HAL
archives-ouvertes.fr

Une modélisation explicite et opérationnelle de la connaissance de Traitement d'Images

Régis Clouard, Abderrahim Elmoataz, Marinette Revenu

▶ **To cite this version:**

Régis Clouard, Abderrahim Elmoataz, Marinette Revenu. Une modélisation explicite et opérationnelle de la connaissance de Traitement d'Images. 11e Congrès Reconnaissance des Formes et Intelligence Artificielle (RFIA), Jan 1998, Clermont-Ferrand, France. II, pp.65-74, 1998. <hal-00822016>

HAL Id: hal-00822016
https://hal.archives-ouvertes.fr/hal-00822016

Submitted on 13 May 2013

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.

Une modélisation explicite et opérationnelle de la connaissance de Traitement d'Images

An explicit and operative modelling of Image Processing knowledge

Régis Clouard Abderrahim Elmoataz Marinette Revenu
Groupe de Recherche en Informatique, Image et Instrumentation de Caen, UPRESA CNRS 6072

GREYC - ISMRA, 6 Bd Maréchal Juin, F-14050 Caen, France
E-mail: Regis.Clouard@greyc.ismra.fr

Résumé

Nous nous intéressons à la spécification d'une modélisation des connaissances de Traitement d'Images qui soit à la fois utilisable par un système à base de connaissances pour résoudre automatiquement des applications, et compréhensible par des experts du Traitement d'images pour faciliter l'activité d'acquisition des connaissances. Nous montrons que l'utilisation du modèle des Sources de Connaissances des Blackboards associé à une approche de la résolution par planification hiérarchique des traitements offrent une réponse intéressante à ces deux objectifs. Ce sont les propriétés de modularité, d'indépendance, d'autonomie et de localisation qui font des Sources de Connaissances des pièces de connaissances qui renforcent la résolution d'applications et facilitent l'acquisition des connaissances.

Mots clés

Traitement d'images, Modélisation des connaissances, Planification, Sources de Connaissances.

Abstract

We are interested in specifying an Image Processing knowledge modelling, that is both computable by a knowledge-based system in order to automatically solve applications, and understandable by Image Processing experts in order to enhance the knowledge acquisition. We point out that the use of the Blackboard Knowledge Sources model associated with a hierarchical planning approach of the resolution leads to an interesting solution for those two objectives. It is the modularity, independence, autonomy and localisation properties that make Knowledge Sources pieces of knowledge that increase the resolution process and make easier the knowledge acquisition.

Keywords

Image Processing, Knowledge Modelling, Planning, Knowledge Sources

1 Introduction

Le problème posé est celui de la spécification d'une modélisation des connaissances de Traitement d'Images, qui soit à la fois *opérationnelle* et *explicite*. Les connaissances concernées sont toutes les connaissances impliquées dans le développement d'une application, mais *indépendantes* des domaines d'application, c'est-à-dire exemptes de toutes références, explicites ou non, aux concepts d'un quelconque domaine d'application (géographique, biomédical...).

- L'indépendance des connaissances aux domaines d'application est un gage de généricité et donc de réutilisabilité de ces connaissances.
- Un modèle opérationnel autorise l'exploitation des connaissances par un système de développement automatique d'applications. On attend d'un tel système qu'il possède des capacités d'auto-configuration lui permettant de décider de manière autonome les traitements à utiliser pour résoudre une application définie par un utilisateur.
- Un modèle explicite favorise l'appropriation des connaissances par les experts du Traitement d'Images, pour faciliter l'activité d'acquisition de connaissances.

L'enjeu des travaux présentés doit être vu ici comme une contribution à une formalisation de l'activité de développement d'applications en Traitement d'Images qui précise notamment la façon de décrire un problème et la façon de le résoudre.

Les contraintes précédentes sont directement motivées par l'absence de théories et de méthodes universelles qui systématiseraient le traitement d'une image [17]. Une modélisation purement algorithmique des méthodes de résolution n'est donc pas envisageable. Par contre, des expertises existent qu'il peut être profitable de modéliser et d'intégrer dans un système à base de connaissances, au fur et à mesure de leur mise en évidence. Dans cette perpective, la modélisation des connaissances prend sa source dans l'analyse de la démarche et du raisonnement des experts. Cela implique de savoir modéliser d'une part des stratégies d'analyse et des techniques de traitement en termes de conditions d'applicabilité, d'effets attendus sur les images et d'aptitude à accomplir une tâche donnée, et d'autre part des problèmes en termes d'intentions (objectifs à accomplir) et de contexte (description de l'image à traiter). Le raisonnement de résolution est alors axé sur l'analyse des compétences modélisées (stratégies et techniques) en fonction de la description du problème à résoudre (intentions et contexte), pour composer une solution par sélection et enchaînement de traitements.

Cette approche de la modélisation, basée sur le raisonnement expert n'est pas unique. D'autres approches ont été proposées mais elles s'attachent essentiellement à l'opérationnalisation des connaissances pour permettre par exemple la *compétition de compétences* (ex: [8]) ou le *raisonnement par cas* (ex: [3]).

- La compétition de compétences consiste à exploiter la mise en concurrence de plusieurs stratégies d'analyse

plus ou moins figées exécutées en parallèle, de manière à construire un résultat final par fusion de résultats partiels. Dans ce cas, le raisonnement ne porte plus sur la sélection a priori des traitements à opérer, mais sur l'analyse a posteriori des critères de fusion des données de résultats.

– Le raisonnement par cas est axé sur la remémoration et l'adaptation de plans ou de parties de plans d'analyse construits par les experts pour des résolutions passées. La modélisation des connaissances n'intègre pas ici explicitement le raisonnement mis en œuvre pour construire ces plans. Seuls le résultat (les plans) et les informations nécessaires à sa réutilisation sont modélisés (ex: la nature du problème concerné dans [3]).

Dans ces deux cas, la connaissance n'est pas modélisée dans un souci d'explicitation, ce qui en fait un cadre moins « naturel » pour l'acquisition des connaissances que ce soit pour l'ajout de nouvelles compétences ou pour la maintenance de la base de cas. Nous choisissons, en conséquence, de nous placer résolument dans le cadre défini par l'approche inspirée du raisonnement expert.

Pour présenter notre modélisation, nous discutons au préalable du modèle d'expertise sur lequel nous nous sommes fondés, dans la section 2. Dans la section 3, nous détaillons les caractéristiques de la modélisation, dont nous exposons les avantages pour l'opérationnalisation des connaissances en section 4 et pour l'acquisition des connaissances en section 5. L'utilisation de cette modélisation est illustrée pour la résolution d'une application dans le domaine de la cytologie en section 6.

2 Les connaissances de traitement d'image

Il existe une grande diversité d'approches expertes pour résoudre une application. Chacune d'elles nécessite des catégories et des modélisations de connaissances propres. Nous choisissons ici de nous arrêter sur l'une d'entre elles : le pilotage d'une bibliothèque de programmes.

2.1 La résolution d'applications vue comme le pilotage de programmes

Avec l'avènement des environnements graphiques de développement de programmes tels que Khoros [16], Allegory [19] ou VPL [9], la construction d'applications par les experts prend de plus en plus les formes du pilotage d'une bibliothèque d'opérateurs prédéfinis. Les opérateurs codent des algorithmes de traitement sous la forme de programmes exécutables paramétrables. Ils constituent alors les briques de bases de la construction, qu'il faut sélectionner, paramétrer et assembler pour former des chaînes de traitement adaptées. Le programme final se présente sous la forme d'un graphe d'opérateurs individualisés, dont les liens sont les flux de paramètres et de données images échangés. Ces graphes intègrent toutes les structures de contrôle classiques (if, for, while), permettant par exemple l'ajustement dynamique des valeurs de paramètres d'opérateurs par optimisation de fonction de coût, ou le contrôle d'itération des opérateurs.

Pour les utilisateurs de ces environnements, l'intérêt réside incontestablement dans l'absence d'activité de programmation proprement dite et des contraintes et erreurs qu'elle entraine. Dans un objectif d'automatisation, elle présente aussi l'avantage de rendre possible la construction dynamique de programmes exécutables.

Du point de vue de la résolution de problèmes, le problème du pilotage d'une bibliothèque d'opérateurs peut avantageusement se concevoir comme un problème de planification, à partir du moment où l'on choisit de baser le raisonnement sur les actions à effectuer et non sur les données à produire [18], qui sont plus fluctuantes et incertaines par nature. Cependant, il existe des travaux récents dans lesquels le raisonnement de planification est axé sur l'analyse des données à produire. Par exemple, le système ExTI [10] utilise une modélisation des transformations de données des opérateurs pour opérer une planification par propagation de transformations le long des chaînes d'opérateurs possibles. La solution retenue est une des chaînes qui produit en sortie les résultats décrit par l'utilisateur. Mais, l'utilisation de ces systèmes reste limitée à la résolution d'objectifs ponctuels. La détermination des séquences d'objectifs ponctuels composant un problème complexe est donc entièrement laissée à la charge de l'utilisateur.

La résolution d'un problème basée sur une planification des actions procède alors de quatre activités:

1. La *planification* construit une solution conceptuelle sous la forme d'un plan hiérarchique qui associe en plusieurs niveaux d'abstraction le problème à résoudre à une séquence de traitements primitifs adaptée à sa résolution.
2. L'*instanciation* procède à la construction du graphe d'opérateurs par sélection, paramétrisation et enchaînement d'opérateurs de la bibliothèque, sur la base de la séquence de traitements primitifs précédente.
3. L'*exécution* contrôle l'exécution de chaque opérateur comme celle du graphe total, de manière à produire en sortie les images intermédiaires et finales.
4. L'*évaluation* juge de la pertinence des résultats intermédiaires par rapport aux spécifications du problème et propose des corrections pour les parties de plan non satisfaites. La correction correspond à une replanification (cf. activité 1.) ou à une reparamétrisation (cf. activité 2.).

La résolution complète d'un problème nécessite des cycles d'essais-erreurs autour de ces quatre activités.

Le pilotage d'une bibliothèque d'opérateurs constitue donc une démarche experte réaliste, sur laquelle nous pouvons nous appuyer. Pour preuve, c'est généralement cette démarche de résolution qui est mise en œuvre dans les systèmes d'aide au développement d'applications de traitement d'images type TMO [13].

2.2 Les connaissances à modéliser

Dans le cadre que nous venons de définir, les connaissances à modéliser sont:

– Les *connaissances sur l'image à traiter et sur le domaine d'application* renseignent la nature de l'image et lui donnent un sens, en précisant les informations à considérer comme pertinentes de celles qui ne le sont pas pour le problème courant. Elles sont données sous forme de connaissances factuelles au début ou au

cours de la résolution et sont utilisées à tous les niveaux du raisonnement pour orienter et contrôler les choix. Ces connaissances définissent ce que nous appelons une requête de traitement d'images.

- Les *connaissances sur l'expertise du traitement d'images* référencent les stratégies et les techniques de traitement d'images par leur condition d'utilisation. Elles indiquent comment résoudre un problème par une séquence de traitements élémentaires, en fonction des caractéristiques du contexte.
- Les *connaissances sur les opérateurs de la bibliothèque* détiennent la connaissance sémantique permettant la sélection et la paramétrisation des opérateurs, et la connaissance syntaxique permettant l'appel du programme qui est associé aux opérateurs et la réalisation d'enchaînements syntaxiquement corrects.
- Les *connaissances sur les traitements* autorisent l'évaluation des résultats par la détermination de mesures de comparaison entre les résultats obtenus et ceux à obtenir.
- Les *connaissances de contrôle* de la résolution ont en charge d'orienter la résolution et de résoudre les conflits. Elles sont responsables de la qualité et de la rapidité de convergence vers une solution acceptable.

Une modélisation de ces connaissances doit être capable d'intégrer des compétences reposant sur des références d'origines diverses (ex: traitement du signal, optique, morphologie mathématique) et pouvant se présenter sous forme numérique ou à différents niveaux d'abstraction.

2.3 Modélisation de connaissances pour le pilotage de programmes

Dans la plupart des travaux existants (VSDE [2], OCAPI [5], CONNY[15], MVP [4]), la modélisation des connaissances est entièrement fondée sur une représentation des expertises sous la forme de squelettes de plans. Un squelette de plan référence, par des liens explicites but-opérateurs, les différentes façons connues de résoudre une tâche de traitement d'images selon différentes natures d'image. Il est matérialisé par un arbre de tâches hiérarchiques traduisant un processus de décompositions successives de problèmes en sous-problèmes. Les tâches de l'arbre expriment à un niveau d'abstraction soit des actions à accomplir soit des fonctionnalités à utiliser. Les feuilles de l'arbre sont généralement des opérateurs de la bibliothèque. Seul, le système MVP n'intègre pas directement les opérateurs exécutables dans l'arbre. Les feuilles restent des tâches à accomplir mais d'un niveau très simple. Leur accomplissement se fait par une planification basée sur la propagation de transformations d'images analogue à celle d'ExTI.

A chaque nœud de l'arbre sont référencées les différentes alternatives de décomposition. Le choix d'une alternative à essayer pour une application particulière est géré par un paquet de règles de production attaché aux nœuds. Ces règles sont chargées de classer, de choisir ou d'éliminer des alternatives en fonction des caractéristiques de l'image et des contraintes des tâches. Elles sont de la forme: *Si l'image présente telles caractéristiques alors préférer (ou supprimer) telle alternative*.

Pour cela, *les connaissances sur l'image et le domaine d'application* sont modélisées de façon globale par une liste attributs-valeurs de caractéristiques mesurables symboliques ou numériques (ex: rapport signal/bruit, taille des objets, type de texture...).

Dans les systèmes où la phase d'exécution est véritablement prise en compte, le paramétrage est géré par d'autres règles de production. Ainsi dans Ocapi, le paramétrage est réglé dynamiquement selon des cycles initialisation, évaluation et ajustement. De même, l'évaluation est une extension de ce principe pour les tâches des niveaux supérieurs. L'évaluation automatique est un problème très difficile qui se limite généralement à détecter les aberrations de résultat plus que leur pertinence. Elle procède par remontée des résultats des décompositions puis application des règles locales. *Les connaissances sur les traitements* sont données sous forme de règles expertes du genre: si le résultat présente telles valeurs de caractéristiques (mesurées directement par des opérateurs adaptés) alors décider que le résultat est acceptable ou inacceptable. La gestion des erreurs est faite par backtrack.

Cette modélisation des connaissances conserve une forme relativement naturelle puisque les experts pensent souvent en terme d'étape et de niveaux d'abstraction [18]. Néanmoins, ses limites pour l'opérationnalisation et l'acquisition des connaissances viennent de ce que *les connaissances sur l'expertise du traitement d'images et sur les opérateurs* sont réparties à la fois dans le squelette de plan et dans les règles de choix. D'une part, il est difficile de résoudre le compromis entre construire des squelettes de plans très développés dont il est facile d'écrire puis de maintenir les règles de production localement mais dont les parties devenues très spécifiques sont difficilement réutilisables, et construire des plans plus concis qui favorisent donc la réutilisabilité mais dont les règles de production deviennent difficiles à écrire puis à maintenir au fur et à mesure que le nombre d'alternatives augmente. D'autre part, *les connaissances sur le contrôle* de la résolution sont confondues avec le squelette. La stratégie de résolution est strictement figée et il ne peut y avoir d'adaptation à des contraintes de résolution telle que préférer une solution moins efficace mais plus rapide ou le contraire.

3 Modélisation des connaissances dans Borg

Le système Borg (dont on trouvera une description complète de l'architecture dans [6]) repose sur le pilotage d'une bibliothèque, en l'occurrence Pandore [7], par planification des traitements. La modélisation des connaissances dans ce système se concentre essentiellement autour des Sources de Connaissances du modèle des Blackboards. Nous montrons que cette modélisation associée à un raisonnement en cinq niveaux hiérarchiques apporte des solutions satisfaisantes aux problèmes de l'opérationnalisation et de l'acquisition des connaissances.

3.1 Modèle conceptuel

Comme dans les approches précédentes, la résolution d'une application est vue comme un problème de planification de tâches de traitement.

3.1.1 Modèle de solution

Une solution est représentée par un arbre de tâches, qui associe en plusieurs niveaux de décomposition, la requête à une séquence d'opérateurs adaptée à sa résolution.

Une tâche de l'arbre correspond à une décision de transformation d'images d'entrée en images de sorties. Elle possède toutes les informations nécessaires et suffisantes à sa gestion, avec les attributs suivants:

but	L'identification de la tâche à accomplir par des mots clés.
contraintes	La liste des contraintes associées au but.
entrées	Le chemin pour récupérer les images d'entrée soit dans les sorties de tâches en amont soit dans les entrées de la tâche mère.
décomposition	La relation d'enchaînement des sous-tâches décomposant la tâche (ET, PUIS).
sorties	Le chemin pour récupérer les images de sorties dans les sorties des sous-tâches de la décomposition.
évaluation	Les règles d'évaluation des résultats de la tâche.
résultat	Le jugement constituant le résultat de l'évaluation.

Seules les tâches de niveau Opérateur possèdent deux attributs supplémentaires pour décrire le programme qu'elles réifient:

prototype	La syntaxe d'appel du programme.
mode	Le mode d'exécution de l'opérateur, parmi *normal*, *itération*, *répétition, optimisation*, *satisfaction.*

Une tâche utilise ces attributs pour organiser sa décomposition et gérer les flux de données images à échanger (Figure 1).

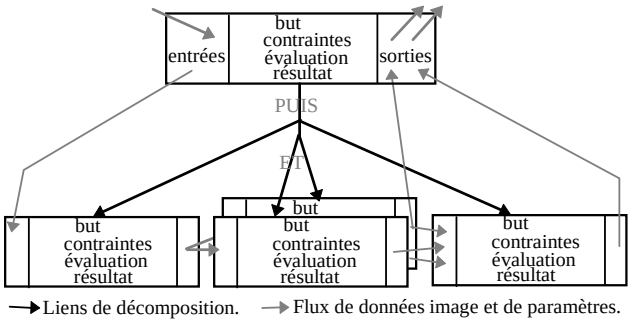

Figure 1. Les attributs gèrent la décomposition d'une tâche.

A la différence des approches précédentes, où l'arbre traduit un *processus de décompositions* successives de problèmes en sous-problèmes, pas toujours plus simples mais qui en constituent les étapes de résolution, l'arbre traduit ici les *abstractions du raisonnement* en trois niveaux consécutifs: intentionnel (définition des objectifs ponctuels), fonctionnel (définition des méthodes de traitements) et opérationnel (définition des outils de traitements). A un même niveau, la solution n'est constituée que de tâches d'un seul niveau d'abstraction. Ce sont en fait deux façons différentes d'appréhender la résolution, mais il est toujours possible de passer d'une représentation à l'autre moyennant une réorganisation de l'arbre [13].

Nous utilisons alors une modélisation des plans de traitements par un arbre de tâches à exactement cinq niveaux d'abstraction, dont les deux premiers correspondent à deux décompositions du niveau intentionnel, le troisième au niveau fonctionnel et les deux derniers au niveau opérationnel:

1. *Requête*: Ce niveau référence les tâches qui constituent la requête initiale de l'utilisateur.
2. *Objectif*: Ce niveau correspond à l'expression d'une stratégie d'analyse en terme d'étapes de traitement. Les tâches de ce niveau utilisent un vocabulaire de type intentionnel (ex: éliminer le fond).
3. *Fonctionnalité*: Ce niveau correspond à l'expression de méthodes générales du traitement d'images. Les tâches se réfèrent grossièrement à des classes d'opérateurs (ex: détection de contours, classification de pixels).
4. *Procédure*: Ce niveau correspond à l'utilisation de techniques de traitement d'images. Les tâches se réfèrent à des opérations de traitement, plus ou moins consensuels, sans qu'ils soient encore question de programme de la bibliothèque (ex: seuillage basé sur la maximisation de la variance interclasse).
5. *Opérateur*: Ce niveau concerne la configuration des programmes disponibles dans la bibliothèque (ex: binarisation, valvarmax).

L'arbre de type (et, puis) qui représente le plan, traduit exactement le processus de raffinements successifs de la requête jusqu'à la séquence d'opérateurs adaptée à sa résolution (Figure 2).

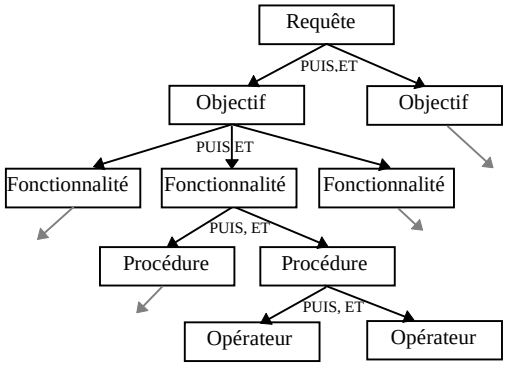

Figure 2. Un plan est un arbre de tâches à exactement 5 niveaux.

A un même niveau, le plan représente une version de la solution pour un niveau d'abstraction, sous la forme d'un graphe de tâche s'échangeant des images (Figure 3).

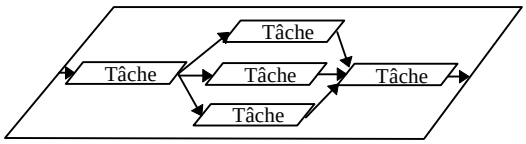

Figure 3. A un niveau, le plan est un graphe de tâches.

Le plan représente une solution unique qui n'inclue pas d'alternatives à l'intérieur. Il est entièrement construit pour un problème déterminé sans qu'il ait jamais existé au préalable, ni même en partie. Sa construction se fait sur une base de données qui lui est entièrement consacrée. Elle est organisée verticalement selon les cinq niveaux d'abstraction et horizontalement selon l'ordre d'exécution.

3.1.2 Modèle de résolution

Toutes les activités de résolution d'une application portent

sur l'arbre de tâches.

1. La planification correspond à la construction de l'arbre solution, par décomposition des tâches d'un niveau en sous-tâches du niveau inférieur. Chaque décomposition implique un choix utilisant des critères de son niveau d'abstraction.
2. L'instanciation se fait dans la continuité de la planification. Chaque procédure du plan produit est décomposée en une séquence d'opérateurs de la bibliothèque, pour lesquels il faut choisir les valeurs de paramètres et le mode d'exécution.
3. L'exécution des opérateurs contrôle l'exécution des programmes associée selon leur mode et leur ordre d'exécution. Les résultats de chaque opérateur sont récupérés et forment ses sorties.
4. L'évaluation procède en deux temps. Les résultats de chaque tâche sont remontés de sa décomposition, puis les règles locales d'évaluation sont appliquées pour former son résultat.

En cas de non-satisfaction, les corrections sont faites en reprenant l'activité de planification sur les parties incriminées.

3.2 La modélisation des connaissances par des Sources de Connaissances

Toute la connaissance de Traitement d'Images impliquée dans la résolution d'une application est éclatée en Sources de Connaissances (SC).

3.2.1 Modèle de SC

Une SC représente une action ponctuelle de la résolution agissant sur le plan solution soit pour participer à sa construction, son instanciation, son exécution ou son évaluation. Chaque SC n'est qu'une partie de la connaissance totale. Il faut nécessairement la coopération et la compétition de plusieurs SCs pour résoudre complètement une application.

Une Source de Connaissance possède toute la connaissance nécessaire au pilotage de son action en terme de condition d'applicabilité, d'aptitude à atteindre l'objectif auquel elle participe et d'adaptabilité au contexte courant. Il est impératif qu'elle respecte les propriétés de modularité, d'autonomie, d'indépendance et de localisation.

- *Modularité*: Une SC code une action et une seule. Néanmoins, elle possède toutes les connaissances nécessaires à l'adaptation de son action aux particularités du contexte. Il n'y a de granularité a priori sur la connaissance à embarquer dans la SC, sauf que le problème du contrôle doit être résolu à l'intérieur. Cela se traduit par le fait que les choix à l'intérieur de la SC ne doivent pas être heuristiques mais déterministes.
- *Indépendance*: Toutes les SCs s'ignorent mutuellement. Elles utilisent des données du plan dont elles ne connaissent pas les créatrices et produisent des données sur le plan dont elles ne connaissent pas les futures consommatrices.
- *Autonomie*: Une SC détient toutes les connaissances nécessaires pour déterminer quand elle peut contribuer à la résolution et pour estimer a priori les performances de sa contribution.
- *Localisation*: Chaque SC est liée à une tâche, ou au plus à un niveau, qu'elle est responsable de savoir reconnaître. Ce qui fait que l'action et le raisonnement mis en œuvre pour son pilotage ne renferment que des connaissances à un niveau d'abstraction unique.

Nous montrerons que ce sont ces propriétés des SCs qui rendent l'opérationnalisation des connaissances plus efficace et l'acquisition des connaissances plus facile.

3.2.2 Format d'une Source de Connaissances

Toutes les SCs de la base de connaissances sont décrites par les mêmes parties suivantes:

Déclencheur	Prédicats décidant de l'opportunité de la contribution.
Précondition	Prédicats décrivant les conditions d'applicabilité.
Évitement	Prédicats décrivant l'inutilité de la contribution.
Action	Règles de production agissant par modification du plan.
Importance	Règle de production estimant la priorité de la contribution.
Crédibilité	Règle de production estimant l'adéquation entre la contribution et les conditions spécifiées par les contraintes et le contexte.
Fiabilité	Règle de production estimant la probabilité d'obtenir les résultats escomptés compte-tenu du contexte.
Complexité	Règle de production estimant le coût de mise en œuvre de la méthode en terme de nombre d'opérations et de paramètres à ajuster.

Par la partie *déclencheur*, une SC est attachée à un niveau d'abstraction ou à une tâche déterminée d'un niveau, pouvant avoir des contraintes spécifiques. Elle confère aux SCs un comportement dirigé par les événements puisqu'elle ne porte que sur les modifications produites sur le plan par l'exécution des SCs.

Les parties *précondition* et *évitement* permettent de gérer les alternatives de décomposition d'une tâche. La précondition interdit à une SC de s'exécuter tant qu'une autre SC est en cours d'essai. La partie évitement la supprime de la liste des alternatives quand une SC produit des résultats acceptables ou que la tâche est supprimée du plan.

La partie *action* agit par création, suppression ou modification d'attributs de tâches dans le plan.

Les quatre caractéristiques *importance*, *crédibilité*, *fiabilité* et *complexité* calculent dynamiquement une estimation des performances de la contribution de la SC. Ce sont des règles de production dont la condition porte sur l'analyse des valeurs de caractéristiques des images et des contraintes associées à la tâche, et dont l'action produit une valeur symbolique parmi l'intervalle prédéfini:

[très-faible, faible, moyen, fort, très-fort].

Lorsqu'une SC raisonne sur une caractéristique qui n'est pas renseignée dans le contexte d'application, deux cas peuvent se présenter. Soit la caractéristique est essentielle au raisonnement de la SC, auquel cas la valeur est demandée interactivement à l'utilisateur puis stockée dans le contexte, soit elle n'est pas essentielle auquel cas cette caractéristique n'est pas prise en compte. Le distinguo est fait par l'expert au moment de l'écriture de la SC, en

choisissant pour cela une méthode différente d'accès à la valeur de la caractéristique.

3.2.3 Spécification d'une requête

Compte-tenu des informations utilisées par les SCs, une requête doit combiner la déclaration des intentions sur les images en termes de tâches contraintes à accomplir (qui constituent les tâches du niveau requête) et la description du contexte d'application en terme de caractéristiques mesurables (qui sont utilisées par les règles de production des parties importance, crédibilité, fiabilité et action des SCs).

Le vocabulaire utilisé doit proscrire toute référence aux domaines d'application pour se limiter à celui du Traitement d'images, du Traitement du signal, de la morphologie mathématique. Ceci est une conséquence de la contrainte de départ d'indépendance aux domaines.

Les tâches font globalement références aux objectifs du traitement d'images (détecter, segmenter, extraire, ...). Elles constituent les entrées dans le plan et doivent être connues des SCs du niveau Requête. Elles sont précisées par trois types de contraintes:

1. Les *critères à optimiser* placent la focalisation d'attention de la tâche sur des caractéristiques d'éléments à privilégier dans le résultat.
   (ex: localisation précise des frontières d'objets).
2. Le *niveau de détail* à atteindre fixe les limites des caractéristiques des éléments du résultat.
   (ex: taille des régions > 36).
3. Les *restrictions* à respecter imposent les limites de l'action de la tâche.
   (ex: ne pas séparer les objets qui se touchent).

La description du contexte est faite en termes de caractéristiques mesurables ou qualifiables. Elle est représentée par une liste d'attributs-valeurs qui combine des informations à trois niveaux de description [12], [1]:

1. Le niveau *physique* renseigne la physique d'acquisition de l'image, c'est-à-dire le capteur, les conditions d'acquisition et les caractéristiques résultantes sur l'image telle que la nature du bruit...
2. Le niveau *perceptif* décrit les entités présentes dans la représentation des images, telles que les caractéristiques des contours, du gradient, des régions...
3. Le niveau *sémantique* définit la notion d'objet pour la scène analysée en terme de propriétés géométriques, topologiques, de contraste, de texture et de relations qu'ils entretiennent entre eux (inclus, juxtaposés...).

Ce contexte est référencé comme une liste globale accessible à toutes les SCs.

3.2.4 Les différents types de SC

La base de connaissances distingue cinq types de SCs pour réaliser les quatre activités de résolution : de *planification*, *d'instanciation*, *d'exécution*, de *description* et *d'évaluation*. A chaque tâche des 3 premiers niveaux sont attachées les SCs de planification, de description et d'évaluation (Figure 4), aux tâches du niveau procédure celles d'instanciation, de description et d'évaluation, et aux tâches du niveau opérateur celles d'exécution.

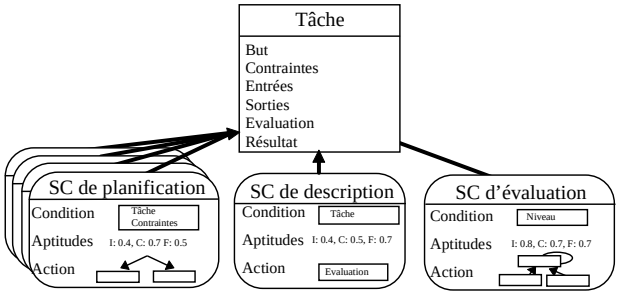

Figure 4. Liens entre SC et tâche pour les 3 premiers niveaux.

En détail :

1. Les SCs de *planification* ont en charge de produire le plan pour les quatre premiers niveaux par décomposition d'une tâche d'un niveau en sous-tâches au niveau inférieur. Une SC de ce type pilote par son action une décomposition unique. La condition est chargée d'identifier la tâche qu'elle peut raffiner. Ses caractéristiques donnent une évaluation de son aptitude à accomplir la tâche en fonction des caractéristiques du contexte et des contraintes.
2. Les Sc. d'*instanciation* ont en charge de sélectionner les opérateurs correspondant à une procédure, puis de fixer une valeur pour chaque paramètre soit avec une constante soit avec la valeur de retour d'un autre opérateur, et enfin de déterminer le mode d'exécution. Il suffit généralement d'une SC de ce type par procédure possible. En effet, la décomposition d'une procédure ne relève bien souvent pas d'un choix mais d'une simple traduction spécifique de la bibliothèque. C'est pourquoi les caractéristiques de ce type de SCs sont donc des constantes communes à toutes.
3. Les SCs d'*exécution* ne font que piloter un seul type d'exécution soit une optimisation, une itération, une répétition ou une satisfaction. La condition porte sur la création d'un opérateur quelconque avec le bon mode d'exécution et l'existence de données d'entrée obtenues par l'exécution des opérateurs qui en sont leurs sorties. Les caractéristiques sont des données intrinsèques qui indiquent simplement qu'il est important d'exécuter un opérateur lorsque cela est possible.
4. Les SCs de *description* pilotent la définition de règles d'évaluation pour une tâche. La condition est responsable d'identifier la tâche. L'action construit les règles en fonction des contraintes de la tâche et du contexte de l'application notamment la description des objets. Les règles d'évaluation sont des règles de production de type "`si alors (sinon)`". La partie condition porte sur des comparaisons de type statistique entre des mesures quantitatives extraites de sorties (obtenues par l'exécution d'opérateurs primitifs) et des mesures de références déduites des spécifications de la tâche. L'action consiste à formuler un résultat de type échec ou succès. Les caractéristiques sont des constantes puisqu'il n'y a qu'une SC de ce type par tâche. Dans le cas où les règles d'évaluation sont impossibles à formuler, la règle par défaut "`IF vrai THEN succès`" est utilisée.

5. Les SCs d'*évaluation* sont chargées de construire les sorties de la tâche avec des sorties de sa décomposition puis d'appliquer les règles d'évaluation pour former le résultat. La condition porte sur l'identification de la tâche et la présence de règles d'évaluation et de sorties dans les décompositions. Les caractéristiques sont des valeurs intrinsèques qui indiquent simplement qu'il est important d'évaluer un résultat qui possède des règles d'évaluation autres que la règle par défaut.

4 L'opérationnalisation des connaissances

L'opérationnalisation des SCs est mise en œuvre par le contrôle de la résolution, dont le problème est de déterminer à chaque instant la SC à exécuter.

4.1 Le contrôle de la résolution

Le choix des SCs à exécuter résulte ici d'un compromis entre la spécification d'un profil de SC adaptée à l'état courant du plan et les SCs effectivement exécutables à cet instant [14].

- La spécification d'un profil de SC se fait d'une part par la mise en place d'une stratégie générale de résolution qui indique à chaque instant sur quelle partie du plan se focaliser, et d'autre part par la définition d'heuristiques générales de comportement qui indiquent quelles sont les caractéristiques des SCs à privilégier. La stratégie d'analyse, choisie par l'expérience, procède en largeur d'abord. Elle s'implante sous la forme de 4 focus d'attention temporaires sur les 4 derniers niveaux du panneau de traitement d'image. Les heuristiques retenues privilégient les SCs de contrôle par rapport aux SCs du domaine, les plus récemment déclenchées, les plus crédibles, les plus fiables, les plus importantes et les moins complexes.
- Les SCs exécutables sont celles qui ont leurs parties déclencheur et précondition vérifiées, et leur partie évitement non vérifiée.

Ces décisions sont utilisées pour calculer une valeur de priorité pour chaque SC exécutable qui représente leur adéquation aux spécifications du contrôle. La résolution d'un problème se fait par des cycles successifs du scheduler qui fonctionne en trois étapes:

```
Tant que le statut du problème <> résolu faire
1) Mise à jour de l'agenda des SCs.
  a) Ajouter les nouvelles SCs déclenchées
  b) Déterminer les SCs exécutables
  c) Supprimer les SCs dont l'évitement est vérifié
  d) Pour chaque SC exécutable faire
       Calculer le coefficient de priorité
2) Choix de la SC à exécuter.
  a) Prendre la SC de plus grande priorité
3) Exécution de la SC.
  a) Exécuter les règles de sa partie action
```

Les décisions de contrôle peuvent être modifiées en fonction de contraintes associées au problème. Par exemple, si le temps d'exécution du traitement est une contrainte pour l'utilisateur, on privilégiera alors les SCs qui agissent du bas vers le haut (évaluation), et on donnera plus d'importance à la complexité qu'à la crédibilité et qu'à la fiabilité. Ces contraintes de contrôle sont paramétrables depuis l'interface du système et ne font pas partie de la requête de traitement d'image.

4.2 Intérêts de la modélisation

Le contrôle et la modélisation des connaissances proposés confèrent à la résolution un comportement incrémental et opportuniste. Elle est incrémental parce que l'agenda des SCs est remis à jour à chaque cycle. Les SCs sont choisies en fonction de l'état d'avancement de la résolution, ce qui veut dire que la résolution s'influe elle-même. Elle est opportuniste, parce que le choix de la SC à exécuter combine une stratégie générale et des préférences intrinsèques sur les caractéristiques. Ainsi, une SC d'exécution peut être préférée à une SC de planification du niveau en focus simplement parce qu'elle est plus importante.

Un comportement incrémental du contrôle présente trois avantages pour la recherche de solutions:

- La solution construite est totalement originale. Cela veut dire qu'il existe des solutions non prévues.
- La réutilisabilité profite de la modularité, de la localisation et de l'autonomie des SCs pour agglomérer les tâches dans le plan. L'action des SCs ne peut appartenir qu'à un seul des cinq niveaux d'abstraction et ne code qu'une action ponctuelle ce qui en fait des pièces de connaissances relativement génériques et adaptées. De plus, ce ne sont pas des règles de choix qui gèrent globalement le paquet d'alternatives de décomposition attachées à une tâche, mais les alternatives qui présentent leurs aptitudes indépendamment les unes des autres.
- La résolution tire avantage d'une base de connaissances comportant des SCs dont les actions se complètent, se concurrencent et se chevauchent.

Un comportement opportuniste permet de faire une gestion sophistiquée des erreurs tant au niveau local qu'au niveau global [18]:

- La gestion locale d'erreurs agit par backtrack. Un échec peut être remonté à la tâche supérieure si aucune des alternatives possibles n'a donné satisfaction, ou si les niveaux de fiabilité et de crédibilité des SCs correspondant à ces alternatives sont plus faibles que celles de la SC d'évaluation correspondante. Par contre, il n'y a pas de réexécution de parties de plan avec de nouvelles contraintes puisque l'on estime ici que chaque SC propose une décomposition optimale.
- La gestion globale d'erreurs agit par détection d'impasse au niveau global. Elle peut être gérée soit par le déclenchement opportuniste de SCs de contrôle générant un échec sur une tâche sans que toutes les alternatives s'y référant aient été essayée soit que les alternatives restantes ont une priorité inférieure aux SCs d'évaluation qui propagent une erreur.

5 L'acquisition des connaissances

L'acquisition des connaissances dénote les activités de maintenance corrective et de maintenance évolutive de la base de connaissances.

5.1 Mode d'acquisition

La maintenance corrective consiste uniquement à :

- Corriger les SCs dont le code est erroné ou incomplet. La découverte d'erreurs avérées se fait lors de la résolution d'application.

La maintenance évolutive consiste à :

- Créer de nouvelles alternatives de décomposition pour une tâche préexistante par une nouvelle SC de planification;
- Ajouter une nouvelle tâche des quatre premiers niveaux, ce qui signifie rajouter plusieurs SCs de planification et une SC de description (les SCs d'évaluation sont déjà toutes définies étant donné qu'elles en détiennent aucune expertise);
- Ajouter un nouvel opérateur ce qui signifie rajouter une nouvelle SC de d'instanciation;
- Ajouter de nouvelles caractéristiques dans le contexte de l'application.

Dans les deux cas, la maintenance est faite au fur et à mesure de la mise en évidence des besoins.

5.2 Intérêts de la modélisation

L'acquisition profite incontestablement des propriétés de modularité, d'indépendance et de localisation des SCs.

- L'écriture d'une nouvelle SC se fait indépendamment des SCs préexistantes que ce soit pour la partie action ou pour les caractéristiques. Pour ces dernières, les règles de production permettent de calculer une valeur symbolique représentative de la contribution et non pas une valeur de comparaison avec les autres alternatives. Ceci est possible puisque l'on recherche un plan solution satisfaisant et pas non nécessairement optimal. De plus, la granularité des connaissances est faible et limitée à un seul niveau d'abstraction.
- L'ajout de nouvelles SCs se fait sans remettre en cause ce qui existe déjà puisque d'une part le modèle de résolution supporte une base de connaissances incohérente et d'autre part ce sont les alternatives qui reconnaissent la tâche sur laquelle elles agissent, et non les règles attachées à la tâche qui gèrent globalement les alternatives. Dans notre cas, l'ajout se fait sans rien modifier des autres alternatives, dans l'autre cas cela nécessiterait de revoir les règles de production pour intégrer la gestion de la nouvelle alternative.
- L'ajout de nouvelles caractéristiques dans le contexte se fait sans remettre en cause les SCs qui existent déjà, puisqu'une caractéristique apparaît avec la première SC qui l'utilise.

Par contre, la modélisation ne permet pas aux experts d'avoir une vision globale du contenu de la base de connaissances ni de connaître les tâches et les caractéristiques déjà reconnues par les SCs. C'est pourquoi, notre équipe développe en parallèle un système d'acquisition des connaissances [13] permettant en autres d'appréhender les plans générables par Borg.

6 Exemple d'une base de connaissances pour le traitement d'images de cytologie

L'exemple utilisé reprend une étude menée par [11], dans le cadre de l'optimisation d'un analyseur d'images dédié à la détection des anomalies de ploïdies des tumeurs cancéreuses humaines. La requête de traitement d'image correspondante (présentée Figure 5) est d'isoler les objets cellulaires qui reposent sur le fond des images en vue de leur classification ultérieure en catégories cellulaires (tâche non prise en compte ici). Les contraintes précisent qu'il faut respecter la distinction entre objets isolés et amas d'objets, et qu'il faut localiser leur frontière avec précision. Le contexte met en évidence que le fond et les objets sont caractérisés et séparables par leur niveau de gris, et que les objets sont de forme convexe et de taille moyenne.

```
Tâches: (Isolate objects from the background)
Contraintes:
(Object-Separation= partial)        ; Niveau de détail
(Boundary-Localization= precise)    ; Critère à optimiser
(Clusters= to-be-kept)              ; Restrictions
Image:("/images/cytology11.pan")
Contexte:
(; Description physique
(length-of-image= 256)
(width-of-image= 256)
(noise= low)
(imaging-device= microscope)
(quality= sharp)
; Description perceptive
(type-of-image= density)
(background?= yes)
(background= homogenous)
(gray-level= discriminative)
(boundaries-contrast= high)
(texture?= no)
; Description sémantique
(background-color= light)
(object-number-of-classes= 1)
(object-size= average)
(object-color= dark)
(object-repartition= dissociated)
(object-mini-size= 36)
```

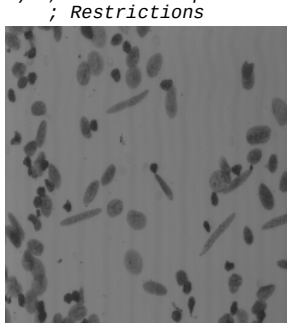

Figure 5. Extrait d'une requête possible pour la segmentation d'images de cytologie de l'œsophage.

La Figure 6 présente le plan solution construit en conséquence. Ce plan traduit une stratégie d'analyse descendante consistant à d'abord éliminer le fond de l'image, pour ne garder que les régions correspondant aux zones d'intérêt, puis à rechercher les objets dans ces régions.

On peut noter dans ce plan, que le seuil de l'opérateur *Binarization* (O5) est obtenu par optimisation de la localisation des frontières. L'intervalle des valeurs potentielles

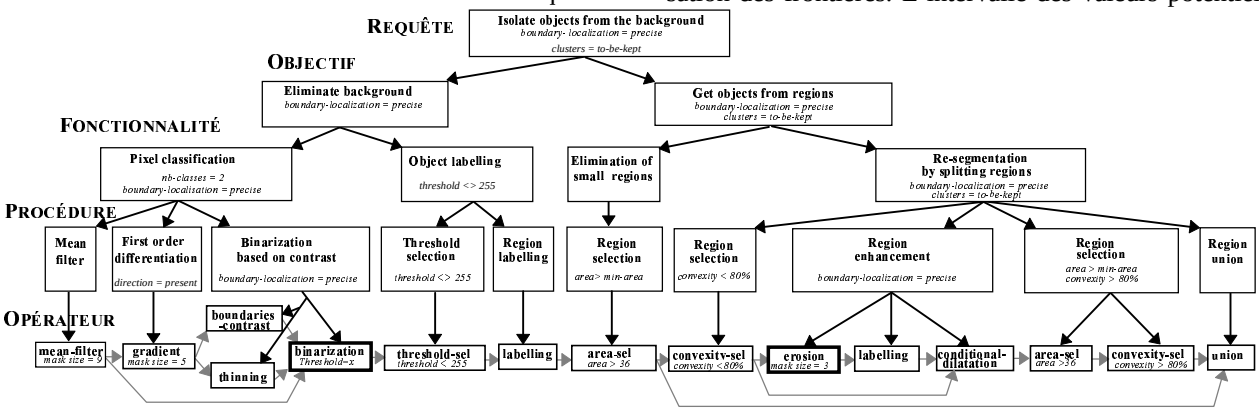

Figure 6. Un exemple de plan possible pour la requête Figure 5.

est construit autour de la valeur retournée par l'opérateur *Boundary-Contrast* (O3) et la fonction d'évaluation utilisée indique que le seuil est maximum lorsqu'est maximum le nombre de points communs entre les frontières des régions obtenues et les contours obtenus par l'opérateur *Thinning* (O4):

```
intervalle=[5-(valeur(O3)), 5+(valeur(O3))]
évaluation=(nbpts (and (sortie(O5))(sortie(O4))))
```

L'opérateur *Erosion* est itéré 2 fois pour essayer de séparer les régions qui se touchent partiellement sans séparer celles qui se chevauchent franchement. Ce nombre d'itérations est calculé à partir de la valeur de l'attribut *object-min-size* du contexte. L'évaluation du résultat de la tâche *Pixel-Classification* est faite à partir de la règle:

```
IF (nbclass(sortie))=2) THEN succès ELSE échec
```

qui consiste à vérifier le nombre de classes obtenues en sortie.

La Figure 7 visualise quelques SCs utilisées pour construire le plan.

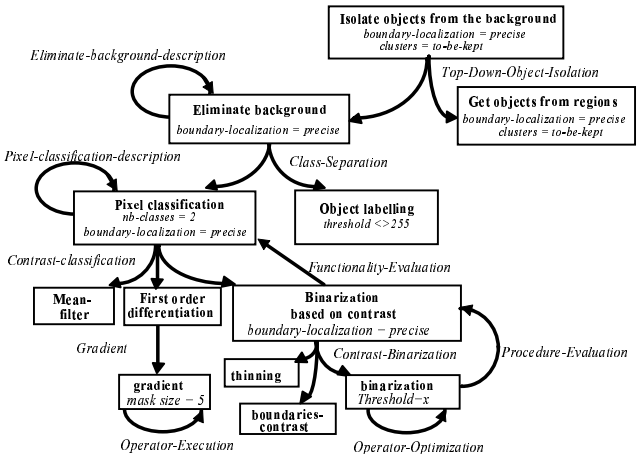

Figure 7. Quelques unes des SCs utilisées pour résoudre l'application.

On peut noter que la SC *Top-Down-Object-Isolation* implante la stratégie d'analyse descendante: d'abord éliminer le fond, puis construire les objets par séparation des régions obtenues précédemment. Elle a été choisie car ses valeurs de crédibilité et de fiabilité sont fortes lorsque le fond est homogène et séparable des objets. Une alternative aurait été de commencer par détecter des primitives régions (par exemple à partir de la détection des minima régionaux) puis de reconstruire les objets et les amas à partir des ces régions primitives. La SC *Class-Separation* a été choisie car les deux classes fond et objet sont séparables par leur niveau de gris. Elle propose d'éliminer le fond par classification des pixels en deux classes, puis de supprimer la région correspondant au fond. La SC *Constrast-Classification* (Figure 8) propose une classification basée sur le contraste. Elle a été choisie parce que le contraste aux frontières est bien marqué. Une alternative toute aussi probable compte tenu du contexte, aurait été de prendre la variance interclasse comme critère. La première SC a été choisie simplement parce qu'elle était la première essayée. La mise en œuvre est faite par la SC d'instanciation *Contrast-Binarization* (Figure 10). On y retrouve aussi la SC de description *Pixel-Contrast-Classification* (Figure 9) qui construit une règle pour vérifier qu'en sortie on obtient bien 2 classes de pixels.

```
Nom : Contrast-Classification
Niveau-Entrée : Fonctionnalité
Niveau-Sortie : Procédure
Condition-Vars : (F)
Déclencheur : ∃ 1 fonctionnalité F tq
        F.but = (pixel classification)
        et (nb-classes=2) ∈ F.contraintes
Précondition : F.décomposition =()
Evitement : F.statut=inactif ou F.résultat=succès
Importance : moyen
Crédibilité : IF (gray-level=discriminative) ∈F.contexte
           THEN fort ELSE très-faible
Fiabilité :IF (object-boundary=contrasted)∈ F.contexte
          and (object-texture=homogeneous)∈ F.contexte
           THEN fort ELSE très-faible
Complexité : faible
Action :
  Créer Procédure P1
   P1.but = (mean filter)
   P1.contraintes = ()
   P1.entrées = (1ère de F.entrées)
  Créer Procédure P2
   P2.but = (first order differentiation)
   P2.contraintes = ((gradient-direction-image=present))
   P2.entrées = (1ère de P1.sorties)
  Créer Procédure P3
   P3.but = (binarization based on contrast)
   P3.contraintes = F.contraintes
   P3.entrées = (1ère de P1.sorties,
                 2ème de P2.sorties,
                 3ème de P2.sorties)
  F.decomposition = (THEN P1 P2 P3)
  F.sorties = ((1ère de P3.sorties)
```

Figure 8. Le code de la SC de planification Contrast-Classification.

```
Nom : Pixel-Classification-Description
Niveau-Entrée : Fonctionnalité
Niveau-Sortie : Fonctionnalité
Condition-Vars : (F)
Déclencheur : ∃ 1 Fonctionnalité F tq
        F.but = (pixel classification)
Précondition : ()
Evitement : F.statut = inactif
Importance : fort
Crédibilité : fort
fiabilité : moyen
Complexité : très-faible
Action :
F.évaluation =
  IF nb-classes ∈ F.contraintes THEN
        "If (nbclass (F.sorties)= valeur(nb-classes))
         Then succès Else échec"
  ELSE "If vrai Then succès"
```

Figure 9. Le code de SC de description Pixel-Classification-Description.

```
Nom : Contrast-Binarization
Niveau-Entrée : Procédure
Niveau-Sortie : Opérateur
Condition-Vars : (P)
Déclencheur : ∃ 1 Procédure P tq
        P.but = (binarization based on contrast)
Precondition : P.décomposition =()
Evitement : P.statut = inactif ou P.résultat = succès
Importance : moyen
Crédibilité : moyen
Fiabilité : moyen
Complexité : fort
Action :
  Créer Opérateur O1
   O1.but = (boundaries contrast)
   O1.contraintes = ()
   O1.mode = normal
   O1.prototype = (()(data → data))
   O1.entrées = (1ère de P.entrées, 2ème de P.sorties)
  Créer Opérateur O2
   O2.but = (thinning)
   O2.contraintes = ()
   O2.mode = normal
   O2.prototype = (()(data data → data))
   O2.entrées = (2ème de P.entrées, 3ème de P.entrées)
  Créer Opérateur O3
   O3.but = (binarization)
   O3.contraintes=(p1=([O1.valeur-2, O1. valeur+2])
                   p2=255)
   O3.mode = optimisation
   O3.evaluation = (nbpts (and (1ère de O2.sorties)
                               (1ère de O3.sorties))
   O3.prototype = ((p1 p2)(data → data))
   O3.entrées = (1ère de P.entrées)
  P.decomposition = (THEN (AND O1 O2) O3)
  P.sorties = (1ère de O3.sorties)
```

Figure 10. Le code de la SC d'instanciation Contrast-Binarization.

7 Conclusion

Nous venons de montrer que la modélisation des connaissances de Traitement d'Images se limitant à des Sources

de Connaissances permet d'apporter des réponses intéressantes pour les deux objectifs initiaux d'opérationnalisation et d'acquisition des connaissances.

Par contre, le système Borg résultant reste encore destiné aux experts du Traitement d'Images (ou à un système d'interprétation) et non aux novices, parce que la formulation d'un problème exige de savoir choisir les bonnes tâches et la bonne description du contexte de l'application. Mais plus que la résolution automatique d'applications, nos motivations profondes au travers de ce système sont de définir un cadre avec lequel nous pouvons définir une ontologie et une formalisation des connaissances de Traitement d'Image pouvant conduire à un système d'apprentissage. Les connaissances expriment ici quand et comment utiliser les méthodes et les techniques de Traitement d'images, à la différence de celles que l'on retrouve dans la littérature, où seules les techniques sont véritablement présentées, et les connaissances expriment essentiellement leurs effets sur les images. Le lecteur doit acquérir seul l'expérience pour savoir quand les utiliser.

L'acquisition des connaisssances ne peut réellement se faire que progressivement autour de la résolution d'applications réelles. Pour cela, notre travail s'inscrit dans le cadre du pôle « Traitement et Analyse d'Images » de Caen. Ce pôle couvre un large domaine d'applications: le biomédical, les sciences des matériaux, la géographie, etc. Il est un champ d'investigation privilégié pour étudier et affiner les concepts proposés et faire coopérer différents types d'experts.

Bibliographie

[1] F. Aubry, V. Chameroy, F. Lavaire, J.P. Ramond, I.E. Saidane, A. Giron, Y. Bizais, A.Todd-Pokropek & R. di Paola, Medical image management using a semantical description approach: image description, *In Computer Assisted Radiology and Surgery,* pp 264-271, Berlin, 1993.

[2] R. Bodington, A Software Environment for the automatic configuration of inspection systems, *In Proc. of Knowledge-Based Systems for reUsing Programs*, pp 100-108, Sophia Antipolis, 1995.

[3] D. Charlebois, D. G. Goodenough & S.Matwin, Case-Based Reasoning in an Intelligent Information System for Forestry, *In Proc SPIE : Intelligent Robots and Computer Vision,* vol 2244, pp 64-74, Orlando, 1994.

[4] S. A. Chien & H. B. Mortensen, Automating Image Processing for scientific Data Analysis of a Large Image Database, *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 18, no. 8, pp 854-859, 1996.

[5] V. Clément & M. Thonnat, A Knowledge-Based Approach to Integration of Image Processing Procedures, *CVGIP: Image Understanding,* vol. 57, no 2, pp 164-184, 1993.

[6] R. Clouard, C. Porquet, A. Elmoataz & M. Revenu, Why building Knowledge-Based Image Segmentation is so difficult, *In Proc. of Knowledge-Based Systems for reUsing Programs*, pp 137-148, Sophia-Antipolis, 1995.

[7] R. Clouard, A. Elmoataz & F.Angot, *Pandore: Une bibliothèque et un environnement de programmation d'opérateurs de traitement d'image,* Rapport interne du Greyc, Caen, 1997.

[8] B. Charroux, *Analyse d'images: coopération d'opérateurs de segmentation guidée par l'interprétation*, thèse de l'Université Paris XI, 1996.

[9] B. Deen, A graphical user interface for science data processing, *Nasa Science Information Newsletter*, vol. 2, issue 39, pp 22-27, 1996.

[10] P. Dejean, *Un formalisme pour les entités du traitement et de l'analyse des images*, Thèse de l'Université Paul Sabatier de Toulouse, 1996.

[11] A. Elmoataz, M. Revenu & C. Porquet, Segmentation and Classification of Various Types of Cells in Cytological Images, *In Proc. IEE Image Processing and its Applications,* pp 385-388, Maastricht, 1992.

[12] Elmoataz A. *Mécanismes opératoires d'un segmenteur d'images non dédié: définition d'une base d'opérateurs et implantation.* Thèse de l'Université de Caen, 1990.

[13] V. Ficet-Cauchard, Construction interactive d'un modèle conceptuel d'applications de traitement d'images, *In Proc. Rencontre des Jeunes Chercheurs en IA*, pp 95-102, Nantes, 1997.

[14] B. Hayes-Roth, A blackboard architecture for control, *Artificial Intelligence*, vol. 26, pp 251-321, 1985.

[15] C-E. Liedtke & A. Blömer, Architecture of the knowledge based configuration system for image analysis "Conny", *Proc. 11th Inter. Conference on Pattern Recognition*, pp 375-378, The Hague, 1992.

[16] Rasure J. & S. Kubica, The Khoros Application Development Environment, Experimental Environments for Computer Vision and Image Processing. World Scientific, pp 1-32, Singapore, 1992.

[17] T. A. Poggio, C. Koch, & V. Torre. Computational vision and regularisation theory. *Nature*. vol. 317. pp 314-319. 1985.

[18] J. Van dern Elst, *Modélisation des connaissances pour le pilotage de programmes de Traitement d'Images*, Thèse de l'Université de Nice, 1996.

[19] J-R. Vigouroux & D. Mischler, ALLEGORY. Un environnement de programmation pour le Traitement d'Images. *Revue Techn. Thomson-CSF*, vol 24, no. 4, pp 849-865, 1992.