Construction interactive d'un modèle conceptuel d'applications de traitement d'images

Valérie Ficet-Cauchard

Construction interactive d'un modèle conceptuel d'applications de traitement d'images

Valérie FICET

GREYC-ISMRA - 6 bd du Maréchal Juin - F14050 Caen cedex

tél: 31-45-27-21

fax: 31-45-26-98

e-mail: Valerie.Ficet@greyc.ismra.fr

Résumé:

Le Traitement d'Images est un domaine pour lequel les systèmes à base de connaissances peuvent servir de plate-forme expérimentale. Nous proposons un Atelier Logiciel permettant d'acquérir et d'intégrer des connaissances de Traitement d'Images au fur et à mesure de leur mise en évidence par la réalisation d'applications. Cette intégration est organisée dans une architecture réflexive fondée sur les concepts de TÂCHE, MÉTHODE et OUTIL, qui permet de définir un modèle conceptuel opérationnalisable tout en prenant en compte le besoin de coopération expert(s) / système. Notre Atelier s'appuie sur l'existence d'une bibliothèque d'opérateurs de Traitement d'Images qu'il s'agit de sélectionner, d'enchaîner, de paramétrer et de piloter pour résoudre une application particulière en construisant un plan de traitement d'images adapté. Les divers plans de traitement d'images de cytologie et d'histologie déjà construits au sein de cet Atelier Logiciel constituent une première validation de notre approche.

Mots-clés:

- traitement d'images
- pilotage de bibliothèques d'opérateurs
- modèle conceptuel opérationnel

1 - Introduction

Le Traitement d'Images est un domaine qui nécessite la coopération d'experts d'origines diverses: expert du domaine d'application (médecine, cartographie, géologie, ...), expert en Traitement d'Images et concepteur-programmeur. Les systèmes à base de connaissances (SBC) permettent de répondre à ce besoin de coopération entre experts de domaines différents en facilitant la communication et en permettant une acquisition progressive des connaissances. C'est dans l'optique de créer un SBC appliqué au traitement et à l'interprétation d'images qu'a été mis en place le projet "d'Atelier Logiciel d'Intégration de Connaissances en Traitement et Interprétation d'Images" [Revenu 95]. Plutôt que de viser à la création d'un système général de traitement d'images, cet atelier logiciel doit permettre d'acquérir et d'intégrer des connaissances de Traitement d'Images au fur et à mesure de leur mise en évidence par la réalisation d'application. La motivation est non seulement de résoudre des applications réelles, mais aussi de formaliser la connaissance des experts. Les premières expérimentations se font sur des images d'origine biomédicale (histologie et cytologie).

Le travail présenté ici s'inscrit dans le cadre de cet atelier avec comme but la mise au point d'un environnement permettant la construction de plans de Traitement d'Images à l'aide de l'architecture réflexive "TÂCHE-MÉTHODE-OUTIL" (TMO). Après avoir présenté les objectifs de notre atelier (§. 2), nous détaillons les divers types de connaissances qu'il faut pouvoir manipuler et représenter explicitement (§. 3) puis nous présentons l'architecture TMO (§. 4) et son implantation en CLOS. Le dernier paragraphe est consacré à la validation de notre approche et aux directions de recherche envisagées.

2 - Position du problème

L'objectif de l'Équipe Image du GREYC est de traiter informatiquement des images complexes, principalement d'origine biomédicale, dans le but d'aider à la compréhension des phénomènes physiques visualisés au travers de ces images. Il s'agit, par exemple, de fournir des mesures destinées à évaluer l'évolution d'une tumeur à partir d'images microscopiques de cellules. Dans ce cadre, le but de "l'Atelier d'Intégration de Connaissances en Traitement et Interprétation d'Images" est la création d'un environnement informatique qui facilite la construction de programmes de Traitement d'Images. Cet environnement doit permettre la résolution d'une application par sélection, paramétrisation puis enchaînement de codes informatiques exécutables préexistants. Ces codes sont répertoriés dans une bibliothèque et on peut y accéder, soit directement

(commandes UNIX), soit par l'intermédiaire d'un langage informatique.

2.1 - Spécifications

Trois types d'expert entrent en jeu dans la conception de systèmes et la résolution de problèmes de Traitement d'Images : l'expert du domaine d'origine de l'image, l'expert en Traitement d'Images et le concepteur-programmeur. Nous commençons par donner notre définition des rôles de chacun d'entre eux.

L'expert du domaine définit en premier lieu son problème. Pour cela, il fournit une ou plusieurs images (l'objet à traiter), une requête (les objectifs à atteindre) et un contexte d'application qui contient les conditions de traitement de l'objet et qui limite les techniques applicables. Lorsqu'un plan est créé pour résoudre ce problème et qu'il souhaite le tester, le système doit alors lui permettre d'exécuter ce plan en lui proposant des choix à différents niveaux de l'exécution et des retours sur ces choix (par ex. des choix sur les méthodes ou sur les paramètres).

Le rôle de l'expert en Traitement d'Images est de construire un plan pour résoudre le problème. Avec le système que nous proposons, il n'a plus à faire de programmation au sens classique, ni même à connaître le détail des codes informatiques : il crée des plans de traitement en utilisant des blocs prédéfinis. Chaque plan a besoin d'être évalué visuellement par l'expert du domaine. En effet, le Traitement d'Images est un domaine où il n'existe pas de fonctions d'évaluation idéales, et le plan est construit pas à pas suivant une technique "générer - tester".

Le premier rôle du concepteur-programmeur est d'implanter l'architecture de contrôle et les blocs de base lors de la création du système. Par la suite, si le besoin s'en fait sentir, l'expert en Traitement d'Images peut faire appel au concepteur-programmeur pour l'implantation de nouveaux opérateurs.

Les rôles décrits ci-dessus ne sont pas indépendants, ce qui se traduit par des besoins de coopération et de communication entre experts. Le système est ici à la base de la communication. Il offre un cadre pour représenter les connaissances de chacun et permettre le dialogue.

La communication passe d'abord par l'explicitation des opérations : lors de la construction d'une application, une phase descriptive est proposée pour expliciter la décomposition en sous-problèmes (par ex. nature de celle-ci, adéquation à un type d'images, ...). Pour favoriser le dialogue entre les trois types d'experts et le système et pour s'abstraire le plus possible des détails techniques du Traitement d'Images (qui ne peuvent que compliquer le dialogue) nous proposons de programmer au *niveau connaissance* [Newell 82], c'est à dire, de décrire la méthode de résolution en utilisant des termes pertinents pour les experts humains, le concepteur-programmeur et le futur utilisateur. Le modèle conceptuel du système correspondant à cette description leur permet alors d'intervenir dans l'évaluation et la mise au point de cette méthode de résolution [Delouis 93]. Il doit donc être opérationnalisable et compréhensible par les trois types d'experts et s'appuyer sur une description formelle des connaissances basée sur la description symbolique des concepts et des propriétés. Un bon moyen de faciliter cettte communication passe par la mise en place d'une interface graphique qui permet de visualiser les plans de traitement d'images sous forme d'arbres [Willamowski 94] et d'accéder aux images intermédiaires.

Nous allons décrire maintenant comment nous voyons les interactions entre le système et les utilisateurs lors de la construction d'une application, d'une part et lors de son exécution, d'autre part.

Dans le cadre de notre atelier, le premier travail a porté sur la mise au point du générateur automatique de plans BORG. Les difficultés rencontrées pour l'écriture des sources de connaissances nous ont conduits vers une démarche plus pragmatique de construction manuelle des plans de Traitement d'Images. Néanmoins, un des objectifs est de faire coopérer notre système et le générateur automatique de plans. La construction d'un plan pourra se faire soit manuellement par l'expert en Traitement d'Images, soit automatiquement par l'intermédiaire du générateur de plans BORG [Clouard 95a]. Le modèle conceptuel que nous avons choisi nous apporte un formalisme de représentation compatible avec celui de BORG. Ainsi, dans le cas d'une construction automatique, on pourra ensuite améliorer les résultats du plan obtenu en le modifiant manuellement.

Le traitement d'une image est spécifique au domaine d'application : on n'interprète pas de la même façon une image d'origine médicale et une image de cartographie. C'est dans le souci de pouvoir s'appliquer à divers domaines que le système peut faire appel aux experts du domaine. Le système a la possibilité de demander l'intervention de l'utilisateur (expert du domaine ou expert en traitement d'images) lorsqu'il lui manque des données ou pour faire des choix parmi plusieurs techniques. Mais les utilisateurs peuvent également intervenir s'ils veulent des explications ou s'ils veulent modifier les choix faits par le système. Pour cela, les utilisateurs sont directement impliqués dans la boucle de résolution (s'ils le désirent et s'ils ont les connaissances nécessaires). Le système a donc deux modes de fonctionnement dans sa phase d'exécution d'un plan : un mode *automatique* (c'est le système qui fait les choix) et un mode *manuel* (c'est l'utilisateur qui fait les choix).

2.2 - Comparaison avec d'autres approches

Contrairement à des systèmes tels que DIA-ES [Sakaue 85], OCAPI [Clément 93] ou VSDE [Bodington 95] dont l'objectif est la résolution de problèmes, dans notre système, la création manuelle de nouvelles applications n'impose pas l'écriture de règles de production. L'écriture de ces règles pourra se faire par la suite ou ne pas se faire du tout. En phase d'exécution d'un plan, s'il y a conflit entre plusieurs méthodes et absence de règles, un mécanisme de choix par défaut s'appliquera.

Ainsi un expert en Traitement d'Images peut lui-même créer ou modifier ses applications sans que cela ait de répercussion sur les applications existantes.

D'autre part, la possibilité de visualiser des images intermédiaires comme dans VSDE ou dans CONNY [Lidke 92] en fait un outil d'expérimentation pour l'expert en Traitement d'Images et pour l'expert du domaine. Mais contrairement à la façon dont il procède dans ces systèmes, ici, l'expert n'a pas besoin de définir son problème en début d'application : il doit le reconnaître dans une liste prédéfinie et ne donner les critères de choix que lorsque le système en a besoin.

3 - Système à Base de Connaissances pour le Traitement d'Images

Les systèmes à base de connaissances (SBC) permettent la gestion d'informations "abstraites" telles que la description des étapes de résolution d'un problème [Wielinga 92]. Ils assurent la mise en oeuvre des connaissances stratégiques tout en autorisant l'utilisation d'outils informatiques traditionnels et en proposant une forme naturelle d'interaction avec l'utilisateur. En cela, ils répondent à nos besoins en ce qui concerne la coopération des différents experts, l'explicitation des traitements et l'enchaînement des opérateurs de Traitement d'Images. Pour la construction de notre SBC, le choix d'un modèle conceptuel demande une étude approfondie des connaissances mises en jeu.

3.1 - Les Connaissances pour résoudre un problème de Traitement d'Images

Notre système possède trois niveaux de connaissances : le niveau des connaissances du domaine (le Traitement d'Images) et les niveaux des connaissances de contrôle et de métacontrôle pour gérer la modélisation et l'utilisation des connaissances du domaine.

D'après [Clouard 95b], au niveau des connaissances du domaine, on distingue trois types de connaissances :

- des connaissances sur l'application : elles permettent de donner un sens et un sujet à l'image. Il s'agit de connaissances acquises en début d'application ou en cours de résolution s'il y a ambiguïté ou incomplétude,
- des connaissances en Traitement d'Images: elles permettent la détermination des stratégies à utiliser pour accomplir les tâches et évaluer les résultats,
- des connaissances sur les opérateurs: elles permettent leur sélection, la détermination des valeurs de paramètres et la réalisation d'enchaînements syntaxiquement corrects.

Les connaissances de contrôle sont les connaissances qui facilitent l'exploitation des connaissances du domaine. Ce sont elles qui gèrent la résolution des problèmes liés à l'application et leur explicitation. On peut séparer ces connaissances en deux catégories:

- le contrôle du domaine: gestion des plans de Traitement d'Images (leur création, leur modification, leur exécution, ...),
- le contrôle du système: affichage des opérations à effectuer, sélection des traitements accessibles à l'utilisateur, contrôle des enchaînements, ...

Les connaissances de métacontrôle sont les connaissances sur le contrôle du contrôle. Elles définissent les règles de comportement du système, vis à vis de l'utilisateur. En effet, le contrôle sera différent suivant le type de l'utilisateur (on ne propose pas les mêmes activités à un expert et à un novice) et suivant ses désirs (travail en mode automatique ou manuel, explicitation de la méthodologie ou non, ...).

3.2 - Intégration des connaissances

Parmi les approches retenues en Traitement d'Images, nous avons choisi de créer un environnement interactif de travail favorisant le développement incrémental d'applications : cet environnement est le lieu d'acquisition de nouvelles connaissances et de leur opérationnalisation. Cette acquisition de nouvelles connaissances se fait suivant deux axes [Clouard 94] : *intégration procédurale* pour réaliser les traitements et *intégration sémantique* pour les expliciter. D'autre part, on veut recueillir des connaissances du niveau domaine et du niveau contrôle (et peut être du métacontrôle) de façon indépendante mais similaire. Il faut également prendre en compte le fait qu'une requête en Traitement d'Images n'a pas de solution unique : il existe en général plusieurs techniques pour la satisfaire. C'est dans cette multiplicité de solutions que l'intégration de connaissances ontologiques et structurelles prend toute son importance. En effet, ces connaissances vont permettre à l'utilisateur de savoir quand utiliser une technique plutôt qu'une autre.

Dans son étude sur les modèles conceptuels, [Delouis 93] insiste sur l'intérêt :

- de modéliser une expertise selon trois perspectives (les problèmes à résoudre, les méthodes possibles de résolution et les connaissances du domaine),
- d'autoriser la définition de plusieurs méthodes pour résoudre un même problème,
- de pouvoir choisir entre ces méthodes de manière dynamique voire opportuniste,
- de représenter les raisonnements de contrôle et ceux des différents domaines de façon uniforme.

Notre étude des problèmes posés par la conception d'applications de traitement d'images, notre volonté de faire coopérer les différents types d'experts et la prise en compte des quatre points énoncés ci-dessus nous ont amenés à choisir une architecture répondant à ces quatre critères, l'architecture "TÂCHE-MÉTHODE-OUTIL".

4 - Modèle Conceptuel : l'architecture "TÂCHE-MÉTHODE-OUTIL"

Comme son nom l'indique, l'architecture "TÂCHE-MÉTHODE-OUTIL" est basée sur trois notions que nous allons maintenant détailler.

4.1 - Notion de Tâche

Une tâche est la représentation d'un but ou d'un sous-but dans le système. Elle décrit un but à atteindre et rassemble les éléments nécessaires à l'atteinte de ce but : les données à traiter, les types de résultat à produire et les méthodes de résolution connues. Quand une tâche décrit un problème général, elle se décompose en sous-tâches qui décrivent des problèmes plus élémentaires ; ces sous-tâches pouvant être décomposées à leur tour. Une tâche peut être résolue de plusieurs manières, on lui associe donc une ou plusieurs méthodes.

Les tâches servent à représenter tous les buts du système, quelque soit leur niveau :

- niveau *"domaine"*: les tâches représentent un objectif de Traitement d'Images ou une sous-partie de celui-ci. Par exemple, "extraire les regroupements d'objets" est une tâche principale du domaine et "éliminer le fond" en est une sous-tâche (cf. §.6.1).

- niveau *"contrôle domaine"* : les tâches représentent les opérations que l'on veut effectuer sur les plans de traitement d'images. "Exécuter un plan" et "sauvegarder un plan" sont des tâches de ce niveau, "instancier un plan" est une sous-tâche de "exécuter un plan". Ces tâches peuvent être exécutées par le système ou par l'utilisateur.

- niveau *"contrôle système"* : les tâches représentent les opérations de contrôle du bon déroulement des opérations. "Initialiser le système" et "gérer l'interface du système" sont des tâches du "*contrôle système*". Ces tâches ne peuvent être exécutées que par le système.

- niveau *"métacontrôle"* : les tâches représentent les opérations de contrôle sur le contrôle. "Exécuter une tâche" et "choisir une méthode" sont des tâches du métacontrôle. On parlera alors de métatâches.

L'uniformité de représentation des tâches des différents niveaux nous permet de définir l'ensemble {métacontrôle, {contrôle, domaine}} comme un système réflexif.

4.2 - Notion de Méthode

Une méthode spécifie comment une tâche peut être accomplie. Chaque méthode est associée à une seule tâche, mais une tâche peut être associée à plusieurs méthodes (fig. 1-a). Pour cela, on cherche à expliciter, pour chaque méthode, quand il est possible et réellement souhaitable de l'utiliser. Ainsi dans certains contextes, une méthode peut être "déclenchable" car on possède tous les éléments nécessaires pour cela, mais pas vraiment pertinente car pas assez bien adaptée au problème. Le choix de la méthode à utiliser peut être fait, soit par l'utilisateur (mode manuel), soit par le système (mode automatique).

Le corps de la méthode peut prendre deux formes :

- une décomposition en sous-tâches qui prend alors l'aspect d'un arbre "ET - PUIS" (fig. 1 b). Si T1 se décompose en (PUIS (ET T11 T12) T13) par la méthode M1, on exécutera d'abord T11 et T12 (dans l'ordre que l'on veut), puis on exécutera T13. On parle alors de méthode complexe.

- l'appel à un code informatique par l'intermédiaire d'un outil (fig. 1-c). Il s'agit alors de méthode terminale.

Ce sont donc les méthodes qui vont gérer les flux de données entre tâche et sous-tâche et entre tâche et outil.

4.3 - Notion d'Outil

Un outil est la réification d'un code informatique, c'est-à-dire sa représentation pour l'utilisateur en termes conceptuels (but, entrées, paramètres, sorties, syntaxe d'appel, performances, ressources utilisées, ...) avec un lien sur le code permettant sa mise en œuvre. Le code informatique qui lui est associé est vu par l'utilisateur comme une boîte noire dont il connaît seulement la transformation qu'elle effectue sur les entrées pour produire les sorties. L'outil est là pour expliciter les opérations effectuées par le code qui lui est associé, il doit connaître les types des entrées / sorties et la signification exacte de l'opération que le code effectue sur les images.

Au niveau "*domaine*", nos outils représentent essentiellement des opérateurs de base du Traitement d'Images définis dans la bibliothèque d'opérateurs HORUS [Clouard 94] (ex : binariser une image, appliquer un filtre moyenneur uniforme,...), mais ils peuvent également réifier des fonctions quelconques Lisp ou C.

4.4 - Résolution d'un problème.

Lorsqu'il veut exécuter un traitement sur une image, l'utilisateur doit d'abord choisir la tâche correspondante dans un menu. Puis il doit paramétrer le plan associé à cette tâche en fournissant les données (image originale) et les paramètres spécifiques à cette image. Ensuite le système exécute la tâche en proposant des choix à l'utilisateur parmi les méthodes existantes (mode manuel) ou en faisant lui même ces choix en fonctions du contexte (mode automatique). Si les résultats sont insatisfaisants (ou même inexistants), le système propose (ou fait) des retours sur ces choix.

5 - Implantation de l'architecture TMO

L'implantation de l'architecture "TÂCHE-MÉTHODE-OUTIL" a été faite en CLOS [Steel 90]. Avant de décrire en détail les classes TÂCHE, MÉTHODE et OUTIL qui nous ont permis de mettre en place cette architecture (§ 5.1) et d'introduire la notion de DONNÉE (§ 5.2), nous avons été amenés à définir les classes suivantes, pour assurer la gestion globale du système et les interactions avec l'utilisateur :

- AGENDA : pour la gestion des tâches en cours,

- UTILISATEUR : pour prendre en compte les différents types d'utilisateur et leur niveau de compétence et éviter ainsi qu'un utilisateur modifie ou utilise des éléments qui sont en dehors de ses compétences,

- CONTEXTE : pour définir les divers contextes d'utilisation du système (*utilisateur courant, mode d'exécution courant*),

- MENU : pour définir les différents menus et sous-menus que l'on peut présenter à l'utilisateur, en fonction de son niveau de compétence.

5.1 - Les classes TÂCHES, MÉTHODES et OUTILS

Les principaux concepts dont nous avons besoin pour définir notre architecture sont les TÂCHES, les MÉTHODES et les OUTILS. Chacun de ces objets est repéré par un *but* qui doit être assez explicite pour permettre la communication avec l'utilisateur. L'utilisateur a également accès, s'il le désire, à une *description* détaillée de la requête associée à une TÂCHE ou de la stratégie représentée par une MÉTHODE. Les TÂCHES et les OUTILS représentent des opérations à effectuer sur des données: ils possèdent donc chacun une liste d'*entrées,* une liste de *paramètres* et une liste de *sorties*.

Un OUTIL connaît tous les éléments nécessaires à son exécution: le *type* de la fonction à laquelle il fait appel (Lisp, C, opérateur Horus), la *syntaxe* suivant laquelle il appelle cette fonction, le *mode* suivant lequel il doit exécuter la fonction. Trois modes sont disponibles: un mode *"normal"* (exécution simple de la fonction), un mode *"pour"* (exécution de la fonction un nombre prédéfini de fois) et un mode *"optimisation"* qui offre la possibilité de calculer dynamiquement les valeurs des paramètres sur la base de mesures faites directement sur les images. Ce dernier mode reprend les schémas de boucles classiques (le-premier, le-meilleur, tant-que, jusqu-à) et utilise une fonction d'évaluation pour vérifier l'adéquation du résultat aux critères demandés. Notre approche s'inspire ici de [Matsuyama 89] et [Clément 93]. Cette représentation des boucles de contrôle et le fait que nous disposions du code informatique nous donne la possibilité de générer un programme C++ équivalent au plan.

5.2 - Gestion des flux de données

Lors d'une exécution, pour récupérer les données fournies en sorties par les TÂCHES et les OUTILS, on a besoin de savoir quelles décompositions en sous-tâches et quels OUTILS ont été choisis: pour cela, il est nécessaire d'instaurer des liens dynamiques *père - fils* entre une TÂCHE et les SOUS-TÂCHES qui correspondent à la MÉTHODE choisie ou entre une TÂCHE et l'OUTIL auquel elle fait appel. Pour éviter des recherches et des calculs inutiles de données, il faut également savoir si la tâche qui les calcule a déjà été exécutée ou non. Nous avons mis en oeuvre un mode d'évaluation paresseuse des données s'appuyant sur la classe DONNÉE : chaque donnée d'un graphe TMO est une instance de cette classe et sait comment, en cas de besoin, calculer ou récupérer sa valeur. Par exemple, pour calculer sa valeur, la première sortie de la tâche *classifier les pixels* sait qu'elle doit aller chercher la valeur de dans la première sortie de la tâche *binarisation basée sur la variance* ou dans la première sortie de la tâche *binarisation basée sur le contraste*.

5-3 - Les tâches de contrôle et de métacontrôle

Les tâches du métacontrôle (ou métatâches) décrivent comment doit être réalisée une tâche quelconque. Elles définissent le protocole du choix des autres tâches et de leur mode d'exécution. La figure 2 donne le graphe TMO de la tâche principale de notre métacontrôle. Pour des raisons d'efficacité, cette tâche a été implantée de façon procédurale et correspond à la boucle :

```
tant qu'il y a des tâches dans l'agenda faire
   sélectionner une tâche dans l'agenda
   exécuter cette tâche
fin tant que.
```

Les tâches de contrôle sont séparées en deux groupes: les tâches de "*contrôle système*" qui gèrent l'enchaînement des opérations du système et les tâches de "*contrôle domaine*" qui représentent toutes les opérations exécutables sur les tâches du domaine. La tâche principale de "*contrôle système*" est chargée d'initialiser le système et de gérer l'interface en proposant des menus à l'utilisateur et en assurant l'enchaînement des différents menus en fonction des choix de l'utilisateur. Les tâches de "*contrôle domaine*" sont toutes les tâches qui permettent de créer, de modifier ou d'exécuter une tâche, une méthode, un outil. L'aspect réflexif de l'architecture permet la modification de ces tâches de contrôle au même titre que les tâches du domaine.

6 - Validation de l'approche et perspectives

Nous avons réalisé en CLOS une première maquette permettant la construction manuelle de plans et leur exécution. Les plans intégrés à l'aide de notre maquette ont été créés au laboratoire pour traiter des images biomédicales de cytologie et d'histologie fournies par le Centre anticancéreux F. Baclesse de Caen. Ces images sont de bons supports pour tester notre système car ils soulèvent une grande diversité de problèmes concrets.

6.1 - Traitement d'images de cytologie et d'histologie

Dans le domaine des images de cytologie, nous avons intégré un plan ayant pour but la détection des noyaux de cellules présents dans une image. A partir d'une carte de régions, on peut éliminer les noyaux qui ne répondent pas à certains critères de tailles, de formes ou de niveaux de gris pour fournir au biologiste une image qui ne comporte que les noyaux qu'il veut observer. Ce plan a aussi été généré automatiquement par le planificateur BORG [Clouard 95a], ce qui nous a permis de montrer la compatibilité entre les deux aspects de notre Atelier Logiciel concernant la construction et l'exécution manuelle ou automatique de plans de Traitement d'Images.

Dans le domaine d'images d'histologie, souvent beaucoup plus délicates à traiter, nous avons par exemple créé manuellement le plan donné figure 3 pour mettre en évidence les groupements significatifs de noyaux, l'étude de ces groupements permettant de déterminer la présence de lobules tumoraux. La formalisation de la méthode d'analyse de l'image à l'aide d'un graphe TMO a fait émerger de nouvelles idées pour améliorer les traitements, telles que le remplacement d'un module de calcul de distances sur graphe de voisinage par un autre plus performant, et a fourni une vision globale du processus.

6.2 - Conclusion et perspectives

Dans cet article, nous nous sommes attachés à décrire nos différents besoins en ce qui concerne les connaissances mises en jeu et la coopération entre experts de différents domaines, et nous avons montré en quoi l'architecture "TÂCHE-MÉTHODE-OUTIL" nous fournit une solution.

La facilité de codage et la rapidité d'intégration des plans décrits au § 6.1 dans notre maquette, renforcent notre conviction quant à la validité de notre architecture appliquée au traitement d'images. Cette architecture a par ailleurs été validée dans d'autres domaines tels que l'aide à la conduite de projet [Moire 94].

Tout en continuant notre processus d'acquisition de connaissances en intégrant de nouveaux plans de Traitement d'Images, nous réfléchissons à une meilleure représentation des flux de données pour faciliter la création des graphes TMO. Nous comptons nous appuyer pour cela sur la notion d'actigramme tel qu'elle est présentée dans [Matta 95].

Nos recherches passent également par une étude plus approfondie de la coopération dans les systèmes à base de connaissances, en nous appuyant sur les travaux de [Monclar 96] et [Hadj Kacem 96] sur la notion de rôle associé aux agents coopératifs.

La prise en compte de cet aspect coopératif doit passer par la réalisation d'une interface graphique permettant la prise en compte des contraintes liées au type de l'image et adaptées aux différents types d'utilisateurs répertoriés [Brutus 93].

Nous pourrons alors envisager l'intégration du générateur automatique de plans BORG au sein de notre architecture TMO pour permettre la coopération entre les modes automatique et manuel de résolution de problèmes de traitement d'images.

Bibliographie

[Brutus 93] Brutus P., "*Réflexion et construction progressive d'interfaces utilisateur*", Congrès Inforsid, Lille, France, mai 1993, p497-510.

[Clément 93] Clément V. & Thonnat M., "*A Knowledge-Based Approach to Integration of Image Processing Procedures*", CVGIP: Image Understanding, Vol. 57 (2), Academic Press, 1993, p164-184.

[Clouard 94] Clouard R. "*Raisonnement incrémental et opportuniste appliqué à la construction dynamique de plans de Traitement d'Images*", Thèse de doctorat, Université de Caen, France, 1994.

[Clouard 95a] Clouard R., Revenu M., Elmoataz A. & Porquet C., "*A software Workbench for Knowledge Acquisition and Integration in Image Processing*", International Workshop on the Design of Cooperative Systems, Juan-les-Pins, France, janvier1995, p298-313.

[Clouard 95b] Clouard R., Porquet C., Elmoataz A. & Revenu M., "*Why building knowledge-based image segmentation systems is so difficult?*", KBUP'95, Sophia Antipolis, France, novembre 1995, p137-148.

[Delouis 93] Delouis I., "*LISA, un langage réflexif pour la modélisation du contrôle dans les systèmes à base de connaissances, application à la planification des réseaux électriques*", Thèse de doctorat, Université de Paris-sud, Centre d'Orsay, France, 1993.

[Hadj Kacem 96] Hadj Kacem A., "*Les SBC coopératifs, vers une architecture fondée sur une formalisation du contrôle*", RFIA'96, Rennes, France, janvier 1996, p64-72.

[Lidke 92] Lidke C. E. & Blömer A., "*Architecture of the knowledge based configuration system for image analysis : CONNY",*", 11° ICPR, The Hague, Pays Bas, 1992, p375-378.

[Matsuyama 89] Matsuyama T., "*Expert systems for image processing: Knowledge-based composition of image processes*", CVGIP, Vol. 48, 1989, p22-49.

[Matta 95] Matta N., "*Méthodes de Résolution de problèmes : leur explicitation et leur représentation dans MACAO-II*", Thèse de doctorat, Université Paul Sabatier, Toulouse, France, 1995.

[Moire 94] Moire T., "*Etude d'une architecture logicielle basée sur un modèle opérationnel de la connaissance, application à la définition d'un système d'aide à la conduite de projet*", Thèse de doctorat, Université de Caen, France, 1994.

[Monclar 96] Monclar F.R., "*ELICO pour le développement de systèmes à base de connaissances coopératifs*", RFIA'96, Rennes, France, janvier 1996, p73-82.

[Newell 82] Newell A., "*The knowledge level*", Artificial Intelligence, n°18, 1982, p87-127.

[Revenu 95] Revenu M., Clouard R., Elmoataz A. & Porquet C., "*Un Atelier Logiciel d'Acquisition et d'Intégration de Connaissances en Traitement d'Images : Une Approche Méthodologique*", 10è Journées Acquisition, Validation et Apprentissage, Grenoble, France, avril 1995, p315-328.

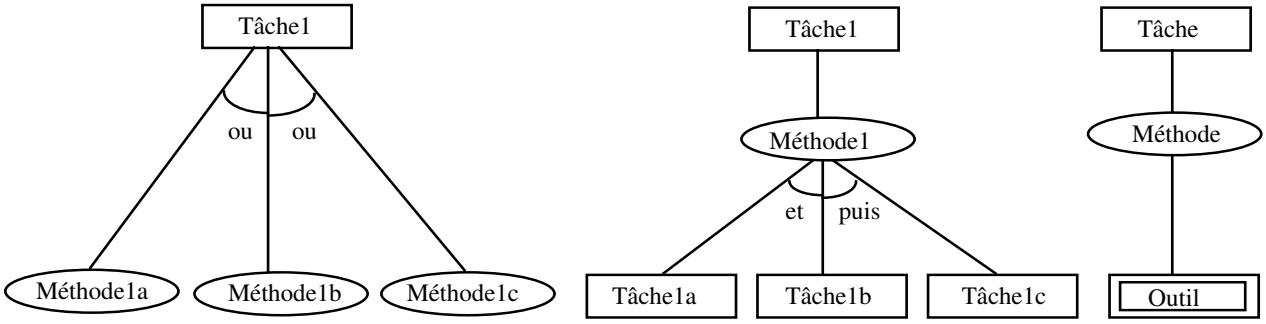
[Robington R] Robington R., "*A software environment for the automatic configuration of inspection systems*", KBUP'95, Sophia Antipolis, France, novembre 1995, p100-108.

[Steel 90] Steel G., "*Common Lisp the language*", Second Edition, Digital Press, 1990.

[Sakaue 85] Sakaue K. & Tamura H., "*Automatic generationof image processing programs by knowledge-based verification*", IEEE on Computer Vision and Pattern Recognition, San Francisco, CA, USA, 1985, p189-192.

[Wielinga 92] Wielinga B.J., Schreiber A.T. & Breuker J.A., "*KADS: a modelling approach to knowledge engineering*", Special Issue: The KADS approach to knowledge engineering, 4, 1992, p5-53.

[Willamowski 94] Willamowski. J., "*Modélisation de tâches pour la résolution de problèmes en coopération système-utilisateur*", RFIA'94, Paris, France, janvier 1994, p305-316.



(1-a) *(1-b)* *(1-c)*

Figure 1: Les différents enchaînements possibles de tâches, méthodes et outils
(1-a): Plusieurs méthodes sont associées à une tâche.
(1-b): Décomposition d'une tâche en sous-tâches.
(1-c): Méthode faisant appel à un outil.

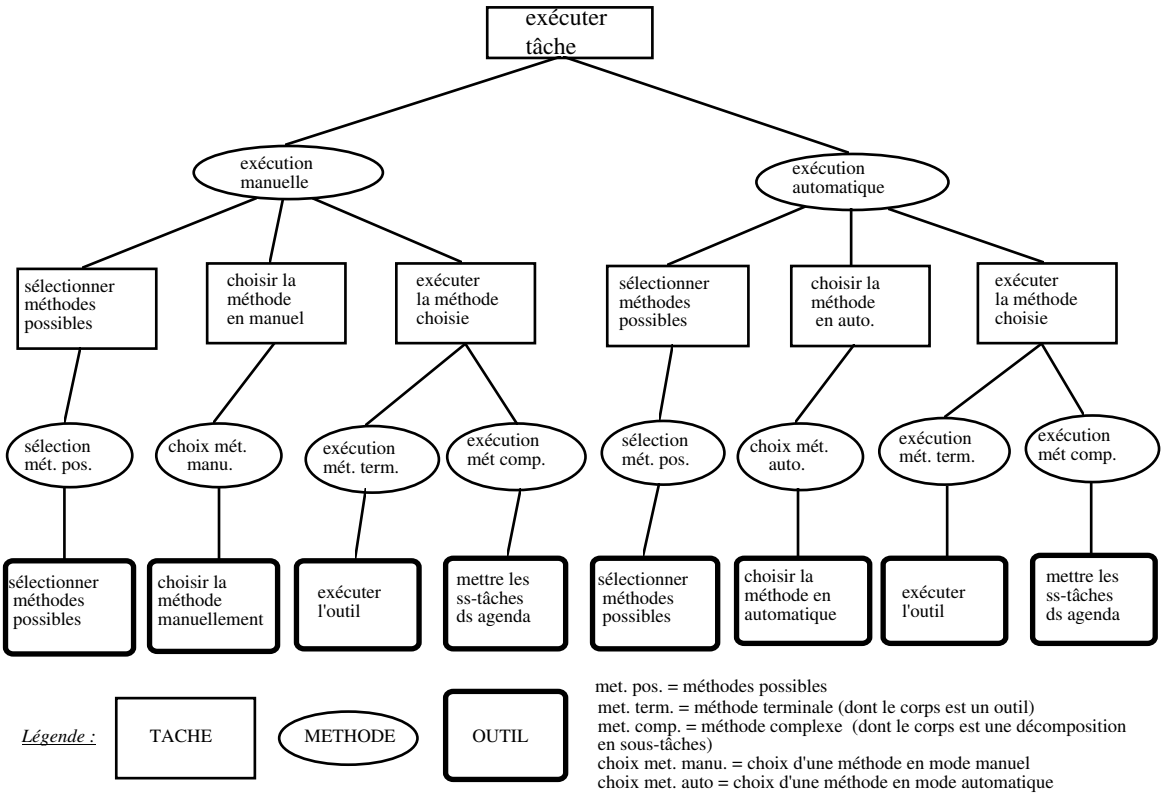


Figure 2: La métatâche "exécuter tâche".

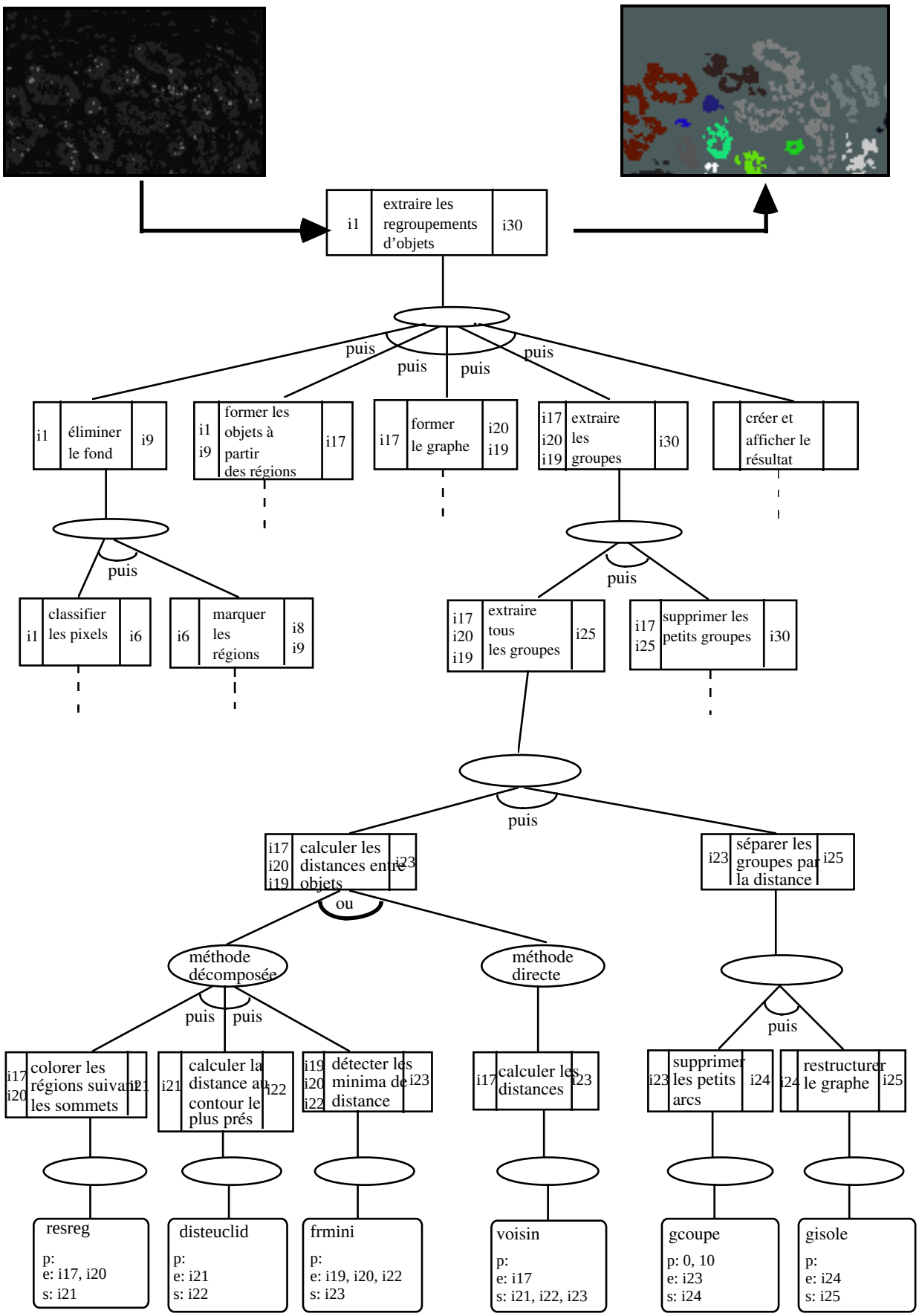


Figure 3: Plan pour la mise en évidence de regroupements significatifs de noyaux de cellules sur une image d'histologie (i1 à i30 représentent les images d'entrée et de sortie des tâches).