Aide à la conception d'applications de Traitement d'Images : une approche basée sur le Raisonnement à Partir de Cas

Valérie Ficet-Cauchard, Marinette Revenu, Christine Porquet

Aide à la conception d'applications de Traitement d'Images : une approche basée sur le Raisonnement à Partir de Cas

Providing assistance during the development of Image Processing applications : a Case-Based Reasoning approach

Valérie FICET-CAUCHARD, Marinette REVENU, Christine PORQUET

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

tél: 02-31-45-27-21 fax: 02-31-45-26-98 e-mail: Valerie.Ficet@greyc.ismra.fr

Résumé

Dans cet article, nous présentons un système interactif pour la construction d'applications de Traitement d'Images. Ce système doit constituer une aide pour l'expert en Traitement d'Images lors de l'élaboration d'une application comme lors de son exécution. Par l'intermédiaire d'une interface graphique, notre système propose à l'utilisateur de modéliser ses applications sous forme d'arbres de tâches et d'expliciter ses stratégies dans le but de les réutiliser. Il lui permet également de représenter et de tester plusieurs techniques pour résoudre une application. Pour accroître l'aide fournie par le système, il est nécessaire d'autoriser la réutilisation de parties de traitement déjà modélisées. Nous nous appuyons pour cela sur des techniques de raisonnement à partir de cas. Nous proposons un algorithme de recherche des cas et des critères qui permettent de classifier une tâche de Traitement d'Images pour savoir quand et comment la réutiliser.

Mots clefs

Raisonnement à partir de cas, Acquisition de connaissances, Traitement d'Images.

Abstract

In this paper, an interactive system for the development of Image Processing applications is described. This system is intended to provide some assistance to Image Processing experts during the working-out as well as the execution of their applications. Through the system's graphical interface, an application can gradually be built and represented as a tree of tasks, and strategies can be made explicit so as to become reusable in future applications. Developers can also represent and try out various techniques to solve their applications. In order to increase the assistance provided by the system, they must also be in a position to reuse parts of preexisting treatments. To implement reusability, Case-Based Reasoning methods are used. Here are proposed an algorithm for the retrieval of appropriate cases and a set of criteria, that enable to classify an Image Processing task in order to know when and how to reuse it.

Keywords

Case-based reasoning, Knowledge acquisition, Image processing

1 Introduction

La multiplicité des applications faisant appel au Traitement d'Images (TI), requiert des experts en TI de plus en plus de rapidité dans l'élaboration d'une solution. L'utilisation de bibliothèques de programmes (appelés opérateurs de TI) leur permet de construire de nouvelles applications sans avoir à tout reprogrammer. Cependant, une application peut nécessiter l'enchaînement de plusieurs dizaines d'opérateurs et le raisonnement associé à cet enchaînement n'est généralement pas explicite, donc pas mémorisé.

L'exploitation « intelligente » de bibliothèques de programmes n'est pas une préoccupation spécifique au domaine du TI. Parmi les systèmes développés à l'aide de l'environnement SCARP [20], beaucoup résolvent les problèmes en créant des enchaînements de programmes élémentaires répertoriés dans des bibliothèques (par ex. : le système Myosis développé pour le diagnostic électromyographique et le système Said mis au point pour le diagnostic vibratoire de plate-forme en haute mer).

Notre but est de construire un système d'aide pour l'expert en TI qui lui permette de réaliser des applications dans n'importe quel domaine (médecine, cartographie, géographie, ...) en réutilisant des parties de traitements (i.e. des enchaînements de plusieurs opérateurs) élaborés pour résoudre d'autres problèmes. Pour cela, il faut qu'il puisse modéliser son raisonnement de façon à savoir quand et comment une partie de traitement est réutilisable pour une autre application.

Dans cet article, nous proposons un système d'aide à la réalisation et à la modélisation d'application de TI, doté d'un module de réutilisation basé sur des techniques de Raisonnement à Partir de Cas (RàPC).

Après avoir exposé nos objectifs et justifié notre démarche (section 2), nous décrivons la partie opérationnelle de notre système d'un point de vue conceptuel et d'un point de vue fonctionnel (section 3). Puis nous détaillons la partie en cours d'implémentation, en donnant l'algorithme et les critères de sélection utilisés pour mettre en œuvre les techniques de RàPC permettant la réutilisation (section 4).

2 Position du problème

Le type d'approche que nous avons choisi pour la mise au point d'applications de TI s'appuie sur des bibliothèques de programmes, appelés opérateurs de TI. Suivant cette approche, construire une application de TI consiste à enchaîner et à paramétrer un ensemble d'opérateurs d'une bibliothèque donnée.

Une première famille de systèmes facilitant l'utilisation de telles bibliothèques regroupe les environnements de programmation graphique. Ces systèmes [11] [13] proposent à l'utilisateur de sélectionner et d'enchaîner des opérateurs mais ils ne lui permettent pas d'expliquer ni de modéliser son raisonnement. Il est donc difficile de réutiliser les éléments de solution construits antérieurement pour d'autres applications.

Une deuxième famille de systèmes de TI utilisant des bibliothèques de programmes est celle des planificateurs automatiques tel que OCAPI [7] ou BORG [8], le générateur de plans mis au point dans notre laboratoire. Ces systèmes possèdent une représentation des connaissances qui autorise leur réutilisation. Cependant les connaissances qui y sont explicitées et représentées le sont plus dans un souci d'opérationnalisation que pour la compréhension de l'utilisateur, ce dernier intervient donc peu dans la résolution.

Dans le but de bénéficier des avantages dus à l'interactivité des environnements de programmation graphique, notre système doit permettre à l'utilisateur de sélectionner et d'enchaîner des opérateurs de TI. Mais il doit également lui donner la possibilité de modéliser et d'expliquer le raisonnement qui l'a amené à cet enchaînement, dans le but de réutiliser la stratégie mise en œuvre, comme le font les systèmes automatiques.

Dans la suite de cette section, nous commençons par présenter les objectifs que nous nous fixons, puis la modélisation que nous proposons pour faciliter la coopération entre les intervenants, enfin nous décrivons les aspects du RàPC qui nous intéressent pour la réutilisation du savoir-faire.

2.1 Des objectifs à deux niveaux

Pour atteindre nos objectifs nous avons procédé en deux étapes, la validation de la première étant nécessaire à la réalisation de la deuxième. La première étape a consisté en l'élaboration d'un système interactif (TMO) permettant à l'utilisateur de construire et d'exécuter une application de TI tout en modélisant le raisonnement mis en jeu dans cette application. Cette première étape correspond à une phase d'acquisition de connaissances. La deuxième étape vise à doter ce système d'un module de RàPC avec lequel l'utilisateur pourra construire ses applications en réutilisant les applications précédemment modélisées. Cette deuxième étape correspond à une phase de réutilisation des connaissances.

Objectifs de TMO

Avec TMO, nous voulons fournir une aide à l'expert en TI lors de la construction et lors de l'exécution d'une application et lui proposer un cadre structurant pour dialoguer avec l'expert du domaine des images. Du fait qu'il existe en général plusieurs techniques pour traiter une image, nous voulons également offrir à l'utilisateur la possibilité de modéliser ces différentes techniques lors de l'élaboration d'une application et de choisir entre ces techniques lors de son exécution.

C'est pourquoi notre système TMO possède les trois fonctionnalités suivantes : construction d'application par assemblage de briques de base, exécution d'application avec intervention de l'utilisateur pour choisir entre plusieurs techniques ou pour ajuster des valeurs de paramètres et stockage d'applications pour pouvoir les rejouer. Les briques de base utilisées pour la construction des applications sont des opérateurs de TI de la bibliothèque PANDORE [9] développée au GREYC. Un opérateur de notre bibliothèque est un programme informatique qui correspond à une opération non décomposable : une opération complexe doit être réalisée par l'enchaînement de différents opérateurs. Chaque opérateur prend en entrées des images et des paramètres et fournit en sortie des images et/ou un résultat numérique. Il est capable d'exécuter le traitement souhaité sur différents types d'images (images de pixels, images d'étiquettes, carte de régions, …). Les paramètres sont des valeurs numériques que l'on peut faire varier pour moduler le comportement de l'opérateur.

Dans TMO, les applications sont modélisées sous forme de plans. Un plan peut être schématisé sous forme d'arbre (fig. 1) et représente non seulement l'enchaînement de l'exécution d'opérateurs de TI correspondant aux feuilles de l'arbre, mais également le raisonnement nécessaire à la création de cet enchaînement, correspondant à des tâches de TI schématisées par les rectangles gris.

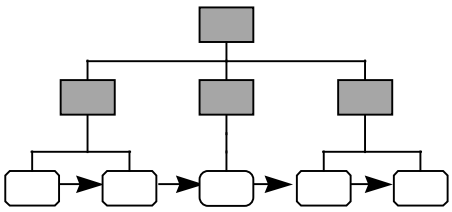

Fig. 1 : représentation d'un plan

Intérêt de la réutilisation

Le traitement d'une image peut nécessiter l'enchaînement d'une cinquantaine d'opérateurs de la bibliothèque qui en comporte près de 150. Pour chacun des opérateurs se pose ensuite le problème du réglage des paramètres. La fabrication manuelle d'un plan conduit donc à une combinatoire énorme à laquelle seul le savoir-faire de l'expert en TI permet de faire face. Les figures 5 à 7 montrent quelques unes des images traitées à l'aide de notre système. La figure 5 présente une image de cytologie dans laquelle on cherche à extraire un type

particulier de cellules. La figure 6 correspond à la résolution d'un problème de tri de pièces industrielles en vrac sur un tapis roulant. Et la figure 7 montre l'extraction de regroupements de noyaux de cellules dans une image d'histologie. Ces applications peuvent être classées suivant plusieurs points de vue : d'une part les figures 5 et 6 correspondent à un même problème, à savoir, « extraire des objets de taille relativement grande, de convexité assez élevée et d'un niveau de gris éloigné de celui du fond de l'image », et d'autre part, les figures 5 et 7 présentent des images issues du milieu biomédical et acquises par le même type de capteur. La comparaison des plans correspondant à ces applications met en évidence des parties communes : dans les cas des figures 5 et 6, on a utilisé la même stratégie (éliminer le bruit, éliminer le fond, extraire les objets, ...) et dans le cas des figures 5 et 7, on a utilisé le même prétraitement. Ces observations nous incitent à mettre en oeuvre une gestion des plans qui permette de créer un nouveau plan en réutilisant des parties de plans définies pour d'autres applications.

Pour mettre en œuvre ce type de réutilisation, nous avons choisi d'employer des techniques de RàPC. Le RàPC résout un nouveau problème en retrouvant et en adaptant des solutions ou des éléments de solution d'un problème précédemment résolu. Il utilise un raisonnement assez proche du raisonnement humain pour pouvoir coopérer avec l'utilisateur et permet de se focaliser sur plusieurs catégories d'informations concernant le plan, que ce soient des informations sur le problème posé, sur les images traitées ou sur la stratégie à adopter.

2.2 Coopération entre les intervenants

En TI, l'expert du domaine d'origine de l'image a un rôle important lors de la mise au point d'une application. En premier lieu, il définit le problème, pour lequel l'expert en TI doit construire une solution, en fournissant une ou plusieurs images (les objets à traiter) et une requête (les objectifs à atteindre). Puis lorsqu'une solution lui est proposée, il doit l'évaluer en visualisant les images résultats. En effet, le TI est un domaine où il n'existe pas de fonction d'évaluation idéale et seul l'expert du domaine peut juger de la validité finale d'une application. Cette validation se fait, soit visuellement, soit en utilisant les résultats obtenus dans un protocole de test plus global (ex : statistiques). Les rôles de l'expert du domaine et de l'expert en TI lors de l'élaboration d'une application sont donc dépendants, ce qui se traduit par des besoins de coopération et de communication entre ces experts. Nous voulons que notre système soit à la base de la communication en offrant un cadre pour représenter et structurer 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 la décomposition, adéquation à un type d'images, ...).

Pour favoriser le dialogue entre les experts et pour s'abstraire le plus possible des détails techniques du TI (qui ne peuvent que compliquer le dialogue) nous proposons de programmer au *niveau connaissance* [16], en aidant l'expert en TI à modéliser ses applications sous forme de plans.

De plus, il est possible d'accéder aux informations sur ces tâches et de visualiser les images intermédiaires, ce qui fait de notre système un outil d'expérimentation et de dialogue pour ces experts.

2.3 Réutilisation du savoir-faire

Pour résoudre un nouveau problème, le RàPC extrait de la mémoire les cas les plus similaires et les plus intéressants et les adapte pour élaborer une solution à ce nouveau problème. Son fonctionnement peut se décomposer en cinq étapes principales qui sont la *sélection,* l'*adaptation,* l'*évaluation,* la *correction* et l'*apprentissage.*

La première étape consiste à sélectionner les cas de la base de cas qui sont les plus proches du problème courant. Pour que cette étape soit efficace, le système doit posséder des moyens de remémoration et d'indexation pour retrouver les meilleurs cas rapidement. Cette recherche peut utiliser une indexation dynamique des cas [18] ou une classification plus statique basée sur des niveaux d'abstraction [1] ou sur la subsomption [19]. L'importance des critères à prendre en compte pouvant différer suivant le type de traitement à effectuer et suivant l'origine de l'image à traiter, nous adopterons plutôt une technique d'indexation dynamique. Le système doit également posséder une mesure de similarité pour effectuer une sélection fine. Cette mesure cherche, soit à maximiser la similarité [5], soit à minimiser l'effort d'adaptation [15]. L'absence de méthode d'évaluation automatique en TI, nous conduit à rechercher une maximisation de la similarité.

La deuxième étape consiste à adapter les solutions proposées par la sélection pour obtenir une nouvelle solution. Pour cela, le système doit prendre en compte toutes les informations fournies par la description du cas courant et évaluer les différences entre celui-ci et les cas sélectionnés. Dans les systèmes où les cas sont abstraits [3], cette étape consiste à affiner les solutions proposées jusqu'à obtention d'une solution complète. Dans les systèmes mémorisant des cas « concrets » [17], des connaissances spécifiques sont utilisées pour modifier localement les solutions proposées et les adapter au problème courant. Une abstraction des cas nous semblant peu adaptée au domaine du TI, nous opterons pour une recherche de cas « concrets ». Notre système étant un système interactif, cette phase pourra se faire en coopération avec l'utilisateur.

Les deux étapes suivantes sont l'évaluation et la correction de la solution proposée. La boucle évaluation/correction permet de valider la solution produite et d'acquérir de nouvelles connaissances en détectant et en corrigeant les erreurs. Le TI étant un domaine où il est difficile de fournir des fonctions d'évaluation, cette validation ne peut être faite que par l'utilisateur.

La dernière étape est celle de l'apprentissage du nouveau cas. Elle permet au système d'acquérir de nouvelles connaissances et utilise les mêmes descriptions et indexations des cas que l'étape de sélection.

3 Description du système TMO

Dans cette section, nous décrivons TMO qui est la partie opérationnelle de notre système et qui permet d'une part de créer et de mémoriser un plan de TI en définissant les différents éléments qui le composent (nœuds et feuilles du plan) ainsi que les relations entre ces éléments, et d'autre part d'exécuter les plans ainsi créés. Dans un premier temps, nous décrivons notre système du point de vue de l'architecture et du modèle conceptuel utilisé, puis dans un deuxième temps du point de vue des fonctionnalités proposées à l'utilisateur.

3.1 Un système à base de connaissances

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.

Les connaissances du domaine modélisent le savoir de l'expert en TI, elles comportent :

- 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 de conception d'application ou en cours de résolution s'il y a ambiguïté ou incomplétude,

- des connaissances en TI : elles représentent les stratégies utilisées pour analyser les images,

- des connaissances sur les opérateurs : telles que l'opération effectuée sur l'image, le nombre et les types des entrées/sorties, la détermination des valeurs des paramètres, etc.

Les connaissances de contrôle sont les connaissances qui gèrent les connaissances du domaine. On peut séparer ces connaissances en deux catégories :

- le contrôle du domaine : gestion des plans de TI (leur création, leur modification, leur exécution, ...),

- le contrôle du système : affichage des opérations à effectuer, sélection des traitements accessibles à l'utilisateur, insertion d'autres modules, ...

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 d'utilisateur (on ne propose pas les mêmes activités à un expert et à un novice) et suivant ses désirs (ex : affichages intermédiaires ou non).

Les trois niveaux de notre système sont représentés uniformément à l'aide de l'architecture « TACHE-METHODE-OUTIL » [10]. Cette représentation uniforme nous permet de gérer les connaissances de contrôle de la même façon que les connaissances du domaine et ainsi d'envisager l'articulation de plusieurs modes de résolution :

- actuellement, résolution manuelle (TMO),

- à court terme, résolution guidée (RàPC),

- à long terme, résolution automatique (BORG).

En résolution de problème, la modélisation sous forme de tâches génériques et décomposables est largement répandue [2] [6]. Dans notre système, une TACHE est la représentation d'un but ou d'un sous-but. 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 résout un problème général, elle se décompose en sous-tâches qui résolvent des problèmes plus élémentaires, ces sous-tâches pouvant être décomposées à leur tour. Une tâche peut être accomplie de plusieurs manières, on lui associe alors une ou plusieurs méthodes (fig. 2). Les tâches servent à représenter tous les buts du système, quel que soit leur niveau. Au niveau *domaine*, elles représentent un objectif de TI ou une sous-partie de celui-ci : dans le plan de la figure 7 « extraire les regroupements d'objets » est une tâche principale et « éliminer le fond d'image » en est une sous-tâche. Au niveau *contrôle domaine,* elles représentent les opérations que l'on veut effectuer sur les plans de TI : « exécuter un plan » et « sauvegarder un plan » sont des tâches de ce niveau, « instancier un plan » est une sous-tâche de « exécuter un plan ». Au niveau *contrôle système*, elles représentent les opérations telles que « initialiser le système » et « gérer l'interface du système », qui sont des opérations de contrôle du bon déroulement des sessions. Enfin au niveau *métacontrôle,* elles représentent des opérations de contrôle sur le contrôle, telles que « exécuter tâche » ou « choisir méthode ».

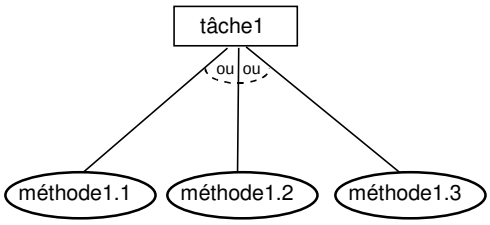

Fig.2 : une tâche associée à plusieurs méthodes

Une METHODE spécifie comment une tâche peut être accomplie. Chaque méthode est associée à une seule tâche, mais une tâche peut être résolue par plusieurs méthodes. Pour cela, on peut expliciter, pour chaque méthode, quand il est possible de l'utiliser. Le corps de la méthode peut prendre deux formes (fig. 3) : soit une décomposition en sous-tâches qui prend l'aspect d'un arbre « PUIS » (on parle alors de méthode complexe), soit l'appel à un code informatique par l'intermédiaire d'un outil (on parle alors de méthode terminale). L'ensemble des sous-tâches à exécuter pour accomplir une tâche est dépendant de la méthode choisie, ce sont donc les

méthodes qui gèrent les flux de données entre tâche et sous-tâches et entre tâche et outil.

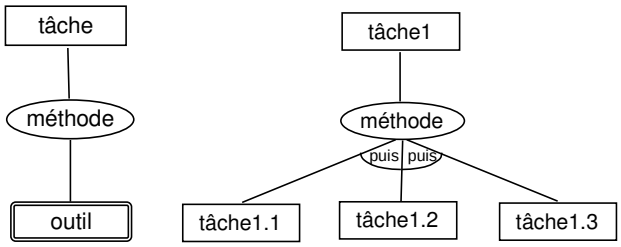

Fig.3 : méthode terminale et méthode complexe

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. Ce code est vu par l'utilisateur comme une boîte noire dont il connaît seulement la nature des transformations qu'elle effectue sur les entrées pour produire les sorties et dont il peut régler les paramètres. Au niveau *domaine*, nos outils représentent essentiellement des opérateurs de base du TI définis dans la bibliothèque d'opérateurs PANDORE (ex : binariser une image, appliquer un filtre moyenneur uniforme,...), mais ils peuvent également réifier des fonctions Lisp ou C externes à la bibliothèque.

L'exécution des tâches est gérée par un agenda. Tant que l'agenda n'est pas vide, une boucle générale sélectionne la première tâche de l'agenda et la tâche de métacontrôle « exécuter tâche » exécute cette tâche en sélectionnant la méthode la plus adaptée au contexte courant ou en demandant à l'utilisateur de choisir parmi les méthodes existantes. Si la méthode sélectionnée est une méthode complexe, les sous-tâches correspondantes sont mises dans l'agenda, si c'est une méthode terminale, on exécute le code associé à l'outil suivant le mode défini dans ce dernier.

3.2 Coopération système / utilisateur

Notre système à base de connaissances est doté d'une interface graphique permettant la mise en œuvre de la coopération homme /machine pour la création et l'exécution de plan. Les différentes fonctionnalités du système sont proposées dans des menus, chacune étant modélisée par une tâche de contrôle.

Création d'un plan

Lorsqu'il veut mettre au point une nouvelle application, l'utilisateur construit ses tâches et ses outils en remplissant les différents champs dans des fenêtres adaptées (fig. 4). Pour cela, il définit un titre (suffisamment expressif pour décrire la tâche ou l'outil sans ambiguïté), le niveau de compétence minimum requis pour l'utilisation de la tâche ou de l'outil et le nombre d'entrées, de sorties, de paramètres et de résultats (les entrées et les sorties correspondent à des images, les paramètres et les résultats à des valeurs numériques). Pour les outils il faut également définir l'opérateur et son mode d'exécution qui correspond à la façon dont l'opérateur ou la fonction est exécuté : mode *normal* (exécution simple de la fonction), mode *pour* (exécution un nombre prédéfini de fois) et mode *optimisation* (exécution conditionnelle). Suivant le cas, sont fournis également le nombre d'itérations ou le mode d'optimisation et la fonction d'évaluation. La fonction d'évaluation effectue des mesures locales sur l'image et permet, suivant le mode d'optimisation de tester qu'une condition est vérifiée ou pas (mode *tant que* et *jusqu'à*) ou de comparer les résultats d'exécution avec différentes valeurs de paramètres (mode *premier* ou *meilleur*). Ensuite, l'utilisateur relie ses tâches et ses outils en définissant des méthodes. Il sélectionne alors la tâche à laquelle appartient la méthode et les sous-tâches ou l'outil dans des listes. Les différents éléments créés sont sauvegardés automatiquement. Il décrit alors les flux de données à l'aide d'une représentation schématique de la partie du plan concernée ; il a également, à ce moment, la possibilité de définir chaque paramètre de l'outil comme étant une valeur fixée pour toute exécution ou comme étant une valeur à demander à l'utilisateur lors de l'exécution (dans ce cas, il peut proposer une valeur par défaut). La possibilité de définir ces différents éléments dans un ordre quelconque autorise une démarche de conception ascendante (par regroupement de sous-tâches), descendante (par décomposition en sous-tâches) ou mixte.

fig.4 : saisie des valeurs des champs d'une tâche

Exécution d'un plan

Pour exécuter un traitement sur une image, l'utilisateur doit d'abord choisir la tâche correspondante dans un menu. Ensuite il doit instancier le plan associé à cette tâche en fournissant l'image initiale. Le système exécute

alors la tâche en proposant des choix à l'utilisateur parmi les méthodes, lorsqu'il en existe plusieurs pour résoudre une tâche, et en demandant les valeurs des paramètres lorsque c'est nécessaire. Pendant l'exécution, l'utilisateur a la possibilité de visualiser les images se trouvant en entrée et en sortie des différentes tâches, et d'accéder aux informations concernant les tâches et les méthodes de résolution par l'intermédiaire d'une représentation graphique du plan sous forme d'arbre de tâches.

3.3 Expérimentations

Six plans créés au laboratoire ont été intégrés dans TMO. Ces plans traitent des problèmes divers sur des images de différentes origines (images de cytologie, images d'histologie, images de visages, images industrielles, images de synthèse). Nous présentons ici deux plans mis au point pour traiter des images biomédicales de cytologie et d'histologie fournies par le Centre de lutte contre le cancer F. Baclesse de Caen.

Dans le domaine des images de cytologie, nous avons mis au point un plan ayant pour but la détection des noyaux de cellules présents dans une image. Après une phase de segmentation région, on peut éliminer les noyaux qui ne répondent pas à certains critères de taille, de forme ou d'intensité pour fournir au biologiste une image qui ne comporte que les noyaux qu'il veut observer.

Dans le domaine des images d'histologie, souvent beaucoup plus délicates à traiter, nous avons par exemple créé interactivement le plan donné figure 8 pour mettre en évidence les groupements significatifs de noyaux, l'étude de ces groupements permettant de délimiter des lobules tumoraux. La modélisation de l'application sous forme 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.

4 Le RàPC comme aide à l'utilisateur

Le raisonnement par cas est un raisonnement humain naturel. En effet, on cherche souvent à résoudre un problème en se rappelant comment on a résolu un problème similaire : on diagnostique une panne ou une maladie parce que l'on est en présence des mêmes symptômes, on établit un plan de travail parce qu'un plan similaire avait donné de bons résultats dans les mêmes circonstances.

Le problème de l'homme lorsqu'il utilise ce style de raisonnement est qu'il possède une mémoire limitée et sélective qui ne lui permet pas toujours de se souvenir du cas le mieux approprié. Le RàPC apporte une solution à ce problème en ce sens qu'il permet de mémoriser tous les cas et de retrouver le mieux adapté, si l'on possède les bons critères de choix. La difficulté est alors d'établir la liste de ces critères.

En s'appuyant sur le même raisonnement que le RàPC, les systèmes d'aide à base de cas [14] assistent l'utilisateur en augmentant sa mémoire et en sélectionnant les cas les plus proches du problème posé, mais, contrairement aux système de RàPC entièrement automatiques, ils laissent l'utilisateur prendre la décision finale : à savoir, sélectionner le cas le plus approprié et décider des besoins d'adaptation.

Dans la suite de cette section, nous présentons sous forme d'algorithme le principe utilisé pour aider l'utilisateur à raisonner à partir de cas, puis nous décrivons les critères de sélection utilisés et enfin, nous détaillons la technique de recherche/adaptation employée.

4.1 Principe

Un cas correspond à un plan qui définit un problème et sa solution. A chaque tâche du plan est associé un ensemble de critères qui caractérisent la position de la tâche dans le plan, le but à atteindre et l'image traitée. Rechercher un cas revient donc à rechercher la tâche racine du plan correspondant au cas.

Le processus de construction d'une application par réutilisation s'appuie sur l'algorithme de recherche/ adaptation présenté ci-dessous. Cet algorithme fournit un plan de traitement en recherchant la tâche racine de ce plan. On a ensuite la possibilité de l'appliquer localement de façon récursive pour modifier une partie de la solution obtenue. Il s'exécute en coopération avec l'utilisateur et utilise des critères classés en deux catégories suivant qu'ils sont liés à la description du problème à résoudre ou au contexte des images traitées. Nous employons donc deux fonctions de similarité : *f1* qui utilise les critères liés à la description du problème et *f2* qui utilise les critères liés au contexte des images.

1- demander à l'utilisateur les valeurs des critères liés à la description du problème

2- rechercher l'ensemble ***T*** des tâches correspondant aux critères désirés en utilisant *f1*.

3- tant que la cardinal de ***T*** n'est pas suffisamment petit faire

   - demander à l'utilisateur les valeurs des critères liés au contexte des images
   - réduire l'ensemble ***T*** en utilisant *f2*

4- demander à l'utilisateur de choisir une tâche parmi l'ensemble des tâches restantes

5- a) si l'utilisateur a sélectionné une tâche, il peut alors adapter le plan correspondant à son problème en relançant l'algorithme sur les sous-tâches qui ne lui conviennent pas ou en créant manuellement de nouvelles parties du plan.

   b) sinon (aucune tâche ne lui convient), demander à l'utilisateur de choisir, parmi celles proposées, une suite de problèmes de TI pour obtenir le schéma d'une décomposition en sous-tâches et relancer l'algorithme sur chaque problème.

4.2 Critères de sélection

Critères liés à la description du problème

Les critères utilisés par *f1* sont ceux qui correspondent à la définition du problème à résoudre. Ils comprennent le ***type*** ou la ***phase de traitement***, la ***définition*** même du problème et le ***niveau d'abstraction*** correspondant.

Suivant le ***niveau d'abstraction*** du problème posé, on prendra en compte soit le ***type de traitement*** (*segmentation d'image, restauration d'image, ...*), soit la ***phase de traitement*** désirée (par ex. pour la segmentation : *pré-traitement, détermination de germes, localisation de contours, ...*).

La ***définition du problème*** correspond à un ensemble de mots clefs que l'on sélectionne parmi ceux répertoriés dans trois listes : une liste de verbes (les opérations à effectuer telles que *détecter*, *classifier*, *binariser, …*), une liste de noms (les objets sur lesquels effectuer l'opération, par exemple *contours*, *régions, étiquettes, …*) et une liste d'adjectifs (pour qualifier l'opération ou les objets, par exemple *local*, *grand*, *partiel, …*).

Enfin les ***niveaux d'abstraction***, dont l'ensemble correspond à un découpage horizontal du plan, sont basés sur ceux définis dans le planificateur automatique BORG [8]. Le premier niveau d'abstraction est le *niveau intentionnel*. Les tâches de ce niveau répondent à la question « quoi faire ? » et portent sur les objectifs du TI. Dans le plan de la figure 7, « extraire les regroupements d'objets », « éliminer le fond », « former les objets à partir des régions » et « extraire les groupes d'objets » sont des tâches du niveau intentionnel. Le deuxième niveau est le *niveau fonctionnel*. Les tâches de ce niveau répondent à la question « comment faire ? » et représentent une technique du TI abstraite des contraintes techniques liées à la réalisation. Dans le plan de la figure 7, « classifier les pixels », « marquer les régions » et « former le graphe » sont des tâches du niveau fonctionnel. Le troisième et dernier niveau est le *niveau opérationnel*. Les tâches de ce niveau répondent à la question « avec quoi ? » et représentent un savoir-faire technique sur le TI qui peut s'implanter sous forme d'algorithme. Ce niveau se rapproche de la notion d'opérateur mais sans être lié à une bibliothèque particulière. Dans le plan de la figure 7, « lissage moyenneur », « gradient », « binarisation basée sur le contraste » et « binarisation basée sur la variance » sont des tâches du niveau opérationnel. Ces trois niveaux sont également très proches des trois niveaux d'abstraction que distingue David Marr [12] dans son étude sur la perception visuelle : spécification des tâches à effectuer (ce qu'il faut calculer et pourquoi), méthodes ou algorithmes (comment réaliser les calculs) et implantation des algorithmes.

L'utilisation de cette première série de critères permet de sélectionner les cas assez proches du cas courant. Si l'ensemble des cas obtenus après cette première sélection comporte un grand nombre de cas, on cherchera à réduire celui-ci en utilisant une deuxième série de critères importants pour le choix du traitement à appliquer : ceux liés au contexte des images.

Critères liés au contexte des images

Parmi les critères liés aux images, certains décrivent la qualité de l'image. Il s'agit en particulier du ***type*** et de la ***quantité de bruit*** que l'on trouve dans l'image et de la ***qualité du contraste***. Ce sont des critères particulièrement importants pour le choix d'un prétraitement.

Les autres critères correspondent plutôt à une description sémantique de l'image et de ce que l'on cherche à mettre en évidence. Ils comprennent ***la présence ou l'absence d'un fond***, l'***aspect*** des objets recherchés (niveau de gris homogène, couleur claire, textures, contours épais, …), la ***forme*** des objets recherchés (convexe, concave, allongé, compact, …), la ***taille relative*** des objets par rapport à l'ensemble des objets présents dans l'image, la ***position*** des objets recherchés (droite, milieu, gauche, bas, centre, haut), et les ***relations*** entre les objets recherchés (proches, connexes, inclus, …).

Utilisation des critères

Chacun de ces critères possède un mode de comparaison retournant une valeur numérique (égalité, supériorité, inclusion, intersection, équivalence, …) et un coefficient d'importance attribué par l'utilisateur. Ce coefficient d'importance permet de déterminer l'ordre de prise en compte des critères par l'algorithme et est utilisé par *f2*.

Tous ces critères ne sont pas renseignés par l'utilisateur : certains, tels que le contraste, peuvent être calculés et d'autres, tels que le bruit ou la présence d'un fond homogène, peuvent être déduits du domaine de provenance de l'image.

4.3 Remémoration et adaptation

La boucle « tant que » de l'étape 3 de l'algorithme demande la valeur d'un nouveau critère jusqu'à l'obtention d'un ensemble de tâches ***T*** qui soit suffisamment petit pour pouvoir en présenter la liste à l'utilisateur. Ce dernier peut alors sélectionner la tâche qui lui parait la mieux adaptée. La deuxième fonction de similarité permet de sélectionner les tâches de l'ensemble ***T*** qui possèdent des valeurs proches pour les critères renseignés. Donc plus le nombre de critères renseignés est important, plus la fonction de similarité permet d'effectuer une sélection fine. L'ordre de renseignement d'un paramètre dépend de son importance et de la capacité de l'utilisateur à fournir les informations. La boucle « tant que » permet de ne renseigner que les critères nécessaires et donc de laisser suffisamment d'importance à l'aspect intuitif qui caractérise la décision de l'utilisateur.

Dans le cas où l'utilisateur sélectionne une des tâches qui lui sont proposées à l'étape 4, il faut ensuite adapter le plan correspondant au problème courant. Adapter le plan consiste à modifier certaines sous-tâches ou certains outils. Ces modifications peuvent être locales (remplacement d'un opérateur par un autre ou modification des valeurs de paramètres) ou globales (remplacement d'un sous-plan). Dans le cas d'une modification globale, on va chercher à remplacer la tâche racine du sous-plan à modifier par une autre. Pour cela on peut relancer le processus de remémoration en considérant

le problème que doit résoudre cette tâche comme étant le nouveau problème courant. Cet aspect « récursif » de l'algorithme permet de réaliser une adaptation du plan à n'importe quel niveau dans l'arbre de tâche.

4.4 Perspectives pour la gestion des échecs

Si aucune des tâches proposées à l'étape 4 ne convient à l'utilisateur, le système peut lui proposer de choisir une démarche de travail. Par exemple, une façon d'aborder un problème de segmentation est d'enchaîner les étapes « prétraiter », « détecter », « localiser » et « regrouper », chacune de ces étapes étant un « problème abstrait » de TI (abstrait dans le sens où il doit être défini plus précisément pour être résolu). On peut donc proposer à l'utilisateur de choisir sa démarche de travail en sélectionnant la « suite de problèmes abstraits » correspondante. Cette notion de « suite de problèmes » est basée sur celle présentée dans CommonKADS [4], dans lequel la réutilisation s'appuie sur une typologie des problèmes et sur le fait qu'il existe des dépendances entre ces différents types de problèmes. Lorsque l'utilisateur a choisi une « suite de problèmes abstraits », le système peut relancer le processus de remémoration/ adaptation sur chacun des problèmes de la suite, en demandant à l'utilisateur de les définir plus précisément.

5 Conclusion

Dans cet article, nous avons décrit un système opérationnel basé sur l'architecture TMO et doté d'une interface graphique autorisant des interactions entre le système et l'utilisateur. La formalisation de la démarche du traiteur d'images à l'aide d'un graphe TMO fournit un support de communication entre l'expert en TI et l'expert du domaine et permet une vision globale du processus.

Nous nous sommes aussi attachés à montrer comment la multiplicité des applications nous conduit à réutiliser des traitements pour accroître l'aide fournie par le système. Nous avons proposé pour cela un algorithme de RàPC et des critères de sélection pour déterminer quand une tâche de TI est réutilisable.

La possibilité d'appliquer plusieurs fois l'algorithme de façon locale et récursive permet à l'utilisateur d'adapter son plan aussi précisément qu'il le désire. Cela permet également de ne pas rester sur un échec dans le cas où il n'existe pas de plan suffisamment approprié au problème.

Plusieurs plans de taille importante ont été étiquetés à l'aide des critères de sélection définis dans cet article. Ces plans répondent à des problèmes variés et s'appliquent dans des images d'origines différentes. La détection de similarités entre ces plans grâce à l'étiquetage confirme la pertinence des critères choisis.

La validation du module de RàPC devra être complétée par une implémentation et des tests informatiques de l'algorithme ; puis par la détermination des différentes suites de problèmes de TI à proposer à l'utilisateur en cas d'échec. Pour confirmer la robustesse du modèle TMO, l'algorithme de RàPC devra être modélisé par un ensemble de tâches de contrôle à l'instar des tâches permettant la création ou l'exécution des plans.

Remerciements

Ce travail a été réalisé dans le cadre du pôle Traitement et Analyse d'Image de Basse Normandie.

Nous remercions Abderrahim ELMOATAZ et François ANGOT qui ont mis au point les plans pour les applications de cytologie et d'histologie mentionnées dans cet article.

Bibliographie

[1] Archibald J. & Hardy C., A Case-Based Reasoning System for Pilot Production Using FMEA Data, *Expersys'96*, Paris, Octobre1996.

[2] Benjamins R. & Pierret-Golbreich C., Assumption of Problem-Solving Methods, *Proceedings of the 6th KEML Workshop*, Paris, 1996.

[3] Bergmann R. & Wilke W., Learning Abstract Planning Cases, *ECML'95*, Heraklion, Crète, Avril 1995.

[4] Breuker J. & Van de Velde W., *CommonKADS Library for Expertise Modelling*, IOS Press, 1994.

[5] Caulier P. & Houriez B., Apport de la modélisation des connaissances à partir de cas pour la capitalisation et la réutilisation de connaissances, *JAVA'95*, Grenoble, Avril 1995.

[6] Chandrasekaran B. & Josephson J.R., The Ontology of Tasks and Methods, *AAAI Fall Symposium*, Stanford University, March 1997.

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

[8] 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, p298-313, Janvier 1995.

[9] Clouard R., Elmoataz A. & Angot F., *PANDORE : une bibliothèque et un environnement de programmation d'opérateurs de traitement d'images*, Rapport interne du GREYC, Caen, France, Mars 1997.

[10] Ficet V., Construction interactive d'un modèle conceptuel d'applications de traitement d'images, *RJC-IA*, Nantes, France, p95-102, Août 1996.

[11] *IRIS Explorer User's Guide*, Silicon Graphics, Inc., Mountain View, California, n°007-1371-020, 1993.

[12] Marr D., *Vision : A computational investigation into the human representation and processing of visual information*, W.H. Freeman and Co, San Francisco, 1982.

[13] Rasure J. & Kubica S., The Khoros Application Development Environment, Experimental Environments for Computer Vision and Image Processing, editor H.I. Christensen and J.L. Crowley, *Word Scientific*, Singapore, 1-32, 1994.

[14] Kolodner J.L., Improving Human Decision Making through Case-Based Decision Aiding, *AI Magazine*, vol. 12, pp52-68, 1991.

[15] Munoz-Avila H. & Hüllen J., Feature Weighting by Explaining Case-Based Problem Solving Episodes, editor Smith I. & Falling B., *EWCBR'96*, Lausanne, Suisse, 1996.
[16] Newell A., The knowledge level, *Artificial Intelligence*, n°18, p87-127, 1982.
[17] Smyth B. & Keane M.T., Retrieval & Adaptation in Déjà-Vu, a Case-Based Reasoning System for Software Design, *AAAI Fall Symposium'95*, MIT Campus, Cambridge, Massachusetts, Novembre 1995.
[18] Veloso M., Munoz-Avila H. & Bergmann R., Case-based planning : selected methods and system, *AI Communication*, vol.9, n°3, Septembre 1996.
[19] Weida R., Knowledge Representation for Plan Recognition, *IJCAI'95*, Montreal, Canada, 1995.
[20] Willamowski J., *Modélisation de tâches pour la résolution de problèmes en coopération système/ utilisateur*, Thèse de doctorat, Université Joseph Fourier, Grenoble, France, Avril 1994.

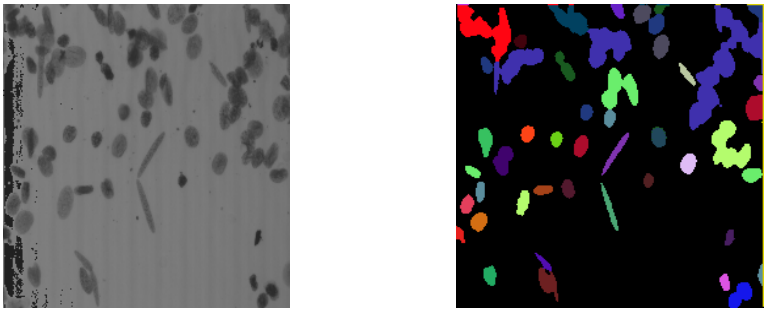

Fig.5 : image d'origine et image de résultat intermédiaire pour la recherche de cellules dans une image de cytologie

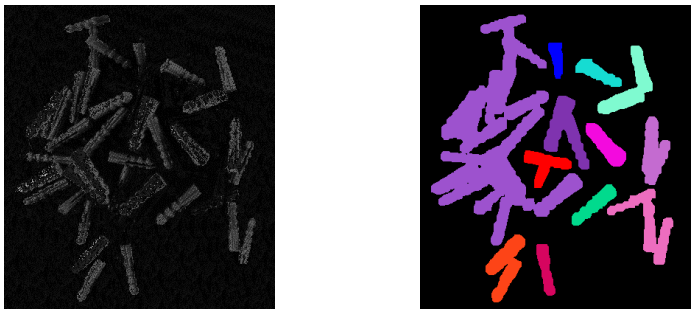

Fig.6 : image d'origine et image de résultat intermédiaire pour le tri de pièces industrielles

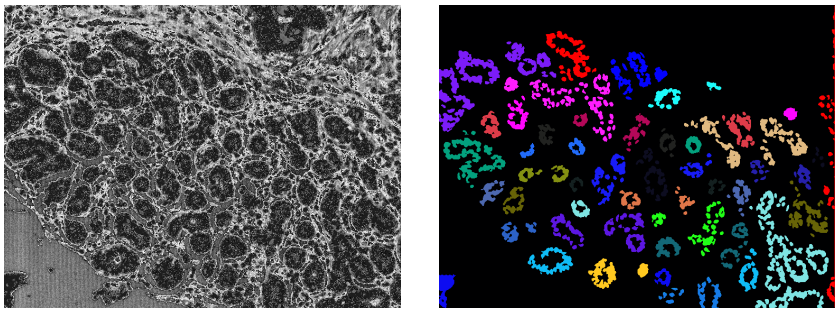

Fig.7 : image d'origine et image résultat de l'extraction de groupements de noyaux dans une image d'histologie

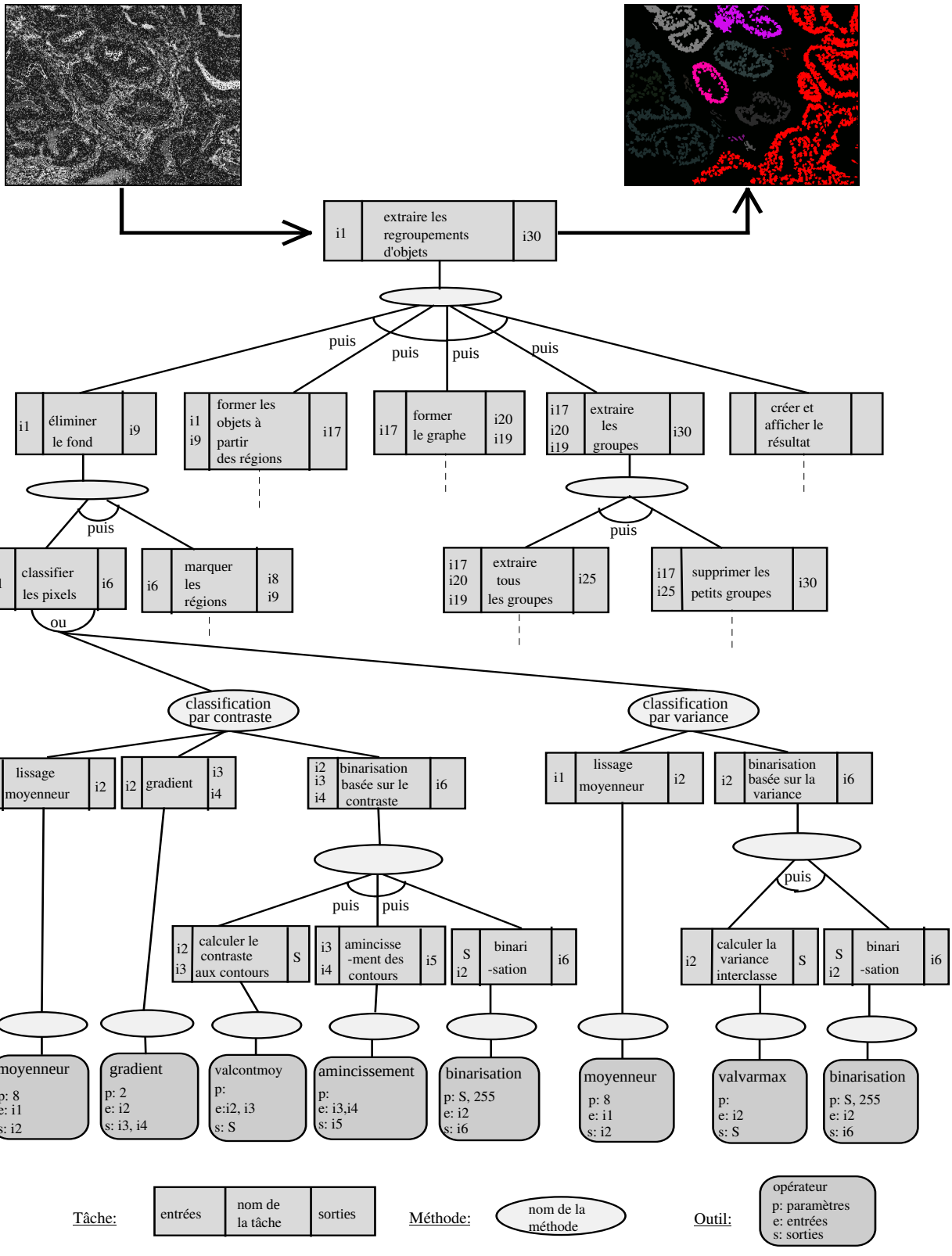

Fig.8 : partie du plan « extraire les regroupements d'objets »