

Pour une ingénierie des connaissances pour le développement d'applications de traitement d'images

Régis Clouard

▶ **To cite this version:**

Régis Clouard. Pour une ingénierie des connaissances pour le développement d'applications de traitement d'images. Traitement des images [eess.IV]. Université de Caen, 2009. tel-01071910

HAL Id: tel-01071910
https://theses.hal.science/tel-01071910

Submitted on 7 Oct 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Université de Caen
Basse-Normandie

UNIVERSITÉ de CAEN/BASSE-NORMANDIE

U.F.R Sciences

Mémoire d'habilitation à diriger des recherches

présenté par

Régis CLOUARD

POUR UNE INGÉNIERIE DES CONNAISSANCES POUR LE DÉVELOPPEMENT D'APPLICATIONS DE TRAITEMENT D'IMAGES

Soutenu le 16 décembre 2009 devant la commission d'examen composée de:

Rapporteurs :	Mme Monique THONNAT	Directeur de recherche INRIA, Sophia-Antipolis
	Mme Marinette REVENU	Professeur des universités, ENSICAEN, Caen
	M Patrice DALLE	Professeur des universités, Paul Sabatier, Toulouse
Examinateurs :	Mme Catherine GARBAY	Directeur de recherche CNRS, LIG, Grenoble
	M Rémy MULLOT	Professeur des universités, La Rochelle

Travail réalisé au GREYC, UMR 6072 CNRS - Université de Caen - ENSICAEN

Remerciements

Qui peut rêver d'un jury plus prestigieux pour juger un travail portant sur la conception de systèmes de traitement d'images avec une approche basée sur les connaissances. Les travaux de recherche de chacun des membres de ce jury ont, d'une manière ou d'une autre, fortement influencé mes travaux de recherche. Je suis particulièrement honoré de leur participation.

Je remercie tous les membres de ce jury d'avoir accepté de juger mon travail. En particulier,

- *je suis reconnaissant à Monique Thonnat, directeur de recherche de l'INRIA, de m'avoir fortement motivé pour soutenir cette habilitation et d'avoir accepté d'en être un des rapporteurs. Les travaux de son équipe ont toujours été pour moi une source d'inspiration, et lors de nos rencontres, nos discussions ont toujours été enrichissantes et motivantes pour la poursuite de mes travaux. Je la remercie de m'avoir impliqué dans le comité de programme de la conférence internationale ICVS (ICV08 et IVS09) et d'un numéro spécial de la revue* Computer Vision and Image Understanding.
- *je remercie Patrice Dalle, professeur à l'IRIT, d'avoir accepté de rapporter sur mon habilitation. Patrice Dalle a toujours témoigné d'un grand intérêt pour mes recherches et m'a prodigué des conseils positifs et utiles pour les mener à bien. Je regrette que les travaux de son équipe sur le thème de conception de système de traitement d'images, pourtant très originaux, soient en suspens à ce jour, faute de candidat doctorant.*
- *je remercie Catherine Garbay, directeur de recherches au CNRS, qui a bien voulu participer à ce jury après avoir déjà accepté de rapporter ma thèse de doctorat. Ses travaux basés sur l'intelligence distribuée prêchent aussi pour une approche par les connaissances de la conception de systèmes de vision.*
- *je remercie Rémy Mullot, avec qui je participe à l'animation du groupe de travail SCATI (en trio avec Didier Coquin), de sa présence dans ce jury. Rémy Mullot m'a initié à l'organisation de la recherche, notamment dans l'animation d'un groupe de travail national et la publication de numéros spéciaux de revue.*
- *enfin, je suis redevable à Marinette Revenu d'avoir toujours encouragé, soutenu et même promotionné mes travaux de recherche depuis mes travaux de thèse. Son soutien a toujours été sans faille.*

J'ai aussi une pensée pour Arnaud Renouf que j'ai encadré pendant ses quatre années de thèse de doctorat et qui poursuit comme responsable R&D dans une entreprise la région nantaise. Je le remercie pour l'apport de son travail de doctorat dans la construction du projet de recherche que je conduis.

Enfin, j'adresse toutes mes amitiés à mes collègues ainsi qu'à tous les doctorants et stagiaires actuels et passés ; et la liste est trop longue pour être citée ici. Toutefois, je souhaite remercier en particulier Abderrahim Elmoataz, représentant du « canal historique » du laboratoire, pour ses éclairages en matière de traitement des images, Sébastien Fourey avec qui j'ai toujours eu des discussions passionnées sur l'enseignement ou d'autres sujets divers, et Daniel Carré qui a été un phare en matière de pédagogie et que la retraite nous a privé.

Enfin, le travail de recherche est extrêmement chronophage et a tendance à beaucoup empiéter sur la sphère privée. Je suis corvéable à Corinne pour sa patience et son soutien. À charge de revanche.

Avant Propos

En tant qu'enseignant-chercheur, mes activités professionnelles se répartissent majoritairement entre les deux missions que sont la recherche et l'enseignement. J'ai toujours accordé une importance égale à chacune de ces deux missions, que je n'envisage d'ailleurs pas de façon séparée.

Sans aucun doute, ma recherche nourrie mes enseignements. Elle apporte une certaine rigueur et maturité scientifique qui permet de dispenser, au delà du savoir nécessaire à la formation d'un ingénieur, un savoir-faire pour organiser et mettre en valeur ce savoir. Elle développe le goût de l'innovation qui pousse, sans cesse, à revisiter le contenu des enseignements.

De l'autre côté, l'enseignement permet une ouverture d'esprit par le fait que l'on enseigne des matières non forcément directement en relation avec sa recherche, et par la prise en compte des attentes souvent très concrètes des industriels en matière de formation des élèves ingénieurs. L'école d'ingénieurs a ceci d'intéressant, qu'elle est en prise directe avec le monde industriel, en particulier au travers de projets et de stages d'élèves ingénieurs.

Comme beaucoup d'autres avant moi, l'écriture d'un mémoire d'habilitation à diriger des recherches m'est d'abord apparue comme une charge supplémentaire dans un emploi du temps déjà conséquent et comme beaucoup d'autres avant moi, le mémoire a été été rédigé pendant la période des vacances universitaires. Mais a posteriori, je considère la rédaction comme une opportunité de faire le point sur un parcours professionnel et sur un projet de recherche. C'est un exercice différent de l'écriture de papiers qui reste ciblée sur une partie seulement de son projet de recherche. L'analyse est ici globale et permet de dégager de nouvelles orientations et perspectives pour le projet.

Ce mémoire est divisé en deux parties indépendantes :

- *La première partie est consacrée à mon parcours professionnel et liste mes activités liées à l'enseignement et les responsabilités assumées.*
- *La seconde présente mes travaux de recherche. Cette partie n'est pas qu'un simple résumé. J'ai voulu au contraire détailler le projet de recherche que je conduis depuis le début de ma recherche. Ce projet porte sur la conception de systèmes pour le traitement d'images avec une approche basée sur la représentation explicite de la connaissance.*

Profitant de la rédaction de ce manuscrit, j'ai voulu souligner le dynamisme du groupe de travail SCATI *« Système Complexe pour l'Analyse et le Traitement d'images » du GDR ISIS et du GDR I3, que je co-anime avec Rémy Mullot et Didier Coquin. Chaque fois que cela a été possible, j'ai privilégié les références à des travaux d'auteurs ayant participé à ce groupe de travail. La volonté est aussi de marquer l'originalité des travaux de recherche menés en France sur le thème de la conception de systèmes d'analyse et de traitement d'images.*

Dans le mémoire, la plupart des schémas d'illustration utilisés dans ce mémoire sont en anglais, dans la mesure où ils ont été repris de papiers publiés en anglais.

Tout ce qui est simple est faux, mais tout ce qui ne l'est pas est inutilisable.

Paul Valéry

Table des matières

Introduction **1**

1 Systèmes de traitement d'images **5**
1 Développement d'application de traitement d'images 5
1.1 Une définition du traitement d'images 5
1.2 Application de traitement d'images 7
1.3 Construction d'application 8
1.4 Exemples d'application 9
2 Conception de système de traitement d'images 13
2.1 Notion de système de traitement d'images 13
2.2 Historique 13
2.3 Constat d'échec 14
2.4 Aujourd'hui : des approches numériques « basées image » 14
3 Vers un système à base de connaissances 15
3.1 Limites des approches numériques 15
3.2 Autopsie de l'échec des approches à base de connaissances 16
3.3 Une proposition de système 17

2 Formulation d'objectifs de traitement d'images **19**
1 Introduction 19
2 La problématique de la formulation 20
2.1 Pourquoi une formulation des objectifs est-elle nécessaire ? 20
2.2 Que doit contenir une formulation ? 21
2.3 État de l'art de la formulation 22
3 Un modèle de formulation 26
3.1 Définition de classes d'images 26
3.2 Spécification de buts 30
4 Une ontologie pour la formulation d'objectifs de traitement d'images . . 32
4.1 Concepts de niveau physique 32
4.2 Concepts de niveau perceptif 33
4.3 Concepts de niveau sémantique 33
4.4 Les concepts de tâche 35
4.5 Les concepts de contrainte 35
4.6 Rôles, restrictions et déductions 37
5 Évaluation du modèle 38
5.1 Différentes formulations d'un même objectif 38

5.2 Différents objectifs pour une même tâche 42
5.3 Exemple d'une application en imagerie biomédicale 42
6 Conclusion 44

3 La génération automatique de programmes d'application 47
1 Introduction 47
2 Problématique 48
2.1 Approches de la génération d'application 48
2.2 Pilotage de codes pour le traitement d'images 51
3 Un système de génération d'application (BORG) 54
3.1 Objectifs et motivations 54
3.2 Architecture du système 55
4 La base de données de traitement d'images 57
4.1 Les niveaux d'abstraction 57
4.2 Représentation d'un plan 59
4.3 Exemple d'un plan en cytologie 62
5 La base de connaissances de traitement d'images 63
5.1 Modèle de sources de connaissances 63
5.2 Sources de connaissances de planification 67
5.3 Sources de connaissances d'instanciation 69
5.4 Sources de connaissances d'exécution 69
5.5 Sources de connaissances de description 69
5.6 Sources de connaissances d'évaluation 69
5.7 Une base de connaissances pour traiter des images de cytologie . 71
6 Le contrôle de résolution 72
6.1 La base de données de contrôle 72
6.2 La base de connaissances de contrôle 72
6.3 Propriétés du contrôle 73
7 Conclusion 73

4 Interaction Homme-Machine 77
1 Introduction 77
2 Le problème de la formulation 78
2.1 Composition d'une formulation 78
2.2 Rôle de l'interface 79
3 Notre approche 81
3.1 Composition des requêtes 82
3.2 D'une représentation extensionnelle du problème à une représentation intensionnelle 84
3.3 Sémantique émergente 85
4 Le cycle d'interaction Homme-Machine 87
4.1 La formulation extensionnelle initiale 88
4.2 Première formulation intensionnelle 93
4.3 Reformulation 94
5 Conclusion 95

5 Acquisition des connaissances **97**
1 Introduction 97
2 Le problème de l'acquisition de connaissances 98
2.1 Nature des connaissances 98
2.2 Approche de l'ingénierie des connaissances 99
3 Un atelier pour l'ingénierie des connaissances 100
3.1 Architecture de l'atelier 100
3.2 Un atelier pour l'acquisition des connaissances 102
3.3 Un atelier pour l'expérimentation 105
4 Conclusion 106

Conclusion et perspectives **109**

Références **117**

Liste des tableaux

1.1 Les six catégories d'objectifs reconnues pour le traitement d'images. . . 6

2.1 Liste des concepts de niveau physique et les descripteurs associés. . . . 34
2.2 Liste des concepts de niveau perceptif et les descripteurs associés. . . . 34
2.3 Liste des catégories de descripteurs perceptifs. 35
2.4 Liste des tâches de traitement d'images de l'ontologie. 36
2.5 Liste des critères à optimiser avec les erreurs acceptables associées pour quelques unes des tâches. 36
2.6 Liste des niveaux de détail avec leurs erreurs acceptables pour quelques unes des tâches. 37
2.7 Liste des contraintes de contrôle référencées dans l'ontologie. 37
2.8 Spécification de l'objectif d'extraction des champs. 39
2.9 Définition de la classe d'images aérienne au niveau physique. 39
2.10 Une définition sémantique possible pour motiver l'utilisation de l'approche par croissance de régions. 41
2.11 Une définition sémantique qui motive l'utilisation de l'approche variationelle. 41
2.12 Une définition sémantique qui motive l'utilisation de l'approche statistique. 42
2.13 Spécification de l'objectif d'extraction d'objets cellulaires 43
2.14 Définition de la classe d'images au niveau physique. 44
2.15 Définition de la classe d'images au niveau sémantique. 45

3.1 Les attributs décrivant une tâche de traitement d'images. 60
3.2 Les attributs définissant les sources de connaissances de traitement d'images. 64
3.3 Exemple de la description de l'efficacité d'une source de connaissances de détection de contours basée sur une différenciation du premier ordre. 67
3.4 Un exemple de source de connaissances de type Planning-KS : Contrast-Classification effectue une classification de pixels à partir d'une binarisation de l'image basée sur la maximisation du contraste aux frontières des régions. 68

3.5 Un exemple de source de connaissances de type Description-KS : Pixel-Classification-Description construit une règle d'évaluation pour le but « pixel classification » et la contrainte « nombre de classe=2 », dans le but de vérifier que le nombre de classes de sortie est bien égal à 2 dans l'image de sortie. 70
3.6 Un exemple d'une source de connaissances de type Evaluation-KS : Functionality-Evaluation construit et évalue le résultat d'une fonctionnalité. 70

4.1 Liste des tâches de post-traitement et les opérations réalisables. 88
4.2 Extrait de la description physique de l'application de cytologie. 90
4.3 Extrait d'une définition extensionnelle de la classe d'images pour une tâche de segmentation : « partition <image> ». 91
4.4 Extrait d'une définition extensionnelle de la classe d'images au niveau sémantique pour la tâche de segmentation : « extract <serous cell nucleus> ». 92
4.5 Extrait d'une définition intensionnelle du concept de {serous cell nucleus} après la phase d'extraction de caractéristiques. 93

Table des figures

1.1 Organisation d'une chaîne d'analyse d'images. 8
1.2 (a) Une image contenant du texte artificiel. (b) Les zones de texte détectées sont repérées par une boîte englobante. 10
1.3 (a) Une image acquise par scanner. (b) L'image après restauration. . . . 11
1.4 (a) Une image initiale acquise par rayons gamma. (b) L'image améliorée mettant en évidence les détails fins. 12
1.5 Le type de caractéristiques utilisées par (Viola and Jones, 2004). 14
1.6 Modélisation d'une voiture par des patchs d'images pris autour des points d'intérêt (Agarwal and Roth, 2002). 15
1.7 Modélisation d'une moto par des régions construites autour des points saillants (Fergus et al., 2003). 15
1.8 Schéma de la procédure de segmentation selon (Leibe and Schiele, 2003). 16
1.9 L'architecture globale du système. 18

2.1 L'objectif de traitement d'images de cette application d'imagerie aérienne est de segmenter l'image pour isoler chaque zone de végétation dans une région. 22
2.2 Deux façons différentes de définir un objet d'intérêt par extension : (a) par blob (b) par patchs d'image. 23
2.3 Exemple de la définition d'un grain de pollen de type « poaceae » faite à partir de « l'Ontologie de Concepts Visuels » proposée par Maillot et al. (Maillot and Thonnat, 2008). 24
2.4 Différentes approches pour la spécification d'objectif par l'exemple : (a) sketch (b) segmentation manuelle (c) gribouillages. 25
2.5 Illustrations de l'hypothèse phénoménologique. (a) Dans cette image aérienne de parking, le concept de bus peut se réduire à un simple rectangle blanc homogène. La différence d'apparence d'une même bouteille vue dans deux position différentes (b) et (c) conduit à distinguer deux objets d'intérêt différents. *Les images b et c sont tirées de la base Columbia Object Image Library (COIL-100).* 27
2.6 Selon l'approche sémiotique, une image est analysée en trois niveaux (1) physique, (2) perceptif et (3) sémantique. 28
2.7 Diagramme UML du modèle conceptuel de la définition d'une classe d'images. 30
2.8 Diagramme UML du modèle conceptuel de la spécification de but de traitement d'images. 31

2.9 L'analyse des effets générés par les différents composants possibles d'une chaîne d'acquisition fournie les concepts du niveau physique pour l'ontologie. 32
2.10 Les concepts du niveau physique correspondent aux effets de la chaîne d'acquisition sur les images d'entrée. 33
2.11 Les concepts du niveau perceptif correspondent aux primitives visuelles. 33
2.12 Une scène de paysage rural est composée d'objets géographiques juxtaposés : étendue d'eau, végétations, zones urbaines et routes, etc. La juxtaposition des objets géographiques est représentée par la relation topologique « externally connected » (EC). 40
2.13 (a) Exemple d'une image de cytologie, et (b) Le résultat souhaité surimposé à l'image initiale. 43
2.14 Les images de cytologie se composent de cellules étalées sur une lame. . 44

3.1 Un programme est un graphe d'opérateurs exécutables paramétrés. A gauche, est donnée la représentation d'une détection de contours utilisant l'algorithme DOG (Difference of Gaussian) sous la forme d'un graphe d'opérateurs et à droite sous la forme d'une séquence des commandes exécutables équivalente. 52
3.2 L'architecture du système BORG. 55
3.3 L'architecture du système BORG. 59
3.4 Flot de données échangé entre une tâche et les sous-tâches de sa décomposition. 61
3.5 Plan permettant de construire une application de cytologie. Les flots d'images au niveau fonctionnalité et au niveau opérateur sont dessinés avec des flèches grises. 62
3.6 Exécution de l'opérateur « binarization » en mode optimisation. 63
3.7 Mesures de l'efficacité d'une source de connaissances. 66
3.8 Liste des sources de connaissances utilisées pour construire l'application de cytologie. 71

4.1 Une représentation des fossés sensoriel et sémantique. 79
4.2 Exemple d'une requête pour la recherche d'images par le contenu. . . . 81
4.3 Architecture du système interactif de génération d'application. 82
4.4 Quatre exemples d'images de l'application de cytologie de séreuse. (*Images fournies par le département de cytologie et d'anatomie pathologique de l'hôpital publique du Cotentin.*) . 83
4.5 L'interface permettant de spécifier les post-traitements. 89
4.6 L'interface permettant de définir le niveau physique de la classe d'images. 90
4.7 Une scène de cytologie est composée de cellules de séreuse et de globules rouges qui reposent sur une muqueuse. Une cellule de séreuse est composée d'un noyau entouré d'un cytoplasme. Dans cet arbre DC, EC, TPP, NTPP, et PO sont des relations topologiques de la grammaire RCC-8. 91
4.8 L'interface permettant de spécifier la tâche. 93
4.9 Interface permettant de corriger la formulation initiale. 94

5.1 L'architecture globale de l'atelier. 101

5.2 Modélisation par tâches de deux façons de localiser les noyaux des cellules de séreuse. La première utilise la détection des contours puis leur fermeture pour obtenir les frontières des primitives régions. La seconde passe par la détection de germes à l'intérieur des régions et par une localisation des régions basée sur une Ligne de Partage des Eaux. 104
5.3 L'interface de construction de plans de l'atelier de génie logiciel PARTHENOS 105
5.4 Utilisation de la notation QOC pour justifier les deux alternatives possibles pour la tâche « find region markers ». 107

Introduction

Cette partie du mémoire synthétise un parcours de recherche qui s'étale sur dix huit ans, depuis le début de ma thèse de doctorat en 1991 jusqu'à aujourd'hui. Il révèle comment l'ensemble de mes activités de recherche ont toujours été motivées par le même objectif : *la conception de systèmes complexes pour le traitement des images permettant à des utilisateurs, non nécessairement spécialistes du traitement d'images, de construire seuls leurs applications*. Cet objectif définit à lui seul les contours du projet nommé PANTHÉON [1], que j'anime au sein de l'équipe IMAGE du GREYC et qui regroupe les travaux théoriques et méthodologiques et les productions logicielles concourant à cet objectif. Même s'il faut envisager cet objectif à long terme, il est dès à présent un moyen de mesurer la convergence des travaux faits dans ce projet, par rapport à la largeur du spectre d'application, au degré d'utilisabilité du système et au niveau d'expertise exigé de l'utilisateur.

De prime abord, un parcours motivé par ce seul objectif peut sembler réducteur, voire révélateur d'une pathologie obsessionnelle, mais en fait la conception de tels systèmes est un thème de recherche pluridisciplinaire, qui couvre des problématiques issues de plusieurs domaines scientifiques comme le Génie Logiciel, le Traitement d'Images et l'Intelligence Artificielle. Ce problème est, en effet, connu pour être particulièrement complexe, parce que le système doit savoir opérer à partir de données d'entrée incomplètes et entachées d'erreurs, d'objectifs non complètement spécifiés et de traitements difficilement maîtrisables. L'élaboration de solutions passe donc par l'intégration de résultats pris dans plusieurs champs thématiques, tels que l'interaction homme-machine, la résolution de problèmes, l'ingénierie des connaissances, les ontologies, la conception logicielle orientée composants ou encore l'apprentissage.

Ce parcours n'est évidemment pas solitaire. Les travaux présentés ici sont le fruit de collaborations, en particulier avec des étudiants de l'école d'ingénieurs et de master que j'ai encadrés au cours de stages ou de projets, avec des collègues du laboratoire ou de groupes de travail et surtout avec Arnaud Renouf au cours de sa thèse. Dans ce mémoire, j'utiliserai donc le « nous » pour présenter les résultats de ces travaux.

Contexte de recherche

Le but de ce mémoire est de faire le point sur la problématique de conception de système complexe pour le traitement d'images et de présenter l'approche que nous

1. http://www.greyc.ensicaen.fr/~regis/Pantheon

façonnons depuis toutes ces années de recherche.

Le domaine étudié se limite résolument au **traitement d'images**, qui est délibérément isolé des domaines qui l'englobent habituellement, comme l'analyse d'images ou la vision artificielle. Plus particulièrement, nous nous focalisons sur le cas du traitement d'images fixes, qu'elles soient 2D ou 3D, en niveaux de gris, en couleur ou multispectrales. À l'évidence, le traitement d'images n'est pas une fin en soi et n'a aucun pouvoir décisionnel. Les résultats d'une application ne doivent pas être tenus comme définitifs, mais comme les données d'entrée élaborées d'une analyse de plus haut niveau visant à l'interprétation, la visualisation, le stockage, ou la transmission d'images. Mais nous voulons regarder le traitement d'images comme un domaine à part entière avec ses propres problèmes et solutions. On ne cherche donc pas à développer un système d'interprétation d'images ou de vision, mais bien un système de traitement d'images, qui pourra ensuite être intégré dans un système plus global d'interprétation ou de vision.

La notion de **système** de traitement d'images est plus ambitieuse que celle de programme informatique dédié à une application particulière parfaitement identifiée. Un système est conçu pour opérer efficacement pour un nombre d'applications différentes et inconnues a priori. Sa conception nécessite donc plus que la mise au point d'un algorithme, aussi sophistiqué soit-il. De ce fait, le propos ne porte pas ici sur l'étude des fondements mathématiques nécessaires à l'élaboration des opérations de traitement des images, mais sur l'étude des fondements épistémologiques nécessaires à la résolution du *problème du contrôle*, c'est-à-dire essentiellement la mise en œuvre de ces algorithmes dans le cadre d'applications concrètes. On ne cherche donc pas à concevoir un algorithme original pour une application cible, mais un programme qui puisse naviguer dans un ensemble d'algorithmes existants pour composer les programmes d'application cibles.

La différence majeure avec la conception de programmes dédiés, et donc la source des difficultés supplémentaires, c'est l'introduction de l'utilisateur comme acteur du système. L'utilisateur est là pour spécifier les termes de l'application particulière à traiter et c'est à partir de cette spécification que le système devra élaborer sa solution. L'efficacité du système repose en grande partie sur sa capacité à prendre en compte l'utilisateur dans la boucle pour construire des programmes d'application adaptés.

Point de vue développé

Le point de vue développé dans ce document soutient que le recours aux connaissances symboliques explicites, et donc à des systèmes à base de connaissances, est une voie prometteuse pour concevoir des systèmes de traitement d'images, dès lors que l'on envisage d'accroître les capacités d'adaptation à tout un ensemble de contextes différents. Ce point de vue semble pourtant aller à l'encontre des approches actuelles, qui s'orientent franchement vers des systèmes purement numériques essentiellement basés sur l'apprentissage supervisé à partir d'exemples. En effet, l'expérience des systèmes à base de connaissances pour concevoir des systèmes de traitement d'images a déjà été faite par de nombreux travaux de recherche, mais le bilan s'est soldé par un échec qui a marqué un coup d'arrêt de ces travaux. La cause principale est connue, c'est l'éternel problème de l'acquisition des connaissances qui suppose l'identification d'une expertise rationnelle formalisable dans le fonctionnement cognitif des experts du traitement d'images. Les approches purement numériques évitent alors le problème de l'acquisition des connaissances. Elles offrent ainsi des

solutions plus réalistes à court terme, pour résoudre des problèmes concrets.

Mais nous prétendons ici que ces approches numériques ne peuvent pas répondre complètement aux besoins de généralité et d'adaptabilité ambitionnés pour les systèmes de traitement d'images, parce que ces approches ne permettent pas de représenter entièrement la sémantique d'une application et prendre la pleine mesure du problème adressé par l'utilisateur. Au même titre que la vision, le traitement d'images est une activité située qui nécessite de représenter les connaissances sur le problème à résoudre (Garbay, 2002). Nous montrerons que ces connaissances ne peuvent pas être intégralement apprises avec des approches purement numériques.

La représentation symbolique explicite des connaissances est, par ailleurs, une contrainte de notre projet, puisqu'au delà de la réalisation effective d'un système de traitement d'images, la motivation est aussi d'élaborer une théorie formelle du développement d'application en traitement d'images qui explique comment s'analyse et se conçoit une application. Cette théorie est la base de la conception du système à base de connaissances.

Une nouvelle approche

Pour aborder à nouveau la conception de systèmes de traitement d'images sous l'angle des systèmes à base de connaissances, nous proposons d'adopter une nouvelle approche et de nouvelles hypothèses fortes pour la conception.

Un premier changement dans l'approche consiste à fonder le système sur l'*interaction homme-machine*, où l'homme et la machine collaborent pour construire une solution : l'homme apportant sa connaissance du problème à traiter et le système apportant ses capacités de traitement. Le second changement porte sur une vision *constructiviste* de l'acquisition des connaissances, basée sur le fait que l'on ne cherche plus à modéliser l'expertise humaine, mais à restituer un processus de construction de solutions, où l'expert n'est qu'une source de connaissances parmi d'autres. Dans ce cadre, les deux hypothèses, *phénoménologique* et de *sémantique émergente*, permettent de considérer qu'il existe bien un domaine du traitement d'images modélisable, avec ses propres connaissances indépendantes des domaines d'application. Cela justifie la construction d'une *ontologie du domaine* du traitement d'images, qui fournit le support de la définition d'un langage de représentation des problèmes d'application.

Organisation du manuscrit

Cette partie du manuscrit présente donc une analyse a posteriori de mes travaux de recherche portant sur la conception de systèmes à base de connaissances pour le traitement d'images. Le plan du mémoire ne suit pas une description chronologique de mes travaux. Mais il est organisé autour des quatre problématiques abordées par la conception de ce système, à savoir la formulation d'objectifs de traitement d'images, la génération automatique de chaînes de traitement, l'interaction homme-machine pour le co-développement d'applications et l'acquisition des connaissances.

Le premier chapitre est une introduction à la notion de système de traitement d'images. Nous définissons d'abord le problème du développement d'applications et soulignons la complexité de la tâche d'automatisation. Nous montrons ensuite que la complexité de la tâche a découragé beaucoup de travaux de recherche qui se sont plutôt réorientés vers la conception de systèmes basés sur l'apprentissage, mais avec

moins d'ambition dans la généralité. C'est justement cette complexité qui justifie l'intérêt que nous portons à la conception de systèmes cognitivistes, dont nous présentons les caractéristiques à la fin du chapitre.

Le second chapitre porte sur l'étude de la formulation d'objectifs de traitement d'images. La formulation est le moyen fourni aux utilisateurs pour exprimer leurs intentions. À partir d'une étude sur les informations que nous identifions comme nécessaires et suffisantes aux développement d'applications, nous proposons une théorie de la formulation qui est ensuite formalisée dans une ontologie de domaine pour fixer le langage de formulation.

Le troisième chapitre concerne la conception d'un générateur de programmes qui agit à partir d'une formulation donnée. Ce générateur est basé sur le paradigme du pilotage de codes prédéfinis et produit ses programmes par assemblage de ces codes. Il est basé sur une architecture à base de connaissances de type Tableau Noir (angl. *Blackboard*), et code ses compétences sous la forme de sources de connaissances modulaires, autonomes, indépendantes et spécialisées.

Le quatrième chapitre est centré sur l'interaction homme-machine pour le développement interactif d'applications de traitement d'images. Les programmes sont construits conjointement par l'utilisateur et le générateur de programmes, dans le but de faire émerger graduellement la sémantique de l'application à développer. Une interface est proposée pour conduire l'interaction entre les deux acteurs.

Le cinquième chapitre traite des problèmes de l'acquisition des connaissances pour le générateur de programmes. Pour cela, nous avons développé un atelier d'ingénierie des connaissances qui permet de capitaliser et d'expérimenter les connaissances de traitement d'images. Les modèles de capitalisation des connaissances proposés forment une théorie du développement d'application, et ils sont à la base de la conception du générateur de programmes. Les connaissances capitalisées dans l'atelier sont ainsi facilement opérationnalisables dans la base de connaissances du générateur.

Enfin, en conclusion, nous résumons notre point de vue sur la conception de système de traitement d'images et présentons nos futures directions de recherches qui bénéficient des travaux déjà entrepris.

CHAPITRE 1

Systèmes de traitement d'images

Ce chapitre est une introduction à la notion de système de traitement d'images telle que nous l'envisageons. Un système correspond à un logiciel capable d'opérer dans une variété de domaines d'application différents. Pour cela, l'introduction de l'utilisateur dans la boucle devient indispensable pour spécifier l'objectif et le contexte de chaque application et pour évaluer les résultats finaux.

Les motivations de notre projet portent sur la réalisation d'un système général capable de s'appliquer à tous les objectifs et domaines d'application du traitement d'images. Notre approche est orientée vers la conception d'un système à base de connaissances, qui utilise une représentation symbolique explicite des connaissances et un raisonnement abstrait pour produire des solutions. Elle s'oppose ainsi aux approches purement numériques « basées image », qui utilisent l'apprentissage supervisé à partir d'exemples pour spécialiser le système à l'application cible. Nous cherchons à montrer ici que, dans l'état actuel des connaissances, seule l'approche basée sur les connaissances permet d'envisager réellement la généralité et une réelle prise en compte des intentions des utilisateurs.

1 Développement d'application de traitement d'images

1.1 Une définition du traitement d'images

Avant toute considération, il est indispensable de se doter d'une définition précise du traitement d'images en l'absence de consensus dans la communauté. Si le traitement d'images est unanimement situé entre la production et l'analyse d'images, les limites de séparation entre chacune de ces disciplines ne sont pas clairement fixées par les définitions existantes. Notre définition est conçue selon le point de vue des objectifs couverts par le domaine et de la nature des données d'entrée :

Définition 1. *Le traitement d'images recouvre tous les objectifs de transformation d'images en images sans interprétation du contenu. Les données d'entrée sont des images iconiques et les données de sorties des images iconiques, intrinsèques ou segmentées.*

Une image iconique est une représentation d'une *scène*[1] par des valeurs d'intensité, c'est-à-dire que les pixels de l'image codent une valeur de niveau de gris, de couleur ou multispectrale. Une image iconique se différencie donc d'une image intrinsèque qui représente des propriétés physiques sur la scène comme l'orientation de surface, la profondeur ou la réflectance des surfaces, et d'une image segmentée qui est une représentation abstraite de la scène faite à partir de primitives telles que les régions, contours ou points d'intérêt. Selon la définition donnée, même si l'image d'entrée est une image segmentée obtenue par un traitement préalable, elle sera considérée comme une image iconique en entrée d'un processus de traitement d'images, c'est-à-dire que les labels seront manipulés comme s'il s'agissait de valeurs d'intensité et perdront leur caractère abstrait.

Si l'on considère les trois transformations de base que sont l'ajout, la modification ou la suppression, appliquées aux images à des fins de pur changement de valeurs d'intensité ou des fins d'augmentation de l'intelligibilité de l'information, cette définition conduit à envisager le traitement d'images comme étant la science qui couvre les six catégories d'objectifs suivantes (*cf.* tableau 1.1) :

- La **restauration** vise à retrouver l'image d'origine à partir de sa version dégradée par ajout d'information au niveau intensité, connaissant le phénomène de dégradation.
- L'**amélioration** cherche à adapter au mieux les données à une exploitation visuelle par modification des valeurs d'intensité.
- La **compression** a pour objectif de réduire la quantité de données nécessaires pour coder une image numérique en supprimant les valeurs d'intensité identifiées comme redondantes.
- La **reconstruction** consiste à créer une nouvelle image par ajout d'une information spatio-temporelle, telle que la forme, le relief ou le mouvement, déduite des valeurs d'intensité d'une vue de la scène ou de plusieurs.
- La **segmentation** propose une représentation abstraite de l'image d'intensité sous la forme de primitives visuelles.
- La **détection** fournit un masque positif des zones d'intérêt de l'image et supprime toutes les autres informations.

Tableau 1.1 – Les six catégories d'objectifs reconnues pour le traitement d'images.

	Transformation	**Augmentation**
Ajout	Restauration	Reconstruction
Modification	Amélioration	Segmentation
Suppression	Compression	Détection

Cette définition est plus restrictive que celles proposées par d'autres auteurs, qui y incluent généralement l'extraction de caractéristiques, telles celles de (Gonzalez and Woods, 2002) ou (Wikipedia, 2009). Le point de vue des objectifs couverts et de la nature des entrées, adopté pour notre définition, vise à se doter d'un crible permettant de savoir reconnaître précisément le type de problèmes adressés par le traitement d'images, de ceux qui relèvent alors de l'analyse d'images. Toutefois, elle ne présume pas de la façon dont sont atteints ces objectifs. Ainsi, pour accomplir un objectif, il est possible de faire appel à des techniques d'extraction de caractéristiques ou des techniques attribuées à d'autres objectifs de traitement d'images. Par exemple,

1. Dans ce document, le terme « scène » désigne un phénomène observé ou calculé qu'il soit naturel ou artificiel.

la compression d'une image peut recourir à des techniques de segmentation et de restauration d'images.

Selon cette définition, le *shape from X* correspond bien à du traitement d'images puisqu'il s'agit d'un objectif de reconstruction dont les images d'entrée sont iconiques et celles de sortie intrinsèques. Mais, la reconstruction d'un réseau routier à partir d'une image de contours, précédemment extraits d'une image aérienne par exemple, ne relève pas du traitement d'images mais de la reconnaissance de formes, puisque les images d'entrée ne doivent pas être considérées comme des images d'intensité mais comme des cartes de contours.

De même, du point de vue du traitement d'images, la détection d'objets dans une image se limite à la détection des zones d'image pouvant correspondre à des parties d'objet ou des amas d'objets identifiables par leur intensité. Mais, la détection plus précise d'objets nécessite de raisonner sur d'autres informations que les seules valeurs d'intensité des images d'entrée – *p. ex.*, les relations de composition ou l'analyse de la forme. Dans ce cas, on a recours à la reconnaissance des formes ou à la classification d'objets.

1.2 Application de traitement d'images

Une fois défini le domaine du traitement d'images, il nous faut définir la notion d'application de traitement d'images.

Définition 2. *Une application de traitement d'images est un logiciel spécialisé dans l'accomplissement d'un objectif de traitement d'images, dont les images d'entrée se conforment à une classe.*

Cette seconde définition détermine clairement une application à partir d'un objectif et d'une classe d'images.

L'objectif se réfère à l'une des six catégories identifiées par la définition 1. Si la définition limite une application à un objectif unique, le cas d'une application à plusieurs objectifs peut toujours se définir comme autant de sous-applications à objectif unique, organisées séquentiellement ou parallèlement. Par exemple, une application de numérisation de livre ancien enchaîne une étape de reconstruction pour une remise à plat de la page par « shape from X », puis une amélioration du rendu visuel en homogénéisant la surface de chaque page avant la compression pour l'archivage numérique.

La classe d'images spécifie un ensemble d'images qui partagent un certain nombre de caractéristiques en terme d'apparence et de sémantique. La définition 2 implique donc que toute modification de l'objectif ou de la classe d'images conduit à envisager une nouvelle application. Mais elle impose également que l'application possède des capacités d'adaptation de son comportement face à la variabilité des images appartenant à la classe.

Une application est donc plus qu'un programme interactif de retouche d'images qui permet à un utilisateur de traiter les images une à une, en choisissant lui-même les transformations à opérer. Elle est au contraire construite pour traiter automatiquement un ensemble d'images sans intervention extérieure, et les traitements doivent être prévus pour toute image pouvant appartenir à la classe.

Dans l'organisation d'une chaîne d'analyse d'images, une application de traitement d'images est positionnée comme un programme intermédiaire qui utilise des images provenant d'un système de production d'images et qui génère des images de

sortie destinées à un système de post-traitements (*cf.* figure 1.1). De cet fait, l'acquisition est responsable de la nature et de la qualité des images iconiques consommées en entrée, et les post-traitements imposent l'objectif et la nature des résultats à produire. Ainsi situé, le rôle du traitement d'images apparaît comme crucial puisqu'il consiste à extraire des images, des informations qualitatives ou quantitatives pour les post-traitements qui suivent, par des procédures de réduction et d'abstraction de l'information initiale, sans perte ni falsification de l'information pertinente.

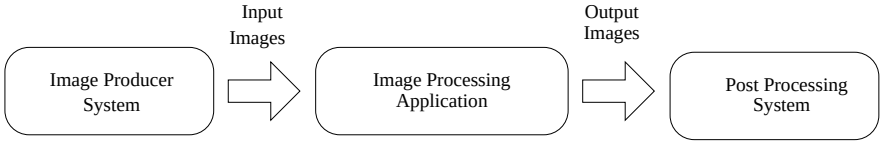


Figure 1.1 – Organisation d'une chaîne d'analyse d'images.

1.3 Construction d'application

Le traitement d'images fournit son savoir sous la forme d'opérateurs de transformations d'images codés par des algorithmes. Les efforts de recherche menés depuis les années 50 ont conduit à la réalisation d'une quantité impressionnante de ces algorithmes et la production semble encore s'accélérer aujourd'hui.

Un opérateur est conçu pour effectuer une transformation plus ou moins ponctuelle des données d'entrée. Chacun est élaboré sur un modèle numérique précis de l'information à traiter, ce qui conditionne fortement son domaine d'applicabilité et son efficacité. Différentes approches de modélisation numérique sont explorées simultanément par les chercheurs en traitement d'images. Pour cela, ils s'appuient sur des théories numériques diverses, parmi lesquelles l'analyse mathématique, les approches variationnelles, statistiques ou géométriques, la théorie des graphes, la morphologie mathématique ou encore les problèmes inverses. Ainsi, les opérateurs de détection de contours basés sur l'approche variationnelle sont adaptés lorsque l'image peut se voir comme une fonction continue par morceaux de l'intensité, telles que les images acquises avec une caméra CDD. Par contre, ils sont peu adaptés quand l'image est de nature statistique comme les images d'échographie. Pour adapter leur comportement à la variabilité des images d'entrée, les opérateurs disposent la plupart du temps de paramètres réglables.

Une application complète nécessite typiquement l'intégration de plusieurs de ces opérateurs pour former des chaînes de traitements successifs, où les sorties des uns servent d'entrée aux suivants dans la chaîne. Par exemple, un processus de segmentation d'images simple enchaîne une détection de contours puis une fermeture de ces contours, et enfin une labelisation des régions délimitées par ces contours.

1.3.1 Un problème complexe et compliqué

Contrairement au développement des opérateurs, le développement d'une application ne peut s'appuyer sur aucune théorie numérique. Elle s'envisage en général de manière empirique comme un problème de sélection, paramétrage et enchaînement d'opérateurs pour créer des chaînes de traitement complètes en fonction de l'objectif et des caractéristiques des images de la classe (Cocquerez and Philipp, 1995).

L'absence de théorie formelle globale en fait une activité *complexe* au sens de la systémique (Simon, 1969; de Rosnay, 1975), dans la mesure où elle fait appel à l'intégration de connaissances, de formes et d'origines diverses, pour résoudre des problèmes dont les sorties ne sont pas en relation causale simple avec les entrées. En effet, premièrement, les informations sur le problème à résoudre sont disponibles à la fois sous forme numérique et symbolique, tandis que les traitements qui les utilisent reposent sur des modèles numériques plus ou moins explicités. Par exemple, l'utilisation d'un détecteur de contours suppose de savoir reconnaître, sur l'image à traiter, le modèle particulier de contours que le détecteur code numériquement – *p. ex.*, la forme, le contraste, la largeur. Deuxièmement, les différentes approches du traitement des images s'appuient sur des points de vue différents sur les connaissances et prennent leurs références dans des domaines théoriques également différents (Garbay, 2001) – *p. ex.*, traitement du signal, analyse numérique, statistiques, optique. Chacune d'elles conduit à une formulation propre des problèmes et de leurs solutions qui rend souvent difficile la comparaison et la coopération d'approches. Enfin, troisièmement, la construction des résultats nécessite de fréquents changements de représentations intermédiaires comme les changements de niveau d'abstraction (*p. ex.*, pixel en contour, contour en région) ou de référentiel (*p. ex.*, spatial, fréquentiel, photométrique) qui ne peuvent pas être décidés à partir des données d'entrée.

La programmation d'une solution de traitement d'images est typiquement une activité *compliquée*. Elle s'inscrit dans un environnement qui comporte un très grand nombre d'éléments à combiner. La masse de données et d'opérations rend les algorithmes de traitement laborieux à coder. Les tests en particulier, ne peuvent s'appuyer ni sur le contrôle visuel des résultats, qui est insuffisant, ni sur l'examen des valeurs des pixels des images de sortie, qui est fastidieux.

Enfin, l'évaluation des résultats est confrontée aux difficultés de définition de figure de mérite. Les critères d'évaluation existent bien, mais ils ne s'expriment pas directement sous forme quantitative. Même quand ils le sont, ils portent sur les résultats finaux et plus rarement sur les résultats intermédiaires. Il n'est alors pas trivial de remonter la chaîne de traitements pour localiser les causes des erreurs. Par exemple, une erreur de localisation des frontières d'un objet peut être due à une description incomplète de l'objet, un mauvais choix de méthode de localisation ou un mauvais réglage de ses paramètres.

A défaut de théorie globale, et en s'appuyant sur le fait qu'une application est un logiciel à part entière, il est alors naturel de se tourner vers le génie logiciel. Mais les méthodes du génie logiciel type « Unified Process » (Kruchten, 2003), Agiles (Ambler and Jeffries, 2002), MDA (Kleppe et al., 2003) et leur langage de modélisation – *p. ex.*, UML (Muller and Gaertner, 2000; Booch et al., 2005) – s'avèrent n'être que de peu d'utilité pour développer des applications de traitement d'images. Leur champ d'action se limite quasiment à l'activité de programmation. Elles n'apportent en effet aucune solution aux difficultés énoncées précédemment et laissent alors le développeur seul avec ses compétences « naturelles » pour découvrir le problème à résoudre, proposer des solutions et les évaluer (Zamperoni, 1996).

1.4 Exemples d'application

Nous donnons ici quelques exemples d'applications montrant qu'une application peut s'écrire comme une chaîne d'opérateurs exécutables. Les scripts sont donnés en Shell Bash et utilisent les opérateurs de la bibliothèque PANDORE (GREYC, 2009).

1.4.1 Détection de texte dans les images

Cette application se situe dans le contexte de l'indexation vidéo. Elle est extraite de la thèse de C. Wolf (Wolf, 2003). L'objectif de l'application de traitement d'images est de détecter les zones d'image susceptibles de contenir du texte artificiel surajouté aux images (*cf.* figure 1.2).



(a) (b)

Figure 1.2 – (a) Une image contenant du texte artificiel. (b) Les zones de texte détectées sont repérées par une boîte englobante.

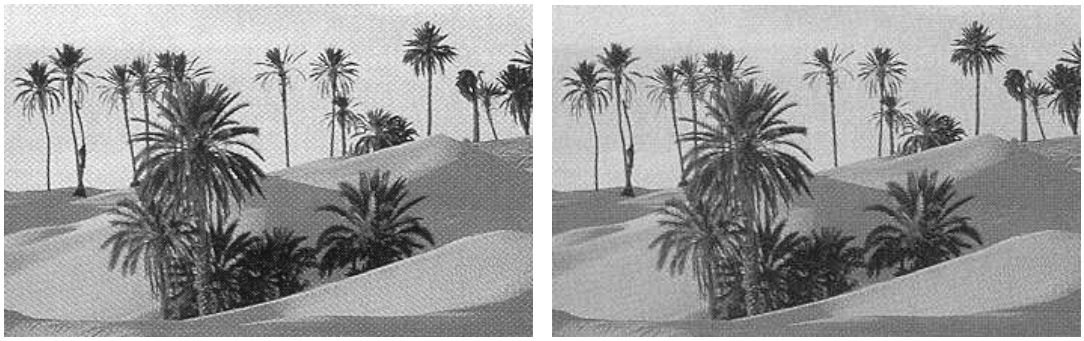
Le script 1.1 permet détecter les zones de texte candidates dans des boîtes englobantes. Ce script traduit une méthode basée sur la recherche des zones d'image ayant une forte densité de contours verticaux.

Script 1.1 – Script Shell pour la détection de texte dans des images.

```
# Vertical gradient
prgb2hsl input.pan i1.pan
pimc2img 2 i1.pan i2.pan
plineardilatation 0 1 i2.pan i3.pan
plinearerosion 0 1 i3.pan i4.pan
pdif i3.pan i4.pan i5.pan
# Horizontal closing
plineardilatation 0 10 i5.pan i6.pan
plinearerosion 0 10 i6.pan i7.pan
# Binarization
pvariancebinarization i7.pan i8.pan
res1=`pstatus`
pinverse i8.pan i9.pan
pmask i7.pan i9.pan i10.pan
pmeanvalue i10.pan
res2=`pstatus`
res3=`echo "scale=1;\$res1 - \$res2" | bc`
res4=`echo "scale=1;0.8 * \$res3" | bc`
res5=`echo "scale=1;\$res4 + \$res2" | bc`
pbinarization \$res5 1e30 i7.pan i11.pan
pgeodesicdilatation 1 1 -1 i8.pan i11.pan i12.pan
# Closing
pdilatation 1 1 i12.pan i13.pan
perosion 1 1 i13.pan i14.pan
# Vertical closing
plinearerosion 2 1 i14.pan i15.pan
plineardilatation 2 1 i15.pan i16.pan
# Eliminate bad candidates
plinearerosion 0 16 i16.pan i17.pan
plineardilatation 0 16 i17.pan i18.pan
plabeling 8 i18.pan i19.pan
pboundingbox i19.pan output.pan
```

1.4.2 Restauration d'images

L'objectif de traitement d'images est ici est de supprimer l'effet de tramage résultant d'une acquisition d'images par scanneur (*cf.* figure 1.3).



(a) (b)

Figure 1.3 – (a) Une image acquise par scanner. (b) L'image après restauration.

La méthode de traitement implémentée dans le script 1.2 consiste à retirer les pics de fréquence dans l'image de la transformée de Fourier de l'image initiale. L'effet de tramage est ici régulier et parfaitement caractérisable dans le domaine fréquentiel. La suppression est faite avec un chapeau haut de forme.

Script 1.2 – Script shell pour la restauration d'images scannées.

```
# Fourier transform
psetcst 0 scannerNdg.pan i1.pan
pfft scannerNdg.pan i1.pan reel.pan imag.pan

# Modulus
pfftshift reel.pan imag.pan i4.pan i5.pan
pmodulus i4.pan i5.pan modulus.pan

# Detect spikes.
plogtransform 0 0 255 modulus.pan | pim2uc – mod.pan
pmedianfiltering 3 mod.pan mod.pan

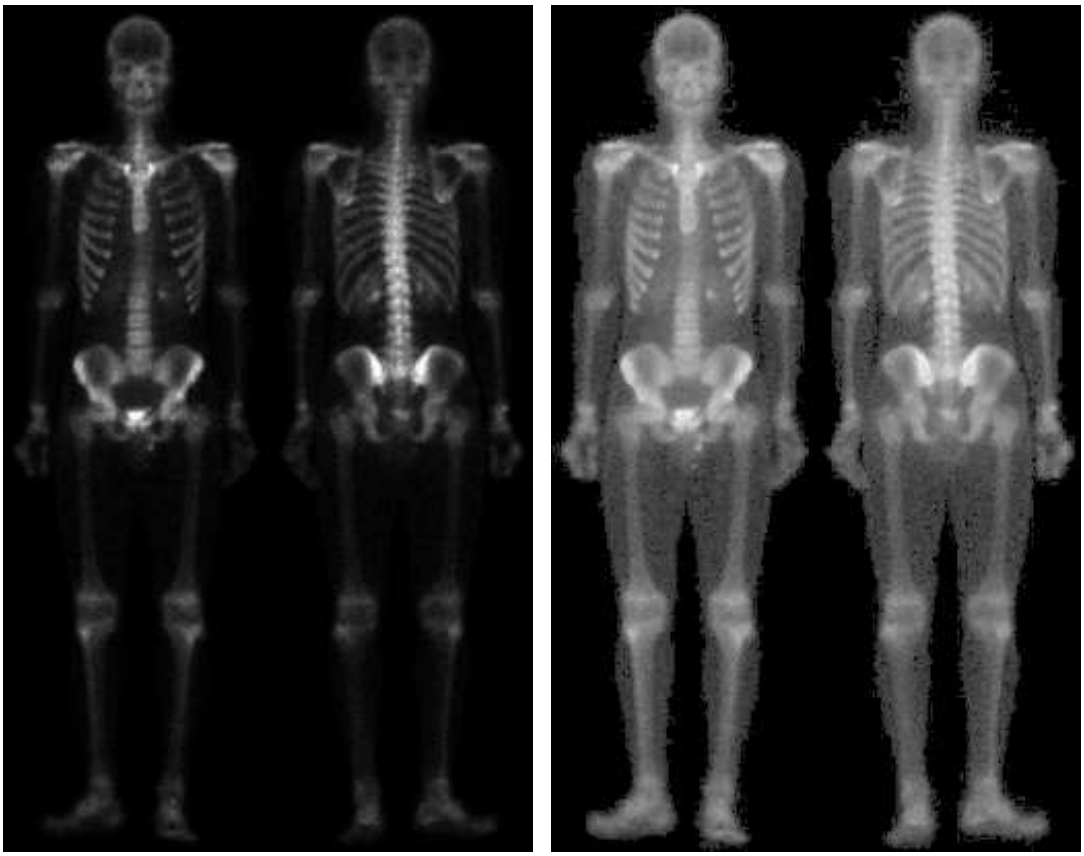
# Top hat to remove spikes.
perosion 1 8 mod.pan | pdilatation 1 8 – | pdif – mod.pan chapeau.pan
pentropybinarization chapeau.pan freq.pan
pshapedesign 1024 512 0 3 23 1024 horiz.pan
pshapedesign 1024 512 0 3 512 12 vertic.pan
por horiz.pan vertic.pan mask1.pan
pinverse mask1.pan mask1.pan
pmask freq.pan mask1.pan mask.pan

# Masking (inversion and shift)
pinverse mask.pan maskScannerNDGv.pan
pfftshift maskScannerNDGv.pan maskScannerNDGv.pan maskReel.pan maskImag.pan
pmask reel.pan maskReel.pan reelMasked.pan
pmask imag.pan maskImag.pan imagMasked.pan

# Inverse Fourier transform
pifft reelMasked.pan imagMasked.pan reelOut.pan ReelOut.pan noise.pan
plineartransform 0 0 255 reelOut.pan reelOut2.pan
pim2uc reelOut2.pan output.pan
```

1.4.3 Amélioration d'images

L'objectif est de rehausser la visualisation des détails fins dans des images acquises par capteur à rayons gamma (*cf.* figure 1.4). Cette application est tirée du livre de R.C. Gonzalez (Gonzalez and Woods, 2002).



(a) (b)

Figure 1.4 – (a) Une image initiale acquise par rayons gamma. (b) L'image améliorée mettant en évidence les détails fins.

Le script 1.3 combine à la fois une technique de « unsharp masking » par Laplacien pour rehausser les détails et par gradient pour favoriser les vrais contours au détriment du bruit.

Script 1.3 – Script shell pour l'amélioration d'images de rayons gamma.

```
psharp 8 1 input.pan i1.pan
psobel input.pan i2.pan
pmeanfiltering 2 i2.pan i3.pan
pmult i1.pan i3.pan i4.pan
plineartransform 0 0 255 i4.pan i5.pan
pim2sl input.pan i6.pan
padd i5.pan i6.pan i7.pan
ppowerlawtransform 0.3 0 255 i7.pan output.pan
```

2 Conception de système de traitement d'images

2.1 Notion de système de traitement d'images

Si les opérateurs ont un caractère générique et peuvent être réutilisés dans plusieurs contextes différents, les applications sont au contraire des programmes dédiés et peu réutilisables tels que. Il est donc nécessaire de construire une nouvelle application pour chaque nouveau problème.

Le très faible taux de réutilisabilité des applications justifie l'intérêt porté aux systèmes de traitement d'images. La vocation d'un système, c'est d'être utilisable dans plusieurs contextes différents. Cela veut dire qu'il faut intégrer au système un moyen de prendre en compte les connaissances sur le contexte d'application spécifique. Pour cela, le recours à l'utilisateur est incontournable (ou à un système d'analyse). Le challenge est donc de proposer des systèmes qui permettent à des utilisateurs de créer eux-mêmes des applications en spécifiant l'objectif et en définissant la classe des images. Dans ce cadre, l'intervention humaine n'est plus celle du développeur de programme informatique mais celle de l'analyste des besoins. La construction de l'application ne se fait plus au niveau code informatique mais au niveau problème.

2.2 Historique

Les chercheurs se sont intéressés très tôt à la conception de systèmes de vision capables d'analyser automatiquement le contenu des images. Ces recherches ont débuté avec l'ambition de la généralité et du tout automatique. Les systèmes les plus emblématiques de ces travaux sont les systèmes d'interprétation d'images VISIONS (Hanson and Riseman, 1978; Draper et al., 1989) et SIGMA (Matsuyama and Hwang, 1990). Mais très rapidement, le talon d'Achille de ces systèmes s'est révélé être la partie bas niveau, c'est-à-dire le traitement des images. Dans ces premiers systèmes, le traitement d'images était réduit à de simples algorithmes généraux de segmentation sans capacité réelle d'adaptation au problème. L'effort de recherche avait été mis sur la partie haut niveau estimant que les erreurs et insuffisances de la partie bas niveau pouvaient être compensées par la partie haut niveau grâce aux connaissances modélisées sur le domaine d'application. Mais, l'analyse de ces systèmes montre au contraire que les manquements de la partie traitement d'images sont rédhibitoires au fonctionnement du système global (Draper et al., 1996).

Dans un deuxième temps, des recherches se sont alors recentrées sur la partie bas niveau pour tenter de la rendre plus adaptative en intégrant dès ce niveau, les connaissances sur le domaine d'application. Plusieurs propositions de systèmes dédiés au traitement d'images ont été faites, principalement dans les années 1990 (Crevier and Lepage, 1997; Clouard et al., 1999b). Ainsi, ont été crées les systèmes SCHEMA (Draper et al., 1989) pour la partie bas niveau de VISIONS et LLVE (Matsuyama, 1989) pour SIGMA. Les propositions les plus ambitieuses ont porté sur la réalisation de systèmes à base de connaissances capables d'automatiser entièrement le développement d'une application pour peu que l'utilisateur soit capable de formuler le problème à résoudre dans le formalisme du système – *p. ex.*, (Liedtke and Blömer, 1992; Clément and Thonnat, 1993; Bodington, 1995; Chien and Mortensen, 1996; Clouard et al., 1999b). Ces systèmes reposent sur une approche cognitiviste basée sur la manipulation d'une représentation symbolique des informations de l'application. Ils profitent d'un raisonnement abstrait à partir de cette description pour construire les résultats.

2.3 Constat d'échec

Mais tous ces systèmes ont montré leurs limites, essentiellement dans le nombre de configurations qu'ils sont capables d'envisager et dans le manque d'adaptabilité à la variétés des images. Les raisons sont connues (Vernon, 2007). Elles se rapportent toutes au goulot d'étranglement de l'acquisition des connaissances et au manque d'ancrage des symboles dans la réalité des données.

De fait, aucun de ces systèmes n'a atteint l'ambition initiale de la généralité, même si un de ces systèmes a quand même été commercialisé (Toriu et al., 1987). Cet échec a donné un violent coup d'arrêt aux travaux de recherche sur le développement de systèmes à base de connaissances pour le traitement d'images.

2.4 Aujourd'hui : des approches numériques « basées image »

Aujourd'hui, les systèmes proposés s'orientent résolument vers des approches purement numériques basés sur l'apprentissage supervisé à partir d'images exemples. Ils évitent ainsi l'étape d'acquisition des connaissances et permettent une meilleure adaptation à la variété des images par un enracinement dans les données.

Beaucoup de systèmes purement numériques ont été développés récemment pour la reconnaissance et la détection d'objets. La plupart sont basés sur une approche qui s'inspire d'une théorie de la vision biologique expliquant la détection d'objet à partir de leurs parties prégnantes et des relations entre ces parties. Le système apprend le modèle des objets d'intérêt à partir d'exemples, qu'il utilise ensuite pour construire automatiquement un détecteur spécialisé, qui devient l'application cible.

Les parties d'objet peuvent être modélisées par des caractéristiques de très bas niveau. Par exemple, Viola et Jones (Viola and Jones, 2004) utilisent des « rectangle features », tels que ceux présentés dans la figure 1.5, qui rappellent les fonctions de base de Haar, pour capturer la présence de contours et autres structures de contraste simples dans le modèle d'objet. Chaque rectangle permet de calculer une valeur pour chaque pixel à partir de la différence entre la somme des pixels voisins dans la partie blanche et celle des pixels dans la partie noire. La classe d'images est représentée par les « rectangle features » qui décrivent le mieux les objets.

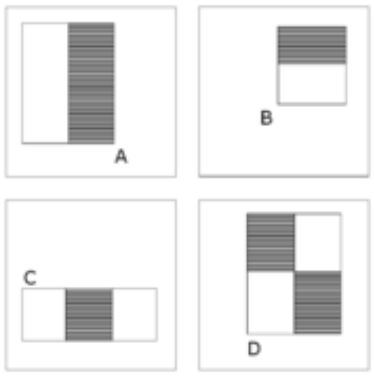


Figure 1.5 – Le type de caractéristiques utilisées par (Viola and Jones, 2004).

Les caractéristiques peuvent s'exprimer sous la forme d'un vocabulaire iconique à base de patchs d'image. Les objets sont modélisés par un ensemble de patchs

pour chacune de ces parties et des relations spatiales entre ces parties. Par exemple, dans le système de Agarwal et al. (Agarwal and Roth, 2002; Agarwal et al., 2004), les patchs de taille 13x13 sont extraits automatiquement autour des points d'intérêts (*cf.* figure 1.6 pour la modélisation d'une voiture), puis regroupées en classes selon leur similitude en intensité pour former un dictionnaire (Jurie and Triggs, 2005). Fergus et al. (Fergus et al., 2003) proposent de sélectionner les parties d'objets à partir de régions circulaires construites autour des points saillants (Kadir and Brady, 2001), et le rayon est contrôlé par l'échelle à laquelle le point apparaît comme saillant (voir la figure 1.7 pour la modélisation d'une moto). Tous ces systèmes reposent sur l'utilisation d'une fonction d'évaluation permettant de sélectionner automatiquement le jeu de caractéristiques pertinentes qui seront utilisées pour calculer la similitude entre patchs.



Figure 1.6 – Modélisation d'une voiture par des patchs d'images pris autour des points d'intérêt (Agarwal and Roth, 2002).

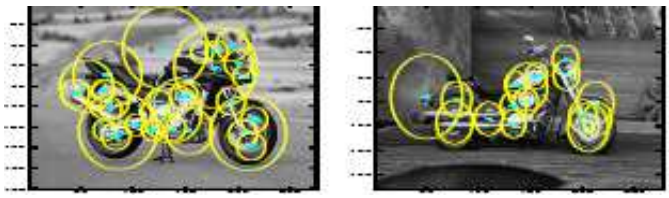


Figure 1.7 – Modélisation d'une moto par des régions construites autour des points saillants (Fergus et al., 2003).

Quelques systèmes ont été développés pour la segmentation d'images. Les mêmes approches que pour la détection d'objet peuvent être réutilisées pour segmenter l'image en objets et fond, en profitant du modèle d'objet pour faire des hypothèses sur la localisation de l'objet (Leibe and Schiele, 2003; Leibe et al., 2008) – la figure 1.8 schématise le processus de segmentation à partir de patchs d'image. D'autres systèmes prennent en charge la segmentation complète des images. Par exemple, Martin et al. (Martin et al., 2006; Martin, 2007) utilisent des exemples de résultats de segmentation construits « à la main » par l'utilisateur pour apprendre quel algorithme, et avec quel paramétrage, est le plus adapté à la segmentation des images de cette classe. La sélection de l'algorithme et de son paramétrage est faite à partir d'une mesure de distance entre le résultat obtenu sur une image test et l'image de référence pour cette image.

3 Vers un système à base de connaissances

3.1 Limites des approches numériques

Mais les approches purement numériques montrent aussi des limites intrinsèques et ne peuvent pas être la solution pour réaliser des systèmes généraux. Première-

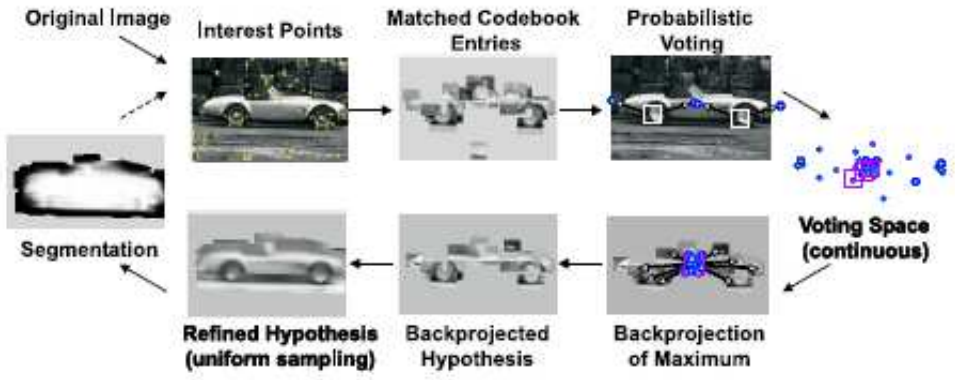


Figure 1.8 – Schéma de la procédure de segmentation selon (Leibe and Schiele, 2003).

ment, elles ne peuvent s'appliquer à tous les problèmes. Ainsi, les approches utilisant les points d'intérêt pour détecter les parties d'objet n'ont pas de sens dans le cas d'images de nature statistique (type image d'échographie). Deuxièmement, elles ne prennent pas pleinement en compte la sémantique de l'application. Une partie de la sémantique est déjà codée implicitement dans ces systèmes et n'est donc plus négociable par l'utilisateur. Elle est codée dans les fonctions de mesure de similarités entre patchs, ou dans les métriques de mesure entre l'image de référence et une image segmentée.

Notre point de vue est de considérer que, si on envisage la généralité, le choix de conception doit s'orienter vers un système à base de connaissances. Plus exactement, nous verrons dans nos perspectives que notre approche est d'utiliser un système hybride, c'est-à-dire un système à base de connaissances pour piloter des sous-systèmes numériques « basés image ». Les connaissances sont utilisées aux plus hauts niveaux pour choisir et configurer des sous-systèmes purement numériques qui vont ensuite être utilisés pour construire les résultats effectifs.

3.2 Autopsie de l'échec des approches à base de connaissances

Mais si les systèmes à base de connaissances sont légitimes, pourquoi n'ont-ils pas permis de réaliser des systèmes viables ?

La première erreur, c'est d'avoir cherché à construire des modèles de connaissances qui soient le reflet du savoir des experts consultés, à partir d'une verbalisation de leurs connaissances et de leur démarche de résolution (Bachimont, 1996). Or, les connaissances de traitement d'images sont tacites et peu verbalisables et la démarche de résolution est essentiellement empirique en l'absence d'une théorie formelle du domaine. L'hypothèse de l'existence d'une expertise rationnelle s'est donc révélée fausse et la tentative d'acquisition des connaissances s'est avérée vaine.

La deuxième erreur, c'est d'avoir cru à l'hypothèse du monde fermé impliqué par la modélisation des connaissances a priori (Draper et al., 1996). L'essentiel de l'effort de modélisation s'est porté sur la représentation des différentes expertises de résolution de problèmes qui ont pu être identifiées dans le domaine du traitement d'images parce que l'on a fait l'hypothèse que le domaine se caractérisait par ses solutions. La base de connaissances est donc composée de solutions toutes faites. La formulation

du problème se réduit alors à choisir une solution dont l'étiquette est libellée par le nom d'une tâche. L'utilisateur a l'impression de poser le problème mais en fait il choisit une solution prédéfinie. De plus, pour rendre ces solutions effectivement opérationnelles, bon nombre de connaissances a priori y sont implicitement codées, ce qui les a rendues peu réutilisables. Les systèmes produits se sont donc révélés peu adaptables parce que les solutions n'étaient pas construites dynamiquement pour le problème traité, mais prédéfinies et simplement ajustées avec les informations du contexte.

3.3 Une proposition de système

Si on veut concevoir un système à base de connaissances, il faut proposer de nouvelles solutions aux problèmes évoqués précédemment. Notre contribution porte sur quatre propositions principales :

- Accorder une place importante à la formulation. Pour cela, nous définissons un langage permettant une formulation précise des besoins de l'utilisateur.
- Construire une solution originale pour chaque application selon un processus incrémental.
- Inscrire la construction de la solution dans une interaction entre le système et l'utilisateur. Il s'agit d'ancrer la formulation dans les données image et de faire émerger la sémantique de l'application sur la base de résultats intermédiaires.
- Utiliser une approche constructiviste de l'acquisition des connaissances, qui soit basée sur une théorie du développement d'application, et plus sur un modèle cognitif des experts.

Notre hypothèse de base est de considérer qu'il existe un domaine du traitement d'images à part entière avec ses propres concepts, qui sont nécessaires et suffisants pour construire une solution. Les connaissances de résolution ne se définissent pas par rapport à un domaine d'application mais bien par rapport au domaine du traitement d'images ; ainsi, il n'y pas de technique de traitement d'images propres aux domaines biomédical ou astronomique, mais ces techniques sont réutilisables en l'état dans d'autres domaines. Cette hypothèse ne doit pas faire croire que nous allons retomber dans l'écueil du monde fermé. Les informations sur le domaine d'application traité sont nécessaires à la définition de l'objectif. Il est donc indispensable d'acquérir ces informations auprès de l'utilisateur pour chaque application. Mais nous prétendons que ces connaissances peuvent être intégralement représentées par des connaissances du traitement d'images, sans perte de sens pour le choix des traitements.

Le système que nous décrivons dans ce mémoire a pour but de générer automatiquement des programmes exécutables pour chaque application. Ces programmes sont alors capables de traiter toute image relevant de l'application. Les choix précédents donnent l'architecture globale du système (*cf.* figure 1.9).

Elle se compose d'une interface de formulation d'objectifs orientée vers l'utilisateur et d'un générateur de programmes qui opère à partir d'une formulation. La formulation d'objectifs se fait à l'aide d'un langage construit à partir d'une ontologie du domaine du traitement d'images. La génération de programmes utilise une bibliothèque d'opérateurs codés sous la forme de codes exécutables. Un programme est construit comme une chaîne d'opérateurs. La construction d'une application consiste à sélectionner les opérateurs, les paramétrer et les enchaîner pour construire une chaîne.

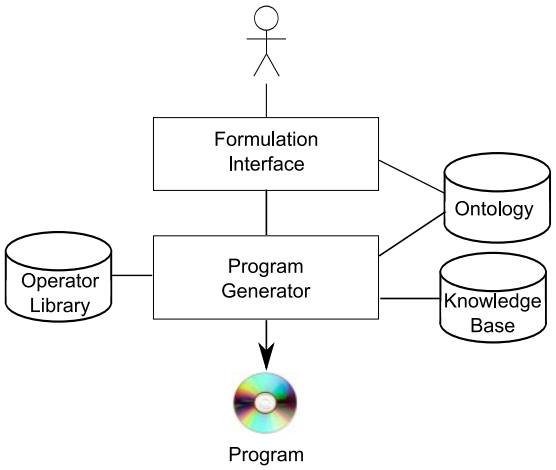


Figure 1.9 – L'architecture globale du système.

Les utilisateurs du système sont considérés comme des experts du traitement d'images dans le sens où ils doivent être capables de formuler un problème de traitement d'images avec les termes du traitement d'images, mais aussi de corriger leur formulation à partir de l'analyse de résultats produits par le système. Par contre, il n'est plus nécessaire qu'il soit informaticien.

Les chapitres suivants détaillent à la fois les problématiques soulevées et les solutions proposées pour chacune de ces parties.

CHAPITRE 2

Formulation d'objectifs de traitement d'images

Contrairement aux méthodes de conception de systèmes de traitement d'images adoptées dans le passé, qui ne s'intéressaient à la formulation des objectifs qu'une fois le modèle du système fixé, il apparaît clairement aujourd'hui que la formulation doit être étudiée très tôt, dans la mesure où elle sous-tend fortement le modèle de production des solutions. En effet, le modèle de formulation définit les seules informations sur le problème à traiter qui peuvent être utilisées pour produire les solutions.

Pour la conception de système à base de connaissances, nous affirmons ici que la formulation doit se faire à l'aide d'un langage capable de représenter la sémantique propre à chaque application, de rendre compte des nuances dans les besoins des utilisateurs et de coller à la réalité des données. Pour cela, le langage doit permettre une définition du problème à partir de descripteurs symboliques, c'est-à-dire de caractéristiques valuées, et de descripteurs iconiques, c'est-à-dire d'exemples pris dans des images. Ce langage est bâti sur une ontologie de domaine qui fixe les primitives et leur relations.

Les résultats présentés ici sont essentiellement issus des travaux de thèse d'Arnaud Renouf que j'ai encadrés entre 2004 et 2007. Son travail a permis d'élaborer un modèle de formulation général et une formalisation sous la forme d'une ontologie de domaine, qui fournit les primitives, munies d'une signification, pour le langage de formulation.

1 Introduction

La formulation d'objectifs est le moyen fourni aux utilisateurs par un système pour décrire les particularités de l'application qu'ils souhaitent construire. La formulation repose sur l'utilisation d'un langage qui permet de représenter les informations fournies par l'utilisateur sous une forme computationnelle. Ce langage définit les informations que l'utilisateur peut spécifier. La puissance d'expressivité de ce langage conditionne donc la largeur du champ d'application et les capacités d'adaptation du système. Tout ce qui ne peut être exprimé par le langage devra être codé implicitement dans le système et ne pourra donc être négocié par l'utilisateur.

Les informations collectées lors de la formulation sont des données essentielles sur l'application. Elles sont utilisées à tous les niveaux du développement du programme solution, que ce soit la conception de la solution, le contrôle de son exécution et l'évaluation de ses résultats :

Conception. Chaque opérateur de traitement d'images est conçu pour un domaine d'utilisation précis. La conception de la solution doit trouver dans la représentation des objectifs, les raisons qui légitiment le choix et le paramétrage des opérations des traitement d'images.

Contrôle. La formulation est aussi un moyen de capturer la variabilité des images à l'intérieur d'une classe. La construction du programme peut ainsi prévoir des mécanismes de contrôle permettant une adaptation automatique au domaine de variation identifié.

Évaluation. L'évaluation des résultats n'a de sens que rapportée à un objectif assigné (Zhang, 1996). L'évaluation quantitative des résultats ne peut se faire qu'en construisant ou en sélectionnant des métriques en référence à l'expression de l'objectif.

Ce chapitre est consacré à la présentation d'un modèle de représentation d'objectifs de traitement d'images. Ce modèle a pour but d'identifier les informations qui sont nécessaires et suffisantes à la construction d'une application fidèle aux intentions des utilisateurs. Une analyse de l'état de l'art de la formulation d'objectifs en traitement d'images dans la section 2, nous conduit à proposer un modèle qui combine une définition des informations sous forme linguistique et iconique. Nous montrons dans la section 3, que ce modèle préconise une spécification du but de l'application et une définition de la classe d'images à traiter. La spécification se fait par l'expression de la tâche à accomplir mais aussi par spécification à partir d'exemples de résultats à obtenir. La définition des informations se fait par une description linguistique des informations pertinentes, mais aussi par description à partir d'imagettes exemples. Ce modèle sert ensuite de support à l'élaboration d'une ontologie de domaine, qui est présentée dans la section 4. Cette ontologie devient le support du langage de représentation des objectifs qui présente l'avantage d'être à la fois opérationnel par un système et compréhensible par les experts de traitement d'images. Les capacités d'expressivité du langage sont mises en évidence dans la section 5, à travers différentes expérimentations visant à éprouver différents axes de la formulation.

2 La problématique de la formulation

2.1 Pourquoi une formulation des objectifs est-elle nécessaire ?

La question se pose de savoir pourquoi une formulation est nécessaire pour mener des traitements sur des images. Pourquoi ne peut-on construire un système qui se contente des images d'entrée pour construire les images de sortie. En quoi les données d'entrée sont-elles insuffisantes ?

Trois raisons expliquent le recours à une formulation. Premièrement, le type et la qualité des résultats à construire ne peuvent se déduire des images d'entrée parce que le traitement d'images n'est pas une fin en soi. C'est l'utilisation des résultats qui sera faite par la suite qui fixe le but de l'application.

Deuxièmement, les données constituant une image sont intrinsèquement incomplètes, dégradées et corrompues (Draper et al., 1996; Dalle, 2000; Clouard, 2004). Le processus d'acquisition est responsable de la création d'images qui sont des représentations sous-contraintes de la scène à analyser parce qu'il provoque la perte d'information – *p. ex.*, perte de la troisième dimension, perte du mouvement ou perte d'une partie de la scène par occlusion –, le mélange de plusieurs facteurs dans la

valeur d'un pixel – *p. ex.*, texture, illumination, géométrie –, l'introduction de fausses valeurs – *p. ex.*, bruit, aberration chromatique –, et l'altération de l'information originale – *p. ex.*, distorsion géométrique, flou.

Troisièmement, le contenu de l'image n'a pas de sens en lui-même. Une image est ambiguë par nature et ne fournit pas d'information sur son contenu. Sans sujet, une image ne permet pas de faire la distinction entre les informations pertinentes et non pertinentes. Ce qui peut être considéré comme pertinent pour une application peut être non pertinent pour une autre. De plus, et contrairement à une idée intuitive, il n'existe pas d'information qui soit intrinsèquement pertinente. Par exemple, une information apparemment aussi simple que le bord d'un objet est difficile à extraire de manière efficace sans connaissance sur la nature de la scène. Le bord d'un objet est souvent modélisé comme un changement brusque dans l'intensité de l'image, mais c'est aussi le cas pour le bruit, les ombres ou les éléments de texture.

En conséquence, les termes du problème correspondant à une application ne sont pas détenus par les images d'entrée seules et doivent donc être formulés en accompagnement des images.

2.2 Que doit contenir une formulation ?

Compte-tenu des informations précédemment identifiées comme absentes des images d'entrée, il est nécessaire de renseigner trois catégories d'information pour décrire un objectif de traitement d'images :

- L'expression de la *finalité* de l'application permet d'attribuer un rôle à l'application de traitement d'images dans la chaîne complète d'analyse d'images.
- La description du *processus d'acquisition d'images* permet de redonner les informations sur la scène qui ont été perdues, altérées, mélangées ou cachées lors de la production d'images.
- L'attribution d'un *sujet* aux images pour assigner une sémantique au contenu de la scène en désignant les informations à considérer comme pertinentes.

Le première catégorie d'information correspond à la spécification du but et les deux suivantes, à la définition de la classe des images d'entrée. Par exemple, considérons une application d'analyse d'images que nous avons développée dans le projet (Coudé, 1996). Cette application vise à l'automatisation de l'analyse diachronique des changements à long terme dans le paysage agricole de la région de Creully dans le Calvados. Un même secteur est comparé d'année en année, toujours à la même saison dans le but de quantifier l'évolution en surface des zones cultivées. La classe d'images est composée d'images aériennes couleur prises toujours avec la même résolution, dont un exemple est donné figure 2.1. L'objectif spécifique du traitement d'images est de segmenter les images dans le but d'isoler chaque zone de végétation dans une région. Ces régions seront ensuite introduites dans un classifieur qui a été entraîné à reconnaître les différentes catégories de zone de végétation : les champs, les forêts, les haies, etc.

On note, sur cet exemple, l'importance de la partie traitement d'images dans les performances du système d'analyse global. La localisation des régions cultivées ne peut se faire qu'à partir des régions extraites par le traitement d'images. De même, on remarque aisément que la délimitation des différentes zones de végétation ne peut pas toujours être uniquement obtenue à partir d'informations purement perceptuelles, c'est-à-dire sans connaissance sur la façon dont elles se manifestent. Par exemple, un champ de blé peut présenter plusieurs couleurs différentes pour son

intérieur si une partie des épis est couchée. Et pourtant, un expert y voit un seul et même champ. La définition d'une zone de végétation exige donc une expertise [1].



Figure 2.1 – L'objectif de traitement d'images de cette application d'imagerie aérienne est de segmenter l'image pour isoler chaque zone de végétation dans une région.

2.3 État de l'art de la formulation

Une analyse de différents systèmes fournissant aux utilisateurs une interface pour formuler des objectifs proposés dans la littérature, conduit à distinguer deux approches pour la représentation de la classe d'images et deux approches pour la représentation des buts.

2.3.1 Définition de classes d'images

La définition d'une classe d'images peut être faite soit par *extension*, par l'intermédiaire d'images exemples, soit par *intension* à travers une description linguistique.

Définition par extension. Les informations a priori sont représentées par des parties d'image exemples. Deux types de parties d'images peuvent être distingués : les *blobs* et les *patchs*.

– Un blob délimite une zone d'image qui désigne soit un objet d'intérêt – *p. ex.*, figure 2.2.a –, soit une partie d'image ciblée. Les blobs peuvent être dessinés manuellement ou obtenus par des outils de segmentation interactifs (Jeon et al., 2003; Bloehdorn et al., 2005). Ils sont ensuite utilisés pour extraire automatiquement des valeurs pour caractériser le concept désigné. Par exemple, les

1. Il est vrai que pour cet exemple nous sommes tous un peu experts parce que nous avons déjà vu des images aériennes de zones rurales.

caractéristiques de couleur, de forme et de taille peuvent être extraites de blobs correspondant à un objet d'intérêt, ou celle du bruit extraites d'une zone d'image supposée homogène dans la scène.
– Un patch d'image est une imagette extraite d'une image exemple qui isole une partie prégnante d'un objet d'intérêt. Souvent, ces patchs sont extraits automatiquement autour des points d'intérêt (Leibe and Schiele, 2003; Agarwal et al., 2004) ou de points saillants (Fergus et al., 2003). Ainsi, un objet est décrit par un ensemble de patchs d'images repérés spatialement – *p. ex.*, figure 2.2b.

Définition par intension. L'information a priori est représentée par une description linguistique. Le langage de description est généralement construit à partir d'une *ontologie de domaine* qui fournit les primitives du langage. La description d'une classe d'images particulière est une *ontologie d'application* qui est faite par sélection et réification de primitives de l'ontologie du domaine (Cãmara et al., 2001). Par exemple, N. Maillot et al. (Maillot et al., 2004; Maillot, 2005; Maillot and Thonnat, 2008) propose « l'Ontologie des Concepts Visuels » qui définit les concepts de texture, couleur, géométrie, et de relations topologiques. La figure 2.3 donne un exemple de la description d'un grain de pollen de graminée avec cette ontologie. Pour mieux rendre compte de la variabilité des manifestations visuelles des objets dans la scène, le langage accepte des valeurs qualitatives à partir de variables linguistiques, aussi bien pour les caractéristiques – *p. ex.*, "pink", "very circular","slighly oblong" (Mezaris et al., 2004) – que les relations spatiales – *p. ex.*, "in front of", "close to" (Hudelot et al., 2008).

Mais la construction de la solution nécessite des valeurs quantitatives. De ce fait, la définition intensionnelle doit aborder le problème de l'enracinement des symboles [2] (angl. *symbol anchoring problem*) dans le but de connecter les symboles linguistiques avec les valeurs de données image. L'enracinement des symboles se définit ainsi :

> « Symbol anchoring is the process of creating and maintaining the correspondence between symbols and sensor data that refer to the same physical objects. » (Coradeschi and Saffiotti, 2003)

L'enracinement des symboles peut être mené en utilisant des dictionnaires tels que le « Color Naming System » (Berk et al., 1982), où l'espace HSL est divisé en 627 couleurs distinctes ; ce principe pouvant être étendu au cas des textures avec le dictionnaire « Texture Naming System dictionary » (Rao and Lohse, 1993). Mais plus souvent, l'enracinement des symboles est appréhendé comme un problème d'apprentissage à partir d'une base de blobs (Li et al., 2007; Maillot and Thonnat, 2008; Hudelot et al., 2008).

Chacune de ces deux approches présente des avantages et des inconvénients. L'avantage d'une définition par extension est de minimiser la quantité d'information a priori à renseigner. Elle réduit la charge cognitive des utilisateurs du système puisque la formulation ne nécessite aucun langage de représentation ; même si la formulation peut s'avérer fastidieuse lorsque le nombre d'images à fournir est élevé. L'inconvénient est que la définition effective d'une classe d'images à partir d'exemples est produite par le système seul, à partir des caractéristiques qui seront extraites des images (que cette liste soit prédéfinie ou obtenue par sélection automatique). Cela signifie qu'une part de la définition est assignée par le système et qu'elle n'est pas maîtrisable par l'utilisateur.

2. Nous avons choisi de traduire « symbol anchoring » par « enracinement des symboles », et de traduire « symbol grounding », que nous verrons au chapitre 4, par « ancrage des symboles ».



Figure 2.2 – Deux façons différentes de définir un objet d'intérêt par extension : (a) par blob (b) par patchs d'image.

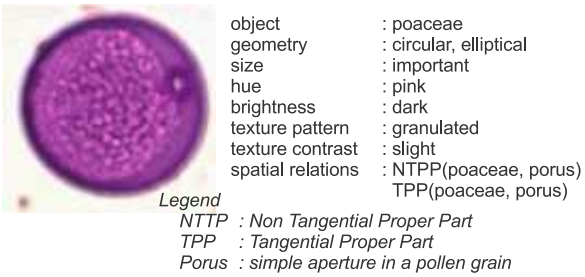


Figure 2.3 – Exemple de la définition d'un grain de pollen de type « poaceae » faite à partir de « l'Ontologie de Concepts Visuels » proposée par Maillot et al. (Maillot and Thonnat, 2008).

L'avantage d'une définition par intension est de mieux refléter l'expertise de l'utilisateur sur la scène. Elle fournit un langage capable de représenter la sémantique de la scène et permet ainsi de mieux capturer la variabilité des images d'entrée. De plus, cette approche est utilisable même pour les séquences d'images (Dasiopoulou et al., 2005). Toutefois, l'inconvénient est que la construction d'une description linguistique est connue pour être difficile (Smeulders et al., 2000; Vernon, 2007).

2.3.2 Spécification de buts

Un but de traitement d'images peut être formulé soit par l'expression de ce que l'on veut faire, sous la forme de *tâches*, soit par ce que l'on veut obtenir, à partir d'*exemples*.

Spécification par tâche. Une tâche décrit une fonctionnalité du système. Il est possible d'associer des contraintes à la tâche pour préciser sa portée. Par exemple, une requête pour le système MVP est « *radiometric correction* » (Chien and Mortensen, 1996) et une requête pour le système LLVE est « *find a rectangle whose area size is between 100 and 200 pixels* » (Matsuyama, 1989). Cette approche nécessite un langage de représentation de l'objectif plus ou moins évolué qui peut être appréhendé par une interface graphique.

Spécification par l'exemple. Un objectif est formulé par l'intermédiaire d'une ou plusieurs images de référence qui contiennent la représentation des résultats à ob-

tenir pour des images tests. Ces exemples sont utilisés par le système pour sélectionner un algorithme prédéfini ou construire une chaîne d'algorithmes qui produit des résultats proches des images de référence. Trois représentations différentes des résultats ont été proposées dans la littérature :

- Les images de référence contiennent des *sketchs* qui sont des tracés faits par l'utilisateur sur des images tests qui donnent des exemples de contours (Hasegawa et al., 1986) ou de régions (Draper et al., 1999) attendus en sortie – *p. ex.*, figure 2.4.a.
- Les images de référence peuvent être définies à partir de *segmentations manuelles* qui donnent les résultats exacts à obtenir pour les images tests (Martin et al., 2006) – *p. ex.*, figure 2.4.b.
- Les images de référence correspondent à des gribouillages qui pointent les régions d'intérêt sans les détourer complètement (Levin et al., 2004; Protière and Sapiro, 2007). Généralement, ces gribouillages sont des contours fermés dessinés directement sur les régions d'intérêt et sur la région du fond complémentaire des objets – *p. ex.*, figure 2.4.c.



Figure 2.4 – Différentes approches pour la spécification d'objectif par l'exemple : (a) sketch (b) segmentation manuelle (c) gribouillages.

Ici encore, chacune de ces deux approches possède son intérêt et son inconvénient. La spécification par tâche présente l'avantage de couvrir tous les objectifs de traitement d'images (Clouard et al., 1999b) : il suffit pour cela de définir le nom d'une tâche. De plus, les tâches permettent de prendre en compte les exigences particulières des utilisateurs à travers des contraintes associées. Cependant, l'inconvénient est que la formulation est qualitative, sans réel lien avec les données image. Cela a deux conséquences importantes : premièrement, la spécification par tâche n'est pas assez explicite, et deuxièmement, il n'existe qu'un nombre fini de formulations d'objectifs différentes. L'avantage de la spécification par l'exemple est que cette formulation est par nature quantitative puisqu'elle prend ses valeurs directement dans les données image. En conséquence, elle prend en compte la variété des formulations d'objectifs. De plus, elle réduit la charge cognitive des utilisateurs parce qu'elle ne nécessite aucun vocabulaire spécialisé. L'inconvénient de cette seconde approche est qu'une image de référence n'est pas suffisante pour formuler toutes les formes d'objectifs de traitement d'images pour au moins trois raisons :

- Cette approche ne s'applique réellement que pour des buts de segmentation, de détection d'images et éventuellement d'amélioration. Les buts de compression, de restauration ou de reconstruction sont plus difficilement couverts.
- Elle ne couvre pas toutes les classes d'images. En particulier, elle semble difficile à mettre en œuvre pour les images 3D et les séquences d'images (bien que l'approche par sketch a été étendue pour une application biomédicale particulière

au cas d'images 3D (Zhou et al., 2001)).
- Elle ne permet pas de varier les contraintes spécifiques attachées à l'objectif telles que « *préférer les fausses détections aux oublis* » ou « *préférer aucun résultat à des résultats non parfaitement satisfaisants* ». Ces contraintes sont en effet imposées implicitement par le système, par exemple dans les métriques de calcul de distance entre un résultat et une référence.

2.3.3 Proposition de formulation

Après avoir analysé les avantages et les inconvénients de chacune des approches pour la spécification de buts et pour la définition de la classe d'images, il apparaît clairement que la définition d'un modèle de formulation complet et générique doit intégrer chacune d'elles. La spécification d'objectifs doit être dirigée par la tâche à accomplir et ancrée dans les données par une spécification par l'exemple. La définition d'une classe d'images doit combiner une définition par intension pour capturer la sémantique de la scène avec une définition par extension qui permet de se rapprocher de la réalité des données.

3 Un modèle de formulation

Dans cette section, nous cherchons à identifier les catégories d'information qui sont nécessaires et suffisantes pour concevoir et évaluer les applications de traitement d'images, et à proposer une représentation de ces informations sous forme computationnelle. Un modèle est développé ; il fournit une conceptualisation de la formulation à partir de ces éléments d'information.

3.1 Définition de classes d'images

Le modèle de définition de classe d'images que nous proposons est basé sur trois hypothèses fortes : les hypothèses phénoménologique, sémiotique et de sémantique différentielle. Comme toute hypothèse, elles sont évidemment simplificatrices, mais nécessaires pour réduire la dimension du problème de la définition et rendre possible la construction d'un modèle computationnel.

3.1.1 L'hypothèse phénoménologique

Parmi les informations qui peuvent être utilisées pour définir une classe d'images, nous arguons que seules les *informations phénoménologiques* sont nécessaires et suffisantes (Renouf et al., 2007b). Les informations phénoménologiques reflètent la manifestation visuelle de la scène. Sous cette hypothèse, le but de la formulation n'est pas de décrire la scène dans sa réalité ontologique mais uniquement par la façon dont elle est perçue à travers les images.

Encore une fois, cette hypothèse est simplificatrice. Le traitement d'images ne se réduit pas complètement à cette hypothèse phénoménologique. Certaines applications requièrent la modélisation d'informations ontologiques qui n'ont pas de manifestation visuelle directe dans les images. Par exemple, la connaissance de l'influence de l'activité métabolique du cerveau sur les aspects instrumentaux permettent de construire des modèles pour guider la segmentation d'images d'IRM fonctionnelles dont le rapport signal sur bruit est extrêmement faible (Fadili, 1999). Mais, ces

connaissances et leur représentation sont strictement dépendantes des domaines d'application et n'ont pas encore de modélisation générale qui permettrait de les intégrer dans une définition de classes d'images générique. Pour ces cas, il est encore nécessaire de développer des logiciels dédiés.

L'hypothèse phénoménologique présente l'avantage de réduire la définition de la classe d'images à une dénotation faite à partir d'indices visuels. Il n'est donc pas nécessaire de représenter les connaissances sur le domaine d'application. Par exemple, l'objet réel « bus » dans l'image aérienne donnée figure 2.5a peut se réduire d'un point de vue phénoménologique à un simple rectangle blanc uniforme dans le contexte de la segmentation de bus à partir d'images aériennes.

Par contre, l'hypothèse présente l'inconvénient de pouvoir nécessiter la définition de plusieurs objets d'intérêt pour un même objet réel, s'il présente des apparences visuelles différentes. Par exemple, si l'on considère un objectif de détection de bouteille, les deux positions différentes de la bouteille données figure 2.5.b et figure 2.5.c peuvent impliquer de considérer deux objets d'intérêt différents s'il n'y a pas de « ressemblance » entre ces deux représentations.

Cette hypothèse plaide pour l'existence d'un noyau d'information spécifique du traitement d'images qui est indépendant de tout domaine d'application mais à partir duquel on peut représenter les informations d'un domaine d'application.



Figure 2.5 – Illustrations de l'hypothèse phénoménologique. (a) Dans cette image aérienne de parking, le concept de bus peut se réduire à un simple rectangle blanc homogène. La différence d'apparence d'une même bouteille vue dans deux position différentes (b) et (c) conduit à distinguer deux objets d'intérêt différents. *Les images b et c sont tirées de la base Columbia Object Image Library (COIL-100).*

3.1.2 L'hypothèse sémiotique

Si on considère les images comme un système de signes mis pour représenter un chose réelle ou artificielle (Joly, 1994), l'analyse sémiotique peut fournir les bases théoriques pour modéliser une classe d'images. Cette approche suggère de distinguer trois niveaux dans l'analyse d'une classe d'images (Renouf, 2007) : (1) le niveau physique, (2) le niveau perceptif et (3) le niveau sémantique (*cf.* figure 2.6).

Le niveau physique. C'est le niveau du signal mesuré qui supporte la représentation de l'image. L'hypothèse phénoménologique conduit à décrire ce niveau par la liste des effets produits sur le signal par la chaîne d'acquisition – *p. ex.*, bruit, distorsion géométrique, illumination –, plutôt que par la liste des éléments de la

chaîne d'acquisition – *p. ex.*, caméra, lentille. Par exemple, savoir que le capteur est une caméra CDD n'est pas une information directement exploitable pour traiter une image. Par contre, savoir que cette caméra produit un bruit de type gaussien est une information directement exploitable. De plus, l'information « caméra CDD » n'est pas une information universelle parce que les effets de la caméra peuvent évoluer dans le temps et différer selon les caméras. Pourtant, ces informations se retrouvent dans plusieurs systèmes, par exemple, Frucci et al. (Frucci et al., 2008) les appellent « nonimage information ». Donc, nous considérons que « bruit » est un concept à représenter au niveau physique mais pas « type de capteur ».

Le niveau perceptif. Ce niveau s'intéresse au rendu visuel du contenu de l'image sans référence directe aux objets d'intérêt présents dans la scène. C'est une définition purement « syntaxique » faite à partir de primitives visuelles telles que les régions, les contours ou les points d'intérêt. La définition à ce niveau correspond à la liste des caractéristiques des différents types de contours, régions, ou points d'intérêt considérés comme pertinents pour l'application.

Le niveau sémantique. Ce niveau est focalisé sur les *objets d'intérêt*. La notion d'objet d'intérêt est à prendre d'un point de vue phénoménologique, c'est-à-dire basée sur l'apparence visuelle. En conséquence, un objet d'intérêt ne correspond pas forcément à un objet de la scène, mais seulement à une partie d'un objet ou au contraire à un ensemble d'objets de la scène. La sémantique de la scène est composée par la dénotation des informations qui doivent être considérées comme pertinentes pour l'identification des différents objets d'intérêt.

Ces informations se réfèrent à la description visuelle individuelle des objets et à la description des relations spatiales. Cela signifie que les objets peuvent être identifiés par leurs caractéristiques intrinsèques ou par leurs relations aux autres objets.

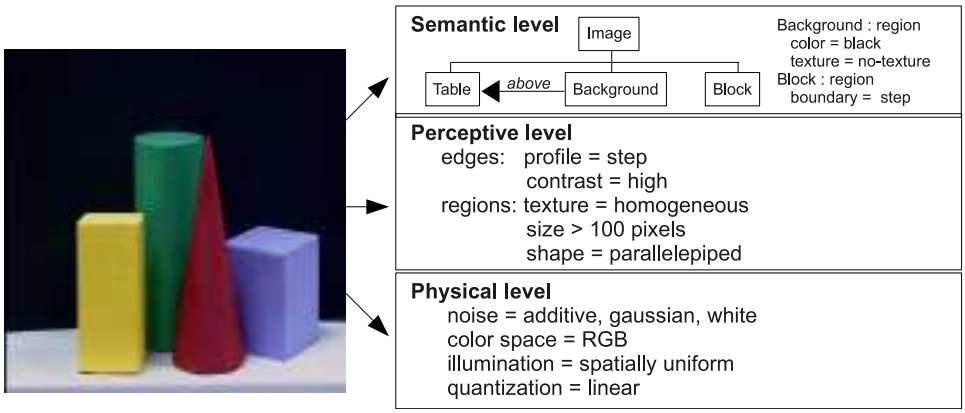


Figure 2.6 – Selon l'approche sémiotique, une image est analysée en trois niveaux (1) physique, (2) perceptif et (3) sémantique.

Il est à noter que ces trois niveaux se retrouvent dans ceux définis par J. Van Den Elst pour la formulation d'objectifs dans le système OCAPI (Van Den Elst, 1996) et par F. Aubry pour l'archivage et la consultation d'images biomédicales dans le système MIMOS (Aubry and Todd-Pokropek, 2001).

3.1.3 L'hypothèse de la sémantique « différentielle »

Pour définir la sémantique d'une scène, nous faisons le choix d'une identification de chaque objet d'intérêt qui la compose, par l'expression de ses différences avec le « fond d'image » et tous les autres objets de la scène. Le fond d'image correspond à tout ce qui n'est pas labellisé comme objet d'intérêt dans la scène. Ce que l'on va chercher à construire, c'est une dénotation discriminante des objets d'intérêt, représentée par une liste de caractéristiques visuelles et de relations spatiales aux autres objets, eux-mêmes discriminables. C'est l'ensemble des éléments de dénotation qui doit être discriminant, et pas nécessairement chaque élément lui-même. Par exemple, la définition « un champ agricole est une région de grande surface avec un faible contraste intérieur et des frontières marquées » semble suffisante pour discriminer les champs par rapport à tous les autres objets géographiques, de même que « les haies sont les régions les plus sombres ».

L'hypothèse écarte toute définition d'objet par différences avec seulement un sous-ensemble d'objets. Par exemple, une définition telle que « un champ est une région plus claire qu'une haie » n'est pas acceptée ; il est nécessaire d'ajouter d'autres éléments de description pour rendre cette description discriminante par rapport à tous les autres objets.

Nous estimons qu'une définition discriminante est suffisante pour construire les traitements et donc qu'il n'est pas nécessaire d'ajouter d'autres expressions plus complexes de la sémantique.

3.1.4 Modèle conceptuel de la définition de classes d'images

Le modèle conceptuel de la définition de classes d'images résultant est donné par le diagramme UML de la figure 2.7. Une classe d'images est définie sur trois niveaux : physique, perceptif et sémantique. Chaque niveau fournie sa propre liste d'éléments de description (*i.e.*, Description Element) dont la description peut être faite par une liste de descripteurs propres (*i.e.*, Descriptor) ou d'images (*i.e.*, Blob).

En fonction des connaissances disponibles sur le problème, la définition de la classe d'images est plus ou moins importante à chaque niveau. En particulier, le niveau perceptif est décrit en l'absence d'information au niveau sémantique puisque la description sémantique est plus informative que la description perceptive. Par exemple, considérons trois applications distinctes :

- Dans le cas de l'application d'imagerie aérienne, une grande quantité d'information est décrite aux niveaux physique et sémantique puisque le système d'acquisition et la scène sont bien connus et les objets sont prédictibles et descriptibles.
- Dans le cas d'une application de recherche d'images par le contenu, peu d'informations sur la chaîne d'acquisition sont connues (en général assimilée à une caméra CDD génératrice d'un bruit gaussien faible) et les objets sont imprédictibles. La définition de la classe d'images est donc limitée à une description au niveau perceptif, souvent en termes de texture ou de forme de régions (Ceccarelli et al., 2006).
- Dans le cas d'une application de robotique, les ingénieurs ont la maîtrise de la chaîne d'acquisition mais les objets sont imprédictibles ou trop variés. De ce fait, ce type d'application est principalement formulé aux niveaux physique et perceptif en termes de contours et de régions.

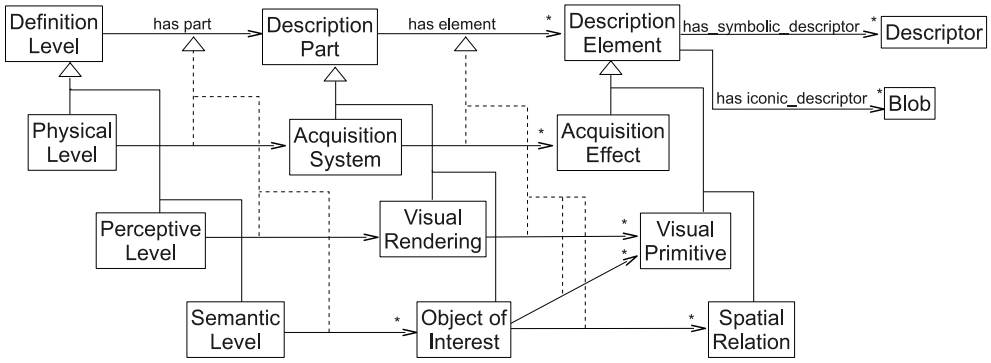


Figure 2.7 – Diagramme UML du modèle conceptuel de la définition d'une classe d'images.

3.1.5 Valeurs de descripteur

Les descripteurs doivent rendre compte du fait que certains éléments de la formulation sont avantageusement décrits à l'aide d'un vocabulaire linguistique (à base de descripteurs à valeurs numériques et symboliques) et d'autres à l'aide d'un vocabulaire iconique (à base de descripteurs à valeurs image).

Parmi les valeurs numériques, nous distinguons les valeurs *simples*, les *intervalles*, les *ensembles*, les *vecteurs* et les *matrices* de valeurs – *p. ex.*, une matrice de correction des distorsions géométriques. Les intervalles, en particulier, sont un moyen de capturer la variabilité des valeurs pour les caractéristiques.

Les valeurs symboliques sont soit des valeurs conventionnelles – *p. ex.*, colorspace : {RGB, HSL, YUV} –, des termes qualitatifs – *p. ex.*, un champ a une grande taille – et des termes superlatifs – *p. ex.*, un champ a la plus grande taille. Conformément à notre hypothèse de sémantique différentielle, nous ne représentons pas les termes comparatifs tels que « plus petit que » ou « plus clair que ». Pour les termes qualitatifs, nous définissons en particulier une variable linguistique *level* qui prend ses valeurs dans une échelle à 6 niveaux[3] : [null, very-low, low, medium, high, very high]. Les valeurs superlatives sont au nombre de deux : [the least, the most].

Les valeurs iconiques sont représentées par des images entières ou par des blobs. Dans notre modèle, les patchs ne sont pas considérés parce qu'ils ne représentent pas une information intelligible facilement manipulable par l'utilisateur pour spécifier ses objectifs. Les patchs constituent plutôt une information de nature calculatoire, extraite et utilisée directement par des algorithmes de traitement dédiés.

3. Une telle échelle de mesure qualitative à 5, 6 ou 7 niveaux se retrouve dans beaucoup de représentation des connaissances, sans réelle justification théorique. Toutefois, George Miller dans son célèbre article « The Magic Number Seven, Plus or Minus Two : Some Limits on our Capacity for Processing Information » (Miller, 1956) en donne certainement une justification expérimentale. En effet, il montré que la quantité d'information discernable par un humain pour mesurer empiriquement un stimulus unidimensionnel (*p. ex.*, la sensation de salé, le volume d'un son, etc.) n'est que d'environ 2,5 à 3 bits, soit 6 à 7 niveaux différents. Au delà, il y a introduction d'erreurs d'appréciation. Une échelle à 6 niveaux est donc une échelle à 5 niveaux graduels, qui présente l'intérêt d'avoir un niveau moyen facilement identifiable, plus une valeur nulle qui représente le 0.

3.2 Spécification de buts

3.2.1 Une spécification combinant tâche et exemples

Compte-tenu du bilan tiré de l'état de l'art (*cf.* § 2.3.3), nous avons choisi de spécifier le but de l'application par une tâche avec un réseau de contraintes associées et des images de référence optionnelles. La tâche est utilisée pour sa capacité à prendre en compte assez précisément les exigences des utilisateurs et les images de référence sont utilisées pour leur capacité à ancrer le problème dans les données. Quatre types de contraintes peuvent être associées à une tâche :

- Les *critères à optimiser* identifient les éléments sur lesquels la tâche doit se focaliser. Des exemples de critères sont : « *maximiser la localisation des frontières des objets* » ou « *maximiser le taux de détection* ».
- Les *niveaux de détail* déterminent les limites hautes et basses de la portée de la tâche. Des exemples pour une tâche de segmentation sont « *séparer les objets qui se touchent* » ou « *mettre les frontières à l'intérieur des régions* ».
- Les *critères de performance* spécifient les ressources autorisées pour l'exécution du programme d'application essentiellement en termes de temps d'exécution.
- Les *critères de qualité* expriment des exigences sur la capacité du système à accomplir l'objectif fixé dans les conditions prévues et les conditions non prévues. Les exemples de critère de qualité sont la *fiabilité* et la *robustesse*.

Dans le but de décider des compromis en cas de doute sur la façon de respecter une contrainte, chaque critère à optimiser et chaque niveau de détail accepte des *erreurs acceptables*. Par exemple, s'il y a doute sur le fait que deux objets se chevauchent ou se touchent, alors une erreur acceptable peut indiquer de préférer la séparation. La conséquence sera, peut-être, de choisir des traitements de type ascendant qui favorisent la sur-segmentation, et donc la séparation, à des traitements de type descendant qui favorisent au contraire la sous-segmentation.

Les images de référence sont utilisées pour compléter la spécification par tâche en apportant des exemples de résultats attendus. Ces références peuvent être des représentations de résultat complet, *p. ex.*, des images segmentées, ou des échantillons de résultat, *p. ex.*, des exemples de contours ou de régions. Ces images peuvent alors être utilisées soit pour choisir les traitements, soit pour évaluer les résultats finaux.

3.2.2 Modèle conceptuel de la spécification de but

Le modèle conceptuel de la spécification de but est résumé par le diagramme UML de la figure 2.8. On retrouve les quatre parties de la composition d'un but : la tâche, les critères à optimiser, les niveaux de détail et les contraintes de contrôle.

4 Une ontologie pour la formulation d'objectifs de traitement d'images

Notre langage de formulation d'objectifs est défini sur une ontologie de domaine de type terminologique. Une telle ontologie est destinée à spécifier les termes utilisés pour représenter les connaissances du domaine et fixer les contraintes sur la structure et le contenu de ces connaissances (*i.e.*, la grammaire et le vocbulaire) (Van Heijst et al., 1997). L'intérêt des ontologies pour la formulation d'objectifs n'est plus à démontrer, en effet :

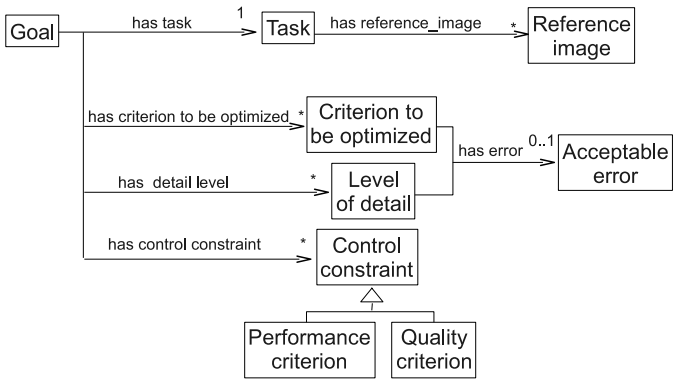


Figure 2.8 – Diagramme UML du modèle conceptuel de la spécification de but de traitement d'images.

> « an ontological query language provides a way of narrowing the semantic gap between users and the system by providing a shared language and hierarchy of concepts for both. » (Town, 2006)

Les ontologies ont été largement utilisées pour concevoir et implémenter l'interprétation haut niveau de scènes (Mezaris et al., 2003; Town, 2006; Neumann and Möller, 2008), mais peu spécifiquement pour le traitement bas niveau, et aucune qui ne couvre tous les objectifs de traitement d'images. C'est pourquoi nous avons développé notre propre ontologie. Dans cette section, nous décrivons une ontologie qui fournit les primitives munies de leur signification pour définir un langage capable de représenter des objectifs de traitement d'images (Renouf et al., 2005; Renouf et al., 2007b; Renouf et al., 2007c). Cette ontologie est un graphe de concepts computationnels organisés sur la base des deux modèles conceptuels présentés dans la section précédente.

4.1 Concepts de niveau physique

Les concepts au niveau physique correspondent aux effets possibles de la chaîne d'acquisition sur la représentation des images. En conséquence, l'analyse des différents éléments pouvant composer une chaîne d'acquisition : éclairage, environnement, système optique, capteur, convertisseur numérique-analogique et stockage, donne la liste de ces effets (*cf.* figure 2.9). Par exemple, un capteur optique peut générer des défauts d'illumination, de géométrie ou de flou sur les images.

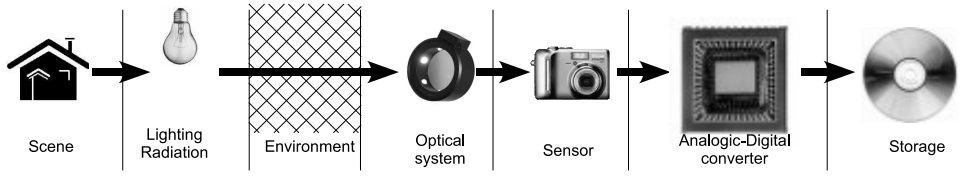


Figure 2.9 – L'analyse des effets générés par les différents composants possibles d'une chaîne d'acquisition fournie les concepts du niveau physique pour l'ontologie.

Nous identifions neuf catégories de concepts (*cf.* figure 2.10) : flou, bruit, co-

lorimétrie, illumination, géométrie, photométrie, échantillonnage, quantification et stockage. Chaque concept est décrit, soit par une liste de descripteurs symboliques comme cela est détaillé dans le tableau 2.1, soit par des blobs ou des images entières. Pour une formulation particulière, seules les primitives qui ont une réelle manifestation dans les images d'entrée doivent être fournies. Plusieurs instances d'une même catégorie peuvent être définies, par exemple, quand différents types de bruit sont présents dans les images.

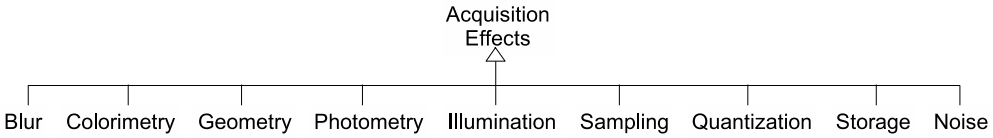


Figure 2.10 – Les concepts du niveau physique correspondent aux effets de la chaîne d'acquisition sur les images d'entrée.

Tableau 2.1 – Liste des concepts de niveau physique et les descripteurs associés.

Blur
Model : *Blob, Image*
Direction : *Numeric* [deg|rad]
Length : *Numeric* [pixel]
Strength : *Level*
Colorimetry
Model : *Blob, Image*
Defect : {Bayer effect, chromatic aberration}
Hue dynamics : *Numeric, Level*
Saturation : *Numeric, Level*
Colorspace : {RGB, HSL, LUV, YUV, gray, binary, multispectral}
Illumination
Model : *Blob, Image*
Illumination spatial : {heterogeneous, homogeneous}
Illumination temporal : {stable, varying}
Illumination defect : {saturation, lag, shift, blooming, smearing, flicker}
Photometry
Model : *Blob, Image*
Global contrast : *Numeric, Level*
Dynamics : *Numeric, Level*
Brightness : *Numeric, Level*
Quantization
Bit per pixel : *Numeric*
Function : {linear, logarithmic}

Geometry
Model : *File*
Defect : {astigmatism, coma, geometric distortion, spherical aberration}
Sampling
Defect : {aliasing, moiré, partial volume effect}
Spatial x-resolution : *Numeric* [%]
Spatial y-resolution : *Numeric* [%]
Spatial z-resolution : *Numeric* [%]
Storage
Defect : {block effect}
Number of bands : *Numeric*
Number of looks : *Numeric*
Type : {synthetic, statistical, iconic}
Noise
Model : *Blob, Image*
Composition : {additive, multiplicative, mixed}
Distribution : {exponential, gaussian, uniform, poisson, rayleigh, impulse}
Power Spectral Density : {white, colored. . . }
Signal Noise Ratio : *Numeric, Level*
Stationarity : {yes, no}
First order : *Numeric, Level*
Second order : *Numeric, Level*
Third order : *Numeric, Level*
Fourth order : *Numeric, Level*

4.2 Concepts de niveau perceptif

Les concepts de niveau perceptif sont les six primitives visuelles (figure 2.11) : région, contour, fond d'image, point d'intérêt, zone d'image et nuage de points. Chaque

primitive visuelle est décrite par des descripteurs symboliques détaillés dans les tableaux 2.2 et 2.3, ou par des blobs.

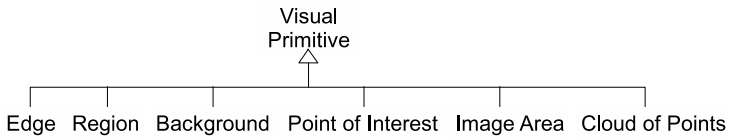


Figure 2.11 – Les concepts du niveau perceptif correspondent aux primitives visuelles.

Tableau 2.2 – Liste des concepts de niveau perceptif et les descripteurs associés.

Visual Primitive	Descriptor Category
Region	Model, Boundary, Photometry, Colorimetry, Texture, Region-Morphology, Orientation, Position, Region-Size, Topology
Edge	Model, Photometry, Colorimetry, Edge-Morphology, Orientation, Position, Edge-Size
Background	Model, Photometry, Colorimetry, Texture
Point of Interest	Model, Morphology, Position, Photometry, Colorimetry
Image Area	Model, Photometry, Colorimetry, Texture, Region-Morphology, Orientation, Position, Region-Size
Cloud of Points	Model, Photometry, Colorimetry, Region-Morphology, Orientation, Position, Region-Size

On peut noter que la représentation de certaines primitives visuelles peut sembler simpliste par rapport aux caractérisations que l'on peut trouver dans la littérature pour ces mêmes primitives. En fait, nous ne gardons ici que les descripteurs que nous avons jugés utiles pour choisir, paramétrer et évaluer les algorithmes de traitement d'images et non ceux qui sont utiles à la construction de ces algorithmes. Par exemple, les caractéristiques de texture, telles que les ondelettes ou les descripteurs d'Haralick ne sont pas utiles pour sélectionner un algorithme de segmentation d'images basé texture. Nous considérons que seules les caractéristiques d'échelle (micro or macro), de type – *p. ex.*, périodique, complexe, point – et d'orientation sont utiles. Les autres caractéristiques pourront être calculées à l'intérieur des algorithmes qui auront été sélectionnés.

Le même raisonnement peut être tenu pour les caractéristiques de couleurs puisqu'il existe beaucoup d'espaces couleur différents. Seul l'espace HSL à été retenu parce qu'il a un sens pour les utilisateurs. D'autres espaces ou combinaisons d'espaces pourront être utilisés dans les algorithmes sélectionnés à partir des caractéristiques couleur données dans l'espace HSL (voir pour cela les travaux portant sur le calcul automatique de l'espace couleur le plus adapté à une tâche de traitement d'images (Vandenbroucke et al., 2003)).

4.3 Concepts de niveau sémantique

Au niveau sémantique, les concepts décrivent les objets d'intérêt individuellement et spécifient leurs relations spatiales. L'ensemble des objets d'intérêt est d'abord organisé dans un *arbre des objets* (Liedtke et al., 1997) en fonction de la relation méronymique « partie-de ». Des objets plus abstraits peuvent aussi être définis pour

Tableau 2.3 – Liste des catégories de descripteurs perceptifs.

Model
Model :*Blob*
Edge-Morphology
Contrast : *Numeric, Level*
Shape : {curve, straight line}
Status : {open, close, loop}
Profile : {roof, ridge, step}
Straightness : *Numeric, Level* [pixel]
Boundary
Nature : { edge, limit of texture, limit of homogeneous regions}
Reference : ***cf.* the visual primitive description : edge**
Photometry
Brightness : *Level, Numeric, Image*
Contrast : *Level, Numeric, Image*
Model : *Patch, Image*
Colorimetry
Hue : *Numeric, Level*
Saturation : *Numeric, Level*
Lightness : *Numeric, Level*
Texture
Direction : *Numeric* [deg|rad], {horizontal, vertical}
Scale : {macro, micro}
Type : {no-texture, contour, dot, complex, periodic}
Topology
Number of holes : *Numeric*

Orientation
Major Axis Angle : *Numeric* [deg|rad], {vertical, horizontal}
Position
Center of mass : *Coordinate*
Edge-Size
Length : *Numeric* [pixel], *Level*
Thickness : *Numeric* [pixel], *Level*
Region-Size
Area : *Numeric*[pixel^2], *Level*
Volume : *Numeric*[pixel^3], *Level*
Bounding Box : *Numeric*[pixel^2|pixel^3], *Level*
Diameter : *Numeric* [pixel], *Level*
Thickness : *Numeric* [pixel], *Level*
Net perimeter : *Numeric* [pixel], *Level*
Convex perimeter : *Numeric* [pixel], *Level*
Region-Morphology
Shape : {square, rectangular, circle, ellipsoid, parallelepiped, cubic, sphere}
Compactness : *Numeric*, *Level*
Convexity : *Numeric*, *Level*
Elongation : *Numeric*, *Level*
Number of angles : *Numeric*
Rectangularity : *Numeric*, *Level*
Roundness : *Numeric*, *Level*
PointOfInterest-Morphology
Junction type : {L, T, X, Y}

factoriser des caractéristiques communes entre objet d'intérêt grâce à la relation hyperonymique « sorte-de ».

Puis la définition d'un objet est faite avec les mêmes six primitives visuelles identifiées au niveau perceptif (*cf.* les tableaux 2.2 et 2.3). Parmi les relations spatiales existantes (Retz-Schmidt, 1988), seules les relations topologiques – *p. ex.*, RCC-8 model (Cohn et al., 1997) pour les images 2D – et les relations spatiales extrinsèques – *p. ex.*, on the left, above, in front of, at a distance of. . . où le référentiel est attaché à l'image – sont représentées dans l'ontologie. À notre connaissance, ce sont les seules relations spatiales qui ont été utilisées jusqu'à présent pour construire des systèmes de traitement d'images, par exemple dans les travaux de Colliot (Colliot et al., 2006), de Deruyver (Deruyver, 2006b; Deruyver, 2006a) et de Brun (Brun and Kropatsch, 2006).

On trouvera dans (Hudelot et al., 2008), une ontologie plus complète des relations spatiales et une méthode pour la quantification floue des valeurs.

4.4 Les concepts de tâche

Les tâches sont spécifiées par un verbe et un argument. Un verbe reflète un but de traitement particulier parmi les catégories d'objectifs identifiés par la définition 1.1.

Les verbes définis dans l'ontologie sont listés dans le tableau 2.4. L'argument associé spécifie l'élément sur lequel porte le verbe. Il peut s'agir d'un objet d'intérêt identifié au niveau sémantique – *p. ex.*, Extract <object> –, une primitive visuelle définie au niveau perceptif – *p. ex.*, Detect <edge> – ou une propriété générale de l'image altérée par le système d'acquisition et caractérisée au niveau physique – *p. ex.*, Correct <noise> or Enhance <colorimetry>.

Tableau 2.4 – Liste des tâches de traitement d'images de l'ontologie.

Objective	Tâche
Compression	Compress
Detection	Detect <object>
	Detect <point of interest>
	Detect <edge>
Enhancement	Enhance <photometry>
	Enhance <colorimetry>
	Enhance <blur>
Segmentation	Extract <object>
	Eliminate <object>
	Partition
Restoration	Correct <noise>
	Correct <photometry>
	Correct <colorimetry>
	Correct <geometry>
	Correct <storage>
	Inpaint
Reconstruction	Reconstruct-shape
	Reconstruct-depth
	Reconstruct-motion

4.5 Les concepts de contrainte

Les contraintes et leurs erreurs acceptables sont représentées par des syntagmes prédéfinis. Ils sont listés dans les tableaux 2.5, 2.6 et 2.7. Plus particulièrement, les valeurs possibles pour le critère de qualité sont au nombre de deux :

- La *fiabilité* restreint le fonctionnement du système aux seules conditions spécifiées par la description de l'objectif. Cela signifie que la qualité des résultats est le critère le plus important et qu'il est préférable de ne pas retourner de résultats plutôt que des résultats partiels. C'est par exemple une contrainte vitale lorsque l'on traite des images en vue de la numérisation du patrimoine culturel.
- La *robustesse* élargit le fonctionnement du système aux conditions non spécifiées. Cela signifie que la production de résultats est le critère le plus important et donc qu'il est préférable de produire des résultats même partiels plutôt que pas de résultat du tout. Par exemple, c'est un critère important dans le cas d'une application de vision robotique où l'on cherche constamment à avoir des informations sur l'environnement, même si elles sont partielles.

Tableau 2.5 – Liste des critères à optimiser avec les erreurs acceptables associées pour quelques unes des tâches.

Tâche	Criterion to be optimized	Acceptable error
Compress	Maximize compression rate Maximize image quality	
Detect	Maximize hits	{Prefer miss to false alarm, Prefer false alarm to miss}
Enhance	Maximize fine detail visualization	
Extract	Maximize hits	{Prefer miss than false alarm, Prefer false alarm than miss}
	Maximize boundary localization	{Prefer boundary inside, Prefer boundary outside}
Partition	Maximize precision	{Prefer sub-segmentation, Prefer over-segmentation}

4.6 Rôles, restrictions et déductions

La famille des logiques de conception permet de spécifier les rôles et les restrictions sur les concepts. Les rôles et les restrictions définis pour et entre les concepts sont utilisés pour vérifier la cohérence syntaxique d'une ontologie d'application construite à partir de l'ontologie de domaine.

Par exemple, les restrictions nous permettent de préciser le rôle *hasArgument* du concept *Task* pour les concepts qui en héritent. Pour le sous-concept *Correct*, nous avons défini la restriction :

$$\text{Correct} \subseteq (\forall\ \text{hasArgument}.(\text{Noise} \cup \text{Blur} \cup \text{Photometry} \cup \text{Colorimetry} \cup \text{Geometry}))$$

Cela signifie que tout individu du concept *Correct* ne peut prendre ses *arguments* que parmi les individus des concepts physiques listés.

Les propriétés des logiques de description permettent de profiter de mécanisme d'inférence pour produire de nouveaux faits dans l'ontologie. L'inférence est exploitée pour proposer une aide interactive lors de la formulation. Ainsi, il est possible de suggérer aux utilisateurs des valeurs par défaut pour les contraintes associées à la tâche une fois les post-traitements décrits. Par exemple, si les post-traitements incluent des mesures de photométrie sur les régions à construire en sortie, le mécanisme de déduction peut suggérer la valeur par défaut « maximize boundary localization » et l'erreur acceptable « prefer boudnary inside » pour éviter de compter des valeurs de pixels sur les bords ou même à l'extérieur de l'objet.

5 Évaluation du modèle

La définition de mesures quantitatives pour évaluer un modèle de formulation est une tâche difficile, étant donné que les résultats expérimentaux dépendent largement des performances du système de génération d'application qui l'utilise. Comme protocole de validation, nous proposons une évaluation plus qualitative : mesurer la puissance d'expression du langage de formulation.

La puissance d'expression du langage est premièrement évaluée par sa capacité à représenter différentes formulations d'une même application, obtenues par rétro-ingénierie de solutions prises dans la littérature. Nous voulons montrer que chaque

Tableau 2.6 – Liste des niveaux de détail avec leurs erreurs acceptables pour quelques unes des tâches.

Task	Level of detail	Acceptable error
Detect	Need all pixels	{Prefer more to less, Prefer less to more}
	Need at least one pixel	
	Need all occurrences	{Prefer more to less, Prefer less to more}
	Exclude borders touching	
Enhance	Do not affect colors rank	
	Do not affect colors ratio	
	Do not add new pixel values	
	Do not affect region shape	
	Do not affect edge profile	
Correct	Do not affect colors rank	
	Do not add new pixel values	
	Do not affect edge profile	
	Do not affect colors ratio	
	Do not affect region shape	
Extract	Put boundary inside / outside	
	Do not separate aggregate	
	Separate all	{Prefer separation, Prefer no separation}
	Separate only just-touching objects	{Prefer separation, Prefer no separation}
	Regularize contours	
	Exclude borders touching	
	Reach sub-pixel precision localization	

Tableau 2.7 – Liste des contraintes de contrôle référencées dans l'ontologie.

Constraint	Category	Qualifier
Performance criterion	Optimization	{run-time, real-time, no-optimization}
Quality criterion	Ability	{reliability, robustness, best compromise}

représentation contient les informations nécessaires et suffisantes pour justifier l'utilisation et la paramétrisation des algorithmes proposés par les auteurs pour traiter l'application en question. Les expérimentations ont été menées par rétro-ingénierie d'une douzaine d'applications différentes (reconnaissances de plaques d'immatriculation, détection de texte dans des images, restauration de film ancien...) fournissant plusieurs tâches par application. Ici, nous traitons de l'exemple de l'imagerie aérienne présentée au § 2.2.

Deuxièmement, la puissance d'expression est évaluée par la capacité du langage de rendre compte de besoins précis. Nous démontrons qu'une même tâche peut conduire à une variété d'objectifs, simplement en variant les arguments, les contraintes et les images de référence associés à la tâche.

5.1 Différentes formulations d'un même objectif

Nous prenons ici le cas d'applications d'imagerie aériennes. C'est un domaine qui a beaucoup été étudié et beaucoup de solutions concrètes ont été proposées dans la littérature (Butenuth et al., 2004; Mueller et al., 2004; Cao et al., 2008; Dubuisson-Jolly and Gupta, 2000; Xu et al., 2003). Nous étudions cinq de ces solutions pour lesquelles nous proposons pour chacune une formulation qui permet de motiver le

choix et le paramétrage des éléments de la solution.

5.1.1 Spécification du but

L'objectif global est la localisation des zones de végétation potentielles. Cet objectif se divise en plusieurs buts de traitement d'images séquentiels :

1. correct <storage> : cet objectif vise à corriger les défauts dus à la compression utilisée pour stocker les images ;
2. eliminate <city> : les zones urbaines nuisent à l'accomplissement de l'objectif suivant ;
3. extract <field> : pour obtenir les régions correspondant aux champs.

Le tableau 2.8 détaille la spécification de l'objectif d'extraction des champs (*i.e.*, Extract <field>). Les contraintes associées sont motivées par le fait que le post-traitement de cette tâche est une classification des régions qui utilise des mesures de photométrie, de morphologie, de localisation et de taille. Le premier critère à optimiser est le taux de détection et l'erreur acceptable associée donne la préférence aux fausses détections, plutôt qu'aux oublis, car les fausses détections pourront être éliminées au cours de l'étape de classification. La localisation des frontières de champ est le deuxième critère à optimiser. En cas de doute sur l'emplacement de la frontière, l'erreur acceptable spécifie d'utiliser les techniques de transformation qui mettent les frontières plutôt à l'intérieur de la région pour éviter les erreurs de mesure photométrique avec des pixels en dehors de la région. Le niveau de détail indique qu'il est nécessaire de séparer les champs qui se touchent. Dans le cas où il est difficile de déterminer si une région correspond à un ou plusieurs champs, l'erreur acceptable indique une préférence pour la séparation, ce qui signifie qu'une sur-segmentation est préférable à une sous-segmentation. Et enfin, parce que la reproductibilité est le point clé de cette application, la robustesse est préférée à la fiabilité.

Tableau 2.8 – Spécification de l'objectif d'extraction des champs.

Descripteur	Valeur
Tâche	Extract <field>
Critère à optimiser (*acceptable error*)	Maximize hits (*prefer false alarm to miss*) Maximize boundary localization (*prefer boundary inside the region*)
Niveau of detail (*acceptable error*)	Separate only just-touching (*prefer separation*)
Performance criterion	Optimization = run-time
Quality criterion	Ability = robustness

5.1.2 Définition physique pour la tâche "correct storage"

La définition de la classe d'images au niveau physique pour la tâche de correction des effets de stockage (*i.e.*, correct<storage>) est présentée dans le tableau 2.9. Dans cette définition, il est à noter que le stockage des images produit un effet de bloc dû à la compression JPEG. La fonction de transfert est définie comme étant linéaire. Le bruit est décrit comme gaussien avec une moyenne nulle et un faible écart-type. Cette définition conduit à choisir un algorithme de lissage qui réduit les effets de bloc. De ce fait, les deux tâches suivantes (*i.e.*, eliminate <city> et extract <field>) utilisent la même définition, mais sans le défaut de stockage.

Tableau 2.9 – Définition de la classe d'images aérienne au niveau physique.

Acquisition effect	Descriptor : Value
Colorimetry	Color space : RGB
Quantization	Function : linear
Noise	Composition : additive Distribution : gaussian Power Spectral Density : white Mean : 0 Standard-deviation : low
Storage	Defect : block effect

5.1.3 Définition sémantique pour la tâche "extract field"

Comme tous les objets d'intérêt sont prévisibles et descriptibles, la définition de la classe d'images se fait plutôt au niveau sémantique qu'au niveau perceptif.

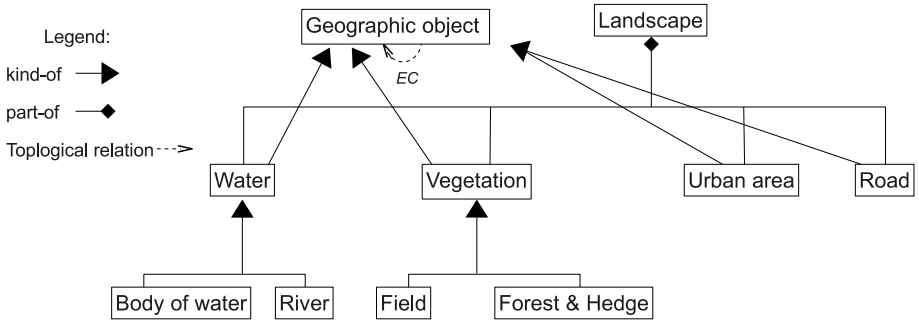


Figure 2.12 – Une scène de paysage rural est composée d'objets géographiques juxtaposés : étendue d'eau, végétations, zones urbaines et routes, etc. La juxtaposition des objets géographiques est représentée par la relation topologique « externally connected » (EC).

L'arbre des objets d'intérêt est donné figure 2.12. Dans cet arbre, une scène géographique est considérée comme étant composée d'objets juxtaposés, indiquée par la relation topologique « externally connected » entre les objets géographiques. Les objets possibles sont les zones d'eau (rivières, étangs), les zones de végétation (champs, haies, forêts), les zones urbaines et les routes.

Nous nous proposons d'étudier trois approches différentes de l'accomplissement de la tâche. Chacune est motivée par un point de vue sur la définition sémantique de l'objet d'intérêt champ.

Approche par croissance de régions. La première approche est basée sur une croissance des régions contrôlée par les contours. Par exemple, M. Butenuth et al. (Butenuth et al., 2004) proposent d'utiliser une segmentation par ligne de partage des eaux, contrôlée par le gradient de l'image, où les germes sont les minima du gradient. Ils justifient l'utilisation de cet algorithme par le fait que les champs sont des régions homogènes (*texture.type : no texture*) et que les frontières des champs sont très contrastées (*edge-contrast : high*]).

M. Mueller et al. (Mueller et al., 2004) proposent de contrôler la croissance de régions avec les contours extraits par une détection de contours préalable. Ils estiment

que les frontières des champs sont bien marquées, avec un fort contraste aux régions voisines (*edge-contrast : high*) et une forme typique longue et droite (*shape : straight line ; length > low*).

Dans ces deux cas, une étape suivante de fusion est effectuée pour fusionner les régions voisines avec une faible différence de niveau de gris (*texture.type : no texture*) et les petites régions dans les plus grandes (*area : level>= low*). Finalement, une dernière étape est entreprise en vue de satisfaire la contrainte de localisation des frontières. M. Buthenuth et al. utilise l'algorithme des snakes. Les snakes sont initialisés avec le plus petit rectangle exinscrit dans chaque champ, puisque les champs sont des régions polygonales avec quatre coins (*shape : parallelepiped*). L'énergie externe est la valeur absolue du gradient (*edge.contrast : high*) et l'énergie interne est fixée de manière à favoriser la rigidité parce que les frontières des champs sont décrites comme de longs contours rectilignes (*shape : straight line ; length > low*).

Le tableau 2.10 résume une définition sémantique qui conduit le système à sélectionner et à paramétrer des algorithmes implémentant l'approche par croissance de région.

Tableau 2.10 – Une définition sémantique possible pour motiver l'utilisation de l'approche par croissance de régions.

Objet d'intérêt	Visual primitive	Descriptor : Value
Field	Region *(Field area)*	Texture.Type : no texture Region-Morphology.Shape : parallelepiped Region-Size.Area : >= low Boundary.Nature : edge
	Edge *(Field boundary)*	Edge-Morphology.Contrast : high Edge-Morphology.Shape : straight line Edge-Size.Length : > low

Approche variationelle. La seconde approche est basée sur les méthodes variationnelles. G. Cao et al. (Cao et al., 2008) utilisent les contours actifs via l'algorithme des ensembles de niveaux (angl. *level sets*) pour partitionner les images. Les ensembles de niveaux sont paramétrés avec des caractéristiques de texture (modélisées par décomposition en ondelettes sur trois niveaux). Les auteurs supposent que les champs sont discriminables par leur micro-texture complexe et invariante à la rotation (*texture.type : complex ; texture.scale : micro ; texture.direction : 0*). Cette première étape fournit une image segmentée avec toutes les régions discriminables par leur texture interne. Il est alors nécessaire de procéder à une sélection des régions par exemple sur la base de leur surface, pour ne garder que les champs potentiels (*area : >= 400 pixels*).

La définition de la classe d'images donnée dans le tableau 2.11 peut être utilisée pour sélectionner et contrôler l'algorithme des ensembles de niveaux.

Approche statistique. La troisième approche est fondée sur une classification supervisée des pixels. Cette approche nécessite une étape d'apprentissage à partir d'images exemples. M-P. Dubuisson-Jolly et al. (Dubuisson-Jolly and Gupta, 2000) utilisent une classification par maximum de vraisemblance qui combine les informations de couleur et de texture. Les caractéristiques de couleur RVB et la micro-texture sont calculées avec le modèle SAR à partir des blobs (*texture.type : complex ;*

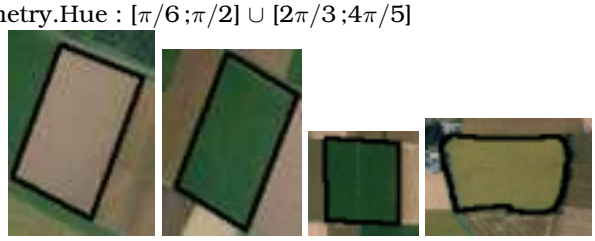
Tableau 2.11 – Une définition sémantique qui motive l'utilisation de l'approche variationelle.

Object	Visual primitive	Descriptor : Value
Field	Region (*field area*)	Texture.Type : complex Texture.Direction : 0 Texture.Scale : micro Region-Size.Area : > 400

texture.scale : micro ; colorimetry.hue : [$\pi/6$;$\pi/2$] $\cup$ [$2\pi/3$;$4\pi/5$] ; model : images). F. Xu et al. (Xu et al., 2003) proposent d'utiliser les SVM pour la classification des pixels. Le SVM est basé sur l'utilisation de dix-sept éléments de micro-texture dont les valeurs sont extraites des blobs (*texture.type : complex ; texture.scale : micro ; model : images*).

La sélection de ces deux approches statistiques est motivée par la définition sémantique donnée dans le tableau 2.12, où les champs sont décrits comme discriminables par leur texture seule, ou par l'utilisation conjointe de la texture et de la couleur.

Tableau 2.12 – Une définition sémantique qui motive l'utilisation de l'approche statistique.

Object	Visual primitive	Descriptor : Value
Field	Region (*field area*)	Texture.Type : complex Texture.Scale : micro Colorimetry.Hue : [$\pi/6$;$\pi/2$] $\cup$ [$2\pi/3$;$4\pi/5$] Model : 

5.2 Différents objectifs pour une même tâche

Le second exemple illustre que la combinaison de la spécification par tâche et par l'exemple permet une grande variété de nuances, ainsi qu'une grande précision dans la spécification des objectifs. La même tâche de base d'amélioration de l'image « enhance » peut être dérivée en différents objectifs :

1. La tâche « enhance <photometry> » couvre toutes les formes d'amélioration d'images sous-exposées, sur-exposées ou faiblement contrastées. La tâche « enchance <colorimetry> » vise à améliorer les images trop délavées ou trop saturées et la tâche « enhance <blur> » correspond à une amélioration de la netteté d'images.
2. Le fait de lui associer une contrainte permet d'affiner la tâche. Par exemple, la tâche « enhance <photometry> » avec le niveau de détail « do not affect color rank » limite les méthodes d'amélioration à la modification de la luminosité et du contraste global, tandis qu'avec le critère à optimiser « maximize fine detail

visualization », la tâche autorise des modifications plus radicales telles celles du filtrage homomorphique.

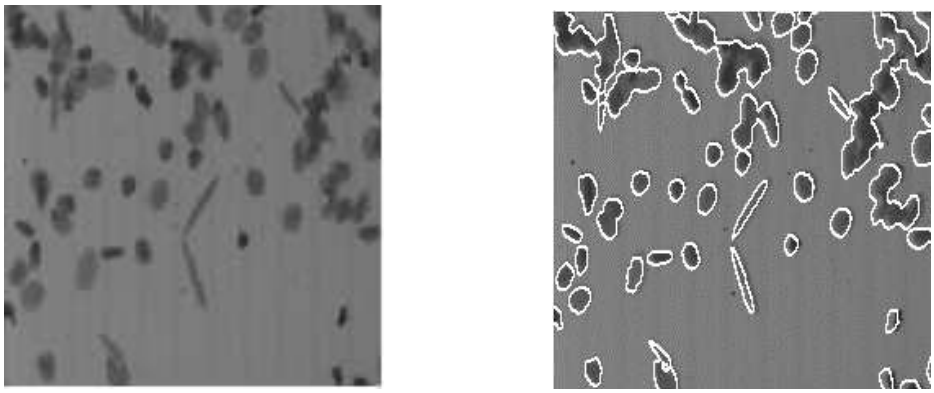
3. Enfin, ajouter des images de référence à la tâche « enhance <photometry> », signifie que l'amélioration de la photométrie peut aussi se faire sur la base des exemples donnés. Il est possible alors d'utiliser des techniques comme le matching d'histogrammes ou l'apprentissage supervisé.

Le langage de spécification de la tâche autorise de nombreuses définitions différentes en même temps qu'il permet un enracinement dans les données.

5.3 Exemple d'une application en imagerie biomédicale

Nous présentons un autre exemple d'application qui sera utilisé comme exemple fil rouge dans la section suivante. Cette application présente l'avantage d'être assez simple et assez didactique. Il s'agit d'une application biomédicale de cytologie décrite dans (Elmoataz et al., 1992). Elle a été étudiée au laboratoire dans le cadre de la réalisation d'un analyseur d'images dédié à la détection d'anomalie de ploïdie dans les tumeurs cancéreuses humaines.

Les images sont des vues en niveaux de gris de lames de cytologie qui contiennent des étalements de cellules humaines d'œsophage – *p. ex.*, figure 2.13. L'objectif de traitement d'images consiste à segmenter l'image pour isoler chaque objet cellulaire dans une région. On fera bien attention de ne pas séparer les amas d'objets et de les conservés dans une région unique. Les régions obtenues, cellules isolés ou amas de cellules, seront ensuite utilisées pour déterminer le type d'objet cellulaire qu'elles désignent – *p. ex.*, lymphocyte, épithéliale, stromale, amas.



(a) (b)

Figure 2.13 – (a) Exemple d'une image de cytologie, et (b) Le résultat souhaité surimposé à l'image initiale.

Nous donnons ici une définition entièrement symbolique. Le tableau 2.13 représente la spécification du but. Il s'agit d'une tâche de segmentation. Les contraintes spécifient qu'il faut faire une distinction entre les objets isolés et les objets en amas et que les frontières doivent être localisées aussi précisément que possible. La figure 2.14 décrit une scène de cytologie par des cellules étalées sur une lame. Le fait que les cellules peuvent se toucher est noté par la relation EC – *p. ex.*, Externally Connected. Les tableaux 2.14 et 2.15 donnent respectivement la définition de

la classe d'images aux niveaux physique et sémantique. La définition physique décrit un flou léger sans direction privilégiée et un bruit blanc gaussien faible. La définition sémantique rend compte du fait que les objets cellulaires sont les régions les plus foncées, de forme convexe, et reposent sur un fond d'image clair.

Tableau 2.13 – Spécification de l'objectif d'extraction d'objets cellulaires

Descripteur	Valeur
Tâche	Extract <cell>
Critère à optimiser (*acceptable error*)	Maximize hits (*prefer false alarm to miss*) Maximize boundary localization (*prefer boundary inside the region*)
Niveau of detail (*acceptable error*)	Separate only just-touching (*prefer no separation*)
Quality criterion	Ability = reliablity

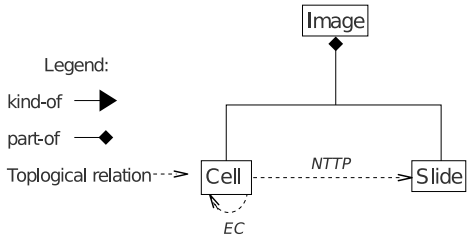


Figure 2.14 – Les images de cytologie se composent de cellules étalées sur une lame.

Tableau 2.14 – Définition de la classe d'images au niveau physique.

Acquisition effect	Descriptor : Value
Colorimetry	Color space : gray
Blur	Strength : low Direction : 0
Noise	Composition : additive Distribution : gaussian Power Spectral Density : white Mean : 0 Standard-deviation : low

6 Conclusion

Dans ce chapitre, un modèle computationel pour la représentation d'objectifs de traitement d'images et une ontologie de domaine correspondante ont été proposés. Par rapport à d'autres travaux qui utilisent les ontologies pour représenter l'information préalable sur les applications d'analyse d'images – *p. ex.*, (Nouvel, 2002; Mezaris et al., 2004; Hudelot, 2005; Town, 2006; Neumann and Möller, 2008; Maillot and Thonnat, 2008), notre contribution est originale dans le sens où nous avons fait le pari de l'existence d'un domaine du traitement d'images avec ses propres concepts. Ainsi, le modèle et l'ontologie définissent un vocabulaire de formulation d'objectifs

Tableau 2.15 – Définition de la classe d'images au niveau sémantique.

Objet d'intérêt	Visual primitive	Descriptor : Value
Cell	Region	Texture.Type : no-texture Region-Morphology.Shape : ellipsoid Region-Morphology.Convexity : very high Region-Size.Area : >= 100 Boundary.Nature : edge Photometry.Brightness : the least (*d*arkness)
Slide	Background	Photometry.Brightness : the most (clearest) Texture.Type : no-texture

indépendant des domaines d'application, mais qui permet de représenter n'importe quel objectif dans n'importe quel domaine. Ceci est possible parce que nous avons basé le modèle de formulation sur les principes et les hypothèses suivants :

1. Une application est définie pour un but et une classe d'images.
2. Le but est établi à partir d'une tâche avec des contraintes et des images de référence optionnelles. La spécification des tâches permet une identification précise de l'objectif et la prise en compte des besoins des utilisateurs. Les images de référence lient la tâche aux données réelles.
3. La définition de la classe d'images repose sur l'hypothèse phénoménologique et est basée sur une dénotation en trois niveaux : physique, perceptif et sémantique. Une dénotation est faite par le biais d'une définition extensionnelle ou intensionnelle de la manifestation visuelle de la scène, selon que les éléments d'information sont plus précisément identifiables par des exemples ou par une description linguistique.

Plusieurs expériences ont été menées par une rétro-ingénierie d'applications existantes (telles que l'imagerie aérienne) et par ingénierie de nouvelles applications pour évaluer la puissance d'expression de notre langage de formulation. Ces expérimentations nous permettent de conclure que, bien que l'expression de la sémantique par une représentation purement syntaxique n'est pas suffisante pour saisir le sens réel de l'image (Santini et al., 2001), elle est néanmoins suffisante pour concevoir des applications de traitement d'images. La sémantique d'une application d'un domaine particulier est représentée par un ensemble de relations entre des structures purement syntaxiques ou purement iconiques du domaine du traitement d'images.

Le langage de formulation est d'une portée générale et profite aussi bien aux systèmes à base de connaissances qu'aux systèmes basés image, mais aussi aux experts de traitement d'images. Il est d'un intérêt fondamental pour le développement d'applications. Tout d'abord, il permet d'aider à produire des applications plus robustes et plus fiables. La représentation de l'objectif est considérée comme un document contractuel, qui reflète un consensus entre les spécialistes du domaine d'application qui fixe le problème et le développeur qui produit le logiciel. Ensuite, il est aussi un moyen pour définir des règles d'évaluation des résultats parce qu'il y a une représentation explicite des éléments que l'utilisateur considère comme importants. Enfin, il favorise la réutilisation, voire même la simple reproduction des solutions existantes.

L'ontologie proposée a été implémentée avec le langage OWL et contient quelques 300 concepts, 20 rôles et plus de 150 restrictions. Elle est disponible à l'adresse : http://www.greyc.ensicaen.fr/~regis/Pantheon/resources.

En terme de perspective, il est nécessaire d'étendre le modèle de l'ontologie aux cas des séquences d'images.

CHAPITRE 3

La génération automatique de programmes d'application

Ce chapitre présente les travaux que nous avons effectués sur la génération automatique de programmes de traitement d'images à partir de formulations d'objectifs faites par des utilisateurs. BORG *est un système qui rentre dans la catégorie des systèmes de pilotage de codes, qui utilisent une bibliothèque d'opérateurs prédéfinis pour construire des chaînes de traitement adaptées aux applications. La construction des chaînes y est abordée comme un problème de planification de tâches. Les plans élaborés servent ensuite de support pour la sélection, le paramétrage et l'enchaînement des traitements effectifs. Ils sont conçus dynamiquement par coopération de sources de connaissances modulaires, autonomes, indépendantes et spécialisées. Le choix des sources de connaissances à exécuter est fait à partir d'un plan de contrôle, lui même construit dynamiquement par des sources de connaissances dédiées au problème du contrôle.*

Ces travaux ont été les premiers entrepris dans le projet. Ils ont débuté avec mon travail de thèse de doctorat et ont évolué jusqu'à aujourd'hui. L'ensemble des autres parties du projet de recherche, présentées dans les autres chapitres, sont nées des besoins et lacunes mis en évidence par ces premiers travaux : la formulation d'objectifs, l'interaction-homme machine et l'acquisition des connaissances.

1 Introduction

Étant donnée une formulation de l'application représentée par une ontologie d'application respectant le modèle proposé dans la section précédente, l'étape suivante consiste à produire un programme exécutable adapté, capable de traiter toutes les images de la classe identifiée.

Dans ce chapitre, nous traitons de la conception de systèmes permettant la génération automatique de programmes d'application à partir d'une formulation. Il y a deux façons de voir cet objectif. Soit construire un système qui produit directement les images résultats pour chaque image à traiter – le programme exécutable est donc le système lui-même et il doit être réexécuté pour chaque image –, soit construire un système qui génère un programme exécutable dédié, capable de traiter les images résultats pour chaque image donnée en entrée. La première façon privilégie l'adaptabilité des traitements à la variété des images de la classe au détriment de la vitesse d'exécution. La seconde est plus performante en temps de traitement, mais il est

plus difficile de prévoir dans un programme impératif tous les cas couvrant la variété des images. La section 2 présente un certain nombre de travaux ayant proposé différentes approches pour l'une ou l'autre de ces façons. Parmi les résultats de ces travaux, le *pilotage de codes exécutables* émerge comme un paradigme de base de la construction de programmes d'application. Un programme y est conçu comme une chaîne d'opérateurs de traitement, où chaque opérateur est représenté par un code exécutable pris dans une bibliothèque prédéfinie. Les images d'entrée subissent successivement les opérations enchaînées pour finalement aboutir aux images résultats. La construction de cette chaîne s'envisage donc un problème de sélection, paramétrage et enchaînement de codes.

Notre approche rentre dans la catégorie des systèmes qui génèrent automatiquement des programmes exécutables autonomes, et elle repose naturellement sur le paradigme du pilotage de codes. L'originalité de notre approche réside dans le fait que le pilotage de codes est abordé comme un problème de planification hiérarchique, opportuniste et incrémentale qui vise à construire un plan d'actions original, pour conduire la sélection, le paramétrage et l'enchaînement des codes, sans que jamais ce plan n'ait intégralement ni partiellement existé dans la base de connaissances du système. Dans la section 3, nous présentons le modèle de notre système basé sur le patron d'architecture « Tableau Noir » (angl. *Blackboard*) de type BB1. Dans la section 4, nous détaillons la représentation d'un plan solution sur la base de données du tableau noir, et dans la section 5, nous décrivons la base de connaissances qui a en charge la construction d'un plan adapté aux données de la formulation. Le problème du contrôle de la résolution est abordé comme un problème à part entière et fait l'objet de la section 6.

2 Problématique

2.1 Approches de la génération d'application

Comme cela a déjà été souligné dans le chapitre 1, le problème de la génération automatique d'applications a beaucoup été étudié surtout dans les années 1990. Différentes approches ont ainsi été proposées pour concevoir des systèmes de génération d'application, dont les plus abouties sont les suivantes :

- La compétition de compétences ;
- La planification linéaire ;
- La construction incrémentale du résultat ;
- L'instanciation de squelettes de plans ;
- Le raisonnement à partir de cas.

2.1.1 Compétition de compétences

L'idée de base de cette approche est d'exploiter la concurrence entre plusieurs stratégies de traitement prédéfinies, codées par des programmes exécutables.

Par exemple, Charroux et al. (Charroux and Philipp, 1995; Charroux et al., 1996) proposent d'exécuter différents algorithmes de segmentation d'images en parallèle, puis de construire le résultat final avec les régions les mieux segmentées par chacun de ces algorithmes. La qualité d'une région est mesurée par son degré d'appartenance à une classe d'objets du domaine, calculé par un classifieur dédié. L'adaptation du système à une application particulière est donc réalisée au travers du classifieur

entraîné pour reconnaître les objets du domaine. Le traitement d'une nouvelle image de la classe nécessite alors de réexécuter le système entièrement.

Martin et al. (Martin et al., 2006; Martin, 2007) proposent de mettre en concurrence les algorithmes de segmentation pour ensuite sélectionner le meilleur, avec le meilleur paramétrage. La sélection se fait par apprentissage à partir d'exemples d'images tests pour lesquelles on fournit la segmentation de référence « idéale » faite à la main. L'algorithme finalement sélectionné, avec son paramétrage, est celui qui minimise une distance entre la segmentation obtenue sur une image test et la segmentation de référence donnée pour cette même image. Cette fois, l'application résultante est donc l'algorithme paramétré sélectionné, qui peut ensuite être utilisé individuellement sur chaque image de classe.

2.1.2 Planification linéaire

Cette approche repose sur la modélisation d'un type d'expression qui peut être propagée le long des chaînes de traitement. Le raisonnement est focalisé sur les opérations à appliquer à l'expression initiale pour construire l'expression finale attendue. Cette expression est en général construite par l'utilisateur (Dejean and Dalle, 1996a; Rost and Münkel, 1998), mais elle peut aussi être construite automatiquement par extraction de caractéristiques à partir d'exemples (Levner et al., 2003a). La génération des chaînes peut être plus ou moins combinatoire. Dans ce cas, chaque opérateur de la chaîne est modélisé par une liste de préconditions et une liste des effets sur l'expression, comme dans le système EXTI (Dejean and Dalle, 1996a; Dejean and Dalle, 1996b; Dejean, 1996; Dalle and Dejean, 1998). La génération des chaînes peut aussi être choisie dans un réseau prédéfini de toutes les chaînes possibles. Dans ce cas, la chaîne correspond à un chemin dans le réseau, décidé par exécution de règles attachées aux nœuds, comme dans le système LLVE (Matsuyama, 1989). Ou encore, la chaîne peut être construite par application de règles de production qui permettent de sélectionner les prochains opérateurs en fonction de l'expression courante, comme dans le système SOLUTION (Rost and Münkel, 1998).

Dans tous ces cas, la chaîne finale est prise comme celle qui retourne l'expression attendue en sortie. L'application finale est la chaîne de traitement construite opérateur par opérateur, qui peut ensuite être utilisée de façon autonome. Draper et al. (Draper et al., 1999) avec le système ADORE et ses variantes MR ADORE(Levner et al., 2003b) proposent de garder différentes alternatives de chaînes possibles dans la représentation du programme. Il utilise un processus de décision markovien pour choisir dynamiquement le bon chemin dans ces chaînes, à partir de caractéristiques extraites automatiquement de l'image à traiter.

2.1.3 Construction incrémentale du résultat

Ici, le système correspond à l'application elle-même et il doit être réexécuté sur chaque image à traiter pour construire les images résultats. La construction du résultat procède par évolution progressive et contrôlée des données d'entrée vers les données désirées en sortie. Cette approche peut être vue comme le dual de l'approche précédente, dans le sens où le raisonnement est focalisé sur les données. Ici, les algorithmes de traitement d'images sont complètement éclatés en un ensemble de règles (Nazif and Levine, 1984) ou d'agents rationnels indépendants (Boucher and Garbay, 1996; Boucher, 1999; Abchiche et al., 2002; Bovemkamp et al., 2004). Dans une telle approche, il n'y a pas de stratégie de génération des chaînes de traitement

qui soit explicitée. Le raisonnement reste focalisé sur l'analyse de l'état courant des données, dans le but de décider des prochains traitements à appliquer dans le cas de règles ou de résoudre le conflit d'accès aux données dans le cas de multi-agents.

2.1.4 Instanciation de squelette de plans

C'est certainement l'approche qui a donné le plus de systèmes de traitement d'images, avec OCAPI (Clément and Thonnat, 1993), VSDE (Théot et al., 1992; Bodington, 1995), CONNY (Liedtke and Blömer, 1992), COLLAGE (Lansky et al., 1995), DIA-Expert (Tamura et al., 1984), MVP (Chien and Mortensen, 1996), le système de M. Schmitt basé morphologie mathématique (Schmitt, 1989), etc.

Les stratégies de traitement sont codées dans des plans hiérarchiques qui associent en plusieurs niveaux de décomposition la tâche correspondant au problème à un ensemble d'opérateurs exécutables qui constituent les éléments d'une chaîne de traitement possible. Les squelettes de plans sont codés par des arbres ET/OU qui indiquent comment une tâche peut être décomposée en sous-tâches. Les feuilles sont les opérateurs exécutables. À chaque nœud, des règles permettent de choisir la branche de décomposition la plus adaptée, et si nécessaire, les valeurs de paramètres en fonction des caractéristiques du problème.

L'application finale est donc le squelette de plan configuré, qui doit être réexécuté pour chaque image à traiter.

2.1.5 Raisonnement à partir de cas

Le raisonnement à partir de cas s'appuie sur la mémorisation pour construire les traitements adaptés. Le principe est de retrouver dans une base, un cas passé « similaire » au nouveau cas à traiter. En traitement d'images, cette approche a été utilisée pour déterminer le bon jeu de paramètres pour configurer un algorithme de traitement général (Perner, 1999; Frucci et al., 2008). Elle a aussi été utilisée pour retrouver directement des plans de traitement – *p. ex.*, (Charlebois et al., 1991; Charlebois, 1997; Ficet-Cauchard et al., 1999; Ficet-Cauchard, 1999). Le raisonnement est basé sur l'analyse d'une expression du problème à résoudre faite par l'utilisateur pour retrouver un cas similaire, qui sera ensuite adapté aux particularités du problème courant (Perner et al., 2005). S'il n'existe pas de cas similaire, alors un nouveau cas doit être appris et stocké dans la base.

L'application finale est l'algorithme correctement paramétré ou la chaîne de traitement retrouvée et adaptée, qui peut s'utiliser individuellement.

2.1.6 Bilan

Au final, aucune de ces approches n'offre de solution meilleure qu'une autre pour concevoir un système réellement viable. Chacune présente des avantages, mais aussi des défauts qui restent encore difficilement surmontables aujourd'hui. En fait, chacune se heurte à l'éternel compromis en Intelligence Artificielle entre l'efficacité du système, que ce soit dans ses capacités d'adaptabilité ou de prise en compte des besoins des utilisateurs, et la nécessité de recourir à une acquisition de connaissances.

Ainsi, l'approche par compétition de compétences est intéressante parce qu'elle ne requiert pas de modélisation explicite d'expertise. Par contre, elle utilise des chaînes de traitements qui sont figées et en nombre fini, dont la seule adaptation repose sur les valeurs de paramètres.

L'approche par planification linéaire évite aussi le problème de la représentation d'une expertise de traitement et permet de créer des chaînes originales pour chaque application. Cependant, elle se heurte à la difficulté de modéliser le problème sous la forme d'une expression propageable le long des chaînes de traitement et surtout à la difficulté de devoir estimer a priori les effets des opérations sur cette expression.

La construction incrémentale du résultat nécessite une phase d'acquisition des connaissances. Mais, le contrôle décentralisé facilite l'acquisition des connaissances, puisqu'il n'est pas nécessaire pour l'ingénieur de la connaissance d'expliciter des stratégies de résolution. Par contre, le processus de résolution global reste complexe à maîtriser parce que la convergence vers une solution n'est garantie que par l'action de règles ou d'agents qui n'ont qu'une vision très locale de leur effets. Chaque règle ou chaque agent est responsable de savoir estimer la valeur de sa contribution par rappoort à l'état courant de la résolution. Cette limite oblige à ajouter des niveaux d'abstraction dans la hiérarchie des règles ou des agents rationnels pour avoir une vision plus globale de la résolution (Boucher, 1999; Bovemkamp et al., 2004).

L'instanciation de squelettes de plans résulte d'un processus de reconnaissance (Nii, 1989) qui fait que la résolution ne cherche pas à chaque étape quelle est la prochaine action à effectuer puisqu'il la connaît : elle fait partie d'un plan connu. L'intérêt est donc de pouvoir produire des solutions même à partir de formulation incomplète du problème et sans avoir besoin de propager les effets des opérateurs. Par contre, cela nécessite de savoir identifier et représenter des expertises de résolution pour chaque type de problème possible.

Le raisonnement à partir de cas est aussi basé sur un processus de reconnaissance, sauf qu'il n'utilise pas d'expertise pour cela. Par contre, le point dur de cette approche réside dans l'adaptation des cas au contexte particulier de l'application. Cela ne peut pas se faire sans attacher une expertise explicite avec les cas inclus dans la base de connaissances, si on veut un tant soit peu d'efficacité (Ficet-Cauchard, 1999). Et dans ce cas, on retombe dans le problème de l'acquisition des connaissances.

2.2 Pilotage de codes pour le traitement d'images

Même si aucune approche n'a fourni de solution définitive, on peut noter que la plupart ont érigé le *pilotage de codes* comme le paradigme de base de la conception d'applications.

2.2.1 Le paradigme de base

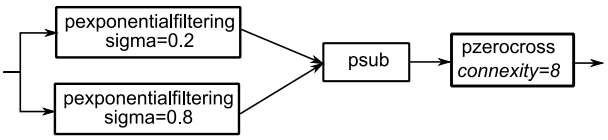
Dans ce paradigme (Thonnat et al., 1999; Thonnat and Moisan, 2000; Moisan and Ziébelin, 2000), les techniques de traitement d'images sont implantées sous forme de programmes exécutables indépendants et regroupés dans une bibliothèque du type de celle de Khoros (Rasure and Kubica, 1994). Un code correspond à un opérateur de traitement d'images qui prend en entrée des images et produit en sorties de nouvelles images ou des valeurs numériques. Il dispose de paramètres qui permettent de varier son comportement.

Un programme est alors représenté par un graphe d'opérateurs orienté. Les liens entre les opérateurs décrivent le réseau d'images et de valeurs de paramètres échangées entre ces opérateurs. Par exemple, la figure 3.1 montre une chaîne d'opérateurs qui effectue une détection de contours par différence de deux gaussiennes, selon l'algorithme de Marr-Hildreth (Marr and Hildreth, 1980). L'exécution d'une chaîne d'opé-

rateurs implique très souvent l'ajustement des valeurs de paramètres et le contrôle du nombre d'itérations des opérateurs afin d'optimiser le résultat sur chaque image.

Le processus complet de construction d'applications selon ce paradigme, nécessite les cinq activités suivantes (Thonnat and Moisan, 1995) :

1. Sélection des opérateurs adaptés ;
2. Détermination des valeurs de paramètre ;
3. Enchaînement des opérateurs pour composer des chaînes ;
4. Exécution de la chaîne ;
5. Contrôle du résultat et corrections éventuelles.




```
pexponentialfiltering 0.2 in.pan i1.pan
pexponentialfiltering 0.8 in.pan i2.pan
psub i1.pan i2.pan i3.pan
pzerocross 8 i3.pan out.pan
```

Figure 3.1 – Un programme est un graphe d'opérateurs exécutables paramétrés. A gauche, est donnée la représentation d'une détection de contours utilisant l'algorithme DOG (Difference of Gaussian) sous la forme d'un graphe d'opérateurs et à droite sous la forme d'une séquence des commandes exécutables équivalente.

2.2.2 Modèle de résolution de problème

Tous les systèmes reposant sur le paradigme du pilotage de codes n'ont pas cherché à automatiser entièrement le processus de construction d'une application. Différents objectifs ont été poursuivis selon le niveau d'implication demandé à l'utilisateur, et par là, le degré d'efficacité recherché. En fait, plus l'utilisateur est impliqué, plus il a de latitude dans la construction. Par contre, plus cela exige de lui une expertise en traitement d'images.

Ainsi, certains travaux ont privilégié l'assistance aux experts dans la construction et le contrôle d'exécution – *p. ex.*, EXPLAIN (Tanaka and Sueda, 1988), le système de Carel (Carel, 1989). D'autres ont essayé d'automatiser totalement ou partiellement le processus de construction de la chaîne et seul le contrôle de l'exécution est laissé à la responsabilité de l'utilisateur – *p. ex.*, DIA-ES (Sakaue and Tamura, 1985), le système de Toriu's (Toriu et al., 1987), EXTI (Dejean and Dalle, 1996a), MVP (Chien and Mortensen, 1996), COLLAGE (Lansky et al., 1995)).

Enfin, seuls les projets avec le plus d'ambition prennent en charge le processus complet jusqu'au contrôle de l'exécution des chaînes. Cela implique aussi l'évaluation et la réparation des chaînes en cours d'exécution – *p. ex.*, CONNY (Liedtke and Blömer, 1992), OCAPI (Clément and Thonnat, 1993), VSDE (Bodington, 1995).

À partir d'une analyse de ces systèmes, cinq règles de conception principales peuvent être dégagées.

Règle 1. *L'expertise de traitement d'images est modélisée par des plans de tâches hiérarchiques.*

Afin de gérer la complexité, la résolution de problèmes est souvent décrite en termes de décomposition en sous-problèmes (Chandrasekaran et al., 1992). La hiérarchie peut prendre la forme d'une décomposition fonctionnelle ou d'une décomposition par niveaux d'abstraction de la tâche initiale à accomplir. Une solution prend la

forme d'un plan qui associe, le long d'une hiérarchie, la tâche à une séquence d'opérateurs adaptée. Toute l'expertise disponible est codée dans des arbres de tâches de type ET/OU – *p. ex.*, CONNY, OCAPI and VSDE qui décrivent les différentes façons possibles d'accomplir une tâche sous la forme d'une chaîne d'opérateurs. Les branches ET marquent un enchaînement séquentiel des sous-tâches. Les branches OU gardent les alternatives possibles de décomposition en sous-tâches.

La hiérarchie est naturelle en traitement d'images. Le traitement d'une image n'est pas un processus monotone, qui fait que chaque opérateur produit des données intermédiaires qui progressent de façon continue et uniforme vers la solution. On considère généralement qu'il faut passer par des changements d'abstraction – *p. ex.*, de pixels en contours ou de régions en graphe d'adjacence –, par des changements d'espace de représentation – *p. ex.*, spatial, fréquentiel, couleur –, mais aussi par des étapes intermédiaires qui ne sont pas directement liées à l'objectif final – *p. ex.*, l'augmentation du contraste ou la réduction du bruit.

Cette modélisation présente deux avantages majeurs :

- Une description des données image à base des caractéristiques est suffisante pour produire une solution. Il n'est pas nécessaire de disposer d'une modélisation exacte de l'environnement ni de la propager le long des chaînes de traitement.
- Les stratégies pour ajuster les paramètres peuvent être directement intégrées dans les plans.

Règle 2. *La construction du graphe d'opérateurs s'apparente à un problème de planification.*

Si l'expertise peut avantageusement se modéliser par des plans hiérarchiques, le processus de construction de chaînes de traitement d'images s'aborde alors naturellement comme un problème de planification. La construction d'une application requiert alors cinq activités :

1. *Planification.* Déterminer un plan adapté ;
2. *Instantiation.* Adapter le plan aux particularités du contexte et produire la chaîne d'opérateurs correspondante ;
3. *Exécution.* Contrôler l'exécution de la chaîne d'opérateurs ;
4. *Évaluation.* Évaluer la pertinence des résultats ;
5. *Correction.* Proposer des corrections pour les parties de plan non satisfaites.

Dans la plupart des systèmes, la détermination du plan se fait par sélection dans une base de squelettes de plan prédéfinis. Les quatre autres activités sont réalisées par des règles de production attachées aux nœuds de l'arbre ET/OU.

Dans le cas du traitement d'images statiques, ces activités peuvent se réaliser séquentiellement ; l'environnement initial, représenté par la formulation du problème, n'évoluant pas au cours de la résolution.

Règle 3. *L'exécution des opérateurs intègrent des structures de contrôle itératives et répétitives.*

Comme la détermination a priori des valeurs de paramètres est un problème complexe (Matsuyama, 1989), il faut disposer de mécanismes pour calculer dynamiquement ces valeurs. Cela est généralement fait par des exécutions successives de l'opérateur avec différentes valeurs de paramètre. Les valeurs potentielles peuvent être

prises autour de valeurs par défaut ou de valeurs obtenues à partir d'autres opérateurs. Les valeurs sélectionnées doivent satisfaire – *p. ex.*, OCAPI – ou optimiser – *p. ex.*, LLVE – des fonctions d'évaluation données. Ces fonctions sont basées soit sur des mesures calculées directement sur les images de sortie – *p. ex.*, « *if number-of-regions<100, then increase the parameter value by 2* » dans OCAPI –, ou à partir de mesures de comparaison entre des images de sortie et des images de référence – *p. ex.*, « *prendre la valeur de seuil qui fournit la meilleure coïncidence entre une image de contour et les frontières des régions obtenues par un opérateur de binarisation* » dans LLVE.

Ce mécanisme doit être explicitement représenté à l'intérieur de l'arbre de tâches. Le flot d'images et le flot de valeur de paramètres doivent être distingués dans l'arbre des tâches.

Règle 4. *L'évaluation est distribuée dans le plan et organisée en hiérarchie.*

L'évaluation automatique est aussi un problème complexe en traitement d'images (Zhang, 1996). Il est alors bénéfique de profiter de la hiérarchie pour distribuer l'évaluation (Clouard et al., 1995b). À chaque tâche, peuvent être associées des règles permettant de vérifier la pertinence de ses résultats – *p. ex.*, « *if :assess too-small object-size and no other-object absent, then :assess ambiguous detection and :failure* » est une règle d'évaluation de la tâche « *coarse-object-detection* » dans OCAPI.

L'intérêt est de pouvoir définir des règles d'évaluation chaque fois que cela est possible et de bénéficier d'une évaluation hiérarchique et distribuée sur le plan : d'abord les résultats de chaque opérateur sont évalués, puis progressivement, l'enchaînement de plusieurs opérateurs et finalement les résultats finaux.

Règle 5. *La correction combine un mécanisme de gestion des erreurs, local et global qui permet la re-spécialisation et la re-planification des parties de plans erronées.*

L'évaluation peut mettre en évidence des tâches non satisfaites. Une phase de correction doit proposer de nouvelles alternatives pour les parties de plan non satisfaites. Un bon mécanisme de gestion des erreurs doit effectuer deux types de correction (Moisan et al., 1995) :

- *Respécialisation* des parties erronées avec de nouvelles valeurs de paramètres.
- *Replanification* par rétro-propagation dans le but de changer les parties erronées avec de nouvelles alternatives.

Une évaluation distribuée rend la localisation des erreurs plus facile. Des règles d'ajustement peuvent être ajoutées pour proposer un nouveau plan d'exécution en changeant les contraintes ou les valeurs de paramètres ou en proposant une rétro-propagation de l'échec vers les niveaux supérieurs.

3 Un système de génération d'application (BORG)

Le système de génération automatique d'applications de traitement d'images que nous avons conçu, se nomme BORG pour « un Blackboard orienté Objet pour la Résolution de problèmes par Génération de plans » (Clouard et al., 1993; Clouard, 1994; Clouard et al., 1999b).

3.1 Objectifs et motivations

BORG est un système à base de connaissances pour la génération d'applications de traitement d'images qui implémente le paradigme du pilotage de codes. Il a été conçu en suivant les prescriptions des règles précédentes.

Le système est sollicité par une formulation de l'objectif faite à l'aide du langage de formulation décrit au chapitre 2 et d'un ensemble d'images tests. Il génère en sortie un graphe d'opérateurs adapté au traitement des images de l'application, qui fournit directement un programme exécutable autonome pouvant être utilisé pour toutes les images de la classe.

3.2 Architecture du système

L'architecture du système, représentée dans la figure 3.2, emprunte celle du Tableau Noir BB1 (Hayes-Roth, 1985) et utilise la bibliothèque d'opérateurs PANDORE. Cette architecture fait la distinction entre la résolution de problèmes de traitement d'images et la résolution du problème du contrôle de la résolution. Pour chaque catégorie, il y a une base de données et une base de connaissances spécifiques.

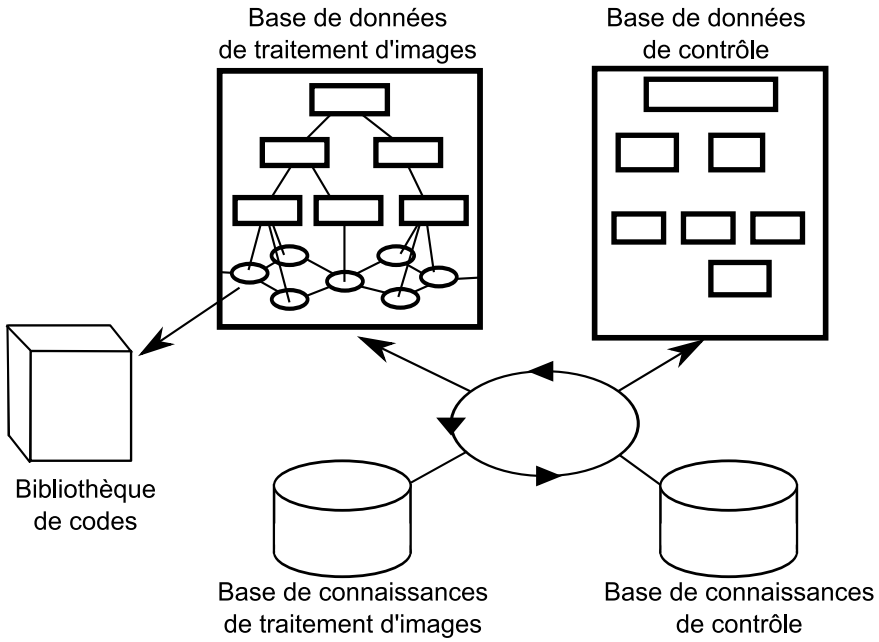


Figure 3.2 – L'architecture du système BORG.

Les bases de données supportent respectivement le plan de traitement d'images et le plan de contrôle de la résolution. Chaque base de connaissances est composée de *Sources de Connaissances*. Une source de connaissances peut être vue comme un module de connaissances de type condition-action qui apporte une brique à la construction de la solution. La partie condition indique les données qui doivent être présentes sur la base de données pour que la partie action puisse s'exécuter. La solution est construite pas à pas par exécution de la partie action de plusieurs sources de connaissances. Différentes alternatives sont représentées par des

sources de connaissances concurrentes. C'est le mécanisme de contrôle qui doit décider quelle alternative utiliser. Une fois le plan créé, il est ensuite essayé sur les images tests, puis évalué. Les parties de plans non satisfaites sont corrigées avec de nouvelles alternatives. Une fois la solution totalement validée, le système peut générer un programme exécutable équivalent au graphe d'opérateurs construit.

3.2.1 La base de données de traitement d'images

La base de données de traitement d'images contient le plan de tâches qui permet de construire le graphe d'opérateurs. Le plan est représenté par un arbre de tâches dont les feuilles sont des opérateurs paramétrés exécutables comme prescrit par la règle 1. Mais ici le plan est entièrement construit sans qu'il n'ait jamais existé explicitement dans la base de connaissances du système. Il n'est donc pas représenté par un arbre de tâches de type ET/OU mais simplement par un arbre de tâches de type ET, puisque c'est directement l'arbre solution.

Au début de la résolution, seule la tâche représentant le problème défini par l'utilisateur est présente sur la base de données. A la fin de la résolution, la base de données contient le plan solution et le graphe d'opérateurs correspondant. Ce graphe est ensuite utilisé pour générer un programme exécutable équivalent.

3.2.2 La base de connaissances de traitement d'images

La construction de l'application relève donc d'un problème de planification conformément à la règle 2. Les sources de connaissances de traitement d'images ont en charge de construire le plan solution, de l'instancier avec des opérateurs, de contrôler son exécution, de l'évaluer et si nécessaire de le corriger en mettant à jour les tâches dans le plan. Chaque source de connaissances n'effectue qu'une seule de ces actions et n'apporte seulement qu'une petite contribution à la solution globale.

3.2.3 La base de données de contrôle

La base de données de contrôle contient les décisions qui seront utilisées pour orienter le choix de la source de connaissances à exécuter à chaque cycle de la résolution. Ce choix résulte d'un compromis entre les actions désirables et les actions faisables (Hayes-Roth, 1985). Les actions désirables sont spécifiées par une stratégie de résolution et des heuristiques de comportement. La stratégie est implémentée par des focus successifs, spécifiant sur quel niveau de la base de données de traitement d'images doit être portée l'attention du moment. Les heuristiques de comportement sont des décisions indépendantes indiquant quel type de source de connaissances doit être privilégié. Les actions faisables sont les sources de connaissances exécutables stockées dans une liste sur la base de données.

3.2.4 La base de connaissances de contrôle

Les sources de connaissances de contrôle peuvent créer, modifier ou détruire des décisions de contrôle pendant le processus de résolution de problèmes. Elles ont pour but de construire à chaque instant le profil de la source de connaissances la plus adaptée à faire progresser la recherche de la solution.

3.2.5 L'ordonnanceur

Les décisions de contrôle sont utilisées par l'ordonnanceur pour calculer la priorité affectée à chaque source de connaissances exécutable. La source de connaissances avec la plus grande priorité est choisie pour l'exécution. Chaque exécution de source de connaissances procède d'un cycle en trois étapes mécaniques :

1. Mettre à jour l'agenda des sources de connaissances activables et calculer leur priorité d'exécution ;
2. Choisir dans l'agenda la source de connaissances avec la plus grande priorité ;
3. Interpréter la source de connaissances choisie.

Il n'y a pas de séparation étanche entre le contrôle de la résolution et la résolution du problème. A chaque cycle de résolution, les sources de connaissances du contrôle sont en concurrence d'exécution avec les sources de connaissances du domaine. Le contrôle de résolution doit donc décider à chaque instant s'il faut modifier le contrôle de résolution lui-même ou faire de la résolution de problème avec l'état courant du contrôle.

3.2.6 La bibliothèque d'opérateurs

Le paradigme du pilotage de codes repose sur le postulat que l'on peut représenter n'importe quelle application de traitement d'images par un réseau d'opérateurs paramétrés et individualisés, s'échangeant des images et des valeurs numériques. La validité de ce postulat est conditionnée d'une part à l'utilisation d'une bibliothèque d'opérateurs représentant un bon compromis entre posséder suffisamment d'opérateurs diversifiés et rester limitée à un nombre raisonnable (Zamperoni, 1996), et d'autre part à l'intégration explicite dans le graphe de mécanismes de contrôle évolués pour réaliser l'adaptation à la variété des images (Matsuyama, 1989).

Pour composer des bibliothèques adaptées au paradigme du pilotage de codes, nous arguons qu'il faut restreindre le plus possible les bibliothèques à des *opérateurs atomiques* (Clouard, 1994). Un opérateur atomique correspond à un programme sur lequel on peut disposer de toute la connaissance sémantique et syntaxique nécessaire à sa sélection, son paramétrage et à l'appréhension de ses effets sur les images. Il n'y a pas de granularité imposée sur le concept d'opérateur. Un opérateur peut être réduit à une opération ponctuelle – *p. ex.*, une addition de deux images – ou à un algorithme très complexe – *p. ex.*, un algorithme de segmentation complet – si :

- ils possèdent chacun un champ d'application parfaitement identifié ;
- ils ont les capacités d'adapter leur comportement à l'étendue de ce champ, ou s'ils présentent des paramètres indépendants permettant d'adapter manuellement leur comportement.

Un algorithme ne sera pas un bon candidat opérateur s'il ne prend pas en compte les particularités de l'environnement dans son exécution, ou s'il possède beaucoup de paramètres corrélés ou trop difficiles à régler.

On peut espérer ainsi que plus un opérateur est atomique, plus il est réutilisable et moins il est nécessaire de disposer d'opérateurs dans la bibliothèque.

4 La base de données de traitement d'images

4.1 Les niveaux d'abstraction

Nous proposons de représenter un plan solution par un arbre de tâches organisé en cinq niveaux d'abstraction. Cette vision s'oppose aux autres approches du pilotage de codes qui utilisent une décomposition fonctionnelle des tâches en sous-tâches. La profondeur d'un arbre peut alors être quelconque. Avec une hiérarchie organisée par abstraction, le lien de décomposition d'une tâche en sous-tâches traduit une relation de raffinement. La différence peut être illustrée avec l'exemple d'une décomposition d'une tâche de segmentation par seuillage binaire. La décomposition fonctionnelle de cette tâche donne deux sous-tâches : « trouver le seuil » puis « binariser ». La tâche « trouver le seuil » est finalement aussi complexe que la tâche de seuillage elle-même. Avec une approche par abstraction, le choix de la valeur de seuil doit forcément être abordé en même temps que le choix de la technique de seuillage. On ne choisit pas un seuillage, mais un seuillage basé sur une technique pour trouver la valeur de seuil, par exemple « *un seuillage qui maximise la variance inter-classe* ». Les sous-tâches sont moins complexes que les tâches parents.

En utilisant une décomposition par niveaux d'abstraction, nous voulons éviter deux écueils dans la représentation des connaissances de planification : associer trop tôt ou trop tard la tâche initiale aux opérateurs. Si la tâche est associée trop tôt, la granularité de la connaissance nécessaire à la planification sera trop importante et donc difficile à définir et à adapter à la variabilité. Si elle est associée trop tard, la granularité des connaissances sera trop fine et donc trop morcelée et difficile à contrôler.

Un autre intérêt d'une organisation de l'arbre en niveaux d'abstraction est qu'elle offre deux visions complémentaires de la solution selon l'axe considéré (Clouard, 2004) (*cf.* figure 3.3) :

- Verticalement, l'arbre représente un plan de traitement. Ce plan explicite les différentes décisions hiérarchiques prises pour construire le graphe d'opérateurs solution.
- Horizontalement, l'arbre représente une hiérarchie de solutions de plus en plus détaillées. A chaque niveau, la solution décrit la chaîne complète des traitements pour le niveau d'abstraction considéré.

Ici, nous prétendons que la représentation de plans de traitement d'images nécessite exactement cinq niveaux d'abstraction successifs, partant de la spécification de l'objectif et débouchant sur son implémentation (*cf.* figure 3.3). De façon décroissante, nous distinguons les niveaux suivants :

Objectif. Un *objectif* représente exactement le problème de traitement d'images soumis au système – *p. ex.*, « *Extraire les noyaux de cellule* ».

Directive. Une *directive* est la spécification d'une étape dans le processus de résolution d'un objectif et résulte du déploiement d'une stratégie de traitement d'images – *p. ex.*, ascendante, descendante, mixte. Elle se formule donc à partir d'un vocabulaire intentionnel emprunté au traitement d'images – *p. ex.*, « *Extraire les marqueurs des régions à segmenter* » ou « *Localiser les régions à partir des germes* ».

Fonctionnalité. Une *fonctionnalité* est la description abstraite d'une classe d'opérations de traitement d'images choisie comme solution pour une directive, telle

que « *Classification de pixels* » ou « *Détection de contours* ». Les contraintes associées agissent comme les paramètres de la fonctionnalité pour en préciser la portée – *p. ex.*, « *Classification de pixels. Nombre de classes = 2* ».

Algorithme. Un *algorithme* décrit l'implantation d'une opération de traitement. Les contraintes associées donnent la façon de paramétrer et de contrôler l'exécution de l'algorithme – *p. ex.*, « *Seuillage binaire, seuil=125* ».

Opérateur Un *opérateur* réifie une commande exécutable de la bibliothèque PANDORE. Il décrit la syntaxe d'appel de la commande et fournit les images d'entrée, de sortie et les valeurs de paramètres nécessaires – *p. ex.*, *pbinarization 125 255 image1.pan image2.pan*.

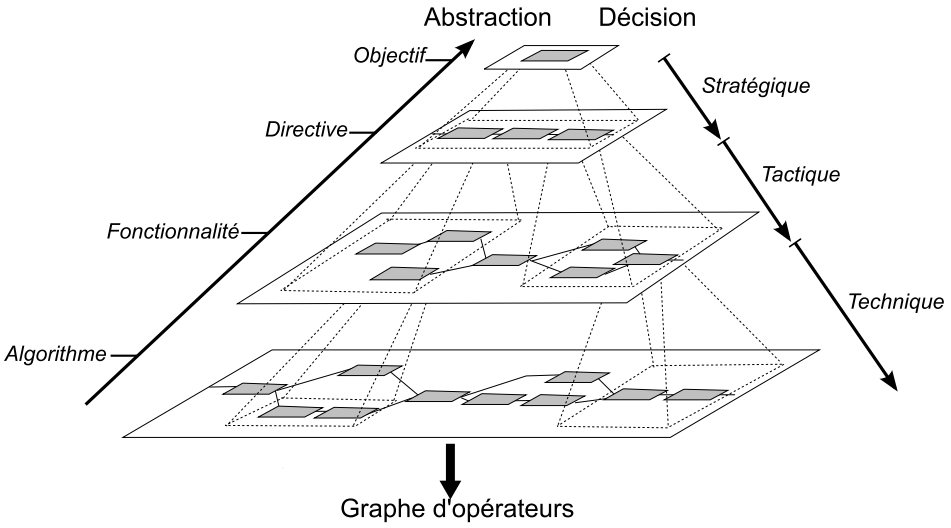


Figure 3.3 – L'architecture du système BORG.

Le choix des cinq niveaux se justifie par le principe d'irréductibilité des niveaux (Pylyshyn, 1984), c'est-à-dire qu'aucun niveau ne peut être confondu dans un autre, et inversement qu'aucun niveau n'en contient d'autres : chaque niveau possède ses propres connaissances qui ne peuvent pas être expliquées par des connaissances des niveaux inférieurs.

Ainsi, le niveau objectif est porteur de concepts du domaine – *p. ex.*, noyaux de cellules – qui disparaissent au niveau directive. Le niveau directive définit des contraintes sur les tâches qui disparaissent au niveau fonctionnalité parce qu'elles sont intégrées implicitement dans le choix d'une décomposition. Le niveau opérateur est spécifique de la bibliothèque d'opérateurs choisie (et peut d'ailleurs être changée avec une bibliothèque présentant les mêmes classes d'opérations).

A l'inverse, les connaissances d'un niveau ne peuvent être expliquées par des connaissances des niveaux supérieurs. D'abord, parce que certaines informations deviennent invisibles en remontant dans les niveaux. C'est le cas, par exemple, des paramètres des algorithmes. Puis,parce que le passage d'un niveau aux niveaux inférieurs relève de choix – *p. ex.*, il existe plusieurs fonctionnalités différentes pour

réaliser la directive « *Extraire les marqueurs des régions* » entre autres en passant par « *Classification des pixels* » ou par « *Détection des H-minima de l'image* ».

4.2 Représentation d'un plan

Les cinq niveaux d'abstraction constituent les cinq niveaux d'organisation de la base de données de traitement d'images. Un plan est représenté par un arbre de tâches de type ET/PUIS sur ces cinq niveaux. Cet arbre marque les relations de décomposition de tâche en sous-tâches aussi bien que les flots de données entre tâches. Les liens de type PUIS regroupe des sous-tâches qui doivent être exécutées séquentiellement (parce que les sorties des unes sont les entrées des suivantes). Les liens de type ET permettent de garder les sous-tâches en parallèle (parce qu'il n'y a pas de dépendance entre leurs entrées et leurs sorties).

Toutes les tâches du plan, quel que soit leur niveau, partagent la même liste d'attributs (*cf.* tableau 3.1). Les attributs *goal*, *constraint* et *image class* pointent vers l'ontologie d'application qui contient la définition de l'objectif de traitement défini par l'utilisateur. Le poids (*weight*) donne une estimation de l'importance de la tâche dans la représentation du plan avec une valeur dans l'intervalle [very-low, low, medium, high, very-high]. La création et la destruction de tâches sont gérées grâce à l'attribut *status* qui prend les valeurs « operative » quand la tâche est active dans la représentation du plan, et « inoperative » quand elle est détruite. Ainsi, il n'y a pas destruction physique des tâches, ce qui permet une analyse a posteriori du raisonnement pour fournir des explications (voir chap. 5). Au niveau opérateur, l'attribut *constraints* contient, soit une valeur pour chacun des paramètres, soit une façon de les récupérer dans des opérateurs en amont.

Tableau 3.1 – Les attributs décrivant une tâche de traitement d'images.

Attribut	Description
Goal	Nom de la la tâche à accomplir.
Constraints	Liste des contraintes attachées à la tâche.
Image class	Pointeur vers la définition de la classe d'images.
Status	Statut de la tâche dans la représentation du plan (*operative*/*inoperative*).
Inputs	Liste des images d'entrée.
Decomposition	La relation de décomposition entre les sous-tâches (ET/PUIS).
Path	Le chemin permettant de récupérer les images de sortie.
Outputs	Liste des images de sortie.
Evaluation	Les règles d'évaluation permettant d'évaluer la qualité des images de sortie.
Result	Le résultat de l'évaluation (*success*/*failure*).
Weight	Estimation de l'importance de la tâche dans la représentation du plan.

4.2.1 Flot de données

Les attributs *inputs* et *outputs* définissent le flot de données (*cf.* figure 3.4). Les images d'entrée d'une tâche correspondent, soit aux entrées de sa tâche parent, soit aux sorties de tâches amonts de la même décomposition. L'attribut *path* est un moyen efficace pour les tâches, de spécifier où trouver ses résultats, parmi les

sorties de ses sous-tâches. Dans le cas d'un opérateur, c'est l'exécution qui produit directement ses sorties.

Une image est représentée ici comme une paire (*image, masque*). Le masque spécifie les zones d'image à traiter. Une même tâche peut être appliquée sur une image entière, ou seulement sur une partie non masquée. Quand les résultats sont remontés dans le plan, il est possible de composer l'image de sortie avec les pixels masqués d'une image et les pixels non masqués d'une autre. De la même façon, tous les opérateurs PANDORE prennent en argument optionnel une carte de régions comme masque.

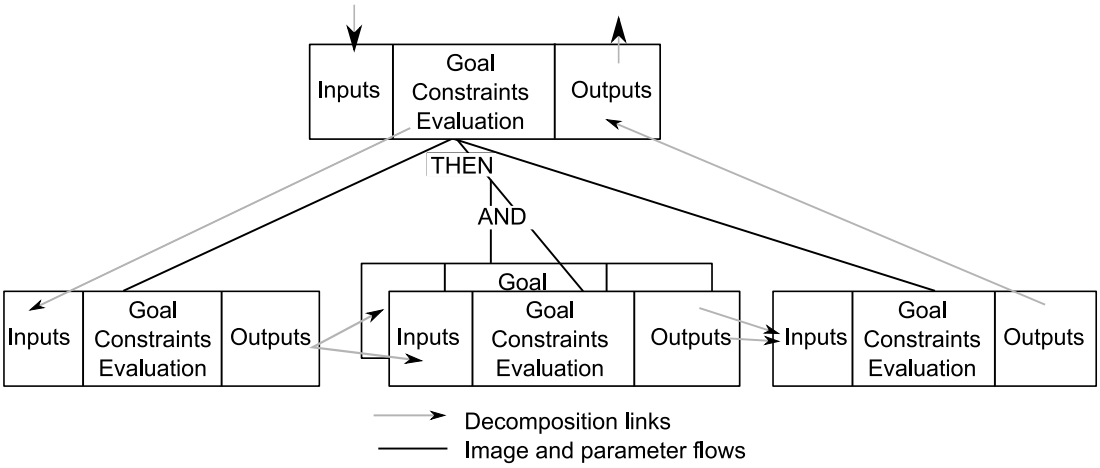


Figure 3.4 – Flot de données échangé entre une tâche et les sous-tâches de sa décomposition.

4.2.2 Mode d'exécution des opérateurs

Les opérateurs sont décrits par trois attributs additionnels : *prototype* contient la syntaxe d'appel des opérateurs et le nombre de ses paramètres, *cost-in-time* donne une estimation de la rapidité d'exécution de l'opérateur avec une valeur parmi [very-low, low, medium, high, very-high], et *mode* indique le mode d'exécution parmi trois (*cf.* règle 3) :

- Le mode *normal* correspond à une exécution unique de l'opérateur ;
- Le mode *loop* correspond à des exécutions successives de l'opérateur avec les mêmes valeurs de paramètre, mais chaque exécution reprend les sorties de l'exécution précédente comme nouvelle entrée ;
- Le mode *optimization* correspond à des exécutions successives de l'opérateur avec les mêmes entrées mais avec différents jeux de valeurs de paramètres. L'opérateur *constraints* spécifie, pour chaque paramètre, l'intervalle de valeurs à essayer et le pas d'incrémentation.

4.2.3 Règles d'évaluation

Les images de sortie d'une tâche sont considérées comme acceptables – *i.e.*, *result=success* – quand elles satisfont les règles d'évaluation définies dans l'attribut *evaluation* (*cf.* règle 4). Chaque tâche est ainsi en position d'évaluer le résultat de sa

propre décomposition en sous-tâches. Il y a deux exceptions à cette règle : au niveau objectif, c'est l'utilisateur qui a défini la requête, qui est en charge d'évaluer le résultat final, et au niveau opérateur, les règles d'évaluation ne sont utilisées que pour contrôler l'exécution des opérateurs. L'évaluation pour les autres niveaux consiste à vérifier que les résultats sont conformes au but, tel qu'il est décrit dans le contexte et les contraintes de la tâche. En général, l'objectif de cette évaluation est essentiellement d'éviter les aberrations de résultat plus que de juger de la qualité intrinsèque, étant donnée la difficulté de définir des mesures de qualité pour les résultats intermédiaires.

Les règles d'évaluation sont des règles de production au format « if then else ». La partie condition porte sur des comparaisons statistiques entre des mesures quantitatives réalisées sur les résultats des tâches et des mesures de référence déduites de la spécification du but. Les mesures sur les résultats sont calculées par des opérateurs PANDORE, pour lesquels il n'y a pas de problème de paramétrage. La partie action des règles consiste simplement en la détermination de la valeur de l'attribut *result* de la tâche, avec la valeur « success » ou « failure ».

4.2.4 Génération du programme exécutable

Le programme exécutable généré est le reflet du graphe d'opérateurs obtenu à la fin de la résolution – *i.e.*, une fois que la tâche initiale a reçu un résultat acceptable. Il est construit avec les mêmes opérateurs que ceux utilisés dans le graphe et inclut aussi les mêmes structures de contrôle.

4.3 Exemple d'un plan en cytologie

La figure 3.5 donne une exemple de plan permettant de construire une application pour celle décrite chap. 2, § 5.3. Ce plan implémente une stratégie de traitement descendante qui consiste d'abord à éliminer la région correspondant au fond d'image, puis à localiser les objets – *i.e.*, les cellules – dans les régions restantes.

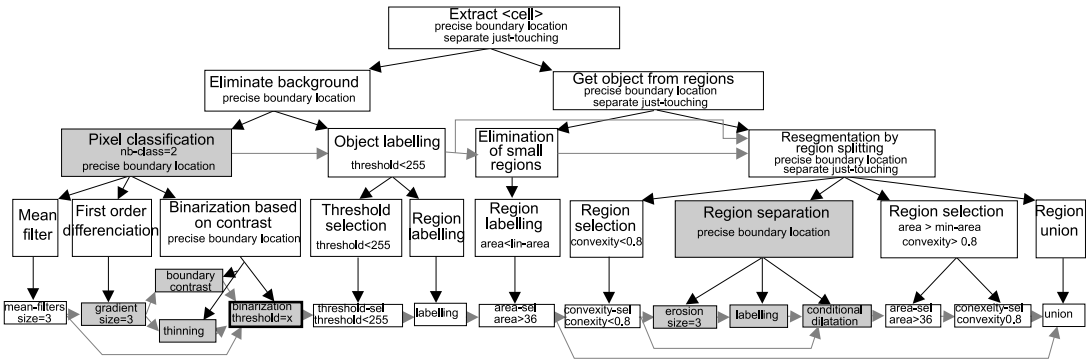


Figure 3.5 – Plan permettant de construire une application de cytologie. Les flots d'images au niveau fonctionnalité et au niveau opérateur sont dessinés avec des flèches grises.

On peut noter dans ce plan que l'opérateur « binarization » est exécuté en mode « optimisation » afin d'optimiser la localisation des frontières des régions (voir figure 3.6). L'intervalle de recherche des valeurs du seuil est construit autour de la

valeur retournée par l'opérateur *boundary-contrast* (notée *d* dans la figure 3.6) et la fonction d'évaluation (notée *f1* dans la figure 3.6) détermine la valeur de seuil qui maximise le nombre de points communs entre les frontières des régions, obtenues par seuillage et les valeurs de gradient qui sont maximale dans la direction du gradient retournées par l'opérateur *thinning*.

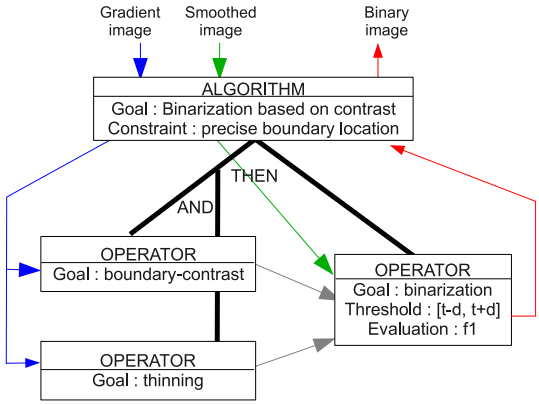


Figure 3.6 – Exécution de l'opérateur « binarization » en mode optimisation.

La séparation des régions utilise la technique de l'ouverture morphologique dans le but de séparer les régions qui se touchent partiellement sans séparer celles qui se chevauchent franchement, mais sans déplacer les frontières obtenues.

Les huit derniers opérateurs prennent dans leur entrée le masque correspondant à la région du fond trouvée dans la première tâche « *Eliminate background* », de façon à focaliser les opérations sur les régions complémentaires du fond.

La règle d'évaluation de la tâche « Pixel classification » vérifie que les images de sorties possèdent bien deux classes, en appelant l'opérateur PANDORE qui compte le nombre de classes dans une image : « *If (pcountclass(output)=2) Then success Else failure* ».

5 La base de connaissances de traitement d'images

Chaque action, contribuant à la construction du plan solution, est effectuée par une source de connaissances. Dans notre base de connaissances, il existe cinq catégories de sources de connaissances : *Planning-KS*, *Instantiation-KS*, *Execution-KS*, *Description-KS*, et *Evaluation-KS*. Avec ces catégories de sources de connaissances, la correction du plan est faite en proposant une nouvelle décomposition pour les tâches erronées, et l'évaluation des résultats est faite en deux temps : d'abord la création des règles d'évaluation par les sources de type *Description-KS*, puis l'évaluation par les sources de type *Evaluation-KS*.

5.1 Modèle de sources de connaissances

Toutes les sources de connaissances de traitement d'images sont représentées par la même liste d'attributs (tableau 3.2). Les sources de connaissances peuvent prendre quatre états différents : *déclenchées*, *exécutables*, *rejetées* et *exécutées*, en

fonction de la partie condition satisfaite. Seules les sources de connaissances ayant les parties *trigger-conditions* et *pre-conditions* vérifiées sont exécutables. La partie action crée, modifie ou détruit des tâches sur le plan. Chaque modification du plan constitue un événement qui déclenchera de nouvelles sources de connaissances.

La base de connaissances est composée de sources de connaissances coopératives et compétitives. Les sources sont coopératives dans le sens où plusieurs sources sont nécessaires pour construire le plan. Elles sont compétitives dans le sens où plusieurs alternatives sont représentées par des sources différentes. Une partie aptitude, représentée par les attributs *Importance*, *Efficiency*, *Complexity* et *Execution-Time*, est utilisée pour estimer les performances a priori de chacune des alternatives.

En pratique, c'est une *KSAR* (Knowledge Source Activation Record) qui est exécutée. Une KSAR est générée à chaque fois qu'un événement déclenche une source. Elle représente la source avec son contexte d'activation, c'est-à-dire les valeurs pour les *condition-variables* et une valeur numérique pour chacune des aptitudes. Une partie des variables est affectée durant le déclenchement et l'autre durant la validation des pré-conditions.

Tableau 3.2 – Les attributs définissant les sources de connaissances de traitement d'images.

Attribut	Description
Trigger-conditions	Predicats décrivant les événements qui déclenchent l'applicabilité.
Pre-conditions	Predicats décrivant les données devant être présentes sur la base de données pour permettre l'exécution.
Obviation-conditions	Predicats décrivant les données du plan qui annulent l'exécution.
Condition-variables	Variables locales utilisées pour l'exécution.
Input-Level	Le niveau de déclenchement.
Output-Level	Le niveau d'action.
Cycle	Le numéro du cycle de déclenchement (âge).
Action	Les règles qui implémentent la contribution sous la forme d'une mise à jour du plan.
Importance	Règles qui quantifient la priorité d'exécution.
Efficiency	Règles qui quantifient l'adéquation de la contribution par rapport aux données du problème.
Complexity	Règles qui quantifient la complexité de la solution apportée en terme de charge cognitive nécessaire à sa mise en œuvre.
Execution-time	Règles qui donnent une estimation du temps d'exécution de la solution proposée.

5.1.1 Propriétés des sources de connaissances

De manière à faciliter l'acquisition des connaissances et à accroître les performances du système, il est essentiel que les sources respectent les propriétés de modularité, d'indépendance, d'autonomie et de spécialisation.

Modularité. Il n'y a pas de granularité a priori sur la taille des connaissances incluses dans la partie action. Simplement, il faut que le problème du contrôle à l'intérieur de la source de connaissances soit complètement résolu (Bachimont, 1992). Cela signifie qu'une source de connaissances code une version unique d'une action sur le plan. Par contre, une source de connaissances doit faire

tous les ajustements nécessaires pour son exécution, en fonction du contexte de l'application et des contraintes

Indépendance. Les sources de connaissances doivent s'ignorer mutuellement. Une source lit des données sur le plan sans savoir quelle source les a créées et produit des modifications sur le plan sans savoir quelle source les utilisera. L'interaction entre les sources n'a lieu qu'à travers les changements sur la base de données.

Autonomie. Une source embarque toute la connaissance nécessaire pour décider quand elle peut contribuer à la résolution et quelle est son aptitude par rapport à l'état courant de la résolution du problème. C'est au contrôle de décider ensuite quelles sources exécuter.

Spécialisation. Une source de connaissances peut prendre ses données dans plusieurs tâches du plan, mais produit son action à un seul niveau du plan. Cela restreint la sémantique des connaissances embarquées à un seul niveau d'abstraction.

5.1.2 Estimation de l'aptitude des sources

L'aptitude des sources de connaissances permet de mesurer la valeur de la contribution proposée. Elle est définie par quatre critères indépendants : l'*importance*, la *complexité*, le *temps d'exécution* et l'*efficacité*.

L'*importance* quantifie la priorité de la contribution dans le processus de construction d'une solution. La *complexité* quantifie la charge cognitive correspondant à la mise en œuvre de la solution développée, par exemple, en terme de nombre d'opérations et de paramètres à gérer. La *vitesse d'exécution* donne une estimation du temps d'exécution de la solution proposée (et non de la source de connaissances). Enfin, l'*efficacité* quantifie l'adéquation entre la contribution de la source et la spécification de la tâche à accomplir.

Chaque attribut contient une fonction qui permet de calculer une valeur numérique entre [0..1], où 0 correspond à une inaptitude totale et 1 à une aptitude totale. Les quatre valeurs caractérisent la contribution de la source pour accomplir une tâche du plan. Elles sont stockées dans la KSAR correspondante.

Si les caractéristiques de complexité, vitesse d'exécution et importance sont des caractéristiques intrinsèques et peuvent être fixées au moment de la création de la source avec des valeurs constantes, la valeur d'efficacité est une caractéristique extrinsèque qui doit être calculée dynamiquement à partir des contraintes associées à la tâche et des caractéristiques de la classe d'images.

Évidemment, puisque les caractéristiques de la classe d'images et les contraintes attendues pour estimer l'efficacité d'une source ne sont pas forcément toutes présentes dans la formulation du problème, il est nécessaire de savoir calculer une valeur d'efficacité, même en l'absence d'une partie de ces informations. C'est pourquoi nous basons le calcul de l'efficacité sur la théorie de l'évidence de Demspter-Shafer (Shafer, 1976), théorie déjà utilisée efficacement pour des problématiques proches par (Bauer, 1994) et (Lefèvre et al., 1996).

Cette théorie permet d'attribuer un degré de confiance à une situation à partir d'un ensemble d'hypothèses, même incomplet. Ce degré s'exprime à l'aide de deux grandeurs, la *crédibilité* et la *plausibilité*. Ces mesures sont établies à partir d'une distribution de probabilité m de l'ensemble Ω de toutes les hypothèses que l'on peut

faire sur le contexte associé à une tâche (classe d'images et contraintes), dans l'intervalle [0,1] et définie telle que : $m(\emptyset) = 0$ et $\sum_{A \subseteq \Omega} m(A) = 1$. Les sous-ensembles $A \subseteq \Omega$ représentent des propositions de mesure de confiance d'une situation donnée qui sont directement supportées par les hypothèses données et encodées par m.

Dans le calcul de l'efficacité d'une source par rapport à l'accomplissement d'une tâche, la crédibilité mesure le degré de certitude totale sur l'efficacité de la source par rapport à l'ensemble des hypothèses formulées dans le contexte A associé à la tâche :

$$Cr_m(A) = \sum_{B \subseteq A} m(B)$$

où $m(B)$ représente la quantité de confiance que l'on peut attribuer exactement à l'hypothèse B dans le calcul de l'efficacité.

La plausibilité mesure le degré de certitude si toutes les hypothèses inconnues allaient dans le sens de l'efficacité. Elle vaut aussi 1 moins la crédibilité du complémentaire de A :

$$Pl_m(A) = \sum_{B \cap A \neq \emptyset} m(B)$$

$$Pl_m(A) = 1 - Cr_m(\neg A)$$

De ce fait, l'efficacité réelle de la source se trouve quelque part entre la crédibilité et la plausibilité, comme cela est illustré par la figure 3.7. Ceci fait que l'attribut *efficacité* d'une source de connaissances donne deux attributs différents *crédibilité* et *plausibilité* dans la KSAR correspondante. Le contrôle peut alors jouer sur deux valeurs pour choisir une source sur la base de son efficacité.

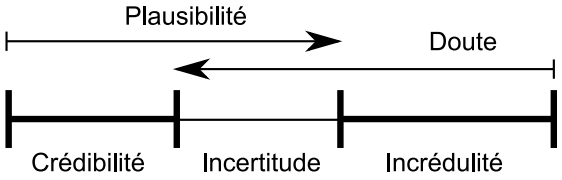


Figure 3.7 – Mesures de l'efficacité d'une source de connaissances.

Nous illustrons ce principe pour le cas d'une source de connaissances de détection de contours, basée sur une différenciation du premier ordre (type Canny-Deriche). Dans le calcul de l'efficacité, il y a deux niveaux : le niveau crédal où les croyances sont modélisées et le niveau pignistique, dans lequel les fonctions de croyance sont transformées en fonctions de probabilités pour la prise de décision.

Au niveau crédal, l'efficacité de la source est décrite comme maximale lorsque le contexte d'exécution, associé à une tâche de détection de contours, présente les caractéristiques données dans la première colonne du tableau 3.3, avec le degré de confiance donné dans la deuxième colonne. Le degré de confiance est donné par une valeur linguistique de l'ensemble {null, very-low, low, medium, high, very-high}. Le dernier critère, *incertitude*, quantifie le degré d'ignorance que l'on a sur l'efficacité de la source. Si l'expert maîtrise parfaitement cette contribution, alors cette valeur = null. Il correspond en fait à la masse qui n'a pu être attribuée à aucune partie de Ω pour la caractérisation de l'efficacité.

En utilisant une quantification des valeurs linguistiques telle que null=0,0, very-low=0,2, low=0,4, medium=0,6, high=0,8 et very high=1,0 ; alors la distribution de probabilité devient celle de la colonne 3 du tableau 3.3. Après normalisation entre 0 et 1, on obtient la distribution de masse donnée par la colonne 4.

Tableau 3.3 – Exemple de la description de l'efficacité d'une source de connaissances de détection de contours basée sur une différenciation du premier ordre.

Critère	Importance	Degré	Degré normalisé
$''good\ localization'' \in constraints$	medium	0,6	0,1
$''good\ detection'' \in constraints$	medium	0,6	0,1
$noise.SignalNoiseRatio >= low$	very-high	1,0	0,166
$noise.PowerSpectralDensity = white$	medium	0,6	0,1
$region.texture = no\ texture$	high	0,8	0,133
$edge.morphology.contrast >= low$	high	0,8	0,133
$edge.morphology.profile = step$	high	0,8	0,133
$edge.orientation \in \{vertical, horizontal\}$	low	0,4	0,066
Incertitude	*low*	0,4	0,066
Total		6	1

Au niveau pignistique, étant donné la spécification d'une tâche de détection de contours dans le plan avec le contexte c_1 simplifié suivant :

c_1={ *constraint="good localization",*
noise.PowerSpectralDensity=white,
noise.SignalNoiseRatio=high,
edge.morphology.profile=step,
edge.morphology.contrast=very low,
edge.orientation=0}

On remarque que toutes les critères ne sont pas renseignés (detection et texture) et que les critères de contraste et d'orientation ne sont pas satisfaits. On obtient alors les crédibilité et la plausibilité suivantes :

$$\begin{aligned} Cr_m(c_1) &= 0,1(localization) + 0,1(PSD) + 0,166(SNR) + 0,133(profile) \\ &= 0,5 \\ Pl_m(c_1) &= 1 - Cr_m(\neg c_1) \\ &= 1 - (0,133(contraste) + 0,066(orientation) \\ &= 0,80 \end{aligned}$$

Pour ce contexte, le score d'efficacité de la source est donc borné par $[0,5; 0,8]$, ce qui en fait une source moyennement efficace (0,5), avec quelques contre-indications (0,20) et beaucoup d'incertitude (0,3).

5.2 Sources de connaissances de planification

Chaque source de connaissances de planification (voir un exemple tableau 3.4) code un savoir-faire pour accomplir une tâche donnée, en proposant une décomposition de cette tâche en une séquence ordonnée de sous-tâches aux niveaux inférieurs. Pour chaque sous-tâche, le but, les contraintes, le poids et le flot de données doivent

Tableau 3.4 – Un exemple de source de connaissances de type Planning-KS : Contrast-Classification effectue une classification de pixels à partir d'une binarisation de l'image basée sur la maximisation du contraste aux frontières des régions.

Attributs	Values
Trigger-conditions	There is a new functionality F where F's goal = (pixel classification) and (nb-classes=2) $\in$ F's contraintes.
Pre-conditions	F's decomposition =()
Obviation-conditions	F's status = inoperative or F's result = success
Condition-variables	(F)
Input-Level	Functionality
Output-Level	Algorithm
Action	Create Algorithm P1 P1's goal = (mean filter) P1's contraintes = () P1's weight = medium P1's inputs = (1^{st} of F's inputs) Create Algorithm P2 P2's goal = (first order differentiation) P2's contraintes = () P2's weight = medium P2's inputs = (1^{st} of P1's outputs) Create Algorithm P3 P3's goal = (binarization based on contrast) P3's contraintes = F's contraintes P3's weight = medium P3's inputs = (1^{st} of P1's outputs, 2^{nd} of P2's outputs, 3^{rd} of P2's outputs) F's decomposition = (THEN P1 P2 P3) F's path = ((1^{st} pixel image of P3's outputs, 1^{st} region map of P3's outputs))
Importance	Medium
Efficiency	((background.texture=no-texture high) (object.edge.morphology.contrast>low high) (object.region.texture=no-texture high) (object.region.lightness<>() very-high) (background.region.lightness<>() very-high) (uncertainty low))
Complexity	Low

être complètement spécifiés. Il y a autant de Planning-KSs attachées à une tâche qu'il y a de façons connues de décomposer cette tâche.

Selon les niveaux d'entrée et de sortie, une source de connaissances de type Planning-KS aborde des décisions de sémantiques différentes. Le raffinement d'une solution d'un niveau d'abstraction au niveau suivant aborde successivement les décisions (*cf.* figure 3.3) :

Stratégique. Une source de connaissances qui décompose une tâche de niveau objectif en sous-tâches au niveau directive, correspond au déploiement d'une stratégie de traitement d'images, type ascendante, descendante, mixte, *p. ex.*, La stratégie de S. Beucher (Beucher, 1992) pour localiser les régions procède de manière ascendante par les directives suivantes : « *Extraire les marqueurs des*

régions » puis « *Construire les régions à partir des marqueurs à l'aide d'une Ligne de Partage des Eaux* ».

Tactique. Une tactique est un engagement sur la façon de mettre en œuvre une directive sous la forme d'une séquence de fonctionnalités, *p. ex.*, La tactique de M. Coster & J-L. Chermant (Coster and Chermant, 1985) pour « *Séparer des régions convexes* » consiste à faire un « *Calcul de l'image de distance aux frontières des régions* » puis « *Extraction des extréma régionaux de la fonction distance* » puis « *Localisation par ligne de partage des eaux à partir des extréma détectés sur l'inverse de l'image de distance* ».

Technique. La détermination des algorithmes se fait par la mise en place de techniques de traitement d'images, *p. ex.*, pour corriger une hétérogénéité spatiale d'illumination, J. Russ (Russ, 1995) propose une technique qui consiste à faire « *Division de l'image à traiter par une image de fond sans objet* » puis « *Recadrage des valeurs pour reprendre le même domaine de valeurs que l'image initiale* ».

5.3 Sources de connaissances d'instanciation

Les sources de connaissances de type Instantiation-KS sont très similaires aux sources de planification, dans le sens où une partie de leur action consiste aussi à faire de la décomposition de tâches, ici d'algorithmes en opérateurs PANDORE. Mais, elles ont aussi en charge l'estimation du coût en temps de l'exécution de l'opérateur, de l'ajustement des valeurs de leurs paramètres ou la façon de les récupérer d'autres opérateurs et la détermination des fonctions d'évaluation pour les opérateurs qui doivent être exécutés en mode « loop » ou en mode « optimization ».

5.4 Sources de connaissances d'exécution

Les sources de type Execution-KS sont responsables de l'exécution des opérateurs PANDORE. Leur rôle est de construire la ligne de commande à envoyer à l'interpréteur Shell, puis de récupérer leurs résultats. Il existe trois sources de connaissances d'exécution, une par mode d'exécution : « normal », « loop » et « optimization ».

5.5 Sources de connaissances de description

Les sources de connaissances de description (voir un exemple tableau 3.5) ont en charge la construction des règles d'évaluation pour les tâches. Il n'y a qu'une seule source de connaissances de ce type pour chaque tâche, parce que l'évaluation ne consiste qu'en une validation, c'est-à-dire que les règles d'évaluation ne dépendent que du contexte et des contraintes des tâches, et pas des alternatives de décomposition qui peuvent être proposées. Nos règles d'évaluation sont construites par appel d'opérateurs d'évaluation PANDORE.

La construction dynamique des règles profite de la nature hiérarchique du plan pour produire des règles adaptées à chaque niveau. En général, aux plus hauts niveaux, les règles portent sur les caractéristiques des objets telles que données dans le contexte d'application, alors qu'aux plus bas niveaux, elles portent plutôt sur des vérifications des contraintes associées au but. Quand aucune règle ne peut être définie, la règle par défaut « IF true THEN success » est utilisée.

Tableau 3.5 – Un exemple de source de connaissances de type Description-KS : Pixel-Classification-Description construit une règle d'évaluation pour le but « pixel classification » et la contrainte « nombre de classe=2 », dans le but de vérifier que le nombre de classes de sortie est bien égal à 2 dans l'image de sortie.

Attribut	Description
Trigger-conditions	There is a new functionality F where F's goal = (pixel classification)
Pre-conditions	()
Obviation-conditions	F's status = inoperative
Condition-variables	(F)
Input-Level	Functionality
Output-Level	Functionality
Action	F's evaluation = IF the attribute nb-classes $\in$ F's contraintes THEN "If (count-classes(F's outputs)=nb-classes) Then success Else failure" ELSE Default-Rule
Importance	High
Efficiency	((high) (uncertainty low))
Complexity	Very-Low

5.6 Sources de connaissances d'évaluation

Les sources de connaissances d'évaluation (voir un exemple tableau 3.6) sont en charge de construire les résultats des tâches à partir de résultats de leurs sous-tâches, puis d'appliquer les règles d'évaluation définies par les sources de connaissances de type Description-KS. Partant du fait que les sources de connaissances d'évaluation ne contiennent pas d'expertise spécifique, une seule Evaluation-KS est suffisante pour chaque niveau hiérarchique. Les sources d'évaluation sont appliquées dans deux cas : quand toutes les sous-tâches sont jugées acceptables ou quand l'une d'elles est déclarée en échec. Dans le premier cas, les sorties sont obtenues grâce à l'attribut *path*, puis les règles d'évaluation sont appliquées à ces résultats pour vérifier que la décomposition a réussi. En cas d'échec, la source détruit la décomposition courante de la tâche, dans le but de permettre une autre alternative de décomposition s'il en existe. Dans le second cas, on procède comme dans le premier quand l'évaluation échoue.

5.7 Une base de connaissances pour traiter des images de cytologie

La figure 3.8 présente quelques unes des sources de connaissances utilisées pour la construction du plan de la figure 3.5. On notera dans cette base, que la source de connaissances de planification « Top-Down-Object-Extraction-KS » implémente une stratégie d'analyse de type descendante : premièrement, éliminer le fond puis construire les objets à partir des régions restantes. Une autre alternative de type analyse ascendante aurait été possible : premièrement détecter les germes (par exemple en détectant les minima de la fonction de niveau de gris), puis exécuter une croissance de région autour de ces germes. La première stratégie a été préférée, parce que son efficacité est plus élevée que la seconde lorsque le fond est homogène et discriminable par son niveau de gris et que sa complexité est plus faible.

La source de planification « Contrast-Classification-KS » a été sélectionnée, parce

Tableau 3.6 – Un exemple d'une source de connaissances de type Evaluation-KS : Functionality-Evaluation construit et évalue le résultat d'une fonctionnalité.

Attribut	Description
Trigger-conditions	There is a new functionality F where F's path $\neq$ ()
Pre-conditions	IF (FOR all tâches T of F's decomposition, T's result = success) OR (there is one task T of F's decomposition / T's result = failure) AND (F's evaluation $\neq$ ())
Obviation-conditions	F's status = inoperative
Condition-variables	(F)
Input-Level	Functionality
Output-Level	Functionality

Action	IF (FOR all tâches T of F's decomposition DO T's result = success)
	THEN F's outputs = Interpret F's path
	F's result = Application of F's evaluation rules
	IF F's result = failure
	THEN FOR each subtâches ST of F's decomposition DO
	ST's status = inoperative
	ELSE F's result = failure
	FOR each subtâches ST of the F's decomposition DO
	ST's status = inoperative
	F's decomposition = ()

Importance	IF (FOR all tâches T of F's decomposition DO T's result = success) THEN high ELSE very-low
Efficiency	((task.evaluation<>() very-high) (uncertainty null))
Complexity	Very-low

que les deux classes correspondant à l'image de fond et aux objets peuvent être séparées sur la simple base de leurs niveaux de gris. Cette séparation est obtenue grâce à une classification de pixels en deux classes, suivie d'une élimination du fond. Une autre alternative, toute aussi efficace, aurait été d'utiliser la variance inter-classe au lieu du contraste comme critère de classification. Dans notre plan, la première source a été sélectionnée, simplement parce qu'elle a été la première à être jugée. Dans le plan, il convient également de remarquer que la source de description « Pixel-Classification-Description-KS » construit une règle d'évaluation pour vérifier que l'image de sortie contient effectivement deux classes de pixels.

6 Le contrôle de résolution

Le contrôle de la résolution est géré par le tableau noir de contrôle.

6.1 La base de données de contrôle

La base de données contient les décisions de contrôle. Elle est structurée autour des niveaux suivants :

Problème. La tâche de ce niveau représente le problème brut soumis par l'utilisateur. On y retrouve l'objectif de traitement, la définition de la classe d'images et les contraintes de contrôle.

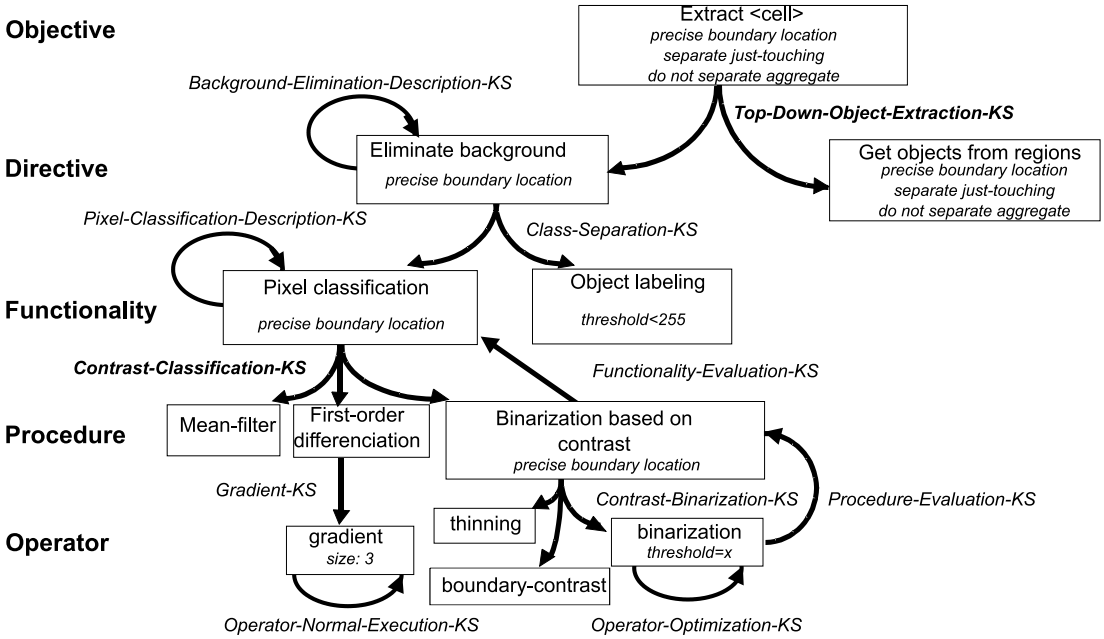


Figure 3.8 – Liste des sources de connaissances utilisées pour construire l'application de cytologie.

Stratégie. Une stratégie décrit les étapes qui rythment le processus de construction de la solution.

Focus. Un focus décrit une étape courante de la stratégie.

Heuristique. Une heuristique décrit le profil des sources de connaissances à privilégier.

Agenda. L'agenda gère la liste des sources de connaissances exécutables.

Élue. Une décision de ce niveau décrit la source qui est choisie pour le cycle en cours.

Les niveaux stratégie, focus et heuristique décrivent le profil de source de connaissance le plus adapté à faire évoluer la solution courante, et les niveaux agenda et élue décrivent la source de connaissances, choisie parmi celles qui sont possibles.

La stratégie de contrôle générale que nous suggérons pour résoudre des problèmes de traitement d'images est une stratégie en profondeur d'abord. Ce choix permet de mieux contrôler la construction de la solution en s'assurant que les premières étapes de traitement sont possibles avant d'entamer le déploiement des suivantes. Cette stratégie est mise en œuvre par des focus d'attention temporaires, successivement sur chacun des niveaux de la base de données de traitement d'images.

Les heuristiques utilisées donnent la préférence :

1. aux sources de connaissances de contrôle sur les sources de connaissances de traitement d'images (préférer réfléchir qu'agir) ;

2. aux sources de connaissances les plus récemment déclenchées, pour un comportement dirigé par les événements, par opposition aux plus anciennes, pour un comportement dirigé par le plan ;
3. en fonction des contraintes de contrôle spécifiées dans la formulation de l'objectif – *p. ex.*, performance et qualité –, les heuristiques peuvent privilégier les sources les plus rapides, ou les sources les plus crédibles, si on recherche une solution fiable, ou les sources les plus plausibles mais les moins complexes, si on recherche une solution robuste.

6.2 La base de connaissances de contrôle

Les sources de connaissances de contrôle construisent le contrôle dynamiquement. Elles ressemblent exactement aux sources de connaissances de traitement d'images avec lesquelles elles partagent le modèle et la liste des attributs. Dans la base de connaissances de contrôle, certaines sources sont génériques car elles ne traitent que du contrôle du contrôle – *p. ex.*, déploiement d'une stratégie sous la forme de focus, mise à jour de l'agenda, détection du succès ou de l'échec de la résolution. D'autres sont spécifiques du domaine du traitement d'images puisqu'elles prennent leur données sur la base de données de traitement d'images – *p. ex.*, choix d'une stratégie, choix des heuristiques.

6.3 Propriétés du contrôle

Grâce à ce type de contrôle, le comportement de résolution est à la fois incrémental et opportuniste. Il est incrémental parce que l'agenda des KSAR en attente est mis à jour à chaque cycle. La sélection des KSAR est basée sur l'état d'avancement de la résolution, ce qui signifie que la résolution s'influence elle-même. Le contrôle est aussi opportuniste parce que le choix de la source de connaissances à exécuter est basé sur la combinaison d'une stratégie générale et de préférences intrinsèques sur des caractéristiques des sources de connaissances. Il ne suit pas strictement une stratégie prédéfinie. Par exemple, une Execution-KS, qui est considérée comme très importante, peut être préférée à une Planning-KS, même si la stratégie est actuellement focalisée sur la planification.

Notre contrôle permet de gérer l'échec local (*cf.* règle 5) par rétro-propagation. Un échec peut être transmis d'une sous-tâche vers sa tâche parent, si aucune des alternatives potentielles n'offre de solution acceptable, ou si l'efficacité des source de connaissances portant ces alternatives est plus faible que celle de la source d'évaluation correspondante. En outre, il n'est pas possible de réexécuter certaines parties d'un plan avec de nouvelles contraintes, contrairement à la prescription de la règle 5, parce que nous considérons ici que chaque source de connaissances propose une décomposition optimale ; si cette décomposition ne réussit pas, cela implique que l'ensemble de la décomposition n'est pas adapté et doit être changé.

Notre contrôle permet aussi de gérer l'échec global (*cf.* règle 5) grâce à la détection des impasses. Il n'est pas nécessaire d'attendre que toutes les alternatives soient exécutées avant que le contrôle ne réagisse. En analysant les valeurs d'aptitude des sources de connaissances en attente, le mécanisme de contrôle peut forcer la propagation de l'échec vers les niveaux supérieurs. Si l'échec est propagé directement à la requête, cela signifie que l'application ne peut être construite par manque d'alternatives fiables.

7 Conclusion

Dans ce chapitre, nous avons étudié le problème de la génération automatique de programmes de traitement d'images. Nous avons conçu un modèle de système qui permet de mettre en œuvre le paradigme du pilotage de codes à partir d'une planification incrémentale et opportuniste. Ce modèle profite des avantages de plusieurs des approches du pilotage de codes proposées dans la littérature, et apporte de nouvelles propositions, dans le but d'améliorer la planification, l'exécution, l'évaluation, le contrôle et la généricité.

Planification. Représenter la connaissance de décomposition par des sources de connaissances apporte une réponse au problème de l'accroissement du nombre d'alternatives associées à une seule tâche, à l'instar des approches multi-agents [1]. Chaque source de connaissances contient seulement la connaissance sur les alternatives de décomposition qu'elle manipule. Elles ont la responsabilité de l'estimation de leur performance, en terme de caractéristiques qui ne dépendent que des données du problème. Cette estimation est complètement indépendante des autres sources de connaissances. C'est le mécanisme de contrôle qui a en charge la sélection des sources à exécuter sur la base de ces valeurs de performance. Ainsi, une augmentation dans la quantité de connaissances n'affecte pas les performances du planifieur, au contraire, elle contribue à l'accroissement de ses capacités. Dans une approche utilisant des règles de choix attachées aux nœuds, la connaissance est axée sur la comparaison des alternatives, et mélange donc l'expertise de résolution et le contrôle de la résolution. Elle résiste alors mal à l'augmentation du nombre d'alternatives, parce que les règles de choix deviennent de plus en plus complexes. Notre approche permet au contraire d'avoir une meilleure visibilité dans l'acquisition et la maintenance des connaissances en séparant les connaissances de résolution de celles du contrôle. La granularité des connaissances est donc indépendante du nombre d'alternatives.

L'autre apport de notre modélisation pour la planification réside dans l'utilisation d'une hiérarchie des connaissances par niveaux d'abstraction. L'acquisition des connaissances ne consiste pas à modéliser des solutions complètes pour résoudre chacun des objectifs, mais plus simplement des briques modulaires, indépendantes et parfaitement localisées de décomposition de tâches pour un niveau d'abstraction donné. Ainsi par exemple, il s'agit de décrire les différentes façons connues de décomposer la fonctionnalité « *détection de contours* » en algorithmes et pour chacune, de définir le contexte d'activation le plus favorable, sans se soucier de la façon dont elles seront utilisées pour construire des plans solutions. Non seulement cela permet de faciliter l'acquisition des connaissances mais cela permet aussi de mieux appréhender la variabilité des objectifs puisque les plans sont construits de manière opportuniste sans que l'ingénieur de la connaissance n'ait eu à prévoir ce cas particulier.

1. Le modèle du tableau noir peut être considéré comme un modèle de type multi-agents, où les agents sont les sources de connaissances. Les sources possèdent, en effet, toutes les caractéristiques des agents : autonomie, modularité, indépendance et spécialisation. La différence conceptuelle provient du mécanisme de collaboration entre ces agents, où les sources ne peuvent communiquer que par l'intermédiaire du tableau noir, alors que les agents rationnels communiquent directement avec eux (avec ses accointances).

Exécution. Notre contrôle de l'exécution des opérateurs intègre explicitement tous les types possibles d'exécution mentionnés dans les autres systèmes : exécutions itératives et répétitives comme dans OCAPI, ajustement des valeurs de paramètres à travers l'optimisation à partir de règles d'évaluation comme dans LLVE, et la simple satisfaction comme dans OCAPI.

Évaluation. Comme dans OCAPI, l'utilisation de règles d'évaluation délocalisées, distribuées dans toutes les tâches rend leur formulation plus facile et permet l'utilisation de critères à plusieurs niveaux d'abstraction. Même s'il reste particulièrement difficile de trouver des règles dans le domaine du traitement d'images, le fait que notre mécanisme d'évaluation soit à la fois hiérarchique et délocalisé, constitue incontestablement une aide à la formulation de ces règles. En outre, BORG fixe cinq niveaux de hiérarchie de tâches, ce qui fournit une typologie naturelle des règles. La formulation de critères d'évaluation est fondée sur le vocabulaire utilisé pour la spécification des tâches avec le même niveau d'abstraction.

Contrôle. La correction des erreurs est décidée dynamiquement par le contrôle. Les alternatives de décomposition d'une tâche sont en compétition avec l'évaluation qui a en charge la propagation de l'échec vers les niveaux supérieurs. Ainsi, seules les alternatives portées par des sources de connaissances ayant une priorité supérieure à celles qui propagent l'échec sont conservées. Le calcul de la priorité est basé sur les préférences définies par l'utilisateur (heuristiques) et l'actuel état de la résolution (focus d'attention). Cela permet de pouvoir combiner des informations de niveau local et global pour décider du contrôle de la résolution. Dans les autres systèmes, ce mécanisme de contrôle est soit statique : essayer toutes les solutions alternatives avant de propager l'échec, soit fait par directement par la règle de choix qui décide elle-même de la propagation de l'échec. Le mélange entre les connaissances de résolution et de contrôle, rend difficile, non seulement, l'acquisition des connaissances, mais aussi l'adaptation du comportement de la résolution aux contraintes de contrôle plus globales qui peuvent être définies avec les objectifs de l'application (robustesse, fiabilité, etc.).

Généricité. L'architecture du système permet d'envisager de piloter d'autres systèmes de traitement d'images, en particulier les systèmes purement numériques basés image, qui possèdent des capacités d'apprentissage à partir d'exemples. Pour piloter de tels systèmes, cette fois le plan peut associer directement la tâche de niveau objectif à des tâches de niveau algorithme sans passer par les niveaux intermédiaires. Le plan se limite alors à un seul opérateur.

Cette architecture permet aussi de profiter de mécanismes d'optimisation de réglage de paramètres, tels que le proposent V. Martin (Martin, 2007) ou R. Landais (Landais, 2006). Cela peut se réaliser assez facilement une fois le graphe d'opérateurs construit. Il s'agit d'exhiber tous les paramètres d'opérateurs à valeur fixe, puis de chercher le jeu de valeurs autour de ces valeurs qui permet d'optimiser une métrique d'évaluation, en comparant les résultats obtenus sur des images tests avec des images de référence.

Toutefois, au même titre que les autres systèmes utilisant un modèle de raisonnement riche en connaissances, le goulot d'étranglement reste l'acquisition des connaissances. Cette question sera traitée au chapitre 5.

CHAPITRE 4

Interaction Homme-Machine

Ce chapitre porte sur le processus complet de construction d'applications par les utilisateurs. Elle montre comment la construction d'applications résulte en fait de cycles d'interaction entre l'utilisateur et le système de génération de programme. Le but est de construire une solution par affinages successifs en utilisant le retour de pertinence, grâce à l'exécution de prototypes intermédiaires sur des images tests.

Au cours de ces cycles, l'utilisateur apprend la sémantique donnée aux primitives du langage de formulation mis à sa disposition, et le système apprend la confiance que l'utilisateur accorde aux éléments de sa formulation.

Les travaux exposés dans ce chapitre sont les plus récents du projet. Le co-développement d'application est apparu comme une évidence pour produire des solutions qui soient vraiment en accord avec les besoins des utilisateurs. Mais cela nécessitait de disposer au préalable d'un système de génération d'application et d'un langage de formulation d'objectifs. La maquette du système d'interaction a été réalisée sur la base du travail de thèse d'Arnaud Renouf, et en partie finalisée par un stage de master de recherche et un stage d'ingénieurs.

1 Introduction

Les travaux sur la formulation d'objectifs présentés au chapitre 2 ont permis d'identifier les informations que nous considérons comme nécessaires et suffisantes pour conduire le raisonnement de développement d'applications. Ces informations sont opérationnalisées dans une ontologie de domaine qui fournit un langage mis à la disposition des utilisateurs pour spécifier leurs besoins. Les formulations résultantes sont directement utilisables par des systèmes de génération de programmes, tel celui présenté au chapitre 3, pour construire des programmes d'application.

Nous nous intéressons maintenant à la façon d'acquérir ces connaissances auprès des utilisateurs pour formuler des objectifs particuliers et construire des programmes d'applications correspondants. Nous montrons dans la section 2, que la formulation est une activité particulièrement complexe en traitement d'images et qu'elle oblige à recourir à une interaction entre le système et l'utilisateur, de manière à construire une application par affinages successifs sur la base de résultats évaluables.

Dans la section 3, nous présentons le modèle conceptuel de l'interaction proposé

qui, pour faciliter la construction d'une formulation linguistique de l'objectif, s'appuie d'abord sur la construction d'une formulation faite à partir d'exemples. La définition linguistique est ensuite affinée pour faire émerger progressivement la sémantique de l'application en profitant des résultats produits par les exécutions des prototypes construits pour chaque version de la formulation.

La section 4 détaille les étapes successives qui rythment le cycle d'interaction et qui visent à guider l'utilisateur dans sa formulation. Ce cycle est alors mis en œuvre par une interface graphique. Le résultat d'une interaction est un programme d'application qui représente le meilleur compromis entre les intentions de l'utilisateur et les capacités de traitement du système.

2 Le problème de la formulation

2.1 Composition d'une formulation

Nous l'avons souligné dans le chapitre 2, une définition linguistique des objectifs permet une meilleure représentation des besoins exprimables par les utilisateurs. Cette définition est aussi directement utilisable pour conduire un raisonnement abstrait de développement d'un programme adapté, tel que présenté au chapitre 3. Mais, toute la difficulté pour les utilisateurs est de sélectionner les descripteurs et d'affecter les valeurs qui traduisent au mieux leurs besoins.

Or, la spécification de ces informations est regardée comme une activité particulièrement complexe, puisqu'elle est de *nature qualitative* et qu'elle résulte de choix subjectifs. Cela implique qu'il ne peut pas exister de formulation exhaustive et exacte de l'application, mais simplement une caractérisation approchée du comportement désiré (Reichgelt, 1991). C'est la conséquence des fossés sensoriel et sémantique, qui séparent la scène réelle de sa représentation linguistique (*cf.* figure 4.1).

Le fossé sensoriel est la différence entre les objets dans la scène et leur représentation dans les images :

> « The sensory gap is the gap between the object in the world and the information in a (computational) description derived from a recording of that scene. » (Smeulders et al., 2000, p. 1352)

Le fossé sémantique est la différence entre l'interprétation de la scène que quelqu'un peut faire à partir d'une représentation image et l'interprétation faite à partir d'une description de l'image en terme de caractéristiques bas niveau :

> « The semantic gap is the lack of coincidence between the information that one can extract from the visual data and the interpretation that the same data have for a user in a given situation. » (Smeulders et al., 2000, p. 1353)

En conséquence, formuler signifie fournir les informations qui permettent de franchir ces deux fossés avec le moins de perte possible sur l'apparence et la sémantique [1] de la scène réelle, pour permettre la construction d'un programme d'application satisfaisant.

Une formulation est la représentation d'un point de vue personnel sur le problème à résoudre, exprimé par un utilisateur. De ce fait, il n'y a donc aucun moyen automatique de vérifier la pertinence d'une formulation ou de détecter toute dérive conceptuelle. On ne peut qu'identifier six propriétés qualitatives de ce que doit être une « bonne » formulation (Renouf, 2007) :

1. au sens phénoménologique fixé au chapitre 2.

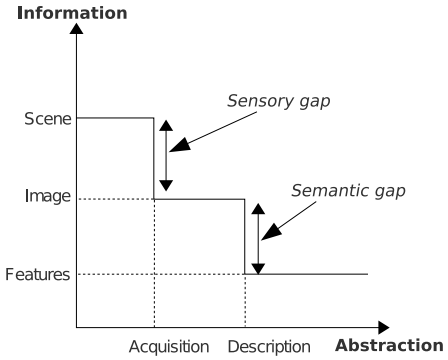


Figure 4.1 – Une représentation des fossés sensoriel et sémantique.

- *représentative.* La formulation garde une forte correspondance avec les objets réels ;
- *discriminante.* Elle contient assez d'information pour discriminer les objets entre eux ;
- *robuste.* Elle représente bien la variabilité des images et des objets ;
- *compacte.* Elle est réduite aux informations essentielles ;
- *précise.* Elle permet une bonne identification des objets ;
- *dense.* Elle caractérise tous les objets de la classe d'images et leur variabilité.

2.2 Rôle de l'interface

C'est donc le rôle de l'interface du système de guider l'utilisateur pour l'amener à construire une « bonne » formulation. Cette interface doit être plus qu'un simple panneau de contrôle créé a posteriori et qui expose les éléments identifiés comme nécessaires pour mettre en œuvre les solutions préexistantes stockées dans la base de connaissances. Par exemple, le système MVP (Chien and Mortensen, 1996) présente une interface graphique avec la liste des tâches reconnues par le système et des champs textes pour rentrer les valeurs des options possibles. Le système VSDE (Bodington, 1995) propose une interface plus sophistiquée. Elle utilise une série de questions pour spécifier la tâche à accomplir et des images exemples à partir desquelles sont automatiquement extraites les valeurs des caractéristiques d'une liste prédéfinie. Mais dans tous ces cas, les utilisateurs croient formuler leur problème à travers l'interface mais, en fait, ils ne font que choisir une solution toute faite. Les solutions sont cachées sous un nom de tâche prédéfinie et les paramètres de configuration de la solution sont cachés sous des caractéristiques images prédéfinies.

Le rôle assigné à l'interface doit être plus important. En effet pour faire face à des problèmes non prévus et avec la plus grande pertinence, la formulation des objectifs ne doit pas être guidée par des contraintes de mise en œuvre de solutions prédéfinies, mais au contraire guidée par une logique d'interaction plus proche des intentions des utilisateurs. Les solutions doivent alors être constuites dynamiquement par complémentarité entre l'utilisateur et la machine (Zou and Nagy, 2006). Dans ce couple, l'utilisateur apporte la connaissance du problème à résoudre et son habileté à évaluer les résultats finaux, tandis que la machine apporte sa capacité à effectuer des traitements sur les images. Le lien entre l'utilisateur et la machine est fait par l'intermédiaire d'un langage d'interaction.

Cette idée d'interaction homme-machine a été mise en œuvre depuis longtemps dans le cadre de la segmentation interactive d'images – *p. ex.*, le système de J.K. Udupa (Udupa, 1982). Mais, elle reste difficile à implémenter pour la construction interactive d'applications de traitement d'images. La principale difficulté provient de le nécessité de construire une solution qui soit adaptée à toute une classe d'images, et plus seulement à une image. Cela exige une définition du problème qui puisse capturer la variabilité des données dans les images.

Quelques travaux ont toutefois été entrepris avec l'idée de la complémentarité entre l'utilisateur et le système pour le développement itératif d'applications de traitement d'images basé sur une description linguistique de l'application, notamment en France (Capdevielle, 1995; Capdevielle, 1995; Saidali et al., 2002; Saidali et al., 2002; Hudelot and Thonnat, 2003; Hudelot, 2005). On notera en particulier l'approche originale proposée par l'équipe autour de P. Dalle (Nouvel, 2002; Nouvel and Dalle, 2002; Abchiche et al., 2002), qui fonde la formulation d'objectifs sur une construction interactive de la définition des concepts du domaine d'application. L'utilisateur compose ses concepts par spécialisation de concepts plus primitifs créés à partir des résultats produits par l'enchaînement d'opérateurs de traitement d'images. Par exemple, le concept de « grand os du poignet » est une spécialisation de « grand objet clair », qui est lui-même un « objet clair » muni d'une mesure de taille. Enfin, un « objet clair » est le résultat d'un opération de seuillage de l'image initiale.

S. Santini et al. (Santini et al., 2001) proposent une autre vision de la formulation par interaction, dans le cas particulier de la recherche d'images par le contenu. Cette fois, la formulation, n'utilise pas de représentation explicite du problème (ni symbolique, ni iconique). L'idée est de profiter de l'interaction pour faire émerger progressivement la sémantique de la recherche, c'est-à-dire définir quelles sont les informations de l'image exemple qui doivent être recherchées dans les autres images, sans qu'il n'y ait de représentation de l'objectif. Par exemple, dans l'image figure 4.2, le but de la recherche est-il de retrouver des images du Mont Saint-Michel ou de course à pied ? Pour répondre à cette question, les auteurs proposent d'utiliser l'interaction pour guider le système dans la spécification des informations qu'il faut considérer comme pertinentes. À partir de la requête, le système retourne des exemples d'images qu'il considère comme « semblables » au sens d'un ensemble de valeurs de caractéristiques communes. Grâce à un langage graphique, l'utilisateur a la possibilité de construire des regroupements d'images pour préciser le concept de la recherche – *p. ex.*, le concept de Mont Saint-Michel ou celui de course à pied – conduisant alors le système à changer sa définition interne de la notion de ressemblance.

Même s'il n'utilise pas le nom, cette approche est proche du « paradigme de l'énaction », tel que défini par F. Varela (Varela et al., 1991). L'interaction énactive ne se fonde ni sur une représentation symbolique, ni sur une représentation iconique de l'information d'interaction, ici entre l'utilisateur et le système. Elle est centrée sur l'action pour construire progressivement une définition interne de la sémantique du problème à résoudre dans le système, qui n'est a priori pas accessible à l'utilisateur.

Toutefois, à l'heure actuelle, ce paradigme reste difficile à mettre en œuvre pour concevoir des systèmes cognitifs et ses apports restent encore limités (Vernon, 2007). Mais des travaux de recherche sont en cours, notamment autour des interfaces homme-machine (Raymaekers, 2009), ou en incubation, notamment autour de la reconnaissance des formes, comme le révèle l'atelier « Enaction : une voie possible pour un dialogue entre sciences informatiques et sciences cognitives » organisé par J. Labiche lors de RFIA 2008.



Figure 4.2 – Exemple d'une requête pour la recherche d'images par le contenu.

3 Notre approche

Par rapport à l'approche de l'équipe de P. Dalle (Nouvel, 2002; Nouvel and Dalle, 2002), que l'on peut considérer comme ascendante, notre approche est plutôt descendante et procède par sélection des caractéristiques qui définissent au mieux les concepts du domaine parmi toutes celles possibles. Un concept est représenté par une primitive visuelle et l'ontologie de domaine fournit l'ensemble des caractéristiques qui lui sont associables. Le rôle de l'utilisateur est alors de sélectionner la bonne primitive visuelle puis les bonnes caractéristiques et de leur assigner les bonnes valeurs.

Le système de traitement d'images complet consiste en deux sous-systèmes (Renouf et al., 2007a) schématisé par la figure 4.3 :

- un sous-système de formulation basé sur une interface graphique nommé HERMÈS ;
- un sous-système de génération de programmes qui opère à partir d'une formulation du problème. On retrouve à ce niveau, le système BORG présenté dans le chapitre précédent.

Pour construire une application, le système fonctionne en deux temps. Il est d'abord utilisé « off-line » pour construire la formulation de l'objectif et un programme d'application correspondant. Puis, le système configuré est utilisé « on-line » pour traiter en routine toute image de la classe, ou si un programme a été généré à partir du graphe d'opérateurs, c'est le programme qui est directement utilisé en routine pour traiter les images.

C'est en mode « off-line » que le système utilise l'interaction Homme-Machine ; en mode « on-line » le système, ou le programme généré, fonctionne automatiquement. L'utilisateur formule l'objectif de traitement d'images par le biais de l'interface du système de formulation HERMÈS. Le système de formulation sollicite ensuite le système de génération de programmes par une requête qui contient la formulation et

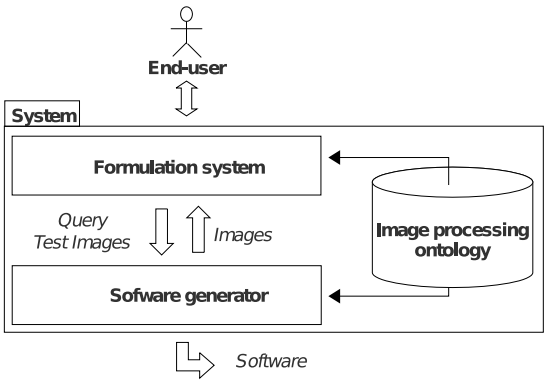


Figure 4.3 – Architecture du système interactif de génération d'application.

des images tests. Le système de génération construit l'application correspondante et retourne les images résultats obtenues par l'exécution de l'application sur les images tests. A la vue des résultats, l'utilisateur peut reconsidérer la formulation et soumettre une nouvelle requête au générateur de programme avec une représentation affinée du problème. Le cycle de formulation s'arrête dès que l'utilisateur estime que les images résultats sont acceptables.

Dans l'état actuel du système, les utilisateurs sont considérés comme des experts du traitement d'images, parce qu'ils doivent manipuler des symboles ayant une interprétation en traitement d'images – *p. ex.*, contours, régions, couleur, bruit, quantification – et surtout savoir interpréter leurs implications sur la construction des résultats.

3.1 Composition des requêtes

La représentation d'une requête est faite à partir de l'ontologie de domaine présentée au chapitre 2. Elle correspond donc à une ontologie d'application qui instancie des descripteurs linguistiques de l'ontologie de domaine.

Le générateur d'application, tel que présenté dans le chapitre 3, requiert une représentation linguistique du problème qui soit, de plus, ancrée dans les données images pour conduire un raisonnement abstrait de construction d'une solution. Une expression linguistique est une représentation intensionnelle explicite du but et de la sémantique de l'application, construite comme un ensemble organisé de symboles. Les valeurs de symboles doivent être ancrées dans les données pour fournir une représentation directement en relation avec l'apparence visuelle des informations du monde réel (Town, 2006). Schématiquement, le raisonnement abstrait du système de génération d'applications utilise des symboles de la requête pour décider de la stratégie, des tactiques et des techniques à mettre en œuvre pour composer une solution, tandis que leurs valeurs sont utilisées pour paramétrer et contrôler l'exécution de la solution. Par exemple, le système pourrait faire le choix d'une segmentation basée sur la classification couleur des pixels si l'utilisateur décrit les objets d'intérêt par leur couleur ; il pourrait utiliser l'intervalle des valeurs de couleur acceptables pour paramétrer et contrôler la classification des pixels.

3.1.1 Exemple d'une application

Afin de fournir les illustrations de ce chapitre, nous prenons l'exemple d'une application d'analyse d'images biomédicales développée dans l'équipe et décrite dans (Lezoray and Cardot, 2000) Cette application rentre dans le cadre du dépistage du cancer.

Le but est d'assister les cyto-techniciens dans la recherche de cellules suspectes ou anormales à partir d'images de cytologie des séreuses. L'objectif spécifique du traitement d'images est de segmenter l'image pour localiser avec précision les régions correspondant aux cellules présentes sur une lame. Ces régions sont ensuite données en entrée d'un classifieur qui sera entraîné pour déterminer le type de chacune des cellules – *p. ex.*, cellules de séreuse, globules rouges –, à partir de caractéristiques extraites des régions.

La classe d'images (dont la figure 4.4 présente quatre exemples miniaturisés) est composée d'images acquises avec un microscope optique équipé d'une caméra vidéo couleur.

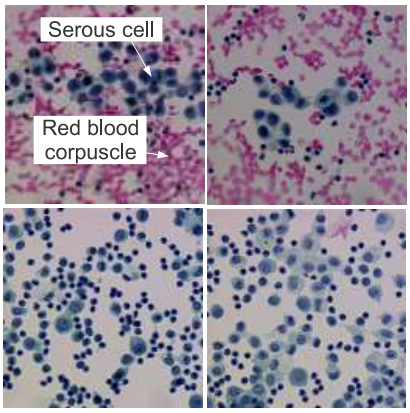


Figure 4.4 – Quatre exemples d'images de l'application de cytologie de séreuse. (*Images fournies par le département de cytologie et d'anatomie pathologique de l'hôpital publique du Cotentin.*)

3.1.2 Le problème du cadre (Frame Problem)

Un problème de traitement d'images concret implique très souvent plus d'une des tâches identifiées au chapitre 1. Ces tâches sont choisies et organisées, de façon séquentielle ou parallèle, par les utilisateurs, quant à leur pratique de l'analyse du contenu des images. Par exemple, pour l'application de séreuse, le cyto-technicien a estimé que la localisation des cellules de séreuse nécessitait l'enchaînement des quatre tâches suivantes :

1. « *Correct <illumination>* », pour compenser l'hétérogénéité spatiale de l'illumination sur la lame ;
2. « *Detect <serous cell nucleus>* », pour stopper l'application s'il n'y a pas de cellule sur la lame ;
3. « *Eliminate <red blood corpuscles>* », parce que les globules rouges peuvent nuire à la localisation des frontières des cellules ;
4. « *Extract <serous cell nucleus>* », pour localiser les noyaux de cellule de séreuse.

L'enchaînement de plusieurs tâches force le système à mettre à jour la définition de la classe d'images après chaque accomplissement d'une tâche, afin d'accomplir les tâches suivantes. Le problème du cadre (angl. *frame problem*) survient lorsque le système doit décrire les effets que les actions provoquent sur la définition initiale de la classe. Par exemple, la tâche « correct illumination » peut affecter la couleur des globules rouges et donc changer la définition de la classe d'images de la tâche qui suit « Eliminate red blood corpuscles ». Malheureusement, la détermination des effets des opérations de traitement sur les images est connue pour être un problème très difficile, voire impossible.

Pour éviter le problème de cadre, nous considérons chaque tâche séparément et donc ce sont les utilisateurs du système (qui peuvent être d'autres systèmes de plus haut niveau) qui doivent fournir une définition de la classe d'images propre à chacune des tâches.

3.1.3 Le problème de la variabilité

La variabilité des images au sein d'une classe est un problème inhérent en traitement d'images, et contribue à la complexité de la formulation. Considérons par exemple le cas d'une utilisation quotidienne d'une application d'imagerie aérienne. La végétation change significativement au cours de l'année et, par conséquent, les images acquises en hiver n'ont pas les mêmes caractéristiques que les images acquises en été.

Pour réduire cette variabilité, V. Martin et al. (Martin, 2007) proposent de diviser la classe d'images en plusieurs contextes (typiquement moins de six). Pour les images fixes, les contextes sont automatiquement obtenus par un algorithme de clustering non supervisé, tel que « Density-Based Spatial clustering » (Ester et al., 1996). Cet algorithme est basé sur la couleur des histogrammes et est entraîné à partir d'images exemples.

Par conséquent, on doit considérer autant d'applications différentes, et donc autant de formulation d'objectifs, qu'il existe de contextes. Chaque image à traiter doit d'abord être dirigée vers la bonne application grâce au classifieur. Appliqué à notre exemple, le classifieur distingue deux contextes : hémorragique du à la présence des globules rouges telles que les deux images en haut de la figure 4.4, ou homogène sans globule rouge, telles que les deux images en bas de la figure 4.4.

3.2 D'une représentation extensionnelle du problème à une représentation intensionnelle

Même réduite à une tâche et un contexte, une description linguistique de l'objectif, qui soit en plus ancrée dans les données, reste difficile à donner pour un utilisateur. Dans un premier temps, nous proposons d'aider l'utilisateur à produire une définition linguistique de la classe d'images par intension en s'appuyant sur une définition initiale donnée par extension, c'est-à-dire à partir d'images exemples.

3.2.1 Formulation par extension

Une définition extensionnelle d'une classe d'images présente les deux avantages d'être, dans la plupart des cas, relativement facile à fournir par l'utilisateur et d'offrir une solution au problème de l'enracinement des symboles (*cf.* chap. 2, §2.3). La

définition extensionnelle de la classe d'images utilise des blobs pour décrire un objet d'intérêt – *p. ex.*, noyau de cellule de séreuse –, ou une caractéristique physique des images – *p. ex.*, le modèle d'illumination de la scène, le bruit.

Une fois que l'utilisateur a identifié les concepts de son application – *i.e.*, objets d'intérêt et caractéristiques images –, il utilise une interface graphique dédiée qui permet de dessiner les blobs directement sur les images et construire , si nécessaire, la définition extensionnelle pour chacun de ces concepts.

3.2.2 Formulation par intension

La définition intensionnelle est construite en utilisant une extraction automatique de caractéristiques pour chaque concept qui a été décrit avec des blobs dans la définition extensionnelle. On obtient alors une liste de caractéristiques valuées pour le concept décrit. La liste des caractéristiques à extraire est donnée par l'ontologie de domaine et dépend de la primitive visuelle utilisée pour définir le concept. Par exemple, si l'utilisateur décrit, à partir de blobs, le concept de « noyau de cellule de séreuse » comme une région, le système devra automatiquement calculer toutes les caractéristiques des régions telles que la colorimétrie, la photométrie, la morphologie, la taille, etc. La variabilité de l'apparence du concept dans les blobs est capturée par un intervalle de valeurs numériques extrêmes pour chacune des caractéristiques (ou une union d'intervalles).

Une requête soumise au générateur de programme garde ensemble les deux versions de la définition de la classe d'images. La définition par intension est utilisée pour conduire le raisonnement abstrait de construction de la solution, et la définition par extension est utilisée pour mettre en œuvre la solution, où des opérateurs du graphe peuvent utiliser les blobs pour calculer leurs propres caractéristiques lors de la réalisation de leur traitement – *p. ex.*, des opérateurs utilisant un apprentissage supervisé ou des opérateurs de segmentation basés texture.

3.3 Sémantique émergente

La sémantique d'une application est représentée dans notre approche par la liste des concepts identifiés dans le domaine d'application et par une description de ces concepts à l'aide de descripteurs valués. Parce que le système n'a pas connaissance a priori du domaine d'application, il représente, par défaut, chaque concept par la liste de tous les descripteurs associés à la primitive visuelle correspondante, tels que définis dans l'ontologie de domaine. Mais certains descripteurs n'ont pas forcément de sens pour une application spécifique. Par exemple, l'objet d'intérêt « noyau de cellule de séreuse » est décrit, entre autres, par une caractéristique d'orientation après l'extraction automatique de caractéristiques. Mais le cyto-technicien sait que l'orientation n'est pas une caractéristique intrinsèque de l'apparence visuelle des noyaux de cellules de séreuse. Les descripteurs d'orientation doivent donc être éliminés de la formulation.

Ceci fait que, après la phase d'extraction automatique de caractéristiques, le système invite l'utilisateur à construire la sémantique de son application en écartant les descripteurs qu'il juge non pertinents dans la formulation initiale. Cette tâche nécessite évidemment que l'utilisateur soit un expert de traitement d'images.

Toutefois, même pour un expert, cette tâche reste difficile. La difficulté vient du fait que l'utilisateur n'a pas accès à l'interprétation donnée aux descripteurs et donc

à la façon dont ils sont utilisés pour construire la solution. On est confronté ici au fameux problème de l'ancrage des symboles (angl. *symbol grounding problem*), qui est définit comme le problème de l'accès au sens d'un système de symboles (Harnard, 1990). Dans notre système, l'interprétation des symboles de l'ontologie est donnée par l'ingénieur de la connaissance qui est responsable de la base de connaissances du système de génération de programmes et elle n'est pas accessible directement aux utilisateurs.

La sélection de caractéristiques pourrait être faite automatiquement par un algorithme (Guyon and Elisseeff, 2003; Liu and Yu, 2005). Mais nous souhaitons que l'utilisateur reste pleinement maître de sa formulation. Une sélection automatique nécessite l'expression d'une fonction d'évaluation. Cependant, soit cette fonction est fixée et donc une partie de la sémantique est imposée, soit elle est construite pour chaque application, mais cela devient un problème à part entière. La sélection des caractéristiques est donc laissée à l'utilisateur.

3.3.1 Construction émergente de la sémantique

Pour donner accès au sens des descripteurs, il est nécessaire d'inscrire la formulation dans une négociation entre l'utilisateur et le système, sur la base de la construction d'images résultats. Il s'agit de construire la sémantique du problème à partir des symboles de l'ontologie, en procédant par des exécutions successives permettant d'affiner graduellement la sémantique par retour de pertinence.

Cette approche considère que :

> « semantics is an emergent property of the interaction between the user and the system. » (Santini et al., 2001, p. 338)

La sémantique émergente exprime la sémantique à travers une représentation purement syntaxique et ainsi :

> « the semantic understanding is the process of understanding one domain in terms of another. » (Rapaport, 2003, p. 401)

ici le domaine d'application avec les termes du traitement d'images.

Cela implique que l'interaction homme-machine soit faite dans un environnement créé et configuré par l'activité d'interaction elle-même :

- Les utilisateurs apprennent l'interprétation assignée aux symboles de l'ontologie par leur impact sur les résultats produits par l'application générée.
- Le système de génération BORG apprend la confiance (qui peut aussi s'entendre comme la préférence) placée dans chaque caractéristique par l'utilisateur. Par exemple, l'objet « noyau de cellule de séreuse » peut être pertinemment caractérisé par son aspect intérieur et par ses frontières. Mais, on sait que les frontières des noyaux ne sont pas toujours visibles et que par conséquence les caractéristiques de contours sont moins fiables que les caractéristiques des régions pour construire la solution.

Pour intégrer ce principe dans le système de génération, il suffit d'ajouter un degré de confiance dans les hypothèses générées sur le contexte de l'application. Telles qu'elles ont été présentées dans le chapitre 3, les hypothèses étaient toutes considérées comme certaines, et donc avec un poids implicite de 1. Maintenant, elles ont en plus un degré de confiance marqué par un poids dans l'intervalle [0..1]. Ces valeurs sont ensuite utilisées dans le calcul des valeurs de crédibilité et de plausibilité des sources de connaissances.

3.3.2 Le modèle d'interaction

Les cycles d'interaction conduisent à des reformulations qui consistent, pour l'utilisateur, à écarter de la requête les caractéristiques qu'il juge non pertinentes et à pénaliser ou favoriser la confiance dans les descripteurs choisis, et pour BORG à maintenir la liste des valeurs de confiance relatives accordées aux descripteurs de la requête, puis à construire une nouvelle solution. Ce modèle repose sur une hypothèse très forte : le générateur d'applications produit toujours une solution acceptable par rapport au contenu de la requête. Il n'y a pas de remise en cause de la solution produite par le système de génération de programmes. Les seules causes possibles de résultats non satisfaisants ne peuvent provenir que du contenu de la requête et par conséquent d'un mauvais choix de descripteurs fait par l'utilisateur ou d'une mauvaise appréciation de la confiance placée dans les descripteurs par BORG.

Pour assister l'utilisateur dans sa reformulation, le générateur de programmes retourne au système de formulation non seulement la liste des images résultant de l'exécution du programme, mais aussi la liste des caractéristiques utilisées par le générateur pour produire son raisonnement de résolution, ainsi que les caractéristiques qu'il aurait voulu utiliser mais qui n'étaient pas renseignées dans la requête. Ces caractéristiques peuvent alors être exploitées par l'utilisateur pour mesurer le degré de responsabilité de chaque descripteur dans l'échec de la solution et ainsi aider à la reformulation. Seuls les descripteurs impliqués dans la conception du plan sont retournés, c'est-à-dire ceux utilisés pour décider des stratégies, des tactiques et des techniques parce que nous considérons que les descripteurs utilisés pour sélectionner et paramétrer les opérateurs ont moins de responsabilité dans la qualité de la solution finale. Nous verrons, en particulier en conclusion, qu'il est possible a posteriori, d'utiliser un mécanisme de recherche automatique du paramétrage optimal.

Comme conséquence de l'approche par émergence de la sémantique, le logiciel final n'est pas le résultat de la dernière formulation, mais le résultat de la série des formulations faites par interaction, parce que durant le processus de formulation, le système a aussi appris la confiance attribuée à chaque descripteur choisi et décrit par l'utilisateur.

4 Le cycle d'interaction Homme-Machine

La formulation complète d'une application est organisée par un cycle d'itérations autour de cinq étapes dérivées du modèle de formulation défini :

1. L'utilisateur formule une représentation initiale du problème sous forme extensionnelle ;
2. Le système de formulation exécute une extraction automatique de caractéristiques à partir de cette formulation extensionnelle ;
3. L'utilisateur sélectionne des caractéristiques qu'il juge pertinentes et donne une liste d'images pour composer la première requête ;
4. BORG génère un programme d'application et retourne les images résultats de son exécution à partir des images tests, plus les caractéristiques effectivement utilisées par le raisonnement et celles qu'il aurait pu utiliser mais qui n'étaient pas présentes dans la requête ;
5. En cas de résultats non satisfaisants, l'utilisateur reformule son problème en écartant ou ajoutant de nouvelles caractéristiques et en pénalisant ou favorisant

la confiance dans les caractéristiques. Le cycle continue en général à l'étape 4, mais peut repartir à l'étape 1 en cas de modification profonde.

Chacune de ces étapes est détaillée dans les sections suivantes.

4.1 La formulation extensionnelle initiale

Dans le but d'aider l'utilisateur à construire sa première formulation extensionnelle, le système de formulation utilise cinq écrans séquentiels destinés à :

1. La spécification des post-traitements qui seront appliqués ensuite aux résultats. C'est un point d'entrée assez naturel qui vise à spécifier le contexte d'utilisation de l'application.
2. La définition de la classe d'images au niveau physique.
3. La définition de la classe d'images au niveau sémantique ou au niveau perceptif, selon que les objets d'intérêt sont descriptibles ou pas (*cf.* chap. 2, § 3.1.4).
4. La spécification de la tâche et de ses contraintes.
5. Et finalement, la détermination des contraintes de contrôle (qualité et performances de l'application).

4.1.1 Spécification des post-traitements

Les post-traitements sont les opérations qui seront effectuées sur les résultats produits par la future application de traitement d'images. Ces informations ne font pas partie, à propement parlé, de la formulation de traitement d'images, mais sont utiles pour fournir une assistance durant la spécification de la tâche (*cf.* § 4.1.5).

L'interface présente un panneau avec 13 catégories différentes de post-traitement que l'utilisateur peut sélectionner. Pour chaque post-traitement, l'utilisateur peut indiquer les opérations qui y seront effectuées (voir le tableau 4.1).

Tableau 4.1 – Liste des tâches de post-traitement et les opérations réalisables.

Post-processing tasks	Post-Processing operations
Classification	Counting
Image comparaison	Printing
Quantization	Displaying
Template matching	Size measurement
Registration	Orientation measurement
Scene reconstruction	Radiometry measurement
Detection-tracking	Morphology measurement
Pattern recognition	Topometry measurement
Art visualization	Topology measurement
Visual inquiry	Spatio-tempral measurement
Transmission	Compression rate measurement
Archival storage	

Pour notre exemple fil rouge, la tâche « extract <serous cell nucleus> » est suivie par une classification des régions dans le but de détecter les cellules anormales ou suspectes (*cf.* figure 4.5). Cette classification est basée sur des mesures de taille, de morphologie et de radiométrie.

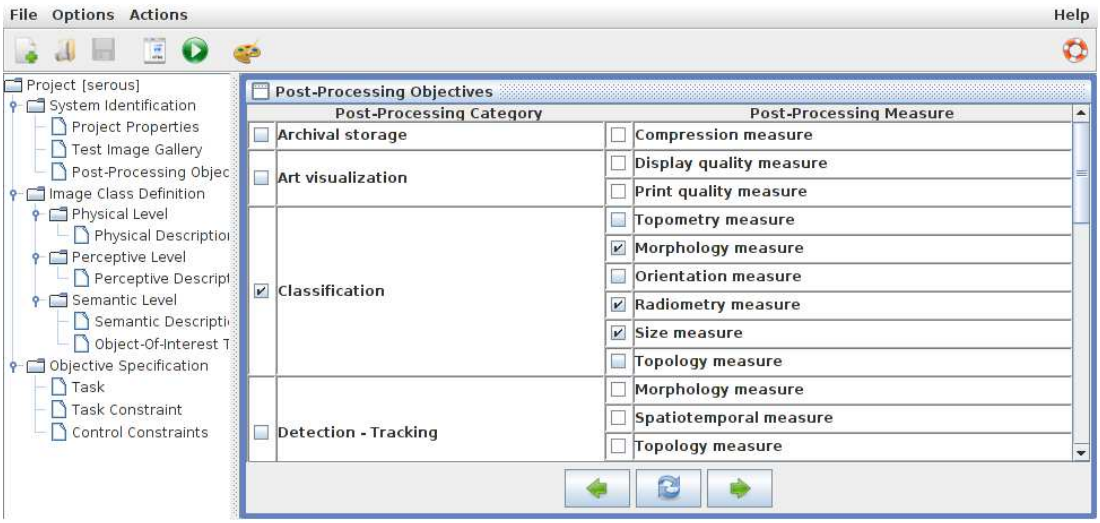


Figure 4.5 – L'interface permettant de spécifier les post-traitements.

4.1.2 Définition physique de la classe d'images

L'utilisateur compose sa définition physique avec les descripteurs qui rendent le mieux compte des effets de la chaîne d'acquisition sur l'apparence visuelle de la scène dans les images d'entrée. L'interface présente tous les descripteurs fournis par l'ontologie sous la forme de tables. L'utilisateur choisit les descripteurs pertinents et affecte les valeurs avec des blobs ou des valeurs numériques ou symboliques. Dans le but d'aider l'utilisateur à donner des valeurs numériques, le système utilise une base de données qui référence les valeurs par défaut pour les systèmes d'acquisition traditionnelles – *p. ex.*, caméra CDD, scanner, microscope.

Par exemple, la définition physique de la tâche « correct <illumination> » est construite à partir de l'interface présentée figure 4.6. Le résultat est donné dans le tableau 4.2. Le cyto-technicien décrit le modèle d'illumination avec une image de champ optique vide – *i.e.*, une lame sans objet cellulaire – puisque l'illumination est constante au cours du temps mais spatialement hétérogène.

4.1.3 Définition perceptive de la classe d'images

Si l'utilisateur choisit de décrire la classe d'images au niveau perceptif, l'interface affiche des tables contenant les descripteurs associés à chaque primitive visuelle comme définis dans l'ontologie. L'utilisateur donne des valeurs numériques ou symboliques s'il le souhaite, ou plus simplement des blobs pour décrire les différentes apparences visuelles de ce qu'il considère comme une région, un fond d'image, un point d'intérêt, ou un contour pour l'application.

L'application de cytologie est décrite au niveau sémantique parce que tous les objets de la scène sont connus. De plus, une description perceptive n'a pas vraiment de sens pour une tâche d'extraction d'un objet puisqu'il n'y a pas de définition d'objet à l'intérieur d'une définition perceptive. Mais si nous supposons une tâche « partition <image> », alors la classe d'images pourrait être décrite avec plusieurs blobs, pour les différentes apparences des régions, contours et du fond d'image (voir tableau 4.3).

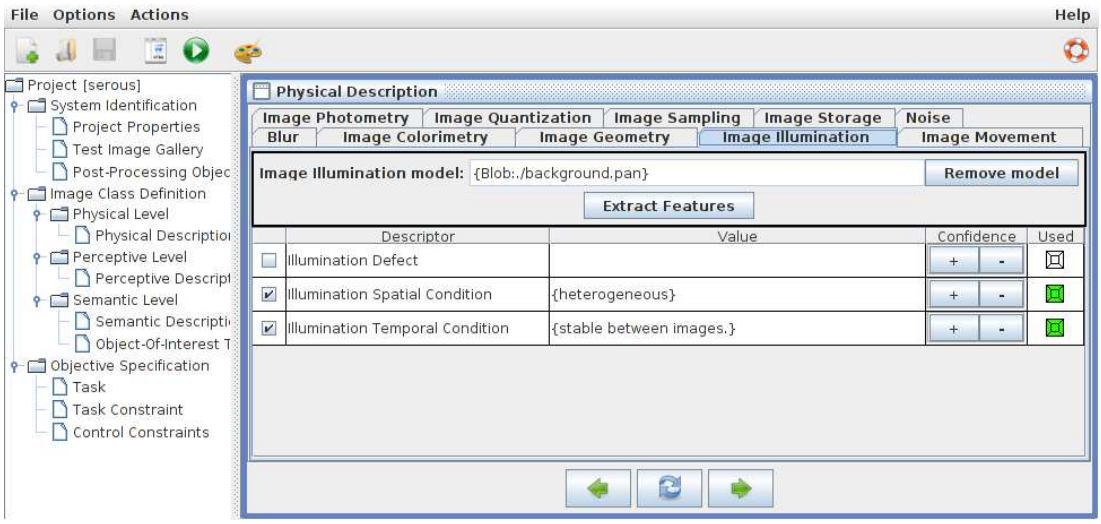


Figure 4.6 – L'interface permettant de définir le niveau physique de la classe d'images.

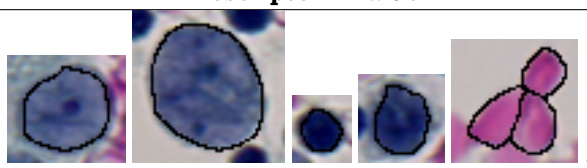
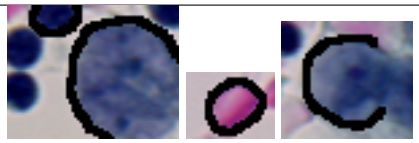
Tableau 4.2 – Extrait de la description physique de l'application de cytologie.

Acquisition effect	**Descriptor = Value**
Illumination	Temporal-Condition = stable Spatial-Condition= heterogeneous Model = 
Colorimetry	Colorspace = RGB
Quantization	Bits-per-Pixel = 24 Function = linear
Noise	Composition = additive Distribution = gauss Power-Spectral-Density = white Mean = 0 Standard-Deviation = very low

4.1.4 Définition sémantique de la classe d'images

Si la classe d'images peut être décrite au niveau sémantique, la définition repose sur la construction de l'arbre des objets d'intérêt (*cf.* chap. 2, § 4.3). Ce niveau renvoie à l'hypothèse de la « sémantique différentielle » où l'on cherche une définition discriminante des concepts. L'arbre des objets aide alors à déterminer la signification de chaque nœud en fonction de ses parents et de ses voisins (Charlet et al., 2006).

Tableau 4.3 – Extrait d'une définition extensionnelle de la classe d'images pour une tâche de segmentation : « partition <image> ».

Visual primitive	Descriptor = Value
Region	Model =  ...
Background	Model =  ...
Edge	Model =  ...

Quatre principes guident la description d'un objet dans l'arbre : les principes de similarité et de différence avec le parent, et les principes de similarité et de différence avec les frères.

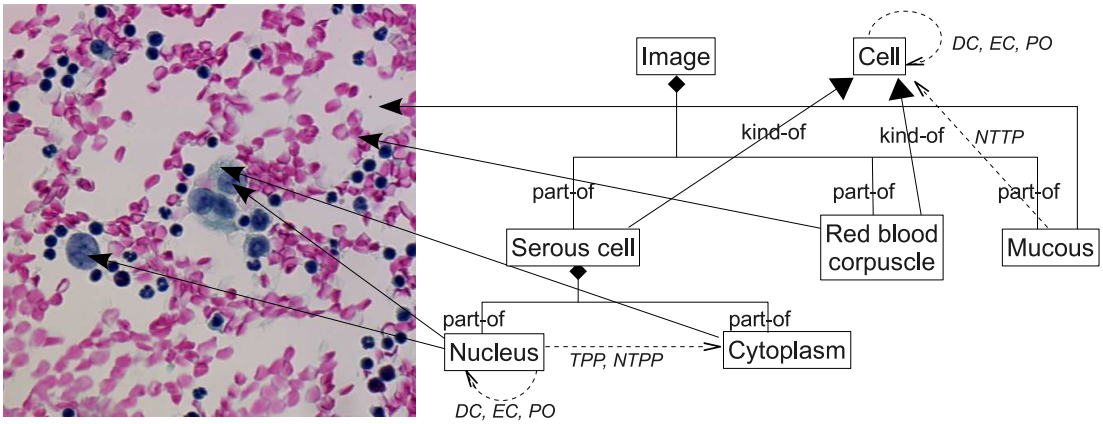


Figure 4.7 – Une scène de cytologie est composée de cellules de séreuse et de globules rouges qui reposent sur une muqueuse. Une cellule de séreuse est composée d'un noyau entouré d'un cytoplasme. Dans cet arbre DC, EC, TPP, NTPP, et PO sont des relations topologiques de la grammaire RCC-8.

L'utilisateur identifie tous les objets d'intérêt qui peuvent apparaître dans les images et les organise hiérarchiquement. Pour cela, il spécifie les relations hyperonyniques et les relations méronymiques entre ces objets :

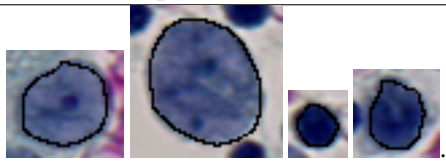
– La relation hyperonymique « sorte-de » définit une taxonomie des objets d'intérêt basée sur les similarités. Cette structure est la base de deux inférences que nous faisons tous les jours : l'identification qui est notre habileté à reconnaître une classe d'objet à partir de ses caractéristiques, et la spécialisation/généralisation qui est notre habileté à organiser les objets en catégories prototypiques.

– La relation méronymique « partie-de » décrit la relation de composition de ces objets en entités et composants. La relation peut ensuite être valuée avec des relations spatiales extrinsèques, type distance et topologie.

En se basant sur cet arbre, l'utilisateur peut ensuite décrire chaque objet d'intérêt à partir de descripteurs, pour lesquels il donne une valeur iconique ou symbolique.

La figure 4.7 est un exemple de l'arbre des objets définissant l'application de cytologie. Le tableau 4.4 donne un extrait d'une définition possible au niveau sémantique. L'objet « Serous cell nucleus » est décrit avec des blobs alors que l'objet « mucous » est décrit avec des valeurs linguistiques comme la région du fond avec l'intensité la plus claire.

Tableau 4.4 – Extrait d'une définition extensionnelle de la classe d'images au niveau sémantique pour la tâche de segmentation : « extract <serous cell nucleus> ».

Object of interest	Visual primitive	Descriptor = Value
Serous cell nucleus	Region	Model =  . . .
Mucous	Background	Photometry.Brightness = the most (*clearest*) Texture.Type = none

4.1.5 Spécification de la tâche

L'utilisateur sélectionne la tâche à accomplir à partir d'une liste présentée par l'interface. Si besoin, le premier argument est choisi dans les définitions physique, perceptive ou sémantique, et le second argument peut être utilisé pour spécifier une liste d'images de référence. Les images de référence sont des exemples de résultats attendus pour les images tests – *p. ex.*, segmentation manuelle complète, image améliorée –, et sont construites à l'aide d'interfaces spécialisées. Pour construire des segmentations de référence, nous utilisons DESCARTES [2] qui est un logiciel de segmentation interactive développé dans l'équipe.

L'utilisateur choisit les contraintes à associer à partir d'une liste définie dans l'ontologie et présentée sur l'interface (*cf.* figure 4.8). À ce niveau, le système peut fournir une aide. Il utilise pour cela les informations sur les post-traitements renseignés au début du cycle de formulation pour suggérer des valeurs possibles pour les contraintes. Par exemple, puisque des mesures de surface des régions seront faites comme post-traitement de la tâche « extract <serous cell nucleus> », le système propose la localisation des frontières comme critère à optimiser.

4.1.6 Les contraintes de contrôle

Finalement, l'utilisateur détermine les types d'optimisation (temps, espace) et de capacité (fiabilité, robustesse) requis pour le programme d'application.

2. http://www.greyc.ensicaen.fr/~luc/SEGMENTE

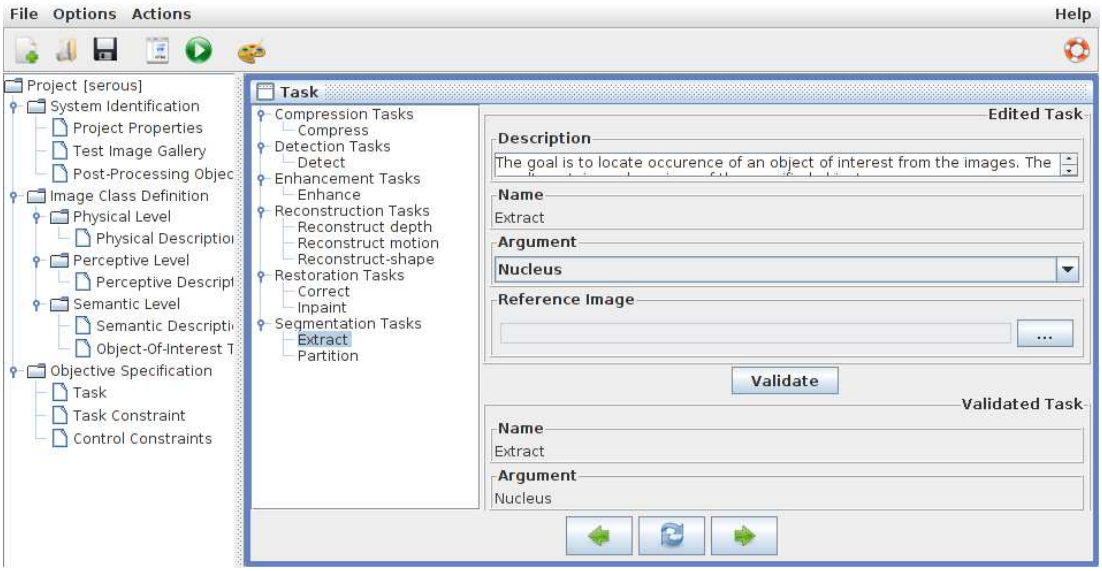


Figure 4.8 – L'interface permettant de spécifier la tâche.

4.2 Première formulation intensionnelle

Une fois donnée cette première formulation principalement extensionnelle, l'utilisateur peut en construire une représentation intensionnelle, en lançant l'extraction automatique de valeurs pour tous les descripteurs valués par des blobs et des images dans la formulation initiale. Par exemple, le tableau 4.5 présente quelques valeurs de descripteurs régions obtenues après extraction dans les blobs donnés dans la définition extensionnelle du tableau 4.4.

Le système de formulation invite ensuite l'utilisateur à corriger cette formulation initiale en écartant les descripteurs qu'il juge non pertinents pour l'application. Pour notre exemple, les descripteurs en italiques dans le tableau 4.5 sont éliminés de la requête par le cyto-technicien puisque les cellules de séreuse ne sont pas caractérisées par leur position, leur rectangularité, leur orientation et le nombre de trous.

Tableau 4.5 – Extrait d'une définition intensionnelle du concept de {serous cell nucleus} après la phase d'extraction de caractéristiques.

Object of interest	Visual primitive	Descriptor = Value
Serous cell nucleus	Region	Photometry.Brightness = [38, 124] Photometry.Contrast = [12, 86]
		Colorimetry.Saturation = [0, 0.609] Colorimetry.Hue = [3.53, 4.71]
		Morphology.Convexity = [0.4, 0.9] *Morphology.Rectangularity = [0.15, 0.81]*
		Size.Area = [65.56, 179.67] Size.Diameter = [8, 35]
		Orientation.Orientation = [0, 345]
		Position.CenterOfMass = [(10, 256), (5, 245)]
		Topology.NumberOfHole = 0

4.3 Reformulation

Durant la reformulation, la plupart des changements ont trait à la définition sémantique de la classe d'images (ou perceptive s'il n'y a pas de définition sémantique). La définition physique est moins une affaire de point de vue puisque la caractérisation des effets de l'acquisition est plus objective.

Après chaque génération de solution, l'interface affiche les images résultantes, et marque les descripteurs utilisés par le système de génération de programme pour produire son raisonnement et les descripteurs non renseignés qu'il aurait voulu utiliser grâce à une signalétique à trois couleurs (*cf.* figure 4.9). À partir de là, l'utilisateur peut reformuler le problème en écartant ou ajoutant des descripteurs dans la nouvelle requête à partir de boîtes à cocher, et en augmentant ou réduisant la confiance dans les descripteurs à partir de boutons trois états (+, -, =).

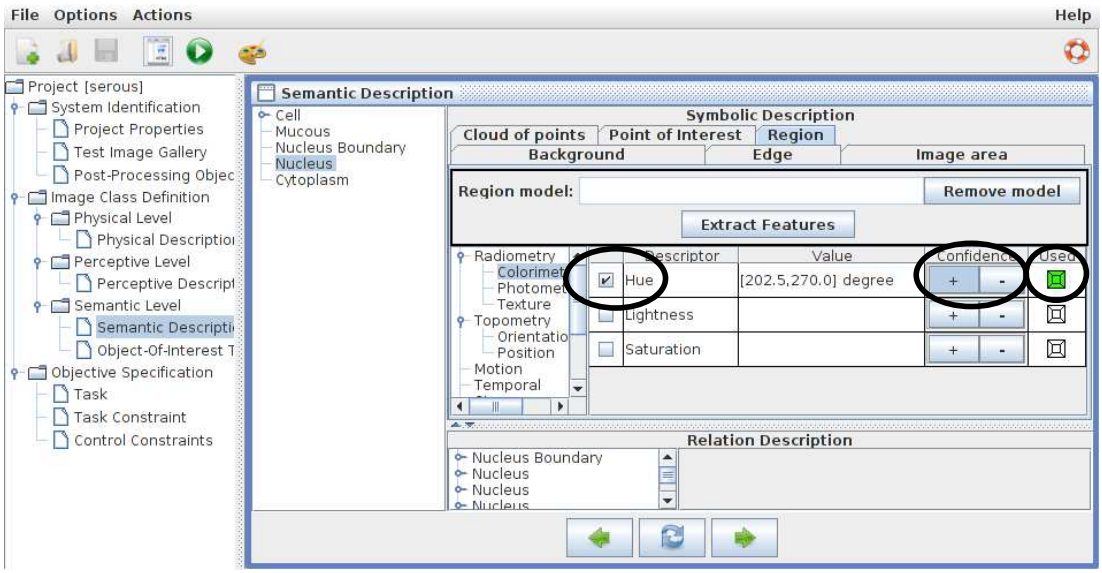


Figure 4.9 – Interface permettant de corriger la formulation initiale.

Quand BORG reçoit la nouvelle requête, il calcule les nouvelles confiances relatives entre les descripteurs avant de construire une nouvelle solution. Pour ce calcul des nouvelles confiances, nous avons choisi d'ajouter à chaque confiance c_i du descripteur d_i :

- +0.2 si le descripteur d_i est marqué par un '+' par l'utilisateur,
- -0.2 le descripteur d_i est marqué par un '-' par l'utilisateur,
- -0 sinon.

Si $c_i < 0$ alors nous fixons $c_i = 0$. Puis, nous effectuons une normalisation par interpolation linéaire pour retrouver les confiances relatives entre 0 et 1 :

$$\forall i, c'_i = \frac{c_i}{\max |c_i|}$$

Le choix de la valeur de correction à 0.2 est purement expérimentale et n'a pas de réelle justification théorique. Toutefois, elle peut trouver son explication, encore une fois, dans les limites des capacités humaines à graduer des stimulus unidimensionnels (par exemple, le volume d'un son, la sensation de salé, l'intensité d'une couleur,

etc.). George Miller (Miller, 1956) a mis en évidence que ces limites semblent être fixées à 5, 6 ou 7 niveaux selon les stimulus considérés. La valeur de 0.2 correspond donc à une échelle à 5 niveaux, inférieure à toutes les limites.

Dans notre exemple, la tâche « extract <serous cell nucleus> » conduit à plusieurs solutions en fonction des confiances (ou préférences) données par l'utilisateur aux descripteurs. Si l'utilisateur donne préférence aux caractéristiques de contours, la solution utilise une détection de contours pour extraire les marqueurs des futures régions, puis une ligne de partage des eaux pour localiser les frontières, puisque les régions comme le fond sont décrits comme homogènes. Si l'utilisateur privilégie les caractéristiques de région et plus particulièrement les caractéristiques de couleur, la solution utilise une classification de pixels basée sur la segmentation couleur pour extraire les marqueurs des futures régions, puis utilise aussi une ligne de partage des eaux pour localiser les frontières des régions à partir des marqueurs obtenus.

5 Conclusion

Il est bien connu que les systèmes cognitivistes souffrent du problème de l'ancrage des symboles (Vernon, 2007; Schierwagen, 2001). C'est pourquoi nous avons proposé d'aborder la formulation d'objectifs à travers l'interaction Homme-Machine. Le modèle d'interaction définit la façon de composer des formulations linguistiques. La première phase de l'interaction vise à construire la définition intensionnelle du problème, difficile à produire de prime abord, à partir d'une formulation extensionnelle, plus facile à composer pour les utilisateurs.

La seconde phase vise à construire la sémantique de l'application par un processus émergent, où les utilisateurs apprennent l'interprétation donnée par l'ingénieur de la connaissances aux symboles du langage de formulation et le système apprend la confiance relative accordée par l'utilisateur à chacun des éléments de la description.

L'interaction fournit ainsi un moyen de franchir les fossés sensoriel et sémantique. Elle conduit graduellement vers une représentation explicite de la sémantique de l'application qui est, de plus, ancrée dans les données image.

La contribution majeure de ce travail réside dans les capacités laissées à l'utilisateur de construire la sémantique de son application de manière explicite et raisonnée. La sémantique de l'application est de nature syntaxique et est représentée par une liste de descripteurs qui décrivent l'objectif avec leur degré de confiance (ou de préférence).

Les approches qui n'utilisent qu'une formulation extensionnelle à base d'images exemples ne laissent pas autant de liberté à l'utilisateur. La sémantique est en grande partie prédéterminée, soit dans une liste de caractéristiques imposées pour décrire les concepts, soit dans une fonction d'évaluation lorsque la liste des caractéristiques est sélectionnée dynamiquement.

Dans notre approche, la liste des descripteurs est entièrement maîtrisée par l'utilisateur. Le rôle de ce dernier est de construire son ontologie d'application en sélectionnant des descripteurs dans l'ontologie du domaine et en donnant les valeurs ou des images exemples pour l'extraction automatique des valeurs. La validation de sa formulation, comme la correction, sont faites sur la base d'images qui sont les résultats de l'exécution de la solution proposée pour la formulation sur des images tests.

Le point faible de cette approche réside dans la gestion de la reformulation en cas de non satisfaction de résultat. Cela exige de fortes compétences en traitement d'images de la part de l'utilisateur, qui doit être capable de déterminer quelles sont les caractéristiques qui doivent être reconsidérées à la vue des résultats finaux.

CHAPITRE 5

Acquisition des connaissances

L'acquisition des connaissances est une activité inhérente à notre approche. C'est intentionnellement que nous cherchons à élaborer une théorie formelle explicite du développement d'applications. Pour cela, l'atelier d'ingénierie des connaissances est la réponse que nous donnons pour aborder la complexité de la tâche en permettant d'étudier le développement d'applications en contexte. Il fournit les outils logiciels pour aider au développement et, en même temps, pour capitaliser les productions sous la forme de corpus structurés. Ces corpus fournissent la « matière première » pour élaborer les modèles originaux du système et enrichir sa base de connaissances.

C'est aussi un lieu d'expérimentation du système à base de connaissances, où la logique de conception embarquée dans le système peut être étudiée et critiquée par les experts.

La construction de l'atelier a nécessité un important travail de développement pour produire tous les outils qui composent cet atelier. Ce travail s'est fait principalement avec l'aide de stagiaires de l'école d'ingénieurs et de l'université. Pratiquement tous les logiciels (au moins ceux qui sont stabilisés) sont disponibles au téléchargement sur le serveur Internet du laboratoire en version code source libre.

1 Introduction

Notre approche de la conception d'applications de traitement d'images est résolument de type cognitiviste. Elle est basée sur une représentation symbolique explicite des connaissances de conception, aussi bien déclaratives (savoir) que procédurales (savoir-faire). Les connaissances de conception distinguent les connaissances du domaine, qui sont ici organisées autour de l'ontologie, celles de construction, d'instanciation et d'évaluation de solutions qui sont représentées sous forme de sources de connaissances modulaires et autonomes. Ainsi modélisées, elles sont à la fois opérationnelles et compréhensibles.

Le prix à payer pour cette approche est la nécessité d'une acquisition des connaissances quasi-continuelle ; à chaque fois que le raisonnement ne dispose pas de toutes les connaissances pour construire une solution acceptable. Cela peut sembler une limite à une utilisation du système en routine, et donc rendre cette approche irréaliste. Mais, premièrement, le système peut se voir comme un système générique spécialisable, qui vise à faciliter la construction d'applications dans des domaines

bien identifiés (Crowley et al., 2007; Abchiche et al., 2002). Avant son utilisation, la base de connaissances du système est enrichie avec différentes solutions de traitement proposées dans la littérature pour résoudre ce genre d'applications – à l'instar de l'exemple d'imagerie aérienne traitée chap. 2, § 5.1, où le système a été configuré avec cinq solutions différentes. Le système est ensuite utilisé en routine pour produire automatiquement des programmes pour des applications concrètes du domaine étudié. Deuxièmement, la motivation de notre projet est bien de maximiser l'explicitation des connaissances dans une perspective d'étude « théorique » de ces connaissances. L'acquisition des connaissances est donc bien une partie intégrante de notre projet.

Nous voulons montrer dans ce chapitre, que le traitement d'images est un domaine particulier pour la problématique de l'acquisition des connaissances, dans le sens où il est difficile, voire impossible, de s'appuyer sur des corpus linguistiques pré-existants. Dans la section 2, nous discutons de la nature des connaissances de conception en traitement d'images et les façons de les capitaliser sous la forme de corpus. Dans la section 3, nous présentons un atelier d'ingénierie des connaissances que nous avons défini comme le lieu de l'acquisition et de l'expérimentation des connaissances.

2 Le problème de l'acquisition de connaissances

2.1 Nature des connaissances

Le domaine du traitement d'images est maintenant connu pour être un domaine où l'activité d'acquisition des connaissances est particulièrement complexe. En effet, le développement d'une application par un expert relève d'une démarche majoritairement empirique faite à partir de connaissances tacites, c'est-à-dire hautement personnelles et fondamentalement non-verbalisables. Ces connaissances sont acquises par l'expérience, c'est pourquoi elles restent difficiles à formaliser et à communiquer (Polanyi, 1958; Nonaka and Takeuchi, 1995).

La construction de solutions mobilise des connaissances expertes qui doivent permettre de surmonter les obstacles suivants :

- La composition d'une chaîne de traitement ne peut pas se décider directement à partir des données d'entrée et de sortie parce que les données de sortie ne sont pas forcément en relation causale avec les données d'entrée. Plusieurs changements de représentations intermédiaires sont en général nécessaires pour composer les sorties, comme les changements d'abstraction (*p. ex.*, pixel en contour, contour en région, région en objet) ou de référentiel (*p. ex.*, spatiale, spectrale, photométrique), qui ne peuvent pas être décidés à partir des seules entrées.
- Le choix et l'adaptation des éléments de traitement numérique de la chaîne nécessitent d'abord de savoir reconnaître dans les données d'entrée, le modèle d'information numérique prédéfini sur lequel ils reposent. La difficulté provient du fossé sémantique qui existe entre l'information disponible, qui est principalement implicite et cachée dans les données, et sa formulation mathématique.
- Les opérations de traitement d'images ont des effets qui sont difficilement prédictibles. Typiquement, les opérateurs sont très sensibles aux valeurs de leurs paramètres et l'expertise requise pour déterminer ces valeurs est souvent obtenue à travers l'expérimentation.

– La coopération et la comparaison d'approches sont rendues délicates par le fait que chacune d'elles reposent sur des a priori différents – *p. ex.*, la forme des objets en morphologie mathématique versus les propriétés du signal pour les approches variationnelles.

En conséquence, il n'y a pas de consensus sur la construction de solution, mais seulement des compromis. Cela interdit la recherche d'une solution optimale et n'autorise que la recherche de solutions acceptables. Pour une même application, une grande variété de solutions reste possible. Elles sont plus ou moins efficaces et leurs caractéristiques dépendent étroitement de la sensibilité scientifique de leur concepteur – *p. ex.*, orientée vers la morphologie mathématique, les statistiques ou l'apprentissage. Ceci explique en grande partie pourquoi un patrimoine de solutions consensuelles n'existe pas encore, malgré les appels répétés de plusieurs chercheurs – voir en particulier la discussion dans CVGIP en 1994 (Haralick et al., 1994), le forum de M. Duff (Duff, 1996) ou le « coup de gueule » de P. Zamperoni dans Pattern Recognition (Zamperoni, 1996).

2.2 Approche de l'ingénierie des connaissances

Lors d'une acquisition des connaissances, ce que l'on on va chercher à expliciter, c'est la part de connaissances dans la pratique des experts. La connaissance s'entend ici comme une croyance vraie et justifiée (Nonaka and Takeuchi, 1997). Il ne s'agit donc pas d'une suite d'astuces, mais de lois et de principes établis.

Notre parti pris est de considérer que ces connaissances ne sont pas pas directement disponibles sous forme linguistique, issues d'interviews ou de la littérature. Nous soutenons que les corpus linguistiques obtenus par interviews se heurtent au mur des connaissances tacites et que la littérature rend compte d'une théorie de la conception d'opérations de traitement d'images, mais pas d'une théorie du développement d'applications. Par exemple, la théorie des équations aux dérivées partielles forme bien une théorie du traitement d'images qui explique comment opèrent les algorithmes basés sur cette théorie, mais elle n'indique pas explicitement quand utiliser ces algorithmes, c'est-à-dire pour quel objectif, pour quelle classe d'images et quelles sont ses répercutions sur les autres éléments de la chaîne. Des travaux comme ceux de l'équipe du LITIS de Rouen ont fait l'hypothèse inverse et ont basé l'élaboration de leur base de connaissances sur l'analyse de corpus linguistiques (interview, articles scientifiques et thèses) (Saidali et al., 2004; Labiche et al., 2007). Ils peuvent alors profiter des théories et des outils de l'analyse linguistique pour constituer le patrimoine de connaissances (Bachimont, 2000; Charlet, 2002).

Au contraire, nous avons choisi d'adopter une approche *constructiviste* de l'acquisition des connaissances (Bachimont, 1996). Il s'agit d'un changement de perspective pour passer d'une modélisation de l'expert et de son fonctionnement cognitif à une modélisation de sa démarche de résolution de problèmes (Gaines et al., 1993). On ne va chercher ici qu'à formaliser les connaissances qui émergent du développement « manuel » d'applications concrètes, en développant des modèles spécifiques. M. Linster résume cette approche ainsi :

> « Early work in knowledge acquisition for knowledge-based systems emphasized a transfer view of knowledge acquisition. Knowledge was considered to be in an expert's head, and the knowledge engineer, using a tool, elicts the knowledge and represents it in a suitable operational formalism. Today, we see knowledge acquisition as a constructive model-building process. » (Linster, 1993, p. 209)

3 Un atelier pour l'ingénierie des connaissances

A défaut de corpus linguistique, il nous est nécessaire de définir seuls nos modèles originaux pour la capitalisation des connaissances.

C'est dans cette perspective que s'inscrit notre atelier d'ingénierie des connaissances (Clouard et al., 1995a; Clouard et al., 1995c; Clouard et al., 1999a; Clouard et al., 2002). Il fournit l'environnement logiciel qui doit permettre d'élaborer les modèles à partir d'interactions entre les experts et l'ingénieur de la connaissance. Parce que nous cherchons à rendre les connaissances à la fois opérationnelles et explicites, l'atelier se présente à la fois comme un outil d'acquisition permettant aux experts d'exprimer et d'échanger les connaissances, et comme un outil d'expérimentation permettant aux experts de critiquer ces connaissances lors de leur mise en œeuvre pour le développement de programmes d'application.

3.1 Architecture de l'atelier

3.1.1 Vue structurelle de l'atelier

L'architecture de l'atelier est donnée figure 5.1. Elle présente deux volets :

- Le volet de droite correspond à la *construction automatique d'applications*. Il est utilisé pour construire des applications à partir du générateur automatique de programmes présenté au chapitre 3. On retrouve de ce côté le système de formulation d'application (HERMES [1]), le système de génération automatique (BORG) et la bibliothèque d'opérateurs (PANDORE [2]).
- Le volet de gauche est dédié à la *construction manuelle d'application*. Ce côté profite aussi du système de formulation HERMES et de la bibliothèque d'opérateurs PANDORE. En plus, l'atelier met à disposition des experts deux logiciels d'aide au développement d'application : une interface de programmation visuelle (ARIANE [3]) pour construire rapidement des programmes exécutables sans faire de programmation et un atelier de génie logiciel (PARTHÉNOS [4]) propre à modéliser la logique de conception des applications.

L'atelier distingue trois types d'acteur :

- *L'utilisateur*, que l'on souhaite à terme ne plus nécessairement être expert en traitement d'images, sollicite l'atelier pour développer une application particulière.
- *L'expert de traitement d'images* est supposé être capable de construire manuellement une application adaptée aux besoins de l'utilisateur.
- *L'ingénieur de la connaissance* est en charge de l'acquisition des connaissances à partir de la production des experts.

Le développement d'une application formulée par un utilisateur devrait pouvoir se faire indifféremment du coté expert ou du coté système à base de connaissances. Les deux volets partagent pour cela les mêmes représentations, dont l'ontologie du domaine, le modèle du livre des connaissances et la bibliothèque d'opérateurs sont les garants.

1. http://www.greyc.ensicaen.fr/~regis/Hermes
2. http://www.greyc.ensicaen.fr/~regis/Pandore
3. http://www.greyc.ensicaen.fr/~regis/Ariane
4. http://www.greyc.ensicaen.fr/~regis/Parthenos

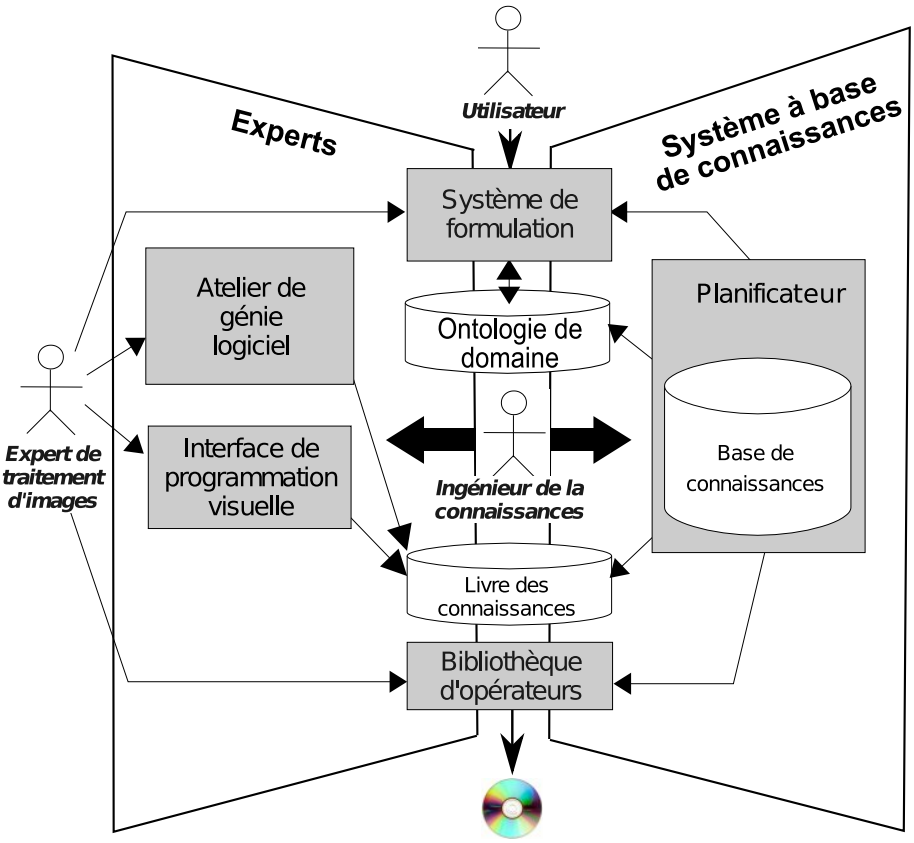


Figure 5.1 – L'architecture globale de l'atelier.

3.1.2 Vue fonctionnelle de l'atelier

L'idée clé de cette séparation reprend celle du paradigme constructiviste : réutiliser les résultats capitalisés dans une partie du projet pour renforcer l'autre.

- L'expérience capitalisée sous la forme de corpus, grâce à l'environnement de développement, est utilisée pour enrichir la base de connaissances du système automatique.
- La modélisation des connaissances proposée pour le système à base de connaissances est utilisée pour rationaliser le processus de capitalisation et affiner les modèles de l'environnement de développement.

A l'intérieur de l'atelier, le développement d'applications est vu comme un processus itératif et incrémental. L'utilisateur collabore avec l'expert, ou le système à base de connaissances, pour développer une application par l'intermédiaire de l'interface de formulation d'objectifs. L'important, c'est d'avoir un résultat très rapidement pour ne pas rompre le cycle d'interaction. La programmation visuelle est ici un outil précieux pour l'expert. Elle permet de créer rapidement des prototypes exécutables sans les contraintes de la programmation (Baroth and Hartsough, 1994; Whitley and Blackwell, 2001).

L'acquisition des connaissances profite de ces développements pour capitaliser les productions. L'ingénieur de la connaissance interagit avec l'expert du traitement

d'images pour expliciter la logique de conception, afin de la capitaliser dans un *livre des connaissances*. Le livre des connaissances, tel que défini par J.L. Ermine (Ermine, 2000b; Ermine, 2000a) dans la méthode MKSM, est en effet un moyen de capitaliser les connaissances tacites à travers différents modèles qui correspondent à différents points de vue sur l'activité de conception : point de vue sur les *Opérations*, l'*Information*, les *Décisions* et les *Connaissances*. Son expérimentation a déjà été faite pour le pilotage de codes (Moisan and Ermine, 2000). Mais ici, nous proposons nos propres modèles de représentation des connaissances pour le livre, aptes à une opérationnalisation immédiates dans notre système tel que présenté au chapitre 3.

3.2 Un atelier pour l'acquisition des connaissances

Dans l'atelier, on se propose d'analyser toute la chaîne de développement, depuis la formalisation des problèmes, la construction d'une solution logicielle, jusqu'à son évaluation, dans le but d'élaborer des modèles qui forment les éléments d'une théorie du domaine du point de vue du développement d'applications. Pour cela, l'atelier doit être le lieu de l'étude de la connaissance en contexte. Selon l'approche constructiviste, c'est l'interaction entre l'expert du traitement d'images et l'ingénieur de la connaissance, par l'intermédiaire des outils logiciels fournis par l'atelier, qui crée la connaissance (Linster, 1993).

3.2.1 Modélisation d'application

Notre vision de la modélisation d'une application de traitement d'images repose sur l'idée force qu'une application s'analyse selon quatre points de vue distincts mais complémentaires. Les quatre points de vue correspondent à une discrétisation de l'application qui isole les différents types de connaissances à capitaliser : la formulation d'objectifs, la définition de classe d'images, la conception de plan de traitement et le codage de programme exécutable. À chaque point de vue est consacré un modèle qui en propose une vue abstraite destinée à capturer la partie de la sémantique de l'application vue sous cet angle.

Notre livre des connaissances distingue donc les quatre modèles : *Système*, *Domaine*, *Tâches* et *Programme*. Ces modèles ont déjà été présentés en détail dans les chapitres précédents puisqu'ils sont à la base de la conception du système.

Le modèle du système. C'est le point de vue externe sur l'application, où l'on s'intéresse à la question « Quel est le problème à résoudre ? ». Ce modèle se fonde sur l'hypothèse systémique qui considère une application comme un système de type boîte noire transformant des images d'entrée en images de sortie. C'est une hypothèse délibérément simplificatrice mais qui permet d'envisager l'application dans sa réalité opératoire. Ce modèle vise ainsi à identifier l'application à travers ses finalités et ses interactions avec son environnement et rend compte du fait qu'une application de traitement d'images n'est pas une fin en soi, mais une partie d'une application plus globale dont elle dépend fonctionnellement.

Les informations de ce modèle sont représentées avec le langage basé sur l'ontologie de domaine introduit au chapitre 2.

Le modèle du domaine. C'est le point de vue contextuel sur l'application, où l'on cherche à découvrir « Quel est le contexte de l'application ? ». La définition de ce modèle repose sur les hypothèses phénoménologique et sémiotique qui amènent

à analyser une classe d'images sur trois niveaux : physique, perceptif et sémantique.

Les informations de ce modèle sont, elles aussi, représentées avec le langage de définition de classe d'images présenté dans le chapitre 2.

Le modèle des tâches. C'est le point de vue des décisions, où l'on tente de comprendre « Comment fonctionne l'application ? ». Ce modèle propose de formaliser une solution conceptuelle sous la forme d'un arbre de tâches hiérarchique qui associe, en plusieurs niveaux d'abstraction, les objectifs initiaux à une chaîne d'algorithmes de traitement. La modélisation de solutions par arbres de tâches permet l'encapsulation et l'abstraction de l'expertise sous forme de granules de connaissances hiérarchiques, définissant ainsi un moyen d'intégration des différentes approches cohabitant en traitement d'images.

La représentation d'une solution utilise plusieurs diagrammes empruntant la notation IDEF-0 (Mayer et al., 1992). C'est une notation basée sur les actigrammes de la méthode de génie logiciel SADT (Structured Analysis and Design Technique) (Marca and Mc Gowan, 1988) et dont la figure 5.2 présente un exemple. Chaque diagramme est consacré à la décomposition d'une tâche en sous-tâches. Une tâche est représentée par une boîte contenant une phrase décrivant l'objet de la tâche et ses contraintes propres. On ajoute au diagramme, un texte de justifications qui expose les raisons ayant motivé le choix de la décomposition proposée, en référence aux éléments du contexte ou des objectifs. Chaque alternative de décomposition fait l'objet d'un diagramme et d'une justification associée. La structure de graphe est conservée par la numérotation séquentielle des diagrammes, et la structure d'arbre par la numérotation arborescente des formulaires (*p. ex.*, A0, A1a, A1a1a).

Le modèle du programme. C'est le point de vue des opérations où l'on s'intéresse à la question « Comment est codée l'application ? ». L'hypothèse de base est celle de la programmation visuelle. C'est à dire qu'un processus de traitement, aussi complexe soit-il, peut toujours se discrétiser en une séquence de traitements ponctuels, dont les liens peuvent se réduire à des images ou des valeurs numériques. La représentation des programmes est faite par graphes d'opérateurs de la bibliothèque PANDORE.

Les modèles du système et du domaine deviennent le support de la simulation et définissent l'axe de la validation de la solution, parce que les résultats sont accessibles à la critique du client et peuvent être confrontés à la définition de l'objectif. Le modèle de tâches définit l'axe de la vérification des solutions et profite d'une graduation des solutions en niveaux d'abstraction. Enfin, le modèle du programme est le moyen de prototypage par excellence qui rend possible la simulation. Il définit l'axe des tests logiciels.

Le livre des connaissances structure quatre types de corpus à partir des quatre modèles :

- la spécification des buts,
- La définition des classes d'images,
- les solutions conceptuelles,
- les programmes sous la forme de graphes d'opérateurs.

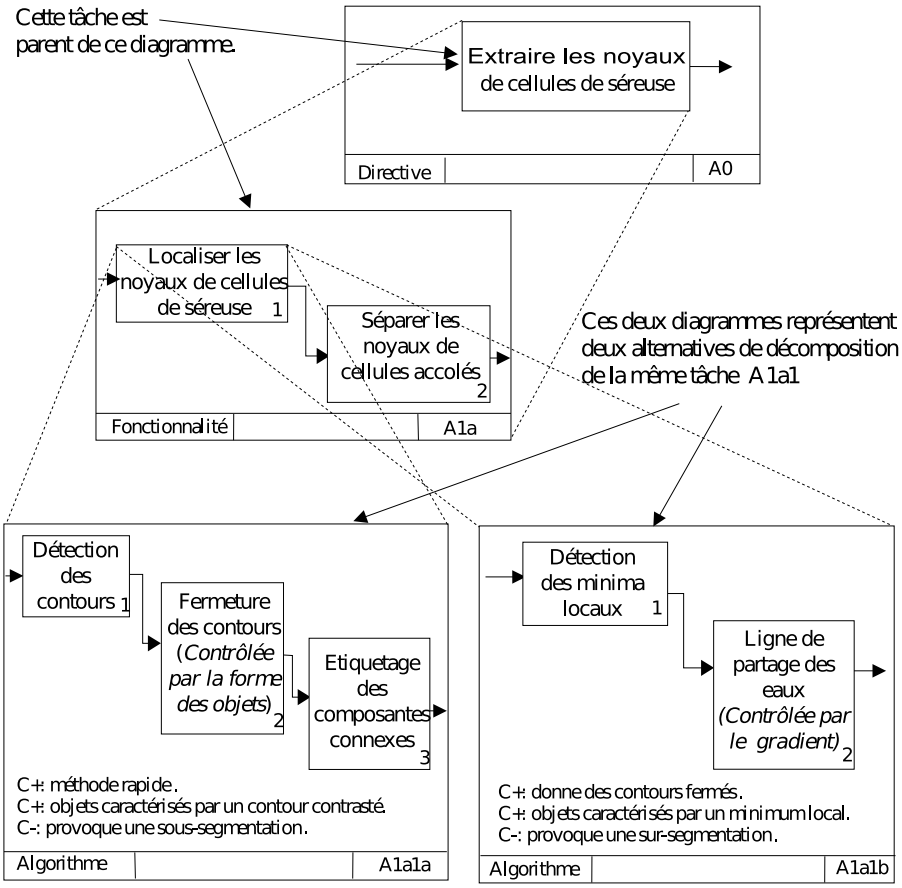


Figure 5.2 – Modélisation par tâches de deux façons de localiser les noyaux des cellules de séreuse. La première utilise la détection des contours puis leur fermeture pour obtenir les frontières des primitives régions. La seconde passe par la détection de germes à l'intérieur des régions et par une localisation des régions basée sur une Ligne de Partage des Eaux.

3.2.2 Méthode d'acquisition

L'atelier de génie logiciel PARTHÉNOS offre des interfaces aux experts pour leur permettre de modéliser leurs solutions pour chaque application, grâce à des représentations diagrammatiques conforment au format IDEF-0. L'interface permet de naviguer dans la représentation de la solution à tarvers les divers formulaires organisés hiérarchiquement, ou voir la solution complète sous la forme d'un plan (voir un exemple d'interface figure 5.3).

Les informations ainsi modélisées et les graphes d'opérateurs correspondants, produits avec l'interface de programmation visuelle ARIANE, composent le contenu du livre des connaissances en même temps que les corpus pour l'acquisition des connaissances.

L'atelier est capable de générer automatiquement les sources de connaissances qui permettent de reproduire chaque plan modélisé. Il utilise la tâche parent du diagramme pour définir la partie condition de la source (la tâche se déclenche s'il existe

sur le plan une tâche de même niveau dont le but est le même), et la décomposition décrite pour produire la partie action (la tâche s'exécute pour produire la même décomposition au niveau inférieur). Enfin, il utilise les critères pour et contre donnés pour justifier la décomposition proposé dans le diagramme pour définir les aptitudes de la source (essentiellement la valeur d'efficacité).

Parthenos a été utilisé, à des fins de validation, pour modéliser sous la forme de diagrammes IDEF-0 (et donc de sources de connaissances), différentes façons de faire de la détection de contours, de la classification de pixels ou de la correction de photométrie, en fonction des caractéristiques des images et des contraintes possibles.

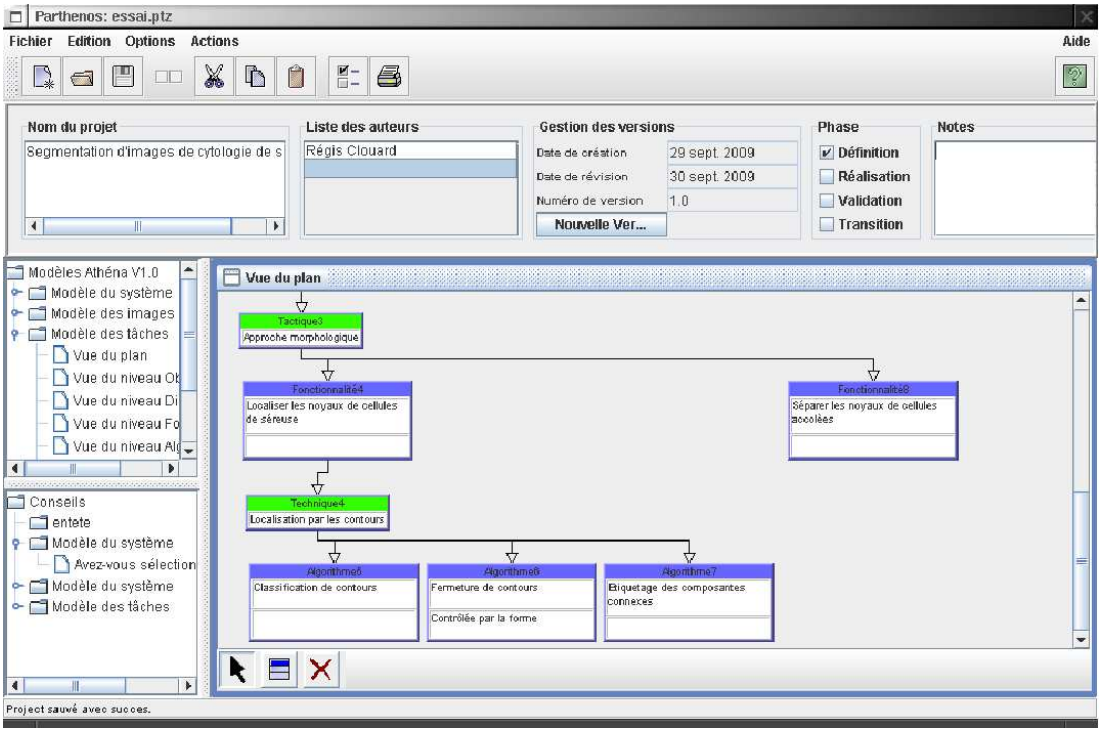


Figure 5.3 – L'interface de construction de plans de l'atelier de génie logiciel PARTHENOS

3.3 Un atelier pour l'expérimentation

Le volet droit de l'atelier permet d'utiliser le système de génération automatique d'applications pour expérimenter les connaissances capitalisées dans la base sur des exemples d'application.

Comme nous l'avons montré dans le chapitre 3, le modèle conceptuel du système de génération de programmes prend ses fondements dans les travaux portant sur la modélisation d'expertises par arbres de tâches pour formaliser les solutions (Chandrasekaran et al., 1992). Mais, il prend aussi ses fondements dans la logique de conception (angl. *Design Rationale*) pour capturer et tracer les raisons qui ont conduit à la conception de ces solutions (Buckingham, 1996). En conséquence, ce modèle permet d'accéder à deux espaces (Lee, 1997) : l'espace des solutions et l'espace des argumentations.

3.3.1 L'espace des solutions

L'espace des solutions permet de répondre aux questions sur le « comment ». Il correspond directement à la base de données du système qui contient les plans. Une solution y est représentée par un arbre de tâches organisé en niveaux d'abstraction, tel qu'il a déjà été présenté au chapite 3. Cet espace contient toutes les informations pour répondre au comment. La décomposition d'une tâche en sous-tâches explique ainis comment atteindre une tâche.

3.3.2 L'espace des argumentations

L'espace des argumentations permet de répondre aux questions sur le « pourquoi ». Une solution est décrite par la logique de conception. La logique de conception identifie les informations à représenter dans cet espace pour capitaliser les raisons qui ont conduit l'élaboration de la solution. Ainsi par exemple, la notation semi-formelle QOC (MacLean et al., 1991) est une notation de la logique de conception basée sur l'argumentation des solutions. Elle présente l'avantage d'être proche de la représentation des solutions par arbre de tâches. QOC manipule trois concepts : une Question (Q) est un but à atteindre, une Option (O) est une alternative de réponse et un Critère justifie (C+) ou infirme (C-) le choix de l'Option comme réponse. A l'aide de ces trois concepts, quatre types de relation sont exprimables et permettent alors d'accéder au raisonnement de conception : Réponse(Q,O), Soulève(O,Q), Argument pour(O,C), Argument contre(O,C). Une question correspond typiquement au but d'une tâche. Une option correspond à une décomposition d'une tâche en sous-tâches. En traitement d'images, ces critères concernent les informations collectées par le modèle du système et par celui du domaine :

- des caractéristiques sur les performances de la méthode en terme de coût-efficacité ;
- des contraintes associées au but d'une tâche ;
- des caractéristiques de la classe d'images ;
- des règles d'évaluation des alternatives.

La figure 5.4 présente une exemple d'utilisation de la logique de conception pour expliquer comment peut se faire la localiser des marqueurs des objets. Il y a deux solutions exposées. La première passe par une classification des pixels basée sur la couleur, et la seconde par une détection des minima régionaux de l'intensité. La première est efficace quand les noyaux sont caractérisés par leur couleur, mais elle peut provoquer une sur-segmentation, alors que la seconde est efficace quand les noyaux sont caractérisés par un minimum régional de la fonction intensité, et en plus elle est rapide.

D'origine, le modèle de Tableau Noir utilisé pour le système de génération automatique contient déjà toutes les informations permettant l'accès à la logique de conception à partir de la notation QOC. L'arbre des tâches fournit les Questions, les sources de connaissances exécutées (en fait les KSAR) fournissent les Options et les parties condition et aptitude des KSAR fournissent les Critères pour et contre. Il suffit alors d'ajouter une interface graphique pour naviguer dans la logique conception d'une application particulière.

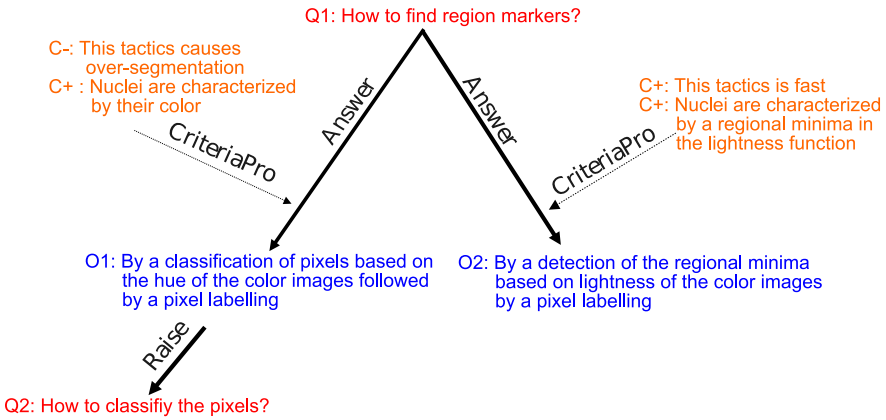


Figure 5.4 – Utilisation de la notation QOC pour justifier les deux alternatives possibles pour la tâche « find region markers ».

4 Conclusion

L'acquisition des connaissances est inhérente à notre approche de la conception de système de traitement d'images. Nous proposons pour cela un atelier d'ingénierie des connaissances qui, sous couvert d'aider les experts à développer des applications en mettant à leur disposition un ensemble d'outils logiciels, constitue un environnement de capitalisation des connaissances mises en jeu lors de ces développements. Ces connaissances sont ensuite utilisées pour concevoir le système à base de connaissances.

Nous avons choisi une approche constructiviste de l'acquisition qui vise à identifier les connaissances qui émergent de l'activité de développement. L'acquisition nécessite de développer des modèles originaux pour capturer toute la sémantique du développement d'application. Nous avons défini quatre modèles : du système, du domaine, de tâche et de programme pour permettent d'analyser l'application selon quatre points de vue. Ils définissent les modèles du livre des connaissances. Mais ils sont aussi utilisés pour concevoir le système à base de connaissances, c'est-à-dire les deux sous-systèmes de formulation et de génération automatique de programmes.

L'intérêt de cet atelier, c'est qu'il est peu intrusif dans la mesure où il augmente peu la charge cognitive des experts de traitement d'images. Par le soutils qu'ils proposent, il peut même, au contraire, permettre d'accroître la production par incitation à la réutilisation et se recentrer sur la couche métier en se débarrassant des problèmes de programmation et de formalisation. Ils assure le couplage entre la spécification du problème et la production de solutions mesurables, évitant ainsi la rupture du dialogue entre l'expert et l'utilisateur. Les connaissances modélisées sont aisément transférables dans la base de connaissances du système puisque le livre et la base de connaissances partagent les mêmes modèles. Ainsi formalisées les connaissances se prêtent plus facilement à la critique, à la révision et à la réutilisation.

Conclusion et perspectives

Arrivé au terme de ce mémoire, je voudrais souligner et justifier notre vision de la conception de systèmes de traitement d'images. Cette vision a été façonnée par l'approche cognitiviste, où l'on cherche à privilégier une représentation symbolique explicite des connaissances. Cela conduit naturellement à concevoir le système comme un système à base de connaissances. Au moins dans un premier temps, ce choix n'a pas tant été dicté par la recherche de performance du système, que par l'élaboration d'une théorie cognitive du développement d'application. Le système, dans ce cas, est un moyen de vérifier la calculabilité de la théorie.

Pour cela, l'ingénierie des connaissances a suivi une approche constructiviste qui fait que l'on ne cherche plus à modéliser les connaissances liées à l'activité cognitive des experts, mais plus simplement celles qui se dégagent de leurs pratiques. Cette approche suppose l'élaboration d'une théorie spécifique du développement d'applications à partir de l'analyse en contexte de l'activité des experts. C'est ensuite, à partir de cette théorie, que sont construits les modèles originaux du système à base de connaissances. En réalité, nos travaux ont d'abord commencé par le développement d'une première version du système. Cela a fourni les premiers modèles de notre théorie. Puis après avoir affiné notre théorie avec de nouveaux modèles, nous avons pu revisiter le système pour accroître ses capacités.

Cette vision tend à prouver qu'il existe bien un domaine du traitement d'images à part entière, qui possède ses propres connaissances et qu'il est alors possible de modéliser. Pour cela, il faut accepter les deux hypothèses, phénoménologique et de sémantique émergente, qui sont à la base de notre théorie. L'hypothèse phénoménologique postule que, seules les informations ayant une mesure dans l'espace de la vue sont nécessaires et suffisantes pour produire une solution. L'hypothèse de la sémantique émergente considère que la sémantique d'une application dans un domaine particulier peut toujours se représenter comme un ensemble de relations entre des structures purement syntaxiques construites uniquement avec des termes du traitement d'images. La représentation de la sémantique doit s'opérer au cours de cycles d'interaction homme-machine avec retour de pertinence, permettant de dépasser les deux problèmes de l'enracinement des symboles (angl. *symbol anchoring problem* et de l'ancrage des symboles (angl. *symbol grounding problem*).

Une théorie du développement d'application

Dans notre théorie, le développement d'une application suit le paradigme du pilotage d'opérateurs qui propose de coder le programme par un graphe d'opérateurs exé-

cutables paramétrés, pris dans une bibliothèque. Le graphe définit l'enchaînement des opérations à effectuer successivement sur les images d'entrée pour construire les images de sortie. À partir de là, notre théorie apporte des contributions originales à au moins trois niveaux : la formulation d'objectifs, la génération automatique de programmes, et l'interaction Homme Machine.

Formulation d'objectifs. La formulation d'objectifs s'entend comme la production d'une description linguistique à base de descripteurs à valeurs symboliques et iconiques, permettant ainsi une définition à la fois intensionnelle et extensionnelle du domaine d'application. Une définition intensionnelle permet de bien représenter la sémantique de l'application à partir d'une description des informations pertinentes et de leur relations. La définition extensionnelle permet de rendre compte de la réalité des données à partir d'exemples pris directement dans les images.

Le langage de formulation est défini sur une ontologie du domaine du traitement d'images qui fixe l'univers du discours. Tel qu'il est défini, ce langage est indépendant de tout système pouvant l'utiliser. Il autorise la construction de solutions aussi bien à partir d'un raisonnement symbolique abstrait qu'à partir d'un apprentissage supervisé.

Génération automatique de programmes. Nous avons fait ici le choix d'un raisonnement symbolique abstrait. La construction d'un graphe d'opérateurs solution est abordé comme un problème de planification hiérarchique de tâches. On cherche, par raffinement de tâches en sous-tâches, à construire un plan de traitement précis pour conduire la sélection, le paramétrage et l'enchaînement des opérateurs.

La construction du plan fait appel à trois niveaux de décision successifs, pour associer un graphe d'opérateurs solution à l'objectif de l'utilisateur : les décisions stratégiques, tactiques et techniques. Ces trois niveaux de décision conduisent à quatre niveaux d'abstraction pour la représentation du plan : les niveaux Directive, Fonctionnalité, Algorithme et Opérateur. Le dernier niveau est celui du graphe d'opérateurs final.

La modélisation des connaissances procédurales sous la forme de sources de connaissances modulaires, indépendantes, autonomes, et spécialisées, facilite l'acquisition des connaissances. Chaque source de connaissances se comporte comme un agent rationnel de niveau « connaissance » au sens de Newell (Newell, 1982). Elle renferme la compétence pour développer une alternative de décomposition d'une tâche en sous-tâches ou en opérateurs paramétrés, et présenter des mesures de son aptitude.

Interaction Homme-Machine. La théorie proposée implique fortement l'utilisateur dans la construction de solutions. La génération d'application se réalise autour d'un processus interactif, qui consiste pour l'utilisateur, à affiner la formulation de son objectif, en profitant de résultats produits à partir des versions précédentes de la formulation.

Au cours des interactions, l'utilisateur apprend la sémantique attribuée par l'ingénieur de la connaissance aux symboles disponibles pour formuler un objectif, à partir des conséquences sur les résultats. Le système de génération de programmes apprend, lui, la confiance que l'utilisateur accorde aux éléments de sa formulation, à partir des modifications apportées successivement par l'utilisateur aux éléments de la formulation.

Un atelier d'ingénierie des connaissances

L'atelier est le lieu de l'analyse de l'activité des experts. Il fournit les outils logiciels permettant d'organiser la production d'application et, en même temps, de capitaliser les productions sous la forme de corpus structurés. Ces corpus fournissent la matière à partir de laquelle il est possible de spécifier le comportement calculatoire du système à base de connaissances.

Cet atelier a été utilisé pour étudier plusieurs applications principalement d'origine biomédicale en raison des liens privilégiés entre le GREYC et les centres de recherche biomédicale voisins : CYCERON (Centre de recherches biomédicales dans le domaine des neurosciences), le centre de lutte contre de cancer F. BACLESSE et le CHU de Caen. Nous y avons aussi étudié plusieurs applications, issues de la littérature, par rétro-ingénierie, comme la détection de texte dans les images, la détection de plaques d'immatriculation, la restauration de films anciens, l'amélioration d'images, la segmentation d'images aériennes, etc.

Ces études nous ont permis de construire un patrimoine de connaissances, représenté dans un livre des connaissances, puis opérationnalisé dans le système à base de connaissances.

Perspectives

Centrée au début sur l'approche *cognitiviste*, notre vision de la conception de système adopte aujourd'hui une approche *hybride* qui ajoute à l'approche *cognitiviste* l'approche *basée image*, telle qu'elle a été présentée chap. 1, § 2.4. L'approche cognitiviste est utilisée pour sa capacité à représenter explicitement la sémantique du problème et la rendre maîtrisable par l'utilisateur. L'approche basée image est utilisée pour sa capacité d'adaptation grâce à l'apprentissage à partir d'exemples réels.

Dans cette nouvelle vision, l'approche cognitiviste est utilisée au plus haut niveau du raisonnement pour piloter, non seulement une bibliothèque d'opérateurs, mais aussi une bibliothèque de sous-systèmes basés sur l'approche image. Le pilotage de sous-systèmes basés image nécessite du système à base de connaissances, qu'il sache reconnaître le contexte d'application de chaque sous-système, mais aussi qu'il puisse spécialiser les sous-systèmes à l'application, en construisant automatiquement les fonctions d'évaluation nécessaires à l'apprentissage qui rendent compte de la sémantique représentée.

La mise en œuvre complète de la nouvelle version du système nécessite un certain nombre de travaux complémentaires à court et moyen termes :

- la construction automatique de fonctions d'évaluation à partir des éléments d'une formulation,
- la recherche du jeu de paramètres optimal,
- l'apprentissage automatique de plans de traitement,
- la reformulation par retour de pertinence.

Construction automatique de fonctions d'évaluation

Maintenant que nous disposons d'une formalisation de la formulation, nous pouvons envisager de construire automatiquement des fonctions d'évaluation des résultats, supervisées ou non, qui tiennent compte de l'objectif et la sémantique de

l'application. L'évaluation supervisée se fait par mesure de distance à une image de référence qui détient la vérité. L'évaluation non supervisée se fait à partir de mesures statistiques de qualité. Dans les deux cas, les mesures sont connues (Chabrier, 2005; Rosenberger et al., 2000), il suffit de les sélectionner en fonction des éléments de la formulation et de les combiner pour construire la métrique finale. Les éléments de la formulation intéressants sont :

- Les contraintes associées à la tâche. Par exemple, si la localisation est importante, alors l'évaluation ajoute une mesure de localisation des frontières.
- La définition des objets. Les descripteurs choisis par l'utilisateur peuvent être utilisés comme mesures de comparaison ou de stabilité.

La construction de fonctions d'évaluation sera abordée comme un problème de pilotage d'une bibliothèque de métriques, et résolu dans un tableau noir dédié.

Recherche du jeu de paramètres optimal

L'approche image peut aussi être utilisée pour déterminer le jeu de paramètres optimal d'un graphe d'opérateurs généré. Les travaux de V. Martin (Martin et al., 2006; Martin, 2007) et R. landais (Landais, 2006) ont déjà montré la faisabilité et proposé des méthodes. Là encore, le système à base de connaissances produit la fonction de calcul de la distance entre un résultat obtenu avec un jeu de paramètres donné et l'image de référence.

Apprentissage automatique

Il s'agit de doter le système à base de connaissances de capacité d'apprentissage pour le rendre plus autonome vis-à-vis de l'ingénieur de la connaissance. L'idée est de profiter directement des graphes d'opérateurs qui ont été construits par les experts au cours de développements manuels d'applications, pour en déduire automatiquement les plans sous-jacents et donc les sources de connaissances correspondantes.

Cette idée a déjà été initiée par le travail de N. Grosdenier et P. Loonis. Les auteurs proposent des mécanismes de découverte de motifs communs entre graphes d'opérateurs, dans le but de construire une abstraction des graphes sous la forme de *Patrons d'Opérateurs*, où « *un patron est une formalisation explicite et hors contexte d'une pratique récurrente des experts.* » (Grosdenier and Loonis, 2008).

Ici, nous comptons profiter de nos modèles de capitalisation des connaissances ainsi que de notre expérience en matière de raisonnement à partir de cas (Ficet-Cauchard et al., 1999; Ficet-Cauchard, 1999) pour développer les capacités d'apprentissage de connaissances explicites. Le raisonnement à partir de cas fournit les méthodes de mémorisation des graphes d'opérateurs en association avec la formulation d'objectifs correspondante, de telle façon qu'ils puissent être réutilisés, au moins en partie, après adaptation, pour d'autres applications.

Reformulation

Cette perspective à plus long terme concernent l'aide à la reformulation ou comment relier l'évaluation des résultats aux éléments de la formulation correspondante. C'est pour l'instant, une activité qui exige de fortes compétences en traitement d'images de la part de l'utilisateur. Nous souhaitons développer les capacités d'interaction du système pour guider l'utilisateur dans sa reformulation.

Pour cela, les approches énactives fournissent les bases méthodologiques, notamment autour de l'utilisation d'actions graphiques, qui permettent l'aide à la reformulation par retour de pertinence.

Références des ressources du projet

Panthéon : Le serveur du projet.
http:/www.greyc.ensicaen.fr/~regis/Pantheon

Pandore : Une bibliothèque et un environnement de développement d'opérateurs de traitement d'images.
http:/www.greyc.ensicaen.fr/~EquipeImage/Pandore

Ariane : Une interface de programmation visuelle.
http:/www.greyc.ensicaen.fr/~regis/Ariane

Hermès : Une interface homme-machine pour la formulation d'objectifs de traitement d'images.
http:/www.greyc.ensicaen.fr/~regis/Hermes

Ontologie : L'ontologie de domaine du traitement d'images orientée développement d'applications.
http :/www.greyc.ensicaen.fr/ regis/Pantheon/resources

Références

Abchiche, Y., Dalle, P., & Magnien, Y. (2002). Construction adaptative de concepts par structuration d'entités de traitement d'images. In : *13ème Congrés de Reconnaissance des Formes et Intelligence Artificielle*, pages 1023–1031.

Agarwal, S., Awan, A., & Roth, D. (2004). Learning to detect objects in images via a sparse, part-based representation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 26(11) :1475–1490.

Agarwal, S. & Roth, D. (2002). Learning a Sparse Representation for Object Detection. In : *7th European Conference on Computer Vision (ECCV))*, pages 113–130.

Ambler, S. W. & Jeffries, R. (2002). *Agile modeling : effective practices for extreme programming and the unified process*. John Wiley & Sons, Inc.

Aubry, F. & Todd-Pokropek, A. (2001). Mimos : A description framework for exchanging medical image processing results. In : *MEDINFO 2001*, R. R. V. Patel & R. Haux, ed., pages 891–895. IOS Press.

Bachimont, B. (1992). *Le contrôle dans les systèmes à base de connaissances*. Hermès.

Bachimont, B. (1996). *Herméneutique matérielle et Artéfacture : des machines qui pensent aux machines qui donnent à penser*. PhD thesis, École Polytechnique, Paris.

Bachimont, B. (2000). *Engagement sémantique et engagement ontologique : conception et réalisation d'ontologies en ingénierie des connaissances*, pages 305–323. Eyrolles.

Baroth, E. & Hartsough, C. (1994). Experience report : Visual programming in the real world. In : *Visual Object-Oriented Programming : Concepts and Environments*, G. &. L. Burnett, ed., pages 21–42. Manning Publications Co.

Bauer, M. (1994). Integrating Probabilistic Reasoning into Plan Recognition. In : *11th European Conference on Artificial Intelligence (ECAI))*, pages 620–624. John Wiley & Sons.

Berk, T., Brownston, L., & Kaufman, A. (1982). A New Color-Naming System for Graphics Languages. *IEEE Computer Graphics and Applications*, 2(3) :37–44.

Beucher, S. (1992). The watershed transformation applied to image segmentation. *Scanning Microscopy International*, suppl. 6 :299–314.

Bloehdorn, S., Petridis, K., Saathoff, C., Simou, N., Tzouvaras, V., Avrithis, Y., Handschuh, S., Kompatsiaris, Y.and Staab, S., & Strintzis, M. (2005). Semantic Annotation of Images and Videos for Multimedia Analysis. In : *Second European Semantic Web Conference (ESWC)*, pages 592–607.

Bodington, R. (1995). A Software Environment for the Automatic Configuration of Inspection Systems. In : *Int. Workshop on Knowledge Based Systems for the reUse*

of Program Libraries (KBUP), pages 100–108.

Booch, G., Rumbaugh, J., & Jacobson, I. (2005). *Unified Modeling Language User Guide, The (2nd Edition) (Addison-Wesley Object Technology Series)*. Addison-Wesley Professional.

Boucher, A. (1999). *Une approche décentralisée et adaptative de la gestion d'informations en vision ; Application à l'interprétation d'images de cellules en mouvement*. PhD thesis, Joseph Fourier University, Grenoble, France.

Boucher, A. & Garbay, C. (1996). A multi-agent system to segment living cells. In : *International Conference on Pattern Recognition*, volume 3, pages 558–562.

Bovemkamp, E., Dijkstra, J., Bosch, J., & J.H.C, R. (2004). Multi-agent segmentation of IVUS images. *Pattern Recognition*, 37 :647–63.

Brun, L. & Kropatsch, W. (2006). Contains and Inside relationships within combinatorial Pyramids. *Pattern Recognition*, 39(4) :515–526.

Buckingham, S. (1996). Design Argumentation as Design Rationale. *The encyclopedia of Computer Science and Technology*, 35(20) :95–128.

Butenuth, M., Straub, B.-M., & Heipke, C. (2004). Automatic Extraction of Field Boundaries from Aerial Imagery. In : *KDNet Symposium on Knowledge-Based Services for the Public Sector*, pages 14–25.

Cãmara, G., Engenhofer, M., Fonseca, F., & Monteiro, A. (2001). What's In An Image ? In : *Int. Conf. on Spatial Information Theory : Foundations of Geographic Information Science*, volume 2205, pages 474–488.

Cao, G., Mao, Z., Yang, X., & Xia, D. (2008). Optical Aerial Image Partitioning Using Level Sets Based on Modified Chan-Vese Model. *Pattern Recognition Letters*, 29(4) :457–464.

Capdevielle, O. (1995). *Formulation d'objectifs de traitement d'images : une exploitation interactive d'un système de planification automatique de chaînes d'opérateurs*. PhD thesis, Paul Sabatier University, Toulouse, France.

Carel, D. (1989). *Système expert d'aide à l'utilisation d'algorithmes de traitement d'images*. PhD thesis, University of Rennes, France.

Ceccarelli, M., Musacchia, F., & Petrosino, A. (2006). Content-based Image Retrieval by a Fuzzy Scale-space Approach. *Int. Journal of Pattern Recognition and Artificial Intelligence*, 20(6) :849–868.

Chabrier, S. (2005). *Contribution à l'évaluation de performances en segmentation d'images*. PhD thesis, University of Bourges, France.

Chandrasekaran, B., Johnson, T., & Smith, J. (1992). Task-structure analysis for knowledge modelling. *Communications of the ACM*, 35(9) :124–137.

Charlebois, D. (1997). *A planning system based on plan re-use and its application to geographical information systems and remote sensing*. PhD thesis, University of Ottawa, Canada.

Charlebois, D., Deguise, J.-C., Goodenough, D., Matwin, S., & Robson, M. (1991). A Case-Based Planner to Automate Development of Expert System Software for Analysis of Remote Sensing Data. In : *IEEE International Geoscience and Remote Sensing Symposium*, volume 3. IEEE.

Charlet, J. (2002). *L'ingénierie des connaissances : développements, résultats et perspectives pour la gestion des connaissances médicales*. Mémoire d'habilitation à diriger des recherches, Pierre and Marie Curie University, Paris 6.

Charlet, J., Bachimont, B., & Jaulent, M.-C. (2006). Building medical ontologies by terminology extraction from texts : An experiment for the intensive care units. *Computers in Biology and medicine*, 36(7) :857–870.

Charroux, B. & Philipp, S. (1995). Interpretation of Aerial Images based on Potential Functions. In : *9th Scandinavian Conf. on Image Analysis*, pages 671–678.

Charroux, B., Philipp, S., & Coquerez, J.-P. (1996). Un système de vision mettant en œuvre une coopération d'opérateurs de segmentation guidée par l'interprétation. In : *Congrès Reconnaissance des Formes et Intelligence Artificielle*, pages 527–537.

Chien, S. & Mortensen, H. (1996). Automating Image Processing for Scientific Data Analysis of a Large Image Database. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 18(8) :854–859.

Clouard, R. (1994). *Raisonnement incrémental et opportuniste appliqué à la construction dynamique de plans de traitement d'images*. PhD thesis, University of Caen, France.

Clouard, R. (2004). Une méthode de développement d'applications de traitement d'images. *Traitement du signal*, 21(4) :277–293.

Clouard, R., Elmoataz, A., & Revenu, M. (1999a). Une méthodologie de développement d'applications de traitement d'images. In : *17e Colloque GRETSI*, pages 323–326.

Clouard, R., Elmoataz, A., & Revenu, M. (2002). Une méthodologie de développement d'applications de traitement d'images. In : *13e congrès Reconnaissance des Formes et Intelligence Artificielle (RFIA)*, pages 1033–1042.

Clouard, R., Porquet, C., Abderrahim, E., & Revenu, M. (1993). Resolution of Image Processing Problems by Dynamic Planning Within the Framework of the Blackboard Model. In : *SPIE Int. Symposium on Intelligent Robot and Computer Vision*, volume 2056, pages 419–429.

Clouard, R., Porquet, C., Abderrahim, E., & Revenu, M. (1995a). A Software Workshop for Knowledge Acquisition in Image Processing. In : *Int. Workshop on the Design of Cooperative Systems*, pages 298–312.

Clouard, R., Porquet, C., Abderrahim, E., & Revenu, M. (1995b). Why building Knowledge-Based Image Segmentation is so difficult ? In : *Int. Workshop on Knowledge-Based systems for the reUse of Program Libraries*, pages 137–148.

Clouard, R., Porquet, C., Elmoataz, A., & Revenu, M. (1999b). Borg : A Knowledge-Based System for Automatic Generation of Image Processing Programs. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 21(2) :128–144.

Clouard, R., Revenu, M., Elmoataz, A., & Porquet, C. (1995c). Un atelier logiciel pour la conception d'applications de segmentation d'images. In : *15e Colloque GRETSI 95*, pages 577–580.

Clément, V. & Thonnat, M. (1993). A Knowledge-Based Approach to Integration of Image Procedures Processing. *Computer Vision, Graphics and Image Processing : Image Understanding*, 57(2) :166–184.

Cocquerez, J.-P. & Philipp, S. (1995). *Analyse d'images : filtrage et segmentation*. Enseignement de la physique. Masson.

Cohn, A., Bennett, B., Gooday, J., & Gotts, N. (1997). RCC : a calculus for Region based Qualitative Spatial Reasoning. *GeoInformatica*, 1(1) :275–316.

Colliot, O., Camara, O., & Bloch, I. (2006). Integration of fuzzy spatial relations in deformable models-Application to brain MRI segmentation. *Pattern Recogn.*, 39(8) :1401–1414.

Coradeschi, S. & Saffiotti, A. (2003). An Introduction to the Anchoring Problem. *Robotics and Autonomous Systems*, 43(2-3) :85–96.

Coster, M. & Chermant, J.-L. (1985). *Précis d'analyse d'images*. Presses du CNRS.

Coudé, S. (1996). Analyse d'images aériennes ; Mise en oeuvre de l'Atelier Logiciel

d'Intégration de connaissances en traitement et en interprétation d'images. Master's thesis, University of Caen.

Crevier, D. & Lepage, R. (1997). Knowledge-Based Image Understanding Systems : a Survey. *Computer Vision and Image Understanding*, 67(2) :161–185.

Crowley, J. L., Hall, D., & Emonet, R. (2007). Autonomic Computer Vision Systems. In : *Int. Conf. on Computer Vision Systems (ICVS'07)*.

Dalle, P. (2000). *Conception de systèmes d'interprétation d'image*. Mémoire d'habilitation à diriger des recherches, Paul Sabatier University, Toulouse, France.

Dalle, P. & Dejean, P. (1998). Planification en traitement d'image : approche basée sur les données. In : *11e congrès Reconnaissance des Formes et Intelligence Artificielle (RFIA)*, pages 75–84.

Dasiopoulou, S., Mezaris, V., Kompatsiaris, I., Papastathis, V.-K., & Strintzis, M. (2005). Knowledge-Assisted Semantic Video Object Detection. *IEEE Trans. on Circuits and Systems for Video Technology*, 15(10) :1210–1224.

de Rosnay, J. (1975). *Le macroscope : vers une vision globale*. Éditions du seuil.

Dejean, P. (1996). *Un formalisme pour les entités du traitement et de l'analyse des images.* PhD thesis, Paul Sabatier University, Toulouse, France.

Dejean, P. & Dalle, P. (1996a). Image analysis operators as concept constructors. In : *IEEE Southwest Symposium on Image Analysis and Interpretation*, pages 66–70.

Dejean, P. & Dalle, P. (1996b). Un langage de description de concepts pour la formulation d'objectifs d'analyse. In : *5èmes Journées ORASIS - ORASIS'96*, pages 219–224.

Deruyver, A. (2006a). Image Interpretation with a semantic graph : labeling oversegmented images and detection of unexpected objects. In : *actes de Aerosense 99 (SPIE) session : Applications and Science of computational intelligence II*, volume 3722, pages 424–432. SPIE.

Deruyver, A. (2006b). Pyramide Adaptative et Graphe sémantique : Un processus de segmentation dirigée par la connaissance. In : *15e congrès Reconnaissance des Formes et Intelligence Artificielle (RFIA)*.

Draper, B., Bins, J., & Baek, K. (1999). ADORE : Adaptive Object Recognition. In : *Int. Conf. on Vision Systems*, pages 522–537.

Draper, B., Collins, R., Brolio, J., Hanson, A., & Riseman, E. (1989). The Schema System. *International Journal of Computer Vision*, 2(3) :209–250.

Draper, B., Hanson, A., & Riseman, E. (1996). Knowledge-Directed Vision : Control, Learning, and Integration. *Proceeding of the IEEE*, 84(11) :1625–1637.

Dubuisson-Jolly, M.-P. & Gupta, A. (2000). Color and texture fusion : application to aerial image segmentation and GIS updating. *Image and Vision Computing*, 18(10) :823–832.

Duff, M. (1996). Forum. *IAPR Newsletter*, 21(1).

Elmoataz, A., Revenu, M., & Porquet, C. (1992). Segmentation and Classification of Various Types of Cells in Cytological Images. In : *IEE Image Processing and its Applications*, pages 385–388.

Ermine, J.-L. (2000a). *La gestion des connaissances*. Hermès.

Ermine, J.-L. (2000b). *Les systèmes de connaissances (2e edition)*. Hermès.

Ester, M., Kriegel, H.-P., Sander, J., & Xu, X. (1996). A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In : *2nd Int. Conf. on Knowledge Discovery and Data Mining*, pages 226–231.

Fadili, M. J. (1999). *Analyse spatio-temporelle des signaux d'activation cérébrale en IRM fonctionnelle*. PhD thesis, University of Caen.

Fergus, R., Perona, P., & Zisserman, A. (2003). Object Class Recognition by Unsupervised Scale-Invariant Learning. In : *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 264–271.

Ficet-Cauchard, V. (1999). *Réalisation d'un système d'aide à la conception d'applications de Traitement d'Images : une approche basée sur le Raisonnement à Partir de Cas*. PhD thesis, University of Caen.

Ficet-Cauchard, V., Porquet, C., & Revenu, M. (1999). CBR for the management and reuse of image-processing expertise : a conversational system. *Engineering Applications of Artificial Intelligence*, 12(6) :733–747.

Frucci, M., Perner, P., & Sanniti di Baja, G. (2008). Case-Based-Reasoning for Image Segmentation. *Int. Journal of Pattern Recognition and Artificial Intelligence*, 22(5) :829–842.

Gaines, B., Dhaw, M., & Woodward, J. (1993). Modelling as Framework for Knowledge Acquisition Methodologies and Tools. *Int. journal of Intelligent Systems*, 8(2) :155–168.

Garbay, C. (2001). Architectures logicielles et contrôle dans les systèmes de vision. In : *Les systèmes de vision*, J.-M. Jolion, ed., chapter 7, pages 197–251. Hermès.

Garbay, C. (2002). Pour une conception distribuée des systèmes de vision. *L'Objet*, 8(4) :71–94.

Gonzalez, R. & Woods, R. (2002). *Digital Image Processing (2nd edition)*. Prentice-Hall Inc.

GREYC (2009). Pandore : Une bibliothèque d'opérateurs de traitement d'images. [Online ; accessed 03-July-2009].

Grosdenier, N. & Loonis, P. (2008). Découverte et acquisition de connaissances métier à l'aide d'une approche fondée sur les arbres de suffixes. In : *16ème atelier de raisonnement à partir de cas (RAPC)*, pages 51–63.

Guyon, I. & Elisseeff, A. (2003). An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3 :1157–1182.

Hanson, A. & Riseman, E. (1978). VISIONS : A computer System for Interpreting Scenes. In : *Computer Vision Systems*, Hanson & Riseman, ed., pages 303–334. Academic Press.

Haralick, R., Cinque, L., Guerra, C., Levialdi, S., Weng, J., Huang, T., Meer, P., Shirai and, Y. Draper, B., & Beveridge, J. (1994). Dialogue : Performance Characterization in Computer Vision. *CVGIP : Image Understanding*, 60 :245–265.

Harnard, S. (1990). The symbol grounding problem. *Physica D*, 42(1-3) :335–346.

Hasegawa, J.-I., Kubota, H., & Toriwaki, J.-I. (1986). Automated Construction of Image Processing Procedures by Sample-Figure Presentation. In : *8th Int. Conf. on Pattern Recognition (ICPR)*, pages 586–588.

Hayes-Roth, B. (1985). A Blackboard architecture for control. *Artificial Intelligence*, 26(3) :251–321.

Hudelot, C. (2005). *Towards a Cognitive Vision Platform for Semantic Image Interpretation ; Application to the Recognition of Biological Organisms*. PhD thesis, University of Nice Sophia Antipolis, France.

Hudelot, C., Atif, J., & Bloch, I. (2008). Fuzzy spatial relation ontology for image interpretation. *Fuzzy Sets Systems*, 159(15) :1929–1951.

Hudelot, C. & Thonnat, M. (2003). A Cognitive Vision Platform for Automatic Recognition of Natural Complex Objects. In : *15th IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, page 398. IEEE Computer Society.

Jeon, J., Lavrenko, V., & Manmatha, R. (2003). Automatic Image Annotation and

Retrieval Using Cross-Media Relevance Models. In : *26th Int. Conf. on Research and Development in Information Retrieval*, pages 119–126.

Joly, M. (1994). *Image et les signes : approche sémiotique de l'image fixe*. Nathan.

Jurie, F. & Triggs, B. (2005). Creating Efficient Codebooks for Visual Recognition. In : *Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV)*, volume 1, pages 604–610. IEEE Computer Society.

Kadir, T. & Brady, M. (2001). Saliency, Scale and Image Description. *International Journal of Computer Vision*, 45 :83–105.

Kleppe, A. G., Warmer, J., & Bast, W. (2003). *MDA Explained : The Model Driven Architecture : Practice and Promise*. Addison-Wesley Longman Publishing Co., Inc.

Kruchten, P. (2003). *The Rational Unified Process : An Introduction*. Addison-Wesley Longman Publishing Co., Inc.

Labiche, J., Trupin, E., Saidali, Y., Dionisi, D., Holzem, M., & Boudon, J. (2007). De l'analyse à l'interprétation d'images de documents. In : *7ème journées francophones Extraction et Gestion des Connaissances, Actes de l'atelier ECOI*, pages 13–24.

Landais, R. (2006). *Compréhension des systèmes d'extraction d'objets dans la vidéo sous l'angle de l'adaptation*. PhD thesis, University of Lyon, France.

Lansky, A., Friedman, M., Getoor, L., Schmidler, S., & Short Jr., N. (1995). The Collage/Khoros links : Planning for image processing tasks. In : *AAAI Spring Symposium : Integrated Planning Applications*, pages 67–76.

Lee, J. (1997). Design Rationale Systems : Understanding the Issues. *IEEE Expert : Intelligent Systems and Their Applications*, 12(3) :78–85.

Lefèvre, V., Pollet, Y., Philipp, S., & Brunessaux, S. (1996). Un système multi-agents pour la fusion de données en analyse d'images. *Traitement du Signal*, 13(1) :99–111.

Leibe, B., Ettlin, A., & Schiele, B. (2008). Learning semantic object parts for object categorization. *Image Vision Computing*, 26(1) :15–26.

Leibe, B. & Schiele, B. (2003). Interleaved Object Categorization and Segmentation. In : *British Machine Vision Conference (BMVC)*, pages 145–161.

Levin, A., Lischinski, D., & Weiss, Y. (2004). Colorization using optimization. *ACM Transaction on Graphics*, 23(3) :689–694.

Levner, I., Bulitko, V., Lihong, L., Lee, G., & Greiner, R. (2003a). Automated Feature Extraction for Object Recognition. In : *Int. Conf on Image and Vision Computing*, pages 309–313.

Levner, I., Bulitko, V., Lihong, L., Lee, G., & Greiner, R. (2003b). Towards Automated Creation of Image Interpretation Systems. In : *16th Australian Joint Conference on Artificial Intelligence*, pages 653–665.

Lezoray, O. & Cardot, H. (2000). Cooperation of Color Pixel Classification Schemes and Color Watershed : a Study for Microscopic Images. *IEEE Trans. on Image Processing*, 11(7) :783–789.

Li, Q., Luo, S., & Zhongzhi, S. (2007). Semantics-Based Art Image Retrieval Using Linguistic Variable. In : *Fourth Int. Conf. on Fuzzy Systems and Knowledge Discovery*, pages 406–410.

Liedtke, C. & Blömer, A. (1992). Architecture of the Knowledge Based Configuration System for Image Analysis "Conny". In : *IEEE Int. Conf. on Pattern Recognition (ICPR)*, pages 375–378.

Liedtke, C., Bückner, J., Grau, O., Growe, S., & Tönjes, R. (1997). AIDA : A System for the Knowledge Based Interpretation of Remote Sensing Data. In : *3rd Int. Airborne*

Remote Sensing Conference & Exhibition, volume II, pages 313–320.

Linster, M. (1993). Closing the Gap between Modeling to Make Sense and Modeling to Implement Systems. *International Journal of Intelligent Systems*, 8(2) :209–230.

Liu, H. & Yu, L. (2005). Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge and Data Engineering*, 17(4) :491–502.

MacLean, A., Young, R., Bellotti, V., & Moran, T. (1991). Questions, Options and Criteria : Element of Design Space Analysis. *Human-Computer Interaction*, 6(3) :201–250.

Maillot, N. (2005). *Ontology Based Object Learning and Recognition*. PhD thesis, University of Nice Sophia Antipolis, France.

Maillot, N. & Thonnat, M. (2008). Ontology-Based Complex Object Recognition. *Image and Vision Computing*, 26(1) :102–113.

Maillot, N., Thonnat, M., & Boucher, A. (2004). Towards Ontology Based Cognitive Vision (Long Version). *Machine Vision and Applications*, 16(1) :33–40.

Marca, D. & Mc Gowan, C. (1988). *SADT : Structured Analysis Design Technique*. Mc Graw Hill.

Marr, D. & Hildreth, E. (1980). Theory of Edge Detection. *Proceedings of the Royal Society of London, Series B, Biological Sciences*, 207(1167) :215–217.

Martin, V. (2007). *Cognitive Vision : Supervised Learning for Image and Video Segmentation*. PhD thesis, University of Nice Sophia Antipolis, France.

Martin, V., Maillot, N., & Thonnat, M. (2006). A Learning Approach for Adaptive Image Segmentation. In : *4th IEEE Int. Conf. on Computer Vision Systems (ICVS)*, pages 40–47.

Matsuyama, T. (1989). Expert Systems for Image Processing : Knowledge-Based Composition of Image Analysis Processes. *Computer Vision, Graphics and Image Processing*, 48(1) :22–49.

Matsuyama, T. & Hwang, V. (1990). *SIGMA : A Knowledge-Based Aerial Image Understanding System*. Perseus Publishing.

Mayer, R., Painter, M., & deWitte, P. (1992). *IDEF Family of Methods for Concurrent Engineering and Business Re-engineering Applications*. Knowledge Based Systems Inc.

Mezaris, V., Kompatsiaris, I., & Strintzis, M. (2003). An Ontology Approach to Object-based Image Retrieval. In : *IEEE Int. Conf. on Image Processing (ICIP03)*, volume II, pages 511–514.

Mezaris, V., Kompatsiaris, I., & Strintzis, M. (2004). Region-based image retrieval using an object ontology and relevance feedback. *Journal on Applied Signal Processing*, 6(1) :886–901.

Miller, G. (1956). The Magic Number Seven, Plus or Minus Two : Some Limits on our Capacity for Processing Information. *Psychological Review*, 63(2) :81–97.

Moisan, S. & Ermine, J.-L. (2000). Gestion opérationnelle des connaissances sur les codes. In : *Journées francophones Ingénierie des Connaissances (IC)*.

Moisan, S., Vincent, R., Van Den Elst, J., & Van Harmelen, F. (1995). Towards an Intelligent Failure Handling Mechanims in Program Supervision. In : *Int. Workshop on Knowledge Based systems for the reUse of Program Libraries (KBUP)*, pages 109–118.

Moisan, S. & Ziébelin, D. (2000). Résolution de problèmes en pilotage de programmes. In : *12ème Congrés Reconnaissance des Formes et Intelligence Artificielle (RFIA)*.

Mueller, M., Segl, K., & Kaufmann, H. (2004). Edge- and region-based segmentation

technique for the extraction of large, man-made objects in high-resolution satellite imagery. *Pattern Recognition*, 37(8) :1619–1628.

Muller, P.-A. & Gaertner, N. (2000). *Modélisation objet avec UML*, (2nd ed.). Eyrolles.

Nazif, A. & Levine, M. (1984). Low Level Image Segmentation : An Expert System. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 6(5) :555–577.

Neumann, B. & Möller, R. (2008). On Scene Interpretation with Description Logics. *Image and Vision Computing*, 26(1) :82–110.

Newell, A. (1982). The Knowledge Level. *Artificial Intelligence*, 19(2) :87–127.

Nii, H. P. (1989). Introduction. In : *Blackboard Architectures and Applications*, V. Jagannathan, R. Dodhiawala, & L. Baum, ed., pages xix–xxix. Academic Press.

Nonaka, I. & Takeuchi, H. (1995). *The Knowledge-Creating Company : How Japanese Companies Create the Dynamics of Innovation*. Oxford University Press.

Nonaka, I. & Takeuchi, H. (1997). *La connaissance créatrice. La dynamique de l'entreprise apprenante*. De Boeck Université.

Nouvel, A. (2002). *Description de concepts par un langage visuel pour un système d'aide à la conception d'applications de traitement d'images*. PhD thesis, Paul Sabatier University, Toulouse, France.

Nouvel, A. & Dalle, P. (2002). Une approche interactive de définition d'ontologies image. In : *13ème Congrés de Reconnaissance des Formes et Intelligence Artificielle*, pages 1023–1031.

Perner, P. (1999). An Architectiure for a CBR Image Segmentation System. *Journal on Engineering Application in Artificial Intelligence*, 12(6) :749–759.

Perner, P., Holt, A., & Richter, M. (2005). Image Processing in Case-Based Reasoning. *The Knowledge Engineering Review*, 20(3) :311–314.

Polanyi, M. (1958). *Personal Knowledge : Towards a Post-Critical Philosophy*. University of Chicago Press.

Protière, A. & Sapiro, G. (2007). Interactive Image Segmentation via Adaptive Weighted Distances. *IEEE Trans. on Image Processing*, 16(4) :1046–1057.

Pylyshyn, Z. (1984). *Computation and Cognition : Towards a Foundation for Cognitive Science*. MIT Press.

Rao, A. & Lohse, G. (1993). Towards a Texture Naming System : Identifying Relevant Dimensions of Texture. In : *4th IEEE Conference of Visualization*, pages 220–227.

Rapaport, W. (2003). What Did You Mean by That ? Misunderstanding, Negotiation, and Syntactic Semantics. *Minds and Machines*, 13(3) :397–427.

Rasure, J. & Kubica, S. (1994). The Khoros Application Development Environment. In : *Experimental Environments for Computer Vision and Image Processing*, H. C. &. J. Crowley, ed., pages 1–32. World Scientific.

Raymaekers, C. (2009). Editorial. Special issue on enactive interfaces. *Interacting with Computers*, 21(1–2) :1–2.

Reichgelt, H. (1991). *Knowledge Representation : An Ai Perspective*. Ablex Publishing Corporation.

Renouf, A. (2007). *Modélisation de la formulation d'applications de traitement d'images*. PhD thesis, University of Caen, France.

Renouf, A., Clouard, R., & Revenu, M. (2005). Un modèle de formulation d'applications de traitement d'images. In : *ORASIS'05*, page 10.

Renouf, A., Clouard, R., & Revenu, M. (2007a). A Platform Dedicated to Knowledge Engineering for the Development of Image Processing Applications. In : *Int. Conf. on Enterprise Information Systems (ICEIS)*, pages 271–276.

Renouf, A., Clouard, R., & Revenu, M. (2007b). How to formulate image processing

applications ? In : *Int. Conf. on Computer Vision Systems (ICVS)*, pages 1–10.

Renouf, A., Clouard, R., & Revenu, M. (2007c). Un système pour la formulation d'applications de traitement d'images. *Traitement du Signal*, 24(5) :337–352.

Retz-Schmidt, G. (1988). Various views on spatial prepositions. *AI Magazine*, 9(1) :95–105.

Rosenberger, C., Chabrier, S., Laurent, H., & Emile, B. (2000). Unsupervised and Supervised Segmentation Evaluation . In : *Advances in Image and Video Segmentation*, Y.-J. Zhang, ed., pages 365–395. IRM Press.

Rost, U. & Münkel, H. (1998). Knowledge-Based Configuration of Image Processing Algorithms. In : *Int. Conference on Computational Intelligence and Multimedia Applications (ICCIMA)*, pages 9–11.

Russ, J. (1995). *The Image Processing Handbook (2nd edition)*. IEEE Press.

Saidali, Y., Adam, S., Ogier, J., Trupin, E., & Labiche, J. (2004). Knowledge Representation and Acquisition for Engineering Document Analysis. In : *Fifth IAPR International Workshop on Graphics Recognition (GREC 2003)*, pages 33–43.

Saidali, Y., Trupin, É., Labiche, J., Baudouin, N., & Holzem, M. (2002). Incremental Modelling and Acquisition of Image Processing knowledge. In : *International Conference WWW/Internet (ICWI))*, pages 184–190.

Sakaue, K. & Tamura, H. (1985). Automatic Generation of Image Processing Programs by Knowledge Verification. In : *IEEE Conference on Computer Vision and Pattern Recognition*, pages 189–192.

Santini, S., Gupt, A., & Jain, R. (2001). Emergent Semantics through Interaction in Image Databases. *IEEE Trans. Knowledge and Data Engineering*, 13(3) :337–351.

Schierwagen, A. (2001). Vision as Computation, or : Does a Computer Vision System Really Assign Meaning to Images. In : *Integrative Systems Approaches to Natural and Social Sciences*, H. M. . J. K. M. Matthies, ed., pages 579–787. Springer-Verlag.

Schmitt, M. (1989). Mathematical Morphological and Artificial Intelligence : an automatic programming system. *Signla Processing*, 16 :389–401.

Shafer, G. (1976). *A Mathematical Theory of Evidence*. Princeton University Press.

Simon, H. (1969). *The Sciences of the Artificial*. The MIT Press.

Smeulders, A., Worring, M., Santini, S., Gupta, A., & Jain, R. (2000). Content-based image retrieval at the end of the early years. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(12) :1349–1380.

Tamura, H., Sato, H., Sakaue, K., & , Kubo, F. (1984). DIA-Expert system and its knowledge representation scheme. *Trans. of Information Processing Society of Japan*, 29(2) :199–207.

Tanaka, T. & Sueda, N. (1988). Knowledge acquisition in image processing expert system EXPLAIN. In : *Int. Workshop on Artificial Intelligence for Industrial Applications*, pages 267–272.

Thonnat, M. & Moisan, S. (1995). Knowledge-based systems for program supervision. In : *Int. Workshop on Knowledge Based systems for the reUse of Program Libraries (KBUP)*, pages 4–8.

Thonnat, M. & Moisan, S. (2000). What can program supervision do for program reuse ? *IEEE Proceedings Software*, 147(5) :179–185.

Thonnat, M., Moisan, S., & Crubézy, M. (1999). Experience in Integrating Image Processing Programs. In : *Intern. Conference on Vision Systems (ICVS)*, H. Christensen, ed., Lecture Notes in Computer Science, pages 200–215. Springer-Verlag.

Théot, C., Gallier, E., & Mischler, D. (1992). VSDE, un environnement de développement automatisé de systèmes de vision. *Revue technique Thomson-CSF*,

24(4) :867–885.

Toriu, T., Iwase, H., & Yoshida, M. (1987). An Expert System for Image Processing. *Fujitsu Science and Technical Journal*, 23(2) :111–118.

Town, C. (2006). Ontological Inference for Image and Video Analysis. *Machine Vision and Applications*, 17(2) :94–115.

Udupa, J. K. (1982). Interactive segmentation and boundary surface formation. *Computer Graphics and Image Process*, 18(1) :213–235.

Van Den Elst, J. (1996). *Knowledge modelling for program supervision in image processing*. PhD thesis, University of Nice-Sophia Antipolis, France.

Van Heijst, G., Schreiber, A., & Wielinga, B. (1997). Using explicit ontologies in KBS development. *Int. J. Hum.-Comput. Stud.*, 46(2-3) :183–292.

Vandenbroucke, N., Macaire, L., & Jack-Postaire, G. (2003). Color image segmentation by pixel classification in an adapted hybrid color space : application to soccer image analysis. *Computer Vision and Image Understanding archive*, 216(2) :190–216.

Varela, F., Thompson, E., & Rosch, E. (1991). *The Embodied Mind : Cognitive Science and Human Experience*. MIT Press.

Vernon, D. (2007). Cognitive vision : The case for embodied perception. *Image and Vision Computing*, 26(1) :127–140.

Viola, P. & Jones, M. J. (2004). Robust Real-time Object Detection. *International Journal of Computer Vision*, 57 :137–154.

Whitley, K. & Blackwell, A. (2001). Visual Programming in the Wild : A survey of LabVIEW programmers. *Journal of Visual Languages and Computing*, 12(4) :435–472.

Wikipedia (2009). Image Processing — Wikipedia, The Free Encyclopedia. [Online ; accessed 03-July-2009].

Wolf, C. (2003). *Text detection in images taken from video sequences for semantic indexing*. PhD thesis, University of Lyon, France.

Xu, F., Li, X., & Yan, Q. (2003). Aerial Images Segmentation based on SVM. In : *Int. Conf. on Machine Learning and Cybernetics*, volume 4, pages 2207–2211.

Zamperoni, P. (1996). Plus ça va, moins ça va. *Pattern Recognition Letters*, 17(7) :671–677.

Zhang, Y. (1996). A Survey on Evaluation Methods for Image Segmentation. *Pattern Recognition*, 29(8) :1335–1346.

Zhou, X.-R., Shimizu, A., Hasegawa, J., Toriwaki, J., Hara, T., & Fujita, H. (2001). Vision Expert System 3D-IMPRESS for Automated Construction of Three Dimensional Image Processing Procedures. In : *Seventh Int. Conf. on Virtual Systems and Multimedia*, pages 527–536.

Zou, J. & Nagy, G. (2006). Human-Computer Interaction for Complexe Pattern Recognition Problems. In : *Data Complexity in Pattern Recognition*, T. K. Ho & M. Basu, ed., pages 271–286. Springer.