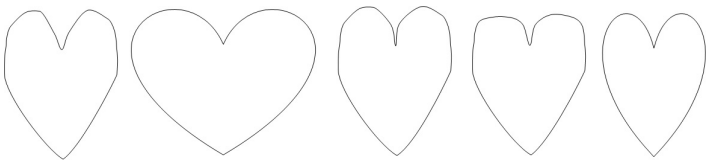


FIGURE 6.19 – Ces différentes formes de cœur ont leur partie supérieure qui ne suit pas une équation précise mais qui est la succession de 4 fonctions monotones.

6.6.3 Reconnaissance d'objets avec des parties ne pouvant pas être définies par une équation connue

Notre but était de montrer que des formes, dont les contours sont décrits entièrement par un ensemble de courbes définies par une équation, peuvent être identifiées de manière sélective, et il est souvent possible d'approximer de nombreuses formes par un ensemble de ces courbes. Toutefois, cette contrainte est parfois trop forte. Par exemple, la partie supérieure d'une forme de cœur n'est pas strictement définie par une équation et peut être dessinée avec de nombreuses variations (Voir Figure 6.19). Un moyen de contraindre une telle courbe est de veiller à ce que les parties des régions composant cette courbe définissent une chaîne de segments ayant un changement de pente monotone. Notre méthode a ainsi pu détecter les courbes formant le contour d'un cœur (voir la 8ème image de la Fig. 6.15). La forme d'une poire, entre aussi dans cette catégorie et peut être décrite par un graphe simple, car cette forme est composée de parties monotones. Nous avons testé notre approche sur des images contenant des poires pour voir si elle peut être étendue à des formes qui sont moins contraintes. La première image est issue d'un test visuel neuropsychologique (celui de la figure 6.14). Les deuxième et troisième images contiennent des fruits (poires et pommes) sur un fond uniforme et la dernière contient une poire dans un environnement naturel. Les images ont été segmentées avec

l'algorithme de détection des lignes de partage des eaux. Cette segmentation fournit l'ensemble des régions à traiter par l'algorithme HAC_{BC}. Il est important de noter que le résultat de cette étape doit être une sur-segmentation. Une sous-segmentation empêcherait la détection de certains objets. Ces images contenaient entre 40 et 575 régions et les poires étaient constituées de 11 à 210 régions. Dans chaque cas, les poires ont été correctement détectées et le graphe modèle a permis la distinction entre une pomme et une poire (Voir Fig. 6.18) et entre une poire et d'autres formes qui se chevauchent. Toutes les poires ont été détectées avec le même graphe et les mêmes valeurs de paramètres, bien que la taille des poires variait d'un facteur allant de 1 à 17. Contrairement aux formes géométriques, la sélectivité n'est pas garantie pour la reconnaissance de ces formes. Cependant, nos tests montrent que le formalisme présenté pour la description et la gestion des contraintes est assez puissant pour être utilisé avec succès dans un contexte plus large que celui des formes géométriques.

6.6.4 Détection du corps calleux dans des images obtenues par résonance magnétique

Nous avons développé un graphe sémantique décrivant les caractéristiques morphologiques du corps calleux en utilisant les principes décrits dans ce chapitre. Le corps calleux est une structure cérébrale bien visible sur des coupes de cerveau de profil. De nombreuses études médicales rapportent que certaines pathologies psychiatriques pourraient être en rapport avec de subtiles anomalies de forme de cette structure. Les tests ont été réalisés sur 40 cerveaux de la base d'images Oasis, téléchargeable sur le site http ://oasis-brains.org/. Les images ont été pré-segmentées avec un algorithme de ligne de partage des eaux (Fig 6.20). Pour construire ce graphe représentant le corps calleux, nous avons défini des nœuds correspondant aux points d'inflexion de la courbe décrivant la forme du corps calleux. Entre chacun de ces points des contraintes équationnelles ont été imposées (Fig 6.21). Sur la Fig 6.22 on peut voir comment l'analyse décompose l'objet dans les différents nœuds du graphe et le résultats final obtenu. Sur les 40 cerveaux analysés seuls les cas pathologiques n'ont pas été correctement détectés (corps calleux sectionné et donc ne suit pas la forme prédéfinie de cette structure anatomique cérébrale). La Fig 6.23 montre quelques résultats obtenus sur 7 images.

6.7 Conclusion

Nous venons de voir que les extensions proposées au cadre des PSC offrent la possibilité de contraindre plus sélectivement les régions segmentées ou les primitives extraites. On

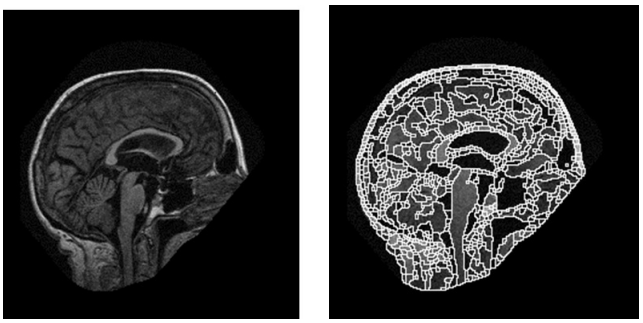


FIGURE 6.20 – A gauche : image d'origine. A droite : image segmentée par un algorithme de lignes de partage des eaux

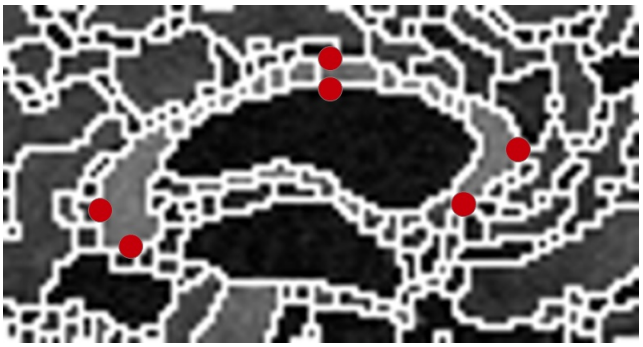


FIGURE 6.21 – Points caractéristiques identifiés dans le graphe modèle du corps calleux

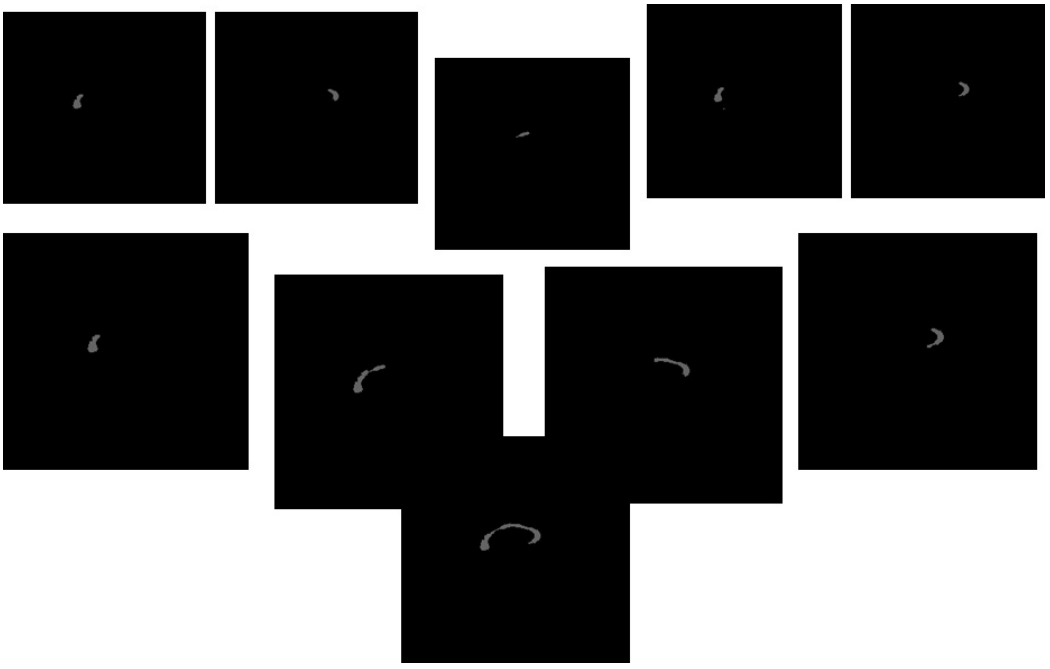


FIGURE 6.22 – En haut : décomposition associée à chaque nœud réalisée par l'analyse. En bas résultat global obtenu

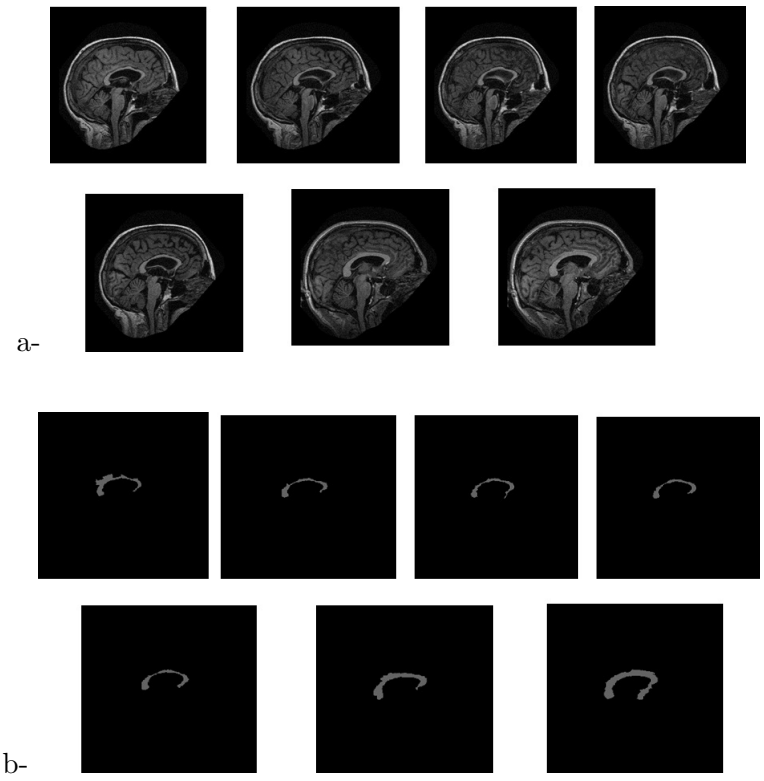


FIGURE 6.23 – a : images d'origines, b : corps calleux détectés dans les images

peut ajouter à ce qui est présenté ici, que nous avons montré au chapitre 4 la possibilité de combiner les contraintes dans des expressions logiques [78], ce qui constitue une façon supplémentaire de décrire des objets de façon plus adaptée et précise. L'avancement conceptuel proposé ici permet d'élargir l'utilisation des PSC dans le domaine de la reconnaissance d'objets. Cette approche augmente les chances de trouver des solutions partielles au problème ouvert de l'interprétation d'image. L'utilisation des PSC dans le cadre de notre formalisme pour représenter des contraintes peut être un outil utile pour l'interprétation des images et ceci pour deux raisons en particulier :

1) Le même cadre permet la reconnaissance des formes, l'analyse de scènes, mais également un raisonnement sur des problèmes sans rapport avec les images, comme la cryptoarithmétique, la planification de réunions ou la résolution de grilles de Sodoku. La possibilité d'avoir un même algorithme pour résoudre des problèmes très divers est assez attrayant, ou du moins intellectuellement satisfaisant. De plus, il est conforme à des hypothèses concernant la représentation symbolique dans le cerveau humain, qui sont soutenus par des données expérimentales [79].

2) La possibilité de travailler sur une représentation explicite (verbale) de la connaissance correspond mieux à la façon de penser de certaines personnes. Par exemple, dans les livres médicaux d'anatomie, la description des organes correspond à la description par graphe que nous utilisons.

Chapitre 7

Conclusion générale et perspectives

Le choix du formalisme des graphes pour interpréter une image nous a initialement motivé par le fait que les neuro-psychologues ont montré que certaines connaissances de type logique ou symbolique sont représentées sous la forme de réseaux de neurones. Ces réseaux sont appelés cartes [80] ou représentations [81]. Les cartes neuronales peuvent ressembler à des graphes et une hiérarchie de ce type de cartes neuronales allant du bas niveau de la vision (rétine) jusqu'au cortex cérébral existe dans le cerveau [82]. Cela semble être une idée intéressante de pouvoir interpréter une image en mimant cette architecture supposée du cerveau. Représenter de façon homogène avec des graphes les relations spatiales entre régions (bas niveau) et les relations spatiales entre les sous-parties d'un objet (haut-niveau) est une façon d'atteindre ce but. Cette raison est intéressante mais pas uniquement d'un point de vue philosophique :

— Dans les livres d'anatomie, la connaissance textuelle est caractérisée par la description de relations spatiales entre les différentes parties anatomiques du corps. Les graphes sont une façon très naturelle de retranscrire cette connaissance.

— Travailler avec un graphe sémantique a l'avantage de présenter la connaissance sous forme déclarative et de séparer la description de la connaissance de haut niveau des règles algorithmiques appliquées pour utiliser cette connaissance (contrôle de la consistance d'arcs). L'insertion de connaissances ne demande pas à un informaticien de traduire cette connaissance sous la forme de règles algorithmiques. La connaissance peut donc être introduite dans le système d'interprétation d'image sans connaissance en programmation et cela rend l'algorithme indépendant de la connaissance.

Les algorithmes que nous avons développés pour travailler avec ces graphes sont intéressants à plusieurs points de vue.

— Ils mettent en correspondance les données et la connaissance sous forme de graphe avec une complexité de temps polynomiale, ce qui est intéressant pour obtenir une analyse en temps réel.

— Ils permettent de gérer des données manquantes et des données non prévues ou des combinaisons logiques d'arcs complexes (représentant par exemple des contraintes alternatives) [83].

Enfin les extensions proposées dans le chapitre 6 permettent de reconnaître des formes géométriques décrites par des équations mathématiques ce qui fournit à l'utilisateur plus de possibilités de description des objets.

7.1 Qu'apporte notre approche ?

Dans ce travail, nous avons proposé d'utiliser le cadre théorique des $FDPSC$ à deux niveaux de contraintes pour reconnaître des scènes complexes mais également pour reconnaître des formes géométriques complexes décrites par un ensemble d'équations mathématiques, dans des images sur-segmentées. Ces fonctionnalités fournissent beaucoup plus de possibilités de décrire des objets, des formes et des relations, de façon à mieux adapter les contraintes que l'utilisateur perçoit et souhaite traduire.
Bien qu'il soit possible de construire de nombreuses contraintes spécifiques, il est intéressant de noter qu'un ensemble de 11 types de contraintes inter-nœud et trois types de contraintes intra-nœud ont été suffisants pour extraire beaucoup de formes différentes dans des environnements variés (Voir Chapitre 6).
Même si ce travail s'intéresse principalement à l'extraction de formes, le formalisme développé permet la description aussi bien d'une scène que de sous-parties d'objets de la même façon que ce qui a été décrit précédemment dans des travaux plus classiques [3, 84]. De plus, non seulement notre formalisme permet de décrire des formes plus précisément et plus sélectivement, mais il permet aussi de combiner les contraintes avec des opérateurs ET/OU [78], ce qui fournit une flexibilité supplémentaire.
Nous avons formellement prouvé que notre méthode détecte n'importe quelle forme dont les sous-parties de bord ont des dérivées monotones et peuvent toutes être décrites par des contraintes équationnelles. Ceci a été illustré sur différentes formes qui possèdent cette propriété. Les expérimentations réalisées sur des images contenant des formes géométriques de différentes tailles et de différentes orientations confirment que ce cadre théorique peut être utilisé avec succès pour détecter des formes spécifiques. Il est intéressant de remarquer que :

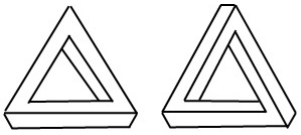


FIGURE 7.1 – Formes possible (à gauche) et impossible (à droite) qui peuvent être discriminées grâce à la vérification de la consistance d'arcs [85]

— Dans de nombreux cas, la description sous la forme d'un graphe permet de construire une représentation des formes avec une large marge de tolérance concernant l'orientation et la taille des formes.

— Le temps de calcul reste acceptable lorsqu'on applique la méthode à un nombre de régions équivalent au nombre de régions segmentées obtenu généralement sur des images de la vie réelle.

— Enfin, notre approche a été capable de détecter des formes dans des environnements bruités, et en particulier, dans le cas de formes superposées, qui est une situation parfois difficile pour le cerveau humain, notamment en cas d'agnosie visuelle. Il a pu être montré que la vérification de la consistance d'arcs et la propagation de contraintes, sont capables de distinguer des formes 3D possibles et des formes 3D impossibles parmi des dessins de formes géométriques [85] (Figure 7.1). C'est également une tâche que les humains peuvent ne pas savoir faire en cas d'agnosie visuelle [86]. La réalisation de ce type de tâches de reconnaissance sur des images d'objets impossibles ou d'objets superposés peut donc être spécifiquement perturbée lors de certaines lésions cérébrales, alors que les autres tâches de reconnaissance visuelles habituelles ne sont pas altérées. Cela suggère que ce type de tâche s'appuie sur une architecture neuronale spécifique et dédiée. Les capacités de notre algorithme à réussir des interprétations d'images dans des contextes difficiles où le cerveau humain peut être spécifiquement en échec si certaines régions cérébrales sont lésées est un résultat intéressant. Cela suggère que notre algorithme correspond à une stratégie de reconnaissance des formes qui diffère des stratégies plus usuelles employées par le cerveau, mais qui existe quand même dans le cerveau comme une stratégie moins commune, avec un rôle complémentaire.

7.2 Comparaison avec d'autres approches dans le contexte des PSC.

Les outils liés au concept de PSC ne sont pas souvent utilisés dans l'interprétation de l'image. Quand on examine les autres approches liées au PSC mais différentes de la nôtre, on constante qu'elles n'offrent pas la possibilité d'exprimer des contraintes de

façon suffisamment puissante pour traduire les contraintes complexes de la réalité visuelle de manière formelle. Ceci est dû à différents facteurs, dont le fait que si l'interprétation d'images est vue comme un problème d'appariement entre un graphe d'adjacence et un graphe sémantique, les algorithmes classiques de satisfaction de contraintes supposent la bijectivité de cet appariement. Cela rend l'interprétation des images sur-segmentées très difficile :

— Une solution pour contourner cette difficulté est de considérer que l'échec de la bijectivité de l'appariement dans le cas d'images sursegmentées est lié à une sur-contrainte. Beaucoup d'extensions des $FDPSC$ ont été proposées [87–89], pour gérer des problème sur-contraints en faisant une relaxation de contraintes (e.g. : contraintes floues, stochastiques, pondérées, probabilistes, hiérarchiques, lexicographiques, etc.). Cependant, ces extensions fournissent des solutions en termes de gestion des contraintes, mais pas en termes de conception de contraintes. Le problème est que si les contraintes ne reflètent pas avec suffisamment de précision les propriétés que les objets doivent satisfaire, peu importe la façon dont ces contraintes sont traitées, il ne sera pas possible de trouver une solution répondant aux attentes. Dans notre cas, nous devons définir des contraintes qui permettent de mieux saisir ce que nous voyons, et c'est ce que nous réalisons avec le formalisme à deux niveaux de contraintes.

— Une autre solution est possible par rapport à l'absence de bijectivité dans l'appariement des graphes de haut et de bas niveaux : Il a été développé de nombreux algorithmes d'appariement many-to-many entre graphes. Cependant, ils fonctionnent en calculant une mesure de similarité entre deux graphes qui supposent que les propriétés associées aux nœuds et aux arcs des deux graphes appartiennent aux mêmes ensembles d'étiquettes de nœuds et d'arcs [90, 91]. Dans notre cas, cela n'est pas vrai, car les propriétés associées aux nœuds et aux arcs du graphe d'adjacence sont des propriétés de bas niveau (relation simple d'adjacence, propriétés calculées sur les valeurs de niveaux de gris des pixels). Au contraire, les propriétés associées aux nœuds et aux arcs du graphe sémantique sont complexes et associées à la signification de l'objet recherché (relations spatiales complexes et propriétés morphologiques). Ces approches ne peuvent donc pas être appliquées dans ce contexte. Les exemples d'extraction de formes présentés dans ce travail ont été obtenus en n'appliquant que la vérification de la consistance d'arcs. Beaucoup de travaux sur les $FDPSC$ supposent que la consistance globale nécessite de combiner la vérification de la consistance d'arcs et le forward checking. Dans nos expériences, probablement en raison du degré élevé de contraintes des objets perçus (c'est peut être pour cette raison intrinsèque que nous les percevons), la cohérence locale est suffisante pour obtenir une cohérence globale.

7.3 Limitations possibles.

Plusieurs problèmes concernant notre approche doivent être pris en considération :

Tout d'abord, la qualité de la connaissance introduite est fondamentale pour obtenir un bon résultat. Des contraintes mal décrites (trop fortes ou trop faibles) ne donneront pas un analyse sémantique satisfaisante. Les exemples donnés dans ce travail montrent que l'ensemble des relations spatiales prédéfinies développé dans le Chapitre 4 [78] permettant de construire un graphe sémantique est capable de produire une consistance sémantique visuellement acceptable pour l'œil humain. Bien sûr, le résultat est fortement lié à la qualité de la construction du graphe sémantique et à la connaissance de l'information que peut fournir la segmentation. Le niveau de détail de la description sémantique a été choisi pour que cela ait un sens.
Par exemple, les contraintes appliquées aux structures cérébrales sont choisies pour permettre l'étiquetage des structures anatomiques les plus grosses. En revanche aucune contrainte n'a été mise pour essayer de diviser le cortex cérébral en gyri (c'est à dire qu'on n'étiquette pas séparément le gyrus temporal, le gyrus occipital, le gyrus frontal, ...), cette subdivision n'ayant pas de sens par rapport à l'information disponible. En effet, subdiviser le cortex en gyri est une tâche complexe, qui ne peut pas être faite uniquement en ajustant les paramètres d'un algorithme de segmentation. Cela est dû à ce que cette subdivision s'applique sur des régions dont les pixels contigus qui ont les mêmes niveaux de gris et il n'y a pas de moyen de diviser ces régions en fonction de valeurs de niveau de gris régionales ou locales. Rien ne permet de garantir qu'une région segmentée ne contienne que des pixels d'un gyrus unique. Cette subdivision est difficile même pour un expert, c'est un processus post-segmentation, une fois que l'ensemble du cortex a été bien isolé et qui nécessite une information 3D.
Cette limitation ne signifie pas qu'on ne puisse segmenter que de très grosses structures, cela dépend en fait de certaines caractéristiques propres de la structure. Par exemple, une petite structure anatomique telle que le septum lucidum a été détectée dans notre analyse d'images. C'est lié au fait que cette structure a des valeurs de niveau de gris différentes du fond. Cela permet de la segmenter avec un ajustement adapté des paramètres de l'algorithme de segmentation. Dans ce cas, la consistance sémantique permet justement d'optimiser ce paramétrage.

Deuxièmement, la qualité de la segmentation dépend de la méthode de segmentation choisie et de ses critères de segmentation (niveaux de gris des régions, contraste, texture locale). Mais pour une méthode donnée, notre algorithme sélectionne la valeur du critère de segmentation (seuil) qui donne le plus petit nombre de régions qui est sémantiquement consistant en fonction du graphe sémantique utilisé. Si les résultats de segmentation obtenus avec différents seuils successifs sont imbriqués dans une relation d'ordre, le résultat

final correspond à une segmentation optimale. Si une telle relation d'ordre ne peut pas être trouvée, plusieurs résultats sémantiquement consistants peuvent être obtenus et le choix de la meilleure segmentation nécessite des connaissances d'expert supplémentaires. Cela dit, notre méthode est au moins capable de fournir une segmentation acceptable en éliminant la sous-segmentation de certaines régions. Si les régions sont sous-segmentées et sémantiquement consistantes, cela suggère que le graphe sémantique n'est pas suffisamment contraint pour éviter cette situation.

Une autre limitation de notre approche est liée à la conception d'un graphe pour des objets complexes. Par exemple, dans la Figure 6.14a, le graphe d'un outil comme une scie égoïne peut être difficile à créer. Bien sûr, le graphe de la scie présente dans la figure peut être construit et permettra la détection de cette scie ou d'autres scies qui lui ressemblent de très près. Cependant, la catégorie de "scie égoïne" n'est pas facilement définie par le type de graphe que nous avons développé, car il existe trop de variantes de formes dans la même catégorie, non seulement des changements de rotation ou de taille, ou des déformations locales comme celles qui sont possibles sur les poires ou sur le trou d'une poignée de scie, mais aussi de nombreuses autres variantes. Malgré cette limitation, de nombreux domaines d'application de la reconnaissance d'image comportent un grand nombre de catégories de formes assez simples à décrire par un graphe en utilisant le formalisme proposé. Par exemple, dans le domaine de la médecine, l'anatomie ne varie pas considérablement et la forme d'un organe donné reste à peu près la même.

Les difficultés à de détecter des points saillants sont une autre limitation de notre approche. Dans la vie réelle, des points saillants marqués peuvent être émoussés en raison de l'approximation de la segmentation. Ainsi, si les contraintes de saillance appliquées à ces points sont abaissées (par exemple, en permettant aux coins supposés saillants d'être des coins un peu arrondis), il risque d'y avoir trop de régions candidates dans les nœuds correspondants, ce qui peut augmenter considérablement le temps de calcul. Cette limite liée à l'accroissement du temps de calcul n'est cependant pas rédhibitoire en pratique : Dans toutes les expériences présentées dans le chapitre six, aucune contrainte n'a été appliquée aux valeurs de niveaux de gris, à la taille ou à la localisation des régions, et les contraintes de métrique étaient faibles. Le fait d'avoir des contraintes lâches et de travailler sur des images avec plusieurs centaines de régions n'a pas empêché d'avoir de bons résultats en des temps raisonnables. Ces résultats confirment l'idée que les contraintes sur les formes et les relations spatiales peuvent suffire à reconnaître un objet dans de nombreux cas. Dans les images avec un nombre beaucoup plus élevé de régions, les contraintes sur les niveaux de gris, la taille ou la localisation des régions peuvent souvent être appliquées pour réduire l'espace de recherche et donc le temps de calcul.

L'occlusion partielle d'un objet est une limitation fréquente pour de nombreux algorithmes d'interprétation d'images et constitue également une limitation dans notre cas. Dans cette situation, tout graphe qui définit précisément les contraintes sur un objet nécessite que ces contraintes soient affaiblies. Dans le chapitre 3 [83] nous avons montré que le formalisme des graphes que nous proposons permet d'affaiblir les contraintes et d'éviter l'échec de la reconnaissance des objets dans de tels cas. Cependant, cet affaiblissement est effectué au prix de la sélectivité et de nombreuses régions peuvent être étiquetées faussement comme appartenant à l'objet cible.

Enfin une dernière limitation est liée à la possibilité que l'algorithme de segmentation initiale puisse produire des régions sous-segmentées. L'extraction de formes peut alors échouer, mais il s'agit d'une limitation générale pour toutes les méthodes d'interprétation d'images appliquées aux images préalablement segmentées. Pour cette raison, les images de la vie réelle testées ont été segmentées avec un algorithme de détection des lignes de partage des eaux qui est une technique très populaire. Cet algorithme a l'avantage de produire des régions hautement segmentées tout en créant rarement des régions sous-segmentées, si le paramètre de l'algorithme définissant la hauteur minimale des bassins versants (hminima) est très proche de sa valeur minimale. L'inconvénient de ce choix est le nombre très important de très petites régions, mais cet inconvénient apparent n'a pas été un vrai problème lorsque notre méthode a été appliquée aux images de test présentées.

7.4 Perspectives

Dans ce travail, nous avons présenté un moyen de traduire le raisonnement symbolique en ajoutant une représentation de la connaissance sous forme de graphe sémantique qui peut être utilisée par un ordinateur.

La construction du graphe sémantique est une étape très importante. Une perspective serait de la faire grâce à une interface graphique, par un expert qui n'a pas de connaissance en informatique et pour un ensemble d'image ayant le même type de contenu. Cela suppose des connaissances a priori qui sont expert dépendantes. Une amélioration serait d'obtenir cela de façon indépendante. Pour atteindre ce but, d'autres équipes [53] ont essayé d'extraire un modèle générique à partir d'un ensemble d'images, ce qui peut libérer la segmentation d'une interaction humaine.

L'utilisation du contrôle sémantique pour piloter un processus de segmentation à l'intérieur d'une pyramide de graphes d'adjacences, abordée au chapitre 5 pourrait être affiné. En fonction de la pertinence sémantique de la fusion, il peut être possible d'autoriser la

fusion de régions classées dans un nœud du graphe sémantique et de geler la fusion d'autres régions classifiées dans d'autres nœuds différents. Si l'on suit cette idée, l'analyse sémantique peut ne pas être utilisée uniquement pour choisir une segmentation mais également pour améliorer la segmentation elle-même en fusionnant des régions qui sont en correspondance avec le même nœud sémantique et qui ne sont pas en correspondance avec d'autres nœuds. Ceci ne peut pas être fait lorsque le nombre de régions est trop grand, car la probabilité de mise en correspondance avec uniquement un nœud est faible. Mais lorsqu'un niveau de segmentation adapté est atteint, cette mise en correspondance many-to-one peut être trouvée pour certains nœuds.

On peut soutenir que d'autres représentations informatiques que celle de graphe sémantique pourraient exister pour traduire un raisonnement symbolique, comme par exemple des représentations neuronales. A l'appui de cette idée, les études sur la cognition des animaux ont montré que les abeilles possèdent certaines capacités d'apprentissage conceptuel et peuvent maîtriser plusieurs concepts[92]. Même si ces insectes communiquent avec un type de langage, leurs capacités de gestion des concepts ne dépendent probablement pas de la combinaison d'éléments de langage. Les mécanismes neuronaux impliqués dans ces capacités de gestion conceptuelle commencent à être révélés [93]. Par conséquent, le raisonnement symbolique peut avoir une traduction neuronale, qui ne suppose pas la manipulation de symboles comme le ferait un mathématicien avec des notations sur papier. Certains auteurs ont suggéré que le raisonnement symbolique est un type spécial de raisonnement incorporé dans lequel les formules arithmétiques et logiques, représentées extérieurement comme des notations, servent de cibles pour des systèmes perceptuels et sensoriels puissants [94]. Cependant, jusqu'à présent, aucune traduction informatique de ces idées n'a été réalisée pour interpréter des images. Tant que notre connaissance sur ces représentations neuronales n'a pas suffisamment progressé, notre approche PSC avec notre formalisme de contraintes possède beaucoup d'avantages pour résoudre des problèmes d'interprétation d'images nécessitant un raisonnement symbolique. En outre, rien n'empêche a priori d'harmoniser étroitement certaines parties de notre représentation avec une représentation neuronale.

Bibliographie

[1] R. A. Brooks. Elephants don't play chess. *Robotics and Autonomous Systems*, 6 : 3–15, 1990.

[2] I. Goodfellow, Y. Bengio, and A. Courville. Deep learning. 2016.

[3] A. Andreopoulos and J. K. Tsotsos. 50 years of object recognition : Direction forward. *Computer Vision and Image Understanding*, 117 :827–891, 2013.

[4] A.Rosenfeld, R. Hummel, and S. Zucker. Scene labeling by relaxation operations. *IEE trans. Systems, Man., Cybernet.*, 6 :420–433, 1976.

[5] A. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8 : 99–118, 1977.

[6] J. Mulder, A. Mackworth, and W. Havens. knowledge structuring and constraint satisfaction : mapsee approach. *IEEE Transaction on P.A.M.I.*, 10 :866–879, 1988.

[7] R. Mohr and G. Masini. Good old discrete relaxation. *Proceedings ECAI-88*, pages 651–656, 1988.

[8] P. Van Hentenryck, Y. Deville, and C.M. Teng. A generic arc-consistency algorithm and its specializations. *Artificial Intelligence*, 57(2) :291–321, 1992.

[9] E.C. Freuder and R.J. Wallace. Partial constraint satisfaction. *Artificial Intelligence*, 58 :21–70, 1992.

[10] J. Benmouffek, Y. Belaid, A. Belaid, and L.D. Minacelli. Rer : un systéme de reconnaissance d'empreintes de rats. *In proceedings of 8iéme congrés AFCET : Reconnaisance des formes et intelligence artificielle*, 1991.

[11] J.V. Mahonney and M.P.J. Fromherz. Interpreting sloppy stick figures by graph rectification and constraint-based matching. *LNCS*, 2390 :222–235, 2002.

[12] O. Nempont, J. Atif, and E. Angeliniand I. Bloch. Structure segmentation and recognition in images guided by structural constraint propagation. *Proceeding of*

the 2008 conference on ECAI 2008 : 18th European Conference on Artificial Intelligence, pages 621–625, 2008.

[13] K. Rothaus, X. Jiang, and P. Rhiem :. Separation of the retinal vascular graph in arteries and veins based upon structural knowledge. *Image Vision Comput.*, 27(7) : 864–875, 2009.

[14] S. Faisan, N. Passat, V. Noblet, R. Chabrier, and C. Meyer. Topology preserving warping of binary images. application to atlas-based skull segmentation. *Medical Image Computing and Computer Assisted Intervention,LNCS*, 5241 :211–218, 2008.

[15] C. Hudelot, J. Atif, and I. Bloch. Ontologies de relations spatiales floues pour l'interprétation d'images. *Rencontres francophones sur la logique floue et ses applications, LFA 2006, Toulouse, France*, pages 363–370, 2006.

[16] O. Lezoray and L. Grady. *Image Processing and Analysis with Graphs : Theory and Practice*. CRC Press, 2012.

[17] M. Vento. A long trip in the charming world of graphs for pattern recognition. *Pattern Recognition*, 48(2) :291–301, 2015.

[18] M. Carcassoni and E.R. Hancock. Weighted graph-matching using modal clusters. In W. Skarbek, editor, *Computer Analysis of Images and Patterns. CAIP 2001*, pages 142–151. Springer.

[19] I. Charon, A. Germa, and O. Hudry. Methodes d'optimisation combinatoire. *Collection Pedagogique de Telecommunication*, 1996.

[20] M. Gondran and M. Minoux. Graphes et algorithmes. *Collection de la Direction des Etudes et Recherches d'Electricité de France*, 1979.

[21] W. Kropatsch and H. Bishof. Digital image analysis. 2001.

[22] A. Perchant. Morphisme de graphes d'attributs flous pour la reconnaissan ce structurelle de scènes. *PhD thesis, Ecole Nationale Supérieure des Télécommunications*, 2000.

[23] L. Miclet. Méthodes structurelles pour la reconnaissance des formes. *CNET, Collection technique et scientifique des télécommunications*, 1984.

[24] D. Ghahraman, A. Wong, and T. Au. Graph monomorphisms algorithms. *IEEE Transactions on Systems, man, and cybernetics*, 10(4) :189–196, 1980.

[25] E. K. Wong. Model matching in robot vision by subgraph isomorphism. *Pattern Recognition*, 25(3) :287–303, 1992.

[26] L. Shapiro and R. Haralick. Structural description and inexact matching. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 3(5) :504–519, 1981.

[27] O. Faugeras and K. Price. Semantic description of aerial images using stochastic labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3(6) : 633–642, 1981.

[28] W. Tsai and K. Fu. Subgraph error-correcting isomorphisms for syntactic pattern recog- nition. *IEEE transactions on Systems, Man and Cybernetics*, 13(1) :48–60, 1983.

[29] A. Sanfeliu and K. Fu. A distance measure between attributed relational graphs for pattern recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, 13 (3) :353–362, 1983.

[30] A. Wong and M. You. Entropy and distance of random graphs with application to structu- ral pattern recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(5) :599–609, 1985.

[31] M. A. Eshera and K. Fu. An image understanding system using attributed symbolic representation and inexact graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(5) :604–618, 1986.

[32] R. Horaud and T. Skordas. Stereo correspondance through feature grouping and maximal cliques. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(11) :1168–1180, 1989.

[33] J. Ullmann. An algorithm for subgraph isomorphism. *Journal of the ACM*, 23(1) : 31–42, 1976.

[34] H. Bunke and B. Messmer. Recent advances in graph matching. *International Journal of Pattern recognition and Artificial Intelligence*, 11(1) :169–203, 1997.

[35] B. Messmer. Efficient graph matching algorithms for preprocessed model g raphs. 1996.

[36] F. Fuchs. Contribution à la reconstruction du bâti en milieu urbain à l'aide d'images aériennes stéréoscopiques à grande échelle. etude d'une approche structurelle. *PhD thesis, Université René Descartes - Paris V, 2001.*

[37] R. Wilson and A. Cross abd E. Hancock. Structural matching with active triangulation. *Computer Vision and Image Understanding*, page 21–38, 1998.

[38] A. Cross and E. Hancock. Graph matching with a dual step em algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11) :1236–1253, 1998.

[39] W. J. Christmas, J. Kittler, and M. Petrou. Structural matching in computer vision using probabilistic relaxation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8) :749–764, 1995.

[40] E. Bengoetxea. Mise en correspondance inexacte de graphes par algorithmes d'estimation des distributions. *PhD thesis, Ecole Nationale Supérieure des Télécommunications*, 2002.

[41] A. Mackworth and E. Freuder. The complexity of some polynomial network consistency algorithms for constraint satisfaction problems. *Artificial Intelligence*, 25 : 65–74, 1985.

[42] R. Mohr and T. Henderson. Arc and path consistency revisited. *Artificial Intelligence*, 28 :225–233, 1986.

[43] C. Bessière. Arc-consistency and arc-consistency again. *Artificial intelligence*, 65 : 179–190, 1994.

[44] A. Deruyver and Y. Hodé. Semantic graph and arc-consistency in true three dimensional image labeling. *in proceedings of IEEE ICIP95*, pages 619–622, 1995.

[45] A. Deruyver and Y. Hodé. Constraint satisfaction problem with bilevel constraint : application to interpretation of over segmented images. *Artificial Intelligence*, 93 : 321–335, 1997.

[46] C. Bessière. Arc-consistency and arc-consistency again. *Artificial intelligence*, 65 : 179–190, 1994.

[47] G. Romanes. Cunnungham's manual of practical anatomy : Volume iii : Head, neck and brain. 1986.

[48] H. Rouvière and A. Delmas. Anatomie humaine. descriptive, topographique et fonctionnelle. système nerveux central, voie et centres nerveux. 2002.

[49] P. Bertolino and A. Montanvert. Multiresolution segmentation using the irregular pyramid. *Proceedings of IEEE ICIP96*, page 357–360, 1996.

[50] C. Hudelot, J. Atif, and I. Bloch. Fuzzy spatial relation ontology for image interpretation. *Fuzzy Sets and Systems*, 159 :1929–1951, 2008.

[51] D. Conte, P. Foggia, C. Sansone, and M. Vento. Thirty years of graph matching in pattern recognition. *International Journal Pattern Recognition and Artificial Intelligence*, 18(3) :265–298, 2004.

[52] J.M. Jolion. Stochastic pyramid revisited. *Pattern recognition Letter*, 24 :1035–1042, 2003.

[53] Y. Keselmann and S. Dickinson. Generic model abstraction from examples. *IEEE Transaction on PAMI*, 27(7) :1141–1156, 2005.

[54] E. Laemmer, A. Deruyver, and A. Sowinska. Watershed and adaptive pyramid for determining the apples maturity state. *Proceeding IEEE ICIP 2002, Rochester USA*, page 789–792, 2002.

[55] I. Bloch. Fuzzy spatial relationships for image processing and interpretation : a review. *Image and Vision Computing*, 23(2) :89–110, 2005.

[56] A.G. Cohn, B. Bennett, and J. Goodayand N. Gotts. Representing and reasoning with qualitative spatial relations about regions. *Spatial and Temporal Reasoning*, page 97–134, 1997.

[57] S. Skiadopoulos and M. Koubarakis. Composing cardinal direction relations. *Artificial Intelligence*, 152 :143–171, 2004.

[58] S. Hai-Bin and L. Wen-Hui. Qualitative spatial relationships cleaning for spatial data mining. *Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, Guangzhou*, page 1851–1857, 2005.

[59] F. Bookstein. Morphometric tools for landmark data : Geometry and biology. 1991.

[60] C. Bauckage, E. Braun, and G. Sagerer. From image features to symbols and vice versa – using graphs to loop data- and model-driven processing in visual assembly recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 18(3) :497–517, 2004.

[61] A. Deruyver and Y. Hodé. Image interpretation with a semantic graph : labeling over-segmented images and detection of unexpected objects. *Proceedings GBR 2001 Ischia*, page 137–148, 2001.

[62] A. Deruyver, Y. Hodé, and J. Jolion. Pyramides adaptatives et graphes sémantiques : segmentation dirigé par la connaissance. *Proceedings of RFIA Conference 2006 (CD)*, 2006.

[63] R. Tadeusiewicz and M. R. Ogiela. Medical image understanding technology.

[64] D. Crevier and R. Lepage. Knowledge-based image understanding systems : A survey. *Computer Vision and Image Understanding*, 67(2) :161–185, 1997.

[65] J. Llados and E. Marti. Structural recognition of hand drawn floor plans. *VI National Symposium on Pattern Recognition and Image Analysis, Cordoba.*

[66] J. M. Jolion and A. Montanvert. The adaptive pyramid : A framework for 2d image analysis. *CVGIP : Image Understanding*, 55(3) :339–348, 1992.

[67] S. Lallich, F. Muhlenbachand, and J.M. Jolion. A test to control a region growing process within a hierarchical graph. *Pattern recognition Letter*, 36 (10) :2201–2211, 2003.

[68] Y. Hazxhimusa and W. Kropatsch. Hierarchy of partitions with dual graph contraction. *LNCS*, 2781 :1611–3349, 2003.

[69] M. Melki and J. Jolion. Building symbolic hierarchical graphs for feature extraction. *LNCS, Proceedings of Graph Based Representation in Pattern Recognition*, 2726 : 277–282, 2003.

[70] R. Glantz, M. Pelillo, and W. G. Kropatsch. Matching segmentation hierarchies. *International Journal of Pattern Recognition and Artificial Intelligence*, 18(3) :397–424, 2004.

[71] A. Petrovic, O. Divorra Escoda, and P. Vandergheynst. Multiresolution segmentation of natural images : From linear to nonlinear scale-space representations. *IEEE Trans. on Image Processing*, 13(8) :1104–1114, 2004.

[72] I. Vanhamel, I. Pratikakis, and H. Sahli. Multiscale gradient watersheds of color images. *IEEE Trans. on Image Processing*, 12(6) :617–626, 2003.

[73] O. Lezoray, C. Meurie, P. Belhomme, and A. Elmoataz. Multi-scale image segmentation in a hierarchy of partitions. *Proceedings of 14th Eusipco 2006, 4-8 September 2006*, 2006.

[74] P. Meer. Stochastic image pyramids. *CVGIP*, 45 :269–294, 1989.

[75] N. Zlatoff, B. Tellez, and A. Baskurt. Image understanding and scene models : A generic framework integrating domain knowledge and gestalt theory. *Proceedings of International Conference on Image Processing 2004*, pages 2355–2358, 2004.

[76] M. Burge and W. G. Kropatsch. A minimal line property preserving representation of line images. *Computing*, 62 :355–368, 1999.

[77] S. Beucher and F. Meyer. The morphological approach to segmentation : the watershed transform. *in : E.R. Dougherty (Ed.), Mathematical Morphology in Image Processing, Marcel Dekker, New York*, pages 433–481, 1993.

[78] A. Deruyver and Y. Hodé. Qualitative spatial relationships for image interpretation by using a conceptual graph. *Image and Vision Computing*, 27 :876–886, 2009.

[79] D. Landy and RL. Goldtone. Formal notations are diagrams : evidence from a production task. *Mem Cognit.*, 35(8) :2033–2040, 2007.

[80] A. R. Damasi. *The Feeling of What Happens, Body and Emotion in the Making of Conciousness*. Hartcourt Brace and Co., 2004.

[81] J. P. Changeux. *L'homme de vérité*. Odile Jacob, 2002.

[82] R. C. Reid. Vision, fundamental neuroscience. pages 821–851.

[83] A. Deruyver, Y. Hodé, and L. Brun. Image interpretation with a conceptual graph : labeling over-segmented images and detection of unexpected objects. *Artificial Intelligence*, 173 :1245–1265, 2009.

[84] R. A. Brooks. Symbolic reasoning among 3-d models and 2-d images. *Artificial Intelligence Journal*, 17(1-3) :285–348, 1981.

[85] D. Waltz. Understanding line drawing of scenes with shadows. *Psychology of computer vision*, pages 19–91, 1975.

[86] Martha J. Farah. Visual agnosia. *MIT Press*, 1990.

[87] P. Meseguer, N. Bouhmala, T. Bouzoubaa, M. Irgens, and M. Sánchez. Current approaches for solving over-constrained problems. *Constraints*, 8(1) :9–39, 2003.

[88] A. Lallouet, J. H. M. Lee, T. W. K. Mak, and J. Yip. Ultra-weak solutions and consistency enforcement in minimax weighted constraint satisfaction. *Constraints*, 20(2) :109–154, 2015.

[89] C. Carbonnel and M. C. Cooper. Tractability in constraint satisfaction problems : a survey. *Constraints*, DOI 10.1007/s10601-015-9198-6 :1–30, 2015.

[90] P. Champlin and C. Solnon. Measuring the similarity of labeled graphs. *5th International Conference on Case-Based Reasoning (5ICCBR), LNCS*, 2689 :80–95, 2003.

[91] A. V. Graciano, R. M. Cesar, Jr., and I. Bloch. Inexact graph matching for facial feature segmentation and recognition in video sequences : Results on face tracking. *CIARP 2003, LNCS*, 2905 :71–78, 2003.

[92] A. Avarguès-Weber and M. Giurfa. Conceptual learning by miniature brains. *Proc Biol Sci.*, 280(1772) :20131907, 2013.

[93] J.M. Devaud, T. Papouin, J. Carcaud, J.C. Sandoz, B. Grünewald, and M. Giurfa. Neural substrate for higher order learning in an insect : Mushroom bodies are necessary for configural discrimination. *Proc Natl Acad Sci U S A.*, 112(43) :E5854–62, 2015.

[94] D. Landy, C. Allen, and C. Zednik. A perceptual account of symbolic reasoning. *Front Psychol. 5 :275. doi : 10.3389/fpsyg.2014.00275. eCollection 2014*, 2014.

Yann HODE

Contribution à l'interprétation d'images et vérification de la consistance d'un graphe

Résumé

Dans cette thèse nous montrons que le raisonnement symbolique associé à la vérification de la consistance d'arc avec propagation de contraintes est un outil efficace pour interpréter les images.

Nous montrons dans un premier temps que ce cadre théorique permet de vérifier l'organisation spatiale de différentes composantes d'un objet complexe dans une image.

Nous proposons ensuite d'étendre l'utilisation de celui-ci à la reconnaissance sélective des formes décrites par des équations mathématiques, grâce à la notion de consistance d'hyper-arc à deux niveaux de contraintes.

La pertinence et la faisabilité de cette approche ont été validées par de multiples tests. En outre, les résultats obtenus sur des images sur-segmentées montrent que la méthode proposée est résistante au bruit, même dans des conditions où les humains (dans certains cas d'agnosie visuelle) peuvent échouer.

Ces résultats soutiennent l'intérêt du raisonnement symbolique dans la compréhension de l'image

Mots clés :

Intelligence artificielle, graphe, propagation de contraintes, vision, reconnaissance de forme, CSP

Résumé en anglais

In this thesis we show that symbolic reasoning associated with arc consistency checking is an efficient tool for images interpretation. We first show that this theoretical framework makes it possible to verify the spatial organization of different components of a complex object in an image.

We then propose to extend the use of this framework to the selective recognition of shapes described by mathematical equations, thanks to the notion of hyper-arc consistency with bi-levels constraint.

The relevance and feasibility of this approach have been validated by multiple tests. In addition, the results obtained on over-segmented images show that the proposed method is noise-resistant, even under conditions where humans (in some cases visual agnosia) may fail.

These results support the interest of symbolic reasoning in image understanding.

Keywords:

Artificial intelligence, graph, constraint propagation, vision, pattern recognition, CSP