HAL
open science

Un système interactif d'aide à la construction d'applications de traitement d'images

Valérie Ficet-Cauchard, Christine Porquet, Marinette Revenu, Régis Clouard

To cite this version:

Valérie Ficet-Cauchard, Christine Porquet, Marinette Revenu, Régis Clouard. Un système interactif d'aide à la construction d'applications de traitement d'images. 16e Colloque GRETSI, 1997, Grenoble, France. pp.949-952. hal-00869596

HAL Id: hal-00869596

https://hal.science/hal-00869596

Submitted on 4 Oct 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.

Un système interactif d'aide à la construction d'applications de traitement d'images

Valérie FICET-CAUCHARD, Christine PORQUET, Marinette REVENU, Régis CLOUARD

GREYC-ISMRA - 6 bd Maréchal Juin - F-14050 Caen cedex - tél: 02-31-45-27-21 - fax: 02-31-45-26-98
e-mail: Valerie.Ficet@greyc.ismra.fr

RÉSUMÉ

Dans le cadre de la mise au point d'applications de traitement d'images, nous proposons un système d'aide fondé sur la modélisation et l'explicitation de la démarche du traiteur d'images. Les traitements à effectuer y sont représentés sous une forme compréhensible et modifiable par l'utilisateur (enchaînements d'opérateurs sous forme de plans), dans un souci de réutilisation de parties entières de traitement. L'organisation de ce système repose sur une modélisation naturelle des connaissances sous forme de Tâches, Méthodes et Outils. La construction et l'exécution des plans de traitement d'images s'effectuent via une interface graphique qui facilite l'expérimentation de diverses techniques sur une image.

ABSTRACT

In this paper, a system that can provide some assistance for developing of Image Processing (IP) applications is described. The idea is to represent and make explicit the thought processes of IP developers. Sequences of IP operators are represented as plans following a twofold motivation : one the one hand making plans easy to understand and modify, and on the other hand enabling to reuse whole parts of plans. The system's organization hinges on a natural knowledge modelling based on Tasks, Methods and Tools. Building and executing IP plans is done through a graphical interface desingned to ease the trying out various experiments of images.

Introduction

Le Traitement d'Image (TI) s'applique maintenant à des secteurs d'activité très variés (biomédical, géographique, industriel) et dans lesquels une multitude de problèmes peuvent être posés, chacun de ces problèmes ayant une solution particulière. Pour aider les experts en TI à mettre au point leurs applications, des bibliothèques d'opérateurs ont été développées dans différents laboratoires, parmi celles-ci on peut citer SIMPA [1], la bibliothèque de méthodes de traitement du signal et des images du GDR-PRC ISIS, ou PANDORE [2], la bibliothèque d'opérateurs de TI développée dans notre laboratoire. Se pose alors le problème de l'aide à l'utilisation de ces bibliothèques : deux familles de systèmes tentent d'y apporter une réponse. La première correspond aux outils de programmation graphique, tels que KHOROS [3] ou VISILOG [4], qui permettent à l'utilisateur de sélectionner et d'enchaîner des opérateurs. Dans ces systèmes, les opérateurs ou les regroupements d'opérateurs effectués par l'utilisateur sont vus comme des boites noires : il n'y a pas d'explication ni de modélisation du raisonnement de l'utilisateur et le travail doit être repris à zéro pour chaque nouvelle application. Il est donc difficile de réutiliser les éléments de solution construits lors d'une application précédente. La deuxième famille correspond aux systèmes de construction automatique d'application tels que les planificateurs automatiques OCAPI [5], BORG [6] et ExTI-SATI [7]. Dans ces systèmes, il n'y a pas ou peu d'interventions de l'utilisateur, hormis pour la formulation des objectifs, et les connaissances qui y sont explicitées et représentées le sont plus dans un souci d'opérationnalisation que pour la compréhension de l'utilisateur.

Le but de nos recherches est la mise au point d'un système d'aide à l'utilisateur qui permette, non seulement à ce dernier de créer ses enchaînements d'opérateurs sous forme de plans, mais aussi d'expliquer et de représenter son raisonnement de façon à pouvoir réutiliser des parties de traitements qui ont été mises au point dans d'autres applications. Pour cela, nous lui proposons de modéliser ses plans à l'aide de tâches (buts ou sous-buts à atteindre), de méthodes (techniques pour atteindre un but) et d'outils (opérateurs réifiés), qui sont les principaux composants de l'architecture « TACHE-METHODE-OUTIL » (TMO) [8].

Spécifications

Le but de « l'Atelier d'Intégration de Connaissances en Traitement et Interprétation d'Images » de l'Équipe Image du GREYC est la création d'un environnement informatique qui facilite la construction de programmes de TI. Cet environnement doit constituer une aide pour l'expert en TI lors de l'élaboration d'une application comme lors de son exécution. Il doit donc comporter une base de plans pour le stockage des applications créées et des modules de contrôle pour gérer la création et l'exécution de ces plans. Il doit aussi offrir une modélisation des traitements qui favorise la réutilisation d'éléments de solution de n'importe quel niveau d'abstraction, car on veut pouvoir réutiliser des parties entières de traitement, par exemple un pré-traitement. Enfin, notre modèle doit permettre de représenter et de tester différentes méthodes pour satisfaire un but ou un sous-but.

La première fonctionnalité de notre système est la construction d'applications. L'expert en TI n'a plus à faire de programmation au sens classique : il programme au « niveau connaissance » en utilisant des blocs prédéfinis. Contrairement aux outils de programmation graphique classiques, dans lesquels l'utilisateur doit obligatoirement adopter une démarche ascendante lors de la construction de son plan, dans notre système, il est possible de choisir entre une démarche ascendante (par regroupement), descendante (par décomposition de problème en sous-problèmes) ou mixte. Contrairement aux systèmes de résolution automatique, dans notre système, la modélisation de

plusieurs méthodes de résolution pour un même problème n'impose pas l'écriture immédiate de règles de choix entre ces méthodes.

La deuxième fonctionnalité est l'exécution d'une application préalablement mise au point à l'aide du système. Pour cela, l'utilisateur doit déterminer le plan à exécuter en reconnaissant son problème dans une liste prédéfinie, ce qui ne l'oblige pas à définir entièrement son problème a priori comme c'est le cas dans les systèmes de résolution automatique. Si des choix doivent être faits entre différentes techniques pendant l'exécution, les critères pour effectuer ces choix ne seront demandés qu'au moment nécessaire.

La possibilité de tester plusieurs de ces techniques pour comparer leurs résultats, de visualiser des images intermédiaires comme dans VSDE [9] ou dans CONNY [10] et d'accéder à la représentation graphique d'un plan font de notre système un outil d'expérimentation pour l'expert en TI. Par ailleurs, le système possède une interface graphique qui permet de visualiser le plan sous forme d'arbre de tâches pendant l'exécution. Une tâche correspond à un but ou a un sous-but de l'application et l'interface graphique permet d'accéder aux informations concernant les tâches et les méthodes de résolution existantes.

Enfin, lorsqu'il a validé son application, l'expert en TI peut soit stocker son plan dans la bibliothèque de plans, soit générer du code C++ pour une exécution en routine.

Architecture du système

Notre système est un système à base de connaissance composé de trois niveaux. Le premier niveau est celui des connaissances du domaine qui contient des connaissances sur l'application en cours (pour donner un sens et un sujet à l'image), des connaissances en TI (qui permettent la détermination des stratégies à utiliser pour atteindre des buts) et des connaissances sur les opérateurs (qui permettent leur sélection, la détermination des valeurs de paramètres et la réalisation d'enchaînements syntaxiquement corrects). Le deuxième niveau est celui des connaissances de contrôle qui facilitent l'exploitation des connaissances du domaine. On peut séparer ces connaissances en deux catégories : les connaissances sur le contrôle du domaine (pour la gestion des plans de TI) et les connaissances sur le contrôle du système (affichage des opérations à effectuer, sélection des traitements accessibles à l'utilisateur, contrôle des enchaînements, ...). Le dernier niveau est celui des connaissances de métacontrôle qui définissent les règles de comportement du système, vis à vis de l'utilisateur: le contrôle sera différent suivant le type de l'utilisateur et suivant ses désirs (explicitation de la méthodologie ou non, ...).

Les trois niveaux de notre système sont représentés à l'aide de l'architecture « TACHE-METHODE-OUTIL » .

Une TACHE 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 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 donc une ou plusieurs méthodes (fig.1). 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 : « extraire les regroupements d'objets » est une tâche principale du domaine 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 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 ». 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 opérations. Enfin au niveau *métacontrôle*, elles représentent des opérations de contrôle sur le contrôle, telles que « exécuter une tâche » ou « choisir une méthode ».

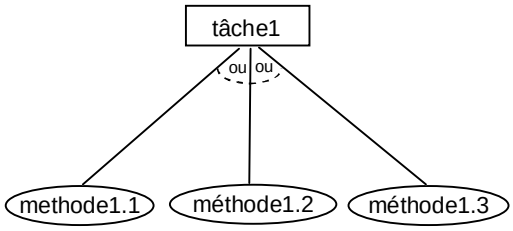

Fig.1 : une tâche associée à plusieurs méthodes

Une METHODE spécifie comment une tâche peut être accomplie. Chaque méthode est associée à une seule tâche, mais une tâche peut être associée à plusieurs méthodes. Pour cela, on cherche à expliciter, pour chaque méthode, quand il est possible et réellement souhaitable de l'utiliser. Le corps de la méthode peut prendre deux formes (fig.2) : soit une décomposition en sous-tâches qui prend l'aspect d'un arbre « ET - 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 exécuter 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.

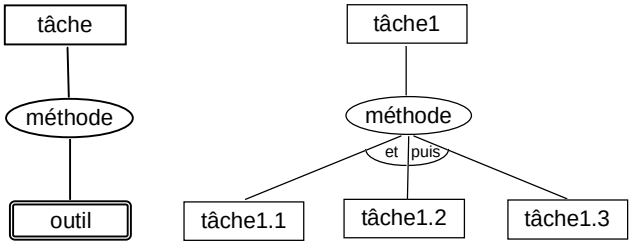

Fig.2 : méthode terminale et méthode complexe

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 les transformations qu'elle effectue sur les entrées pour produire les sorties et dont il peut régler les paramètres. 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 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.

Réalisation et validation

A l'aide de l'interface graphique du système, l'utilisateur construit ses tâches et ses outils en remplissant les différents champs dans des fenêtres adaptées (fig. 3). Puis il les relie en définissant ses méthodes de la même façon. Lorsqu'il veut exécuter un traitement sur une image, l'utilisateur doit d'abord choisir la tâche correspondante dans un menu. Ensuite il doit paramétrer le plan associé à cette tâche en fournissant l'image initiale et des valeurs de paramètres pour les opérateurs. Enfin le système exécute la tâche en proposant des choix à l'utilisateur parmi les méthodes, lorsqu'il en existe plusieurs pour résoudre une tâche.

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. 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 (fig. 4). A partir d'une carte de régions, on peut éliminer les noyaux qui ne répondent pas à certains critères de taille, de forme ou de niveau de gris pour fournir au biologiste une image qui ne comporte que les noyaux qu'il veut observer. Dans le domaine d'images d'histologie, souvent beaucoup plus délicates à traiter, nous avons par exemple créé manuellement un plan pour mettre en évidence les groupements significatifs de noyaux, l'étude de ces groupements permettant de déterminer la présence de lobules tumoraux.

Conclusion

La formalisation du processus d'analyse d'images à l'aide de graphes TMO fournit un support de communication entre le système et l'expert en TI pour la construction et l'exécution d'applications et permet à l'expert d'avoir une vision globale du processus.

Elle permet également de faire 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, ce qui en fait un support pour le partage et la critique des connaissances de différents traiteurs d'images.

Par ailleurs, nous voulons pouvoir réutiliser des éléments quelque soit leur niveau dans le plan (i.e. les tâches comme les outils). Mais un simple nommage des tâches pour en présenter une liste alphabétique deviendrait vite insuffisant. Une étude des techniques de Raisonnement par Cas nous a conduit à essayer de définir une ontologie et des taxonomies des tâches qui permettent de classifier une nouvelle tâche et de la retrouver pour la réutiliser. Nous proposons pour cela de nous baser sur les axes et critères de classification suivants : les 3 niveaux d'abstraction définis dans le planificateur automatique BORG développé dans notre équipe (objectif, fonctionnalité, procédure), les différentes phases de traitement d'une image (pré-traitement, segmentation, caractérisation, resegmentation), les critères physiques, perceptuels et sémantiques liés à une image, les objets du traitement (régions, contours, graphes, ...). Notre travail porte actuellement sur l'affinage et sur les techniques d'utilisation de notre classification des tâches.

Remerciements

Nous remercions M. Abderrahim ELMOATAZ et M. François ANGOT qui ont mis au point les plans pour les applications de cytologie et d'histologie cités précédemment.

Bibliographie

[1] « *SIMPA - Notice d'utilisation version 2.0* », GDR-PRC ISIS, France, 1991.

[2] 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.

[3] 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, 1994, 1-32.

[4] « *Visilog4 Reference Guide* », Noesis, Jouy en Josas, France, 1992.

[5] 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.

[6] 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, janvier 1995, p298-313.

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

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

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

[10] Liedtke C-E. & Blömer A., « *Architecture of the knowledge based configuration system for image analysis: CONNY* », 11th ICPR, The Hague, Netherlands, 1992, p375-378.

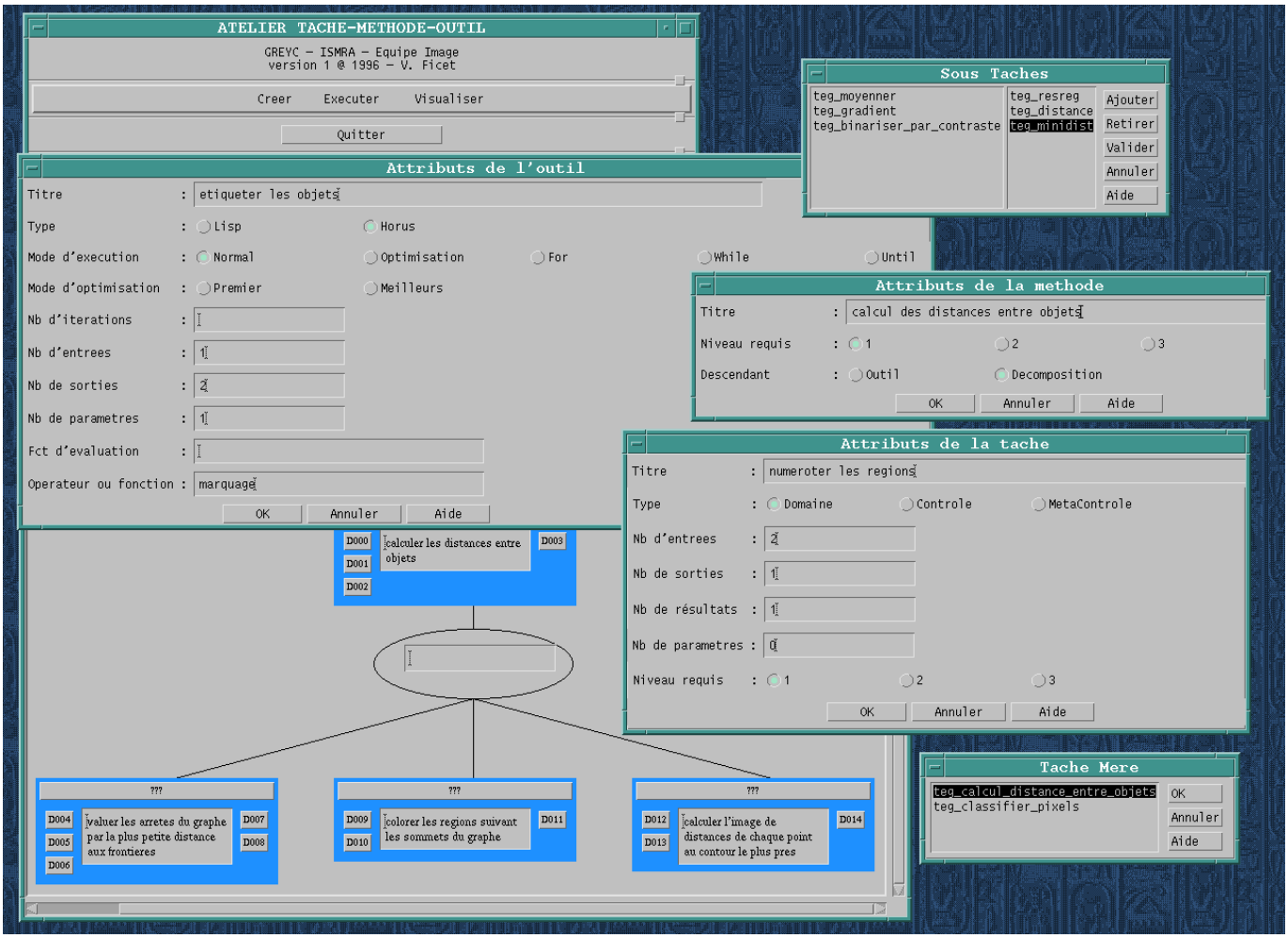

Fig.3 : copie d'écran de l'interface graphique montrant les fenêtres de remplissage des attributs des Tâches, Méthodes et Outils

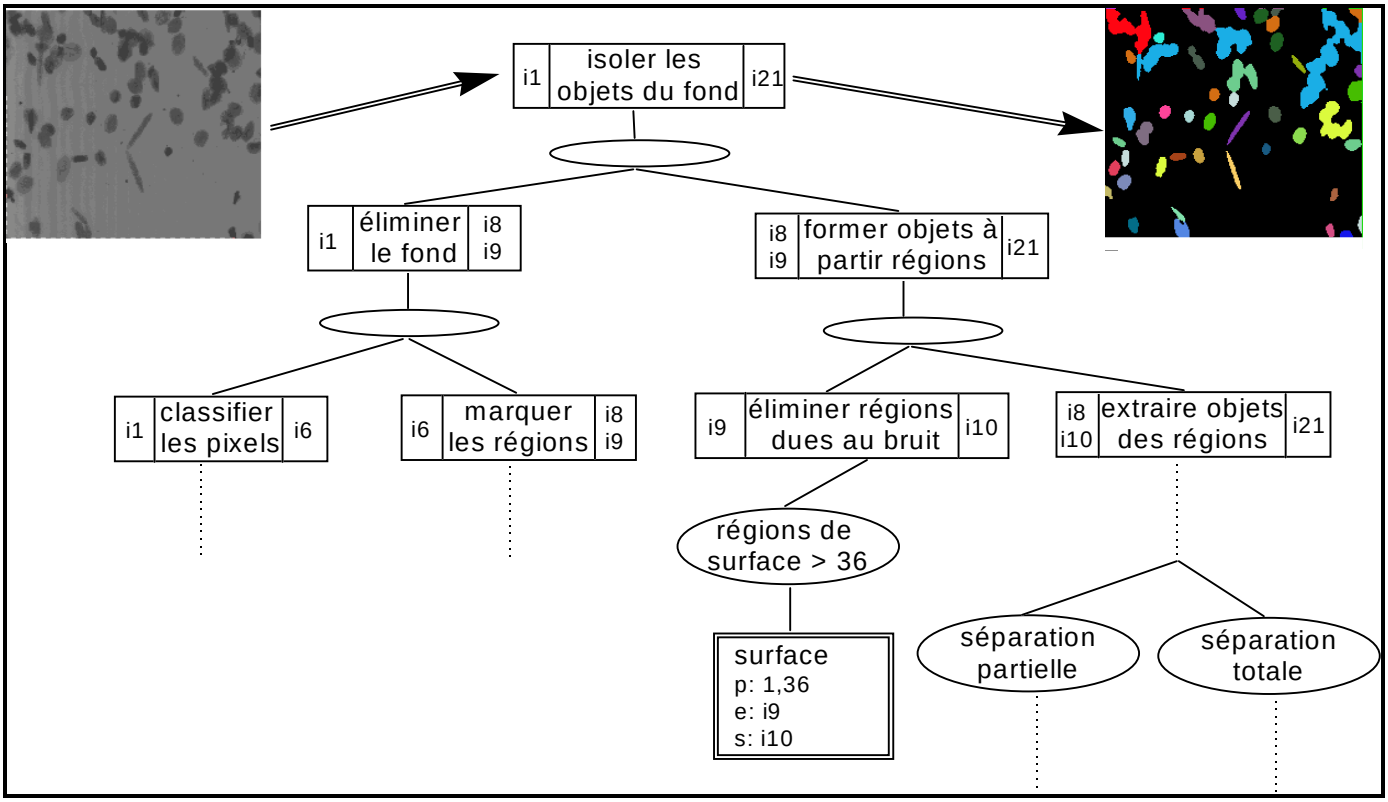

Fig. 4 : Plan partiel pour la détection des noyaux de cellules dans une image de cytologie