Dimitri Vibert

stage (du 1 juin au 31 août 2007)

Développement d'un plugin de segmentation d'image cellulaire pour le logiciel libre ImageJ

Split-And-Merge-Segmentation

Table des matières

1 Introduction

Dans le cadre de ma formation en école d'ingénieur en informatique, il m'a été demandé en deuxième année, d'effectuer un stage en entreprise d'une durée de 3 mois (du 1 juin au 31 août 2007).

Lors de ma recherche de stage, j'ai privilégié certains critères qui me semblaient prépondérants pour le choix de l'entreprise. En effet, je voulais que l'entreprise dans laquelle j'allais effectuer mon stage soit une grosse structure, et le travail proposé, me permette de prendre certaines responsabilités. Après avoir étudié plusieurs réponses d'entreprises favorables a mes demandes de stage, et le CNRS semblant correspondre à la plupart de mes attentes, c'est donc le groupe Montpellier Rio Imaging qui m'a accueilli.

Montpellier RIO Imaging est une plate-forme de soutien des programmes de recherche des Sciences de la Vie et de la Santé qui fait appel à l'imagerie microscopique dans tous ses aspects. Sa mission est de proposer, définir et mettre en œuvre les moyens nécessaires, ainsi que les actions qui s'y rattachent. Elle permet aux chercheurs et scientifiques d'effectuer des acquisitions sur microscope et ensuite de traiter leurs images sur ordinateur.

Mon tuteur m'a alors proposé plusieurs sujets :

- Le premier concernait de la reconstruction 3D à partir de coupes prises au microscope. Ce sujet me paraissait intéressant mais bien au dessus de mes compétences.

- Le deuxième sujet, était de continuer le travail du précédent stagiaire concernant le système de comptage d'utilisation des microscopes. Ce programme permet de savoir quelles machines sont utilisées, à quel instant et combien de temps.

- Enfin le dernier sujet, que j'ai accepté, touchant à la fois au traitement d'image et à la biologie me paraissait être une expérience enrichissante et intéressante.

Le sujet du stage consistait à séparer et compter des cellules sur des images numériques prisent au microscope. Le principal problème étant que les cellules se touchent et qu'il faut trouver des algorithmes et des méthodes mathématiques pour arriver à les séparer au mieux.

Mon stage s'est déroulé dans la salle d'analyse du CRBM (Centre de Recherche en Biochimie Macro moléculaire). C'est là que les scientifiques viennent traiter leurs images sur ordinateur après avoir fait des acquisitions. J'ai travaillé avec une équipe de 6 personnes, le responsable de la plate-forme, le développeur (mon tuteur), un administrateur réseau et trois biologistes. Mais, concernant mon sujet je n'ai eu à faire qu'à mon tuteur et l'administrateur réseau.

N'ayant aucune connaissance en traitement d'image, il a fallut que je me familiarise avec ce domaine. Mon tuteur m'a donc demandé de réaliser une présentation en Anglais de quarante-cinq minutes sur les différentes algorithmes de segmentation d'image existant. Pendant environ deux semaines j'ai effectué des recherches sur internet pour trouver des publications scientifiques afin de réaliser mon exposé avant de le présenter devant le service MRI et quelques personnes du CNRS intéressées par le sujet.

A la suite de cet exposé, j'ai pu commencer a développer. Mon tuteur m'a alors proposé de commencer par un plus petit sujet afin de m'immerger totalement dans les logiciels utilisés et le traitement d'image. J'ai donc passé quatre jours avec celui-ci a développer un algorithme de base de segmentation d'image.

Une fois prêt, j'ai attaqué le sujet du stage. J'ai travaillé principalement seul, mais mon tuteur était toujours à proximité si j'avais des questions ou des problèmes de développement. Toutes les fins d'après midi, il venait voir mon travail et me corriger quelques erreurs. J'en profiter pour lui poser des questions et lui exposer ce que je planifiai de faire le lendemain et le reste de la semaine.

Mais j'ai rencontré plusieurs problèmes tout au long du développement. Tout d'abord le bruit. Il n'est pas évident de travailler dans une salle d'analyse avec des scientifiques qui rentrent et qui sortent toute la journée pour effectuer leurs calculs sur ordinateur.

Un deuxième point difficile fut l'absence de mon tuteur. Celui-ci m'avait prévenu dès le début du stage et même lors de mon entretien qu'il serait beaucoup absent. D'une part avec les vacances et d'autre part pour donner des congrès et s'occuper de sa promotion au sein de MRI. J'ai donc du prendre des initiatives concernant le développement du logiciel. Mais je suis resté bloqué sur certains points et plutôt que de perdre 1 semaine à chercher et attendre le retour de mon tuteur, j'ai pris rendez-vous avec le secteur informatique de l'IGMM (bâtiment en face du CRBM) afin qu'il m'explique et me donne des idées pour continuer. J'ai aussi beaucoup travaillé avec des forums spécialisés dans la programmation et le traitement d'image où je posais de multiples questions.

Ce rapport de stage a pour but de présenter mon travail au sein de la société MRI dans le cadre de mon stage. A cette fin, ce document exposera :

- Une présentation de l'entreprise.
- La mission.
- La présentation du logiciel sur lequel j'ai travaillé.
- Une explication détaillée de la segmentation
- Le premier algorithme réalisé : « région growing », le principe, l'analyse, le développement et le résultat obtenu.
- Le logiciel de développement utilisé ainsi que les différents plug-in
- Le choix de l'algorithme pour le développement du sujet de stage.
- Le deuxième algorithme réalisé : « Split and merge», le principe, l'analyse, le développement, les problèmes rencontrés et le résultat obtenu.
- L'ajout de nouvelles fonctionnalités dans le logiciel.
- une discussion qui relate le déroulement du stage et les difficultés rencontrées.
- une conclusion sur les bénéfices qui ont pu être retirés de ce stage pour l'entreprise ainsi que pour moi-même.

2 Présentation de la société

A La plate forme :

Montpellier RIO Imaging est la plate-forme régionale d'imagerie du Languedoc-Roussillon. Elle est dédiée au soutien de la production de données, de résultats et de connaissances nécessitant des moyens optiques (microscopie et cytométrie). Elle est ouverte à toutes les entités publiques ou privées, sans limitation thématiques ou géographiques.

Elle est distribuée sur 7 sites avec 11 ingénieurs qui environnent 28 stations d'acquisition (microscopes et cytomètres) et 11 stations d'analyse. Elle dispose d'un domaine informatique (mri.cnrs.fr).

En 2006, elle a accueilli 308 utilisateurs (issus de 49 entités clientes) qui ont utilisé 29 121 heures de temps. Elle a également formé 181 stagiaires. MRI dispose d'un outil de comptage et de facturation « temps réel » des temps d'utilisation de son parc machine ainsi que d'un système d'administration informatique de ses comptes utilisateurs.

MRI est aussi plate-forme d'imagerie du Cancêropole Grand Sud Ouest de Montpellier, de Montpellier LR Genopole et du Neuropole Grand Sud en cours de formation.

B Signification du mot RIO :

La politique de concertation entre le CNRS et ses partenaires s'est renforcée dans le cadre de la Réunion Inter Organismes ou RIO, véritable instance de réflexion pour les Sciences du Vivant qui réunit l'INSERM, l'INRA, le CEA et le département des Sciences du Vivant du CNRS.

RIO définit notamment les modalités de soutien aux plates-formes technologiques en Sciences du Vivant, outils indispensables à l'accompagnement des changements d'échelle de la biologie.

C Les partenaires :

MRI disposent de nombreux partenaires :

D L'équipe :

L'équipe est composée de six personnes : un responsable, un développement (mon tuteur), un administrateur réseau et trois biologistes.

Un responsable : **Pierre Travo**

C. Tran-Aupiais	**Olivier Miquel**	**Sylvain de Rossi**	**Volker Bäcker**	**Julien Cau**
				
Optronique (Optique adaptative)	Systèmes réseaux	Optique et systèmes LASER	Conception et développement (MRI Cell Image Analyzer)	Toutes opérations de formation et communication

Durant tout mon stage j'ai travaillé avec mon tuteur (Volker Bäcker) et quelque fois lorsque celui-ci était absent, avec Olivier Miquel. Le reste de l'équipe étant des chercheurs ou biologistes je n'ai eu a faire avec eux concernant mon travail. En revanche, les relations avec les chercheurs et les scientifiques m'ont permis de trouver des contacts pour mon prochain stage de 6 mois de pré-embauche.

E Les mission :

Promouvoir l'utilisation de la microscopie :

En 2006, MRI a permis à ses 308 utilisateurs de tout horizon (CNRS, INSERN, INRA, CIRAD, IRD, Université de Montpellier 1 et 2, CHU de Montpellier, Université de Perpignan etc.) 29121 heures d'utilisation de ses microscopes pour une production de plus de 3 millions d'images.

Participer au développement de la microscopie :

En 2006, MRI conduit deux projets de Recherche/Développement :

- Conception du logiciel d'analyse MRI Cell Analyzer, sur lequel j'ai travaillé.
- Optique adaptative et cultures en trois dimensions.

Former les utilisateurs à la microscopie :

MRI a assuré en 2006, des ateliers et des stages de formations (95 personnes), et retransmet en association avec le rectorat de Montpellier, des expériences de TP avec le lycée Paul Valéry de Sète, par vision conférence.

F Le parc matériel :

MRI est équipé de 5M€ de parc matériel. 23 microscopes, 10 stations d'analyse, une interface en ligne de déconvolution et 4 stations dédiées. 150k€ de frais de maintenance en 2006 et 650 m² de locaux.

G Les sites :


4
sites
Jean-Luc Verdeil, C1 CIRAD
Responsable MRI-CIRAD
CIRAD (Agropolis)
Julien CAU, IR2 CNRS
Responsable MRI-CRBM
Centre de Recherche en Biochimie Macromoléculaire
Responsable adjoint MRI
Nicole Lautrédou, IR2 CNRS
& Chantal Cazevieille, IR2 UM1
Resp. MRI-IURC optique & éléctronique,
Institut Universitaire de Recherche Clinique
Hassan Boukhaddaoui,
IR2 INSERM
Responsable MRI-INM
Institut de Neurosciences de Montpellier

3 Mission

A Le sujet :

Le principal problème sur une image de cellules prise au microscope est la segmentation (séparation) et le comptage des cellules. En effet, lorsque deux ou plusieurs cellules se touchent, les algorithmes courants n'en comptent qu'une seule. Il faut donc trouver une technique permettant de les séparer.

Illustration 1: Segmentation

Cette image illustre très bien le problème, un algorithme de base ne détectera qu'une seule cellule, l'image du centre, alors que la solution optimale serait l'image de droite.

Il existe plusieurs algorithmes pour réaliser cela mais aucun n'est réellement efficace et efficient dans tous les cas. Ci dessous un simple seuillage des intensités ne permet pas de séparer les 2 cellules :

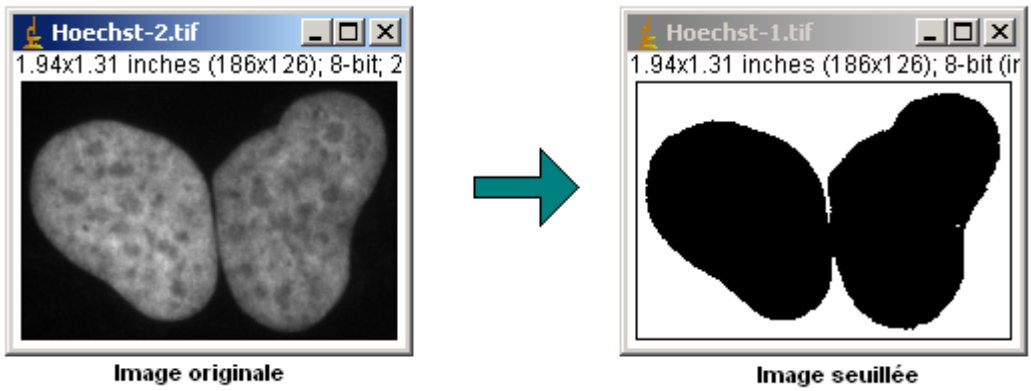


Illustration 2: Seuillage d'intensité d'une cellule

Ce type d'algorithme peut paraître relativement simple mais est en réalité assez complexe car il faut trouver les bonnes combinaisons de conditions (critères d'intensité, de gradient, etc.) pour arriver au meilleur résultat possible. Sachant qu'une bonne combinaison ne sera valable que pour un type d'image et de cellule et pas forcément pour les autres. Par exemple sur une image comme ci-dessous, la segmentation est assez difficile à réaliser car les cellules sont de petites tailles et elles se touchent beaucoup.

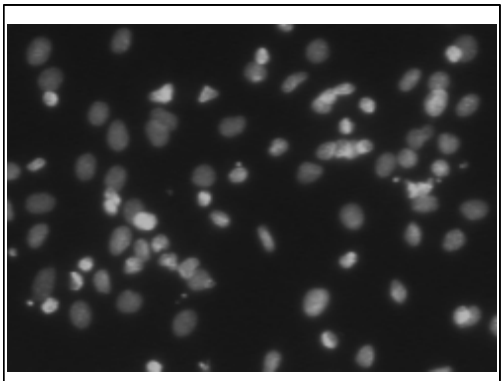


Illustration 3: Image de cellules

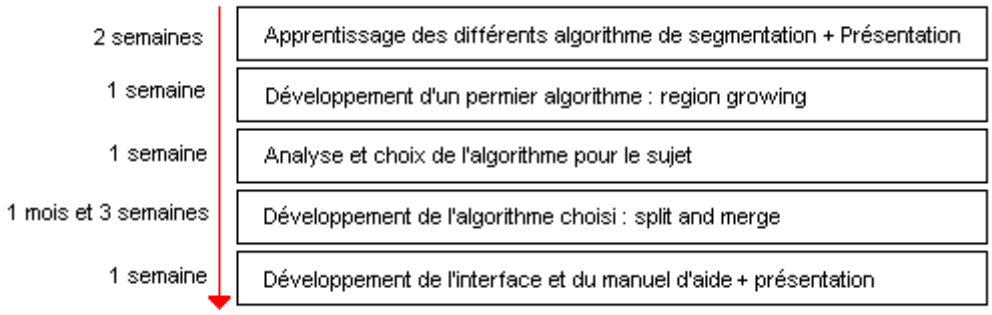
MRI et plus précisément pour tuteur travaille actuellement sur le projet MRI Cell Analyzer qui est un programme d'analyse et de traitement d'image basé sur ImageJ (expliqué après). La mission qui m'a alors été proposée est le développement d'un plug-in (évolutif) pour le logiciel ImageJ permettant de séparer au mieux ces amas de cellules.

Mon travail sera la base de nombreux plug-in que mon tuteur développera par la suite. Il faut donc que mon programme soit bien commenté (en Anglais), facile à comprendre et à maintenir. C'était la contrainte principale de mon sujet.

B Le planning

Concernant le planning des taches, il a été difficile d'en mettre un en place. N'ayant aucune connaissance en traitement d'image,j'ai eu du mal à estimer le temps nécessaire à chaque phase du développement. Le planning s'est affiné au fur et à mesure de mon avancé du projet.

Ci-dessous, le planning prévisionnel de mon stage :



Durant mon stage, je me suis efforcé de respecter au mieux ce planning même si je n'ai pas eu le temps de présenter le plug-in obtenu à la fin.

J'ai donc travaillé sous ImageJ, un logiciel libre développé en Java que je vais maintenant présenter.

4 Le logiciel existant : ImageJ

A Présentation :

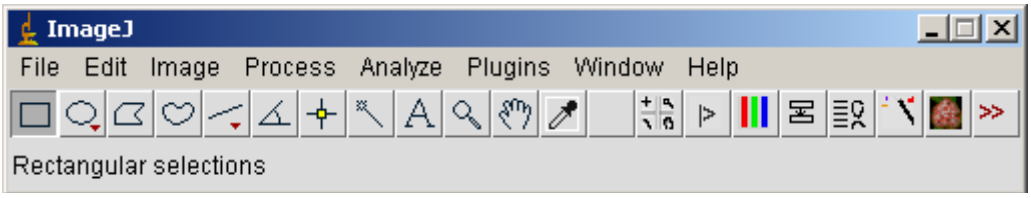


Illustration 4: Interface du logiciel ImageJ

ImageJ est un logiciel de traitement et d'analyse d'images. Le J indique que le programme a été écrit en Java, ce qui en fait un logiciel utilisable sur différents systèmes d'exploitation. Il se présente sous la forme d'une barre de menu qui ouvre des fenêtres de données.

La plupart des opérations courantes de traitement d'images sont réalisables avec ImageJ : visualisation et ajustement de l'histogramme des niveaux de gris, débruitage, correction d'éclairage, détection de contour, transformation de Fourier, seuillage, opérations logiques et arithmétiques entre images...

Des traitements issus de la morphologie mathématique sont aussi disponibles : érosion/dilatation, ligne de partage des eaux, squelettisation...

L'ajout personnalisé de fonction est possible grâce aux plugins à écrire en java.

B Caractéristiques

ImageJ est un logiciel libre : le code source est en accès libre et peut être modifié.

C'est un logiciel multiplateformes, en raison de son fonctionnement sur une machine virtuelle Java (Ce point sera expliqué plus tard). Ses fonctions sont extensibles, de nombreux plug-in existent, qui abordent des domaines jusque là réservés aux logiciels commerciaux.

Par ailleurs, il est possible de combiner les fonctions natives ou ajoutées en créant des macros. La foisonnante diversité des plug-ins disponibles (plus d'une centaine) en fait son avantage principal, mais il est parfois difficile de trouver rapidement une fonction correspondant à un besoin précis.

C Pour info :

ImageJ est un logiciel de traitement d'image assez complexe. Il est constitué de :

- 235 classes
- 3499 fichiers

Lors de mon stage j'ai travaillé sur la version de Montpellier RIO Imaging :

- 426 classes
- 4620 fichiers

Le site officiel du logiciel : http://rsb.info.nih.gov/ij/

5 La segmentation

A Qu'est ce que la segmentation ?

Dès le début de mon stage, n'ayant aucune connaissance en traitement d'image et en microscopie, mon tuteur m'a donné pour mission de réaliser une présentation de quarante-cinq minutes en Anglais sur les techniques de segmentation existante. Après dix jours passés à faire des recherches sur Internet et dans les livres je l'ai présenté devant les membres de l'équipe de MRI ainsi que d'autres personnes du CNRS intéressées par le sujet.

J'ai beaucoup travaillé avec les publications IEEE (site regroupant l'édition et la publication de revues scientifiques). Mon tuteur m'ayant donné accès aux différentes bibliothèques virtuelles du CNRS.

Cet exposé est maintenant disponible sur le site de MRI CNRS à l'adresse suivante :

- http://www.mri.cnrs.fr/mriwiki/uploads/image_segmentation_techniques.pdf

Mais avant d'aller plus loin, il me faut définir exactement ce qu'est la segmentation en traitement d'image. La segmentation est un processus qui consiste à découper une image en régions connexes présentant une homogénéité selon un certain critère, comme par exemple la couleur (l'union de ces régions doit redonner l'image initiale). L'image ci-dessous illustre très bien le principe de la segmentation :

La segmentation est une étape importante pour l'extraction des informations de l'image. Elle fournit une description de haut niveau : chaque région est connectée à ses voisines dans un graphe et chaque région porte une étiquette donnant des informations comme sa taille, sa couleur, sa forme, son orientation. L'image se réduit donc à un graphe composé de noeuds étiquetés, qui contient presque toute l'information utile au système. Les arcs du graphe peuvent être valués précisant si les deux régions connectées sont en simple contact ou si l'une est incluse dans l'autre. D'autres informations peuvent également être stockées comme par exemple le fait qu'une région est au dessus d'une autre. Selon les techniques de segmentation utilisées, la construction de ce graphe peut être plus ou moins complexe.

B Les catégories d'algorithmes :

On regroupe généralement les algorithmes de segmentation en trois grandes classes :

- Segmentation basée sur une approche globale de l'image
- Segmentation basée sur les régions
- Segmentation basée sur les contours

La première catégorie travaille sur des histogrammes de l'image comme par exemple la moyenne des niveaux de gris de l'ensemble des pixels ou la médiane.

On part ici d'un rapport qu'entretient chaque pixel avec des informations calculées sur toute l'image permettant de segmenter l'image en régions. Ces informations permettent de construire des classes de pixels, les pixels appartenant à une même classe et étant connexes forment des régions.

La deuxième catégorie correspond aux algorithmes d'accroissement ou de découpage de région. Les algorithmes par croissance de régions partent d'un premier ensemble de régions, qui peuvent être calculées automatiquement (par exemple, les minima de l'image), ou fournies par un utilisateur de manière interactive. Les régions grandissent ensuite par incorporation des pixels les plus similaires suivant un critère

donné, tel que la différence entre le niveau de gris du pixel considéré et le niveau de gris moyen de la région. Les algorithmes de type décomposition/fusion exploitent les caractéristiques propres de chaque région (surface, intensité lumineuse, colorimétrie, texture, etc.). On cherche des couples de régions candidates à une fusion et on les note en fonction de l'impact que cette fusion aurait sur l'apparence générale de l'image. On fusionne alors les couples de régions les mieux notés, et on réitère jusqu'à ce que les caractéristiques de l'image remplissent une condition prédéfinie.

La troisième catégorie s'intéresse aux contours des objets dans l'image. La plupart de ces algorithmes sont locaux, c'est à dire fonctionnent au niveau du pixel. Des filtres détecteurs de contours sont appliqués à l'image. Le résultat est en général difficile à exploiter sauf pour des images très contrastées. Les contours extraits sont la plupart du temps morcelés et peu précis et il faut utiliser des techniques de reconstruction de contours par interpolation ou connaître a priori la forme de l'objet recherché. Ce type d'algorithme est proche des techniques d'accroissement de région fonctionnant au niveau du pixel. Ces techniques purement locales sont en général trop limitées pour traiter des images bruitées et complexes.

Les algorithmes de segmentation les plus couramment utilisés sont :

- Simple seuillage
- Détection des contours
- Clustering method
- Contours actifs (snakes)
- Accroissement de région
- Split and merge
- Ligne de partage des eaux

Aucune de ces techniques ne marchent sur tous les types d'images et chaque algorithme possède ses avantages et ses inconvénients. C'est pourquoi il m'a fallu faire des recherches afin de mettre en place une technique qui selon moi, serait la plus adaptée à mon sujet.

6 Le choix de l'algorithme :

Suite à mes recherches et à la présentation que j'ai donné, j'ai décidé avec l'accord de mon tuteur, de développer un algorithme déjà existant (qui n'est pas implémenté dans ImageJ) et de l'améliorer. Créer et développer un algorithme étant trop difficile à réaliser dans le temps imparti et avec mon niveau en traitement d'image.

Les algorithmes utilisés par ImageJ sont dit « globaux ». C'est à dire qu'ils travaillent au niveau de l'image et donc perdent en précision. Le but est donc de développer et d'améliorer un algorithme « local » qui travaille au niveau du pixel.

Cet algorithme que j'ai choisi de programmer est le « split and merge » (découpage et fusion). Mais n'ayant encore une fois jamais développé de logiciel de traitement d'image, il m'a fallu commencer avec un plus petit sujet pour « me faire la main ».

Mon tuteur m'a alors proposé de développer une baguette magique de sélection similaire à celle de photoshop. Celle du logiciel imageJ n'étant pas très pratique à utiliser et assez limité.

J'ai donc utilisé un algorithme de segmentation du type « accroissement de région » avec mon tuteur, qui m'a ainsi expliqué les bases du traitement d'image en Java et le fonctionnement de ImageJ. S'immerger dans un programme déjà développé depuis quatre, cinq ans par mon tuteur n'a pas été une chose facile. Je me suis beaucoup aidé de la documentation d'ImageJ sur le site officiel pour éviter de redévelopper des méthodes déjà implémentées dans le logiciel, ou pour trouver des fonctionnalités à utiliser, ainsi que des forums spécialisés sur Internet.

7 Prise en main : Région growing :

Les méthodes d'accroissement de région sont les méthodes de segmentation les plus simples. Le principe : l'algorithme part de petits éléments de l'image qu'il va tenter de regrouper en éléments plus importants. Je vais présenter ici la version de base de l'algorithme d'accroissement de région qui fonctionne en agrégeant des pixels.

A Principe de fonctionnement

Supposons une région de couleur homogène initialement composée de 1 pixel.

On va étendre cette région en incluant les pixels situés sur la frontière et dont la couleur est proche (la variation de couleur est inférieure à un seuil). En répétant cette procédure jusqu'à ce qu'il n'y ai plus de pixel de couleur assez proche sur la frontière, on obtient une région de couleur homogène maximale autour du pixel de départ.

Il y a 2 méthodes possibles. Soit en testant les 4 pixels voisins, soit les 8. Les résultats obtenus pouvant être totalement différent dans les 2 cas.

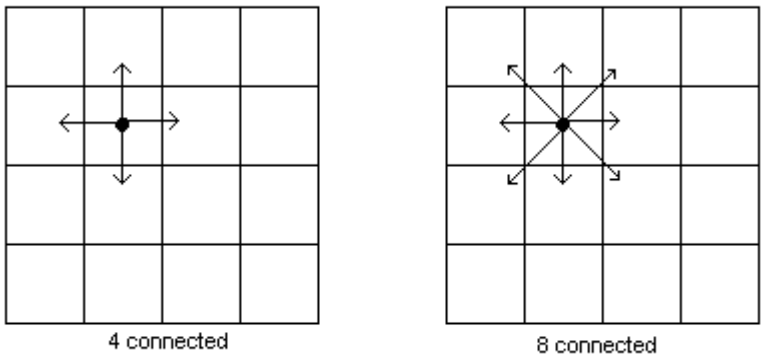


Illustration 5: Méthode de lecture des voisins

Cette méthode présente deux limitations sévères qui n'en font pas une méthode très efficace :

- Les régions obtenues dépendent fortement des pixels d'amorçage choisis et de l'ordre dans lequel les pixels de la frontière sont examinés.
- Le résultat final est très sensible à la valeur du seuil.

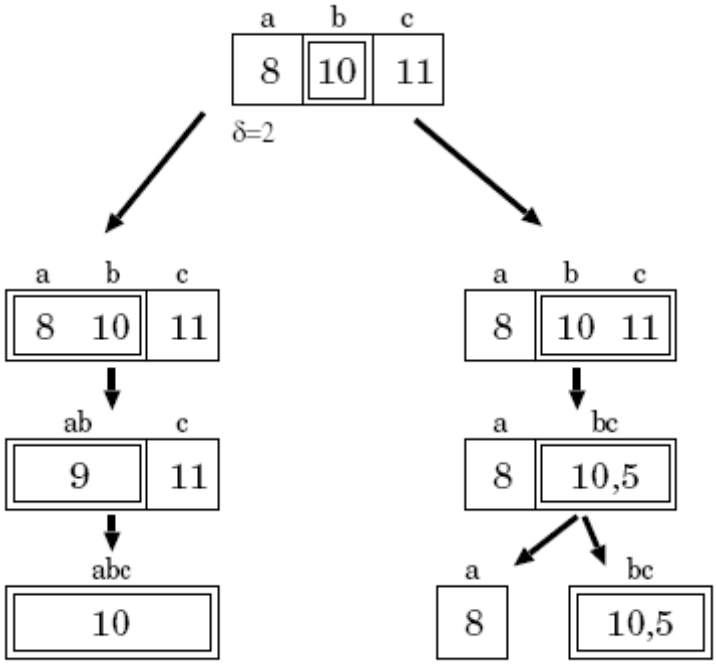


Illustration 6: Un des inconvénients de l'accroissement de région

Pour illustrer le premier problème, considérons trois pixels adjacents **a**, **b** et **c** dont les intensités respectives sont 8,10 et 11. Le seuil de l'intensité est fixé à 2.

La région initiale est constituée du pixel **b**. Deux schémas de regroupement pour les points frontière **a** et **c** sont possibles :

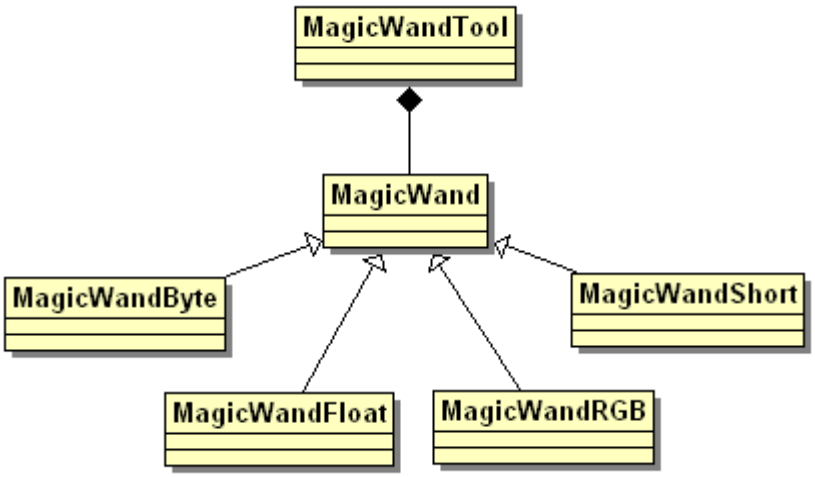
- Le pixel central **b** est l'amorce. Compte tenu du seuil = 2, **a** et **c** sur la frontière devraient être ajoutés à l'amorce **b**. Si l'on commence par tenter d'agréger **a**, le résultat du regroupement, noté [**ab**], a pour intensité moyenne 9 et **c** s'y ajoute ensuite puisque |11 – 9| < seuil . On a donc regroupé **a**, **b** et **c**.
- Si maintenant l'algorithme commence par examiner le point frontière **c**, le groupement de **b** et **c** donne [**bc**] dont l'intensité est 10,5. Le point frontière **a** d'intensité 8 est trop éloigné et il est considéré comme appartenant à une autre région. On obtient donc deux groupements au lieu d'un.

Ce petit exemple illustre combien l'ordre d'examen des points sur la frontière peu influencer sur le résultat de l'algorithme.

B Analyse

Pour développer cet algorithme j'ai tout d'abord commencer par l'analyse UML. Grâce aux multiples publications scientifiques traitant du sujet, il n'a pas été trop difficile de le réaliser.

Diagramme de classe :



Les calculs d'intensité étant différents selon le type d'image, un polymorphisme s'est donc imposé. En effet, la méthode de calcul est différente selon que l'image est de type RGB, Float, Short ou Byte. La classe MagicWand regroupe l'algorithme de calcul d'agrégation des pixels alors que la classe MagicWandTool est celle permettant de gérer le plug-in et son utilisation dans le logiciel ImageJ.

C Développement :

Le squelette principal de l'algorithme se présente de la manière suivante :

```
public Roi createRoi(int x, int y) {
	// ...
	queue.add(firstPoint); // on ajoute le premier point dans la queue
	// ...
	while(!queue.isEmpty()) { // tant que la queue n'est pas vide
		// ...
		Point currentPoint = (Point) queue.remove(0); // on prend le 1er point de la queue
		if(currentPoint.x>=width || currentPoint.x<0 || currentPoint.y>=height || currentPoint.y<0) continue;
		// ...
		float currentIntensity = this.getPixel(currentPoint.x, currentPoint.y);
		// ...
		if(currentIntensity <= (average+threshold) && currentIntensity >= (average-threshold)) { // Si son intensité est comprise dans le seuil
			queue.addAll(this.getNeighbors(currentPoint)); // On ajoute tous ses voisins dans la queue
			totalIntensity += currentIntensity;
			average = totalIntensity / numberOfPoints; // on remet à jour l'intensité
		}
		// ...
	}
	// ...
	return roi;
}
```

D PLUGIN-IN OBTENU :

J'ai rajouté dans l'interface d'ImageJ, une icône permettant de sélectionner le nouvel outil de sélection. Il suffit ensuite à l'utilisateur de cliquer sur l'image pour effectuer une sélection.

De base, le seuil est réglé sur 50, mais il est possible en double cliquant sur l'icône d'ouvrir une petite interface permettant de régler le seuil.

Illustration 7: Sélection du seuil

Les méthodes booléennes d'addition et de soustraction sont aussi possible en utilisant respectivement les touches shift et ctrl. Ainsi, il est possible de faire de la multi-sélection d'objet sur une même image.

La condition utilisée pour l'agrégation d'un pixel est la suivante :

```
if(currentIntensity <= (average+threshold) && currentIntensity >= (average-threshold))
```

Si l'intensité du point que l'on veut ajouté est comprise entre : l'intensité moyenne de la zone moins le seuil et l'intensité moyenne de la zone plus le seuil, alors on ajoute celui-ci. Dans le cas contraire, on le rejette et on le marque comme « déjà testé » pour éviter de repasser dessus.

Sur cette image, le fond (en noir) a été sélectionné. Avec un seuil de 50 pour l'intensité.

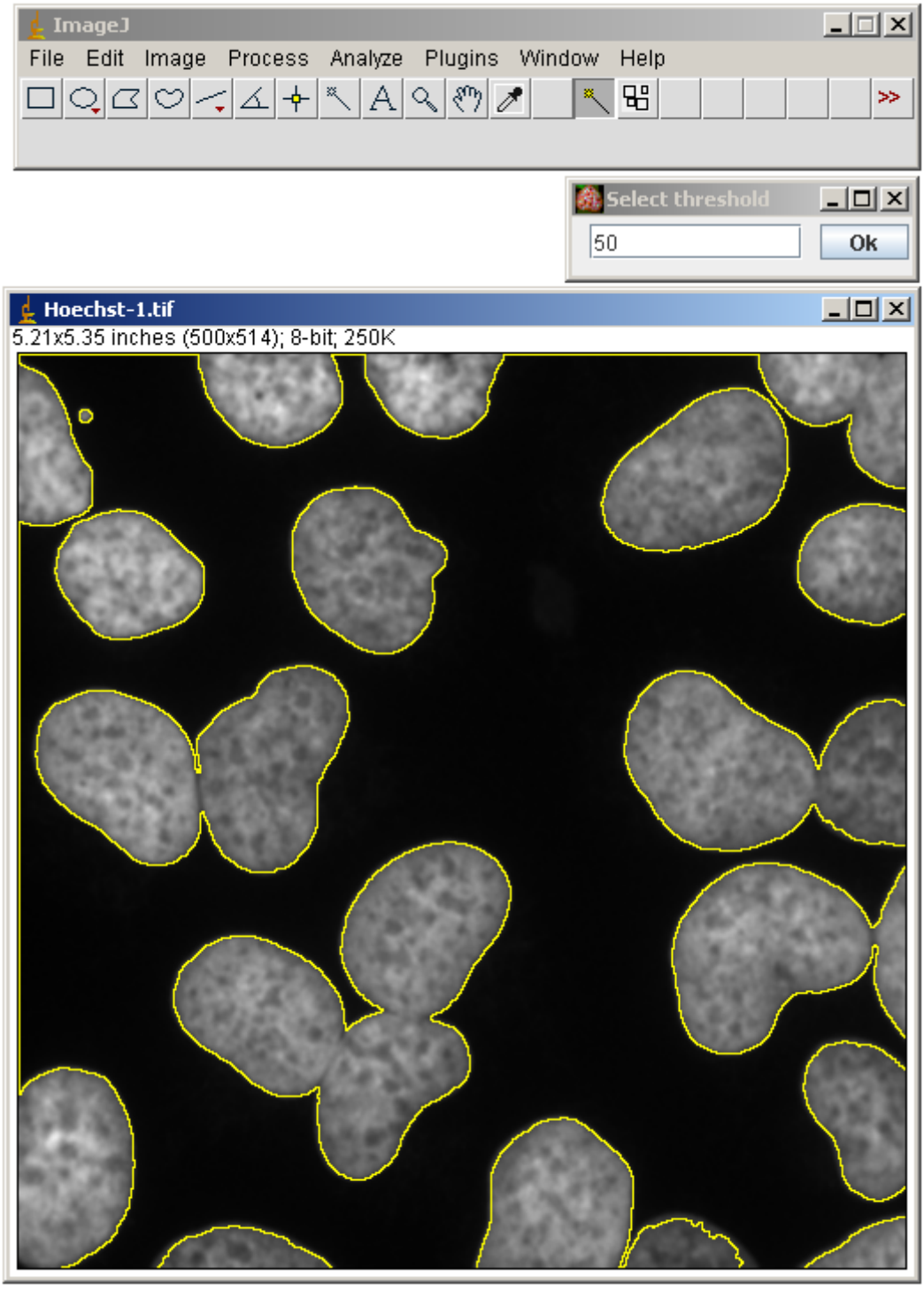


Illustration 8: Exemple d'utilisation de la baguette magique

8 Logiciel de développement utilisé : Eclipse

Pour mener à bien ce développement ainsi que le sujet principal du stage, j'ai travaillé sous Eclipse en utilisant des modules que sont Subversion et Junit.

Eclipse est un environnement de développement intégré (Integrated Development Environment) dont le but est de fournir une plate-forme modulaire pour permettre de réaliser des développements informatiques.

Fruit du travail d'un consortium de grandes entreprises (IBM, Borland, Rational Rose, HP...), il en résulte un IDE performant et openSource qui a su trouver sa place parmi les pointures du marché que sont JBuilder et NetBeans (utilisé à l'EPSI).

La majorité des modules d'Eclipse (plug-in) concernent Java mais des modules sont en cours de développement pour d'autres langages notamment C++, Cobol, mais aussi pour d'autres aspects du développement (base de données, conception UML, ...). Ils sont tous développés en java.

Les modules agissent sur des fichiers qui sont inclus dans l'espace de travail. L'espace de travail regroupe les projets qui contiennent une arborescence de fichiers.

Eclipse fonctionne sur les plate-formes Windows 98/NT/2000/XP et Linux.

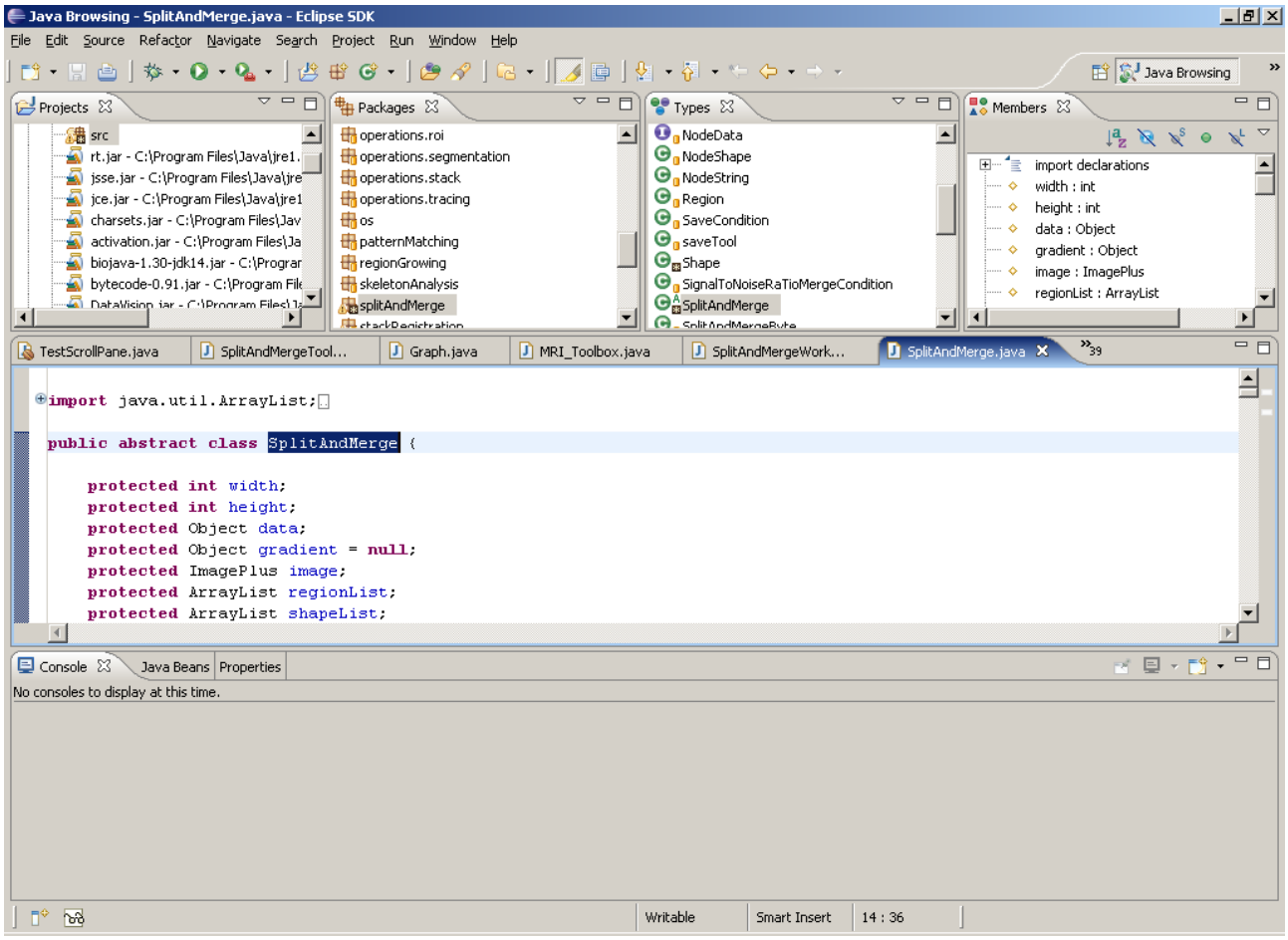


Illustration 9: Interface principale de Eclipse

Au niveau ergonomie, Eclipse n'a rien à envier à ses concurrents. Toutes les fonctionnalités indispensables sont là : création de projet, de template, refactoring, debuggage... Mais, la grande force de cet IDE réside dans l'ouverture de son noyau qui permet l'ajout de très nombreux plugins. Il est par exemple possible d'intégrer facilement un serveur d'application pour le debbugage, un module de déploiement J2EE, un explorateur de bases de données etc...

Eclipse étant un logiciel très complet et possédant de nombreux plug-in, j'ai utiliser Subversion et Junit pour mener à bien mon projet.

9 Les plug- in :

A Subversion (SVN)

Lorsque l'on gère de gros projets, il est difficile de s'y retrouver et de gérer les différentes versions des fichiers. Il faut donc un système de gestion de version.

SubVersion, que j'ai utilisé tout au long de mon stage, a permit de stocker les différentes versions du logiciels sur un serveur configuré à cet effet (serveur SVN). Ainsi il est possible de retrouver toutes les versions modifiées depuis le départ du développement, mais il est surtout possible de revenir à des versions précédentes du logiciel dans le cas d'un mauvais fonctionnement où d'une erreur. Ce module est très intéressant car il permet à chaque membre d'une équipe d'apporter ses améliorations au logiciel et ensuite de les mettre à jour sur le serveur sans perdre les anciennes versions. Bien entendu cette fonctionnalité d'éclipse est d'autant plus intéressante que le projet est complexe et le nombre de personnes concernées important.

Subversion a beaucoup participé à la méthode de travail. En effet, pour chaque modification importante du logiciel correspond un envoi de cette version sur le serveur SVN. Ce serveur comporte un répertoire par projet et chaque projet contient les dossiers trunks, tags et branches.

- branches = versions en développement.
- tags = versions gelées (= releases).
- trunks = branches principales de développement.

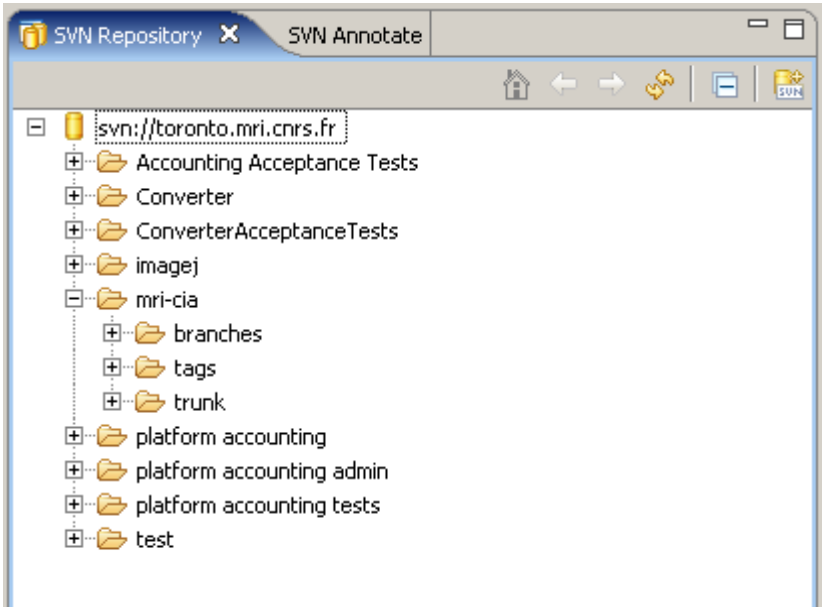


Illustration 10: SVN repository

Ainsi nous avons à disposition toutes les versions au fur et à mesures de l'avancement du projet. Les versions les plus importantes que sont les Releases étant toutes conservées dans le dossier tags.

Les principales caractéristiques de subversion sont :

- La consultation et la possibilité de restauration des anciennes versions d'un fichier.
- Les raisons des modifications apportées et les auteurs de celles-ci.
- Les modifications de version sont stockées sous forme de delta (différence entre deux versions) pour les fichiers texte. Ce qui implique qu'une propagation est proportionnelle aux modifications apportées et non à la taille des données initiales.
- Subversion permet de tracer les versions de répertoires, de fichiers et de droits sur les fichiers.
- Subversion permet de renommer un fichier ou un répertoire tout en conservant son historique.

B Eclipse Junit

Pour mener à bien le projet, j'ai réaliser des procédures de test tout au long du développement. Pour réaliser cela j'ai utilisé un autre plug-in d'Eclipse : Junit.

Junit est un framework (un ensemble de bibliothèques permettant le développement rapide d'application) open source pour réaliser des tests unitaires sur du code Java. Le principal intérêt est de s'assurer que le code répond toujours au besoin même après d'éventuelles modifications.

Le but est d'automatiser les tests. Ceux ci sont exprimés dans des classes sous la forme de test avec leurs résultats attendus. JUnit exécute ces tests et les comparent avec ses résultats.

Avec Junit, l'unité de test est une classe dédiée qui regroupe l'ensemble des tests. Ces cas de test exécutent les tâches suivantes :

- création d'une instance de la classe et de tout autre objet nécessaire aux tests
- appel de la méthode à tester avec les paramètres du test
- comparaison du résultat obtenu avec le résultat attendu : en cas d'échec, une exception est levée

Si tous les tests ont été exécutés avec succès, une ligne verte est affichée.

Cette vue contient deux onglets :

- « Failures » : contient la liste des tests qui ont échoués
- « Hierarchy » : contient une arborescence des tests

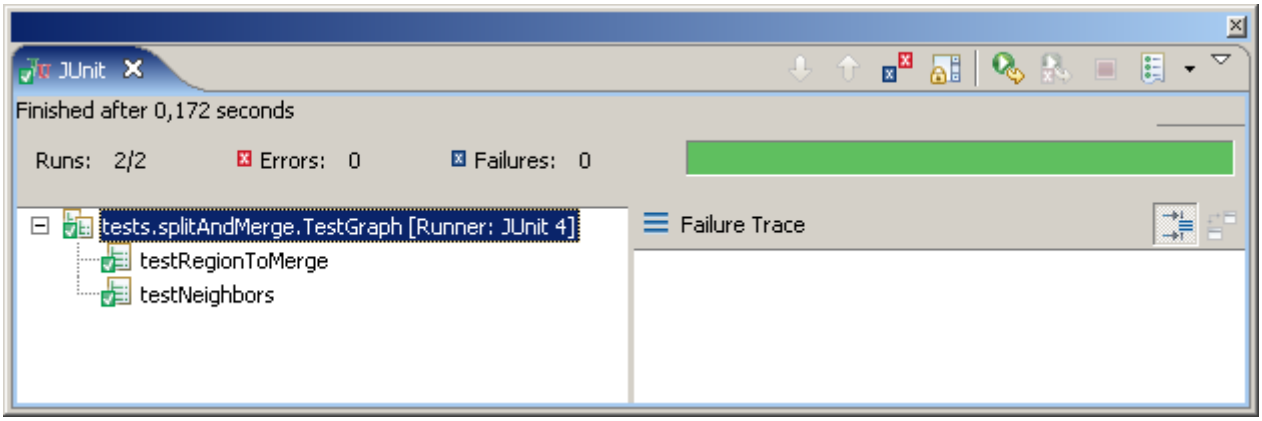


Illustration 11: Résultat JUnit positif

Dans le cas où un ou plusieurs tests échouent, la ligne est rouge et les « failures » sont affichés afin de pouvoir trouver la source de l'erreur et la corriger.

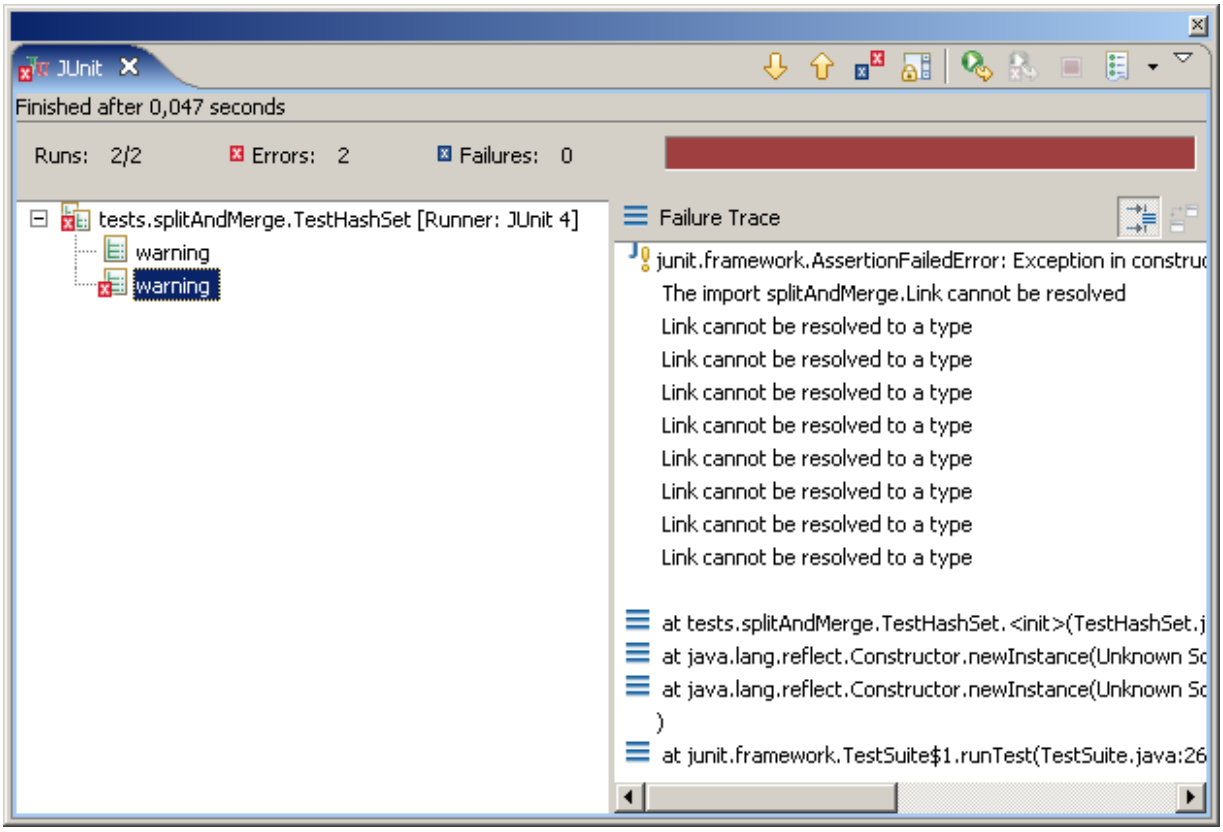


Illustration 12: Résultat JUnit négatif

Ces tests ont bien sur leurs limites puisqu'ils ne renvoient que le résultat de ce dont le développeur veut prouver, et pas la source de l'erreur. Cela n'exclut donc pas quelques problèmes en cours de développement mais l'expérience dans ce domaine rend les tests plus pertinents.

10 Le langage utilisé : Java

ImageJ étant développé en Java, il est évident que le choix du langage n'a pas été difficile.

Java est un langage de développement créé par Sun qui a obtenu une très grande notoriété ces dernières années, notamment grâce à sa portabilité et son affinité avec internet. De plus c'est un langage libre et donc gratuit qui permet de développer des applications complexes.

Sa portabilité, sa robustesse et la richesse de ses API fournies en standard, par des tiers commerciaux ou libres sont ses principales qualités. Son plus gros défaut est sa relative lenteur d'exécution des programmes mais ce facteur s'améliore au fil des versions des machines virtuelles.

L'avantage le plus significatif pour un logiciel Java est probablement sa portabilité, puisqu'il peut fonctionner sur n'importe quelle machine disposant d'une Java Virtual Machine (JVM). Cette portabilité est fondamentale sur Internet ou même en réseau où un nombre important de machines et systèmes d'exploitation sont différents.

Dans un environnement, le code Java n'est conçu pour aucune plate-forme spécifique mais est une sorte de code intermédiaire qui doit être traduit en code natif machine par un programme particulier faisant office d'interprète : Java Virtual Machine. Java est réellement indépendant de toute plate forme contrairement à un langage C ou C++ qui nécessite une recompilation.

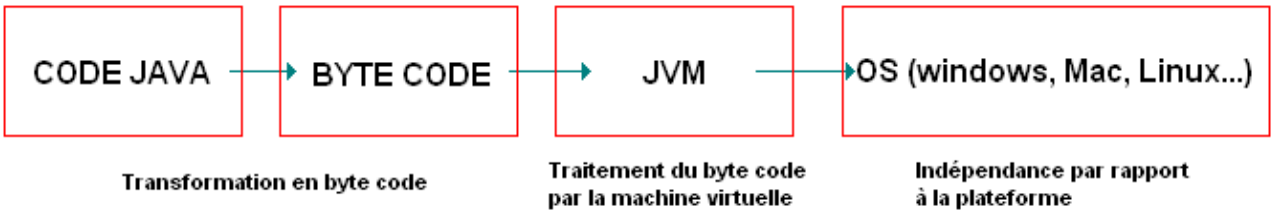


Illustration 13: Fonctionnement de la JVM Java

Qui plus est Java gère lui-même les accès mémoire, ce qui évite les fuites de mémoire et augmente considérablement la sécurité en limitant les failles .

11 L'algorithme choisi : Split and merge

Une fois la baguette magique développée, j'ai pu entrer dans le vif du sujet et l'attaquer à proprement parlé.

Cet algorithme s'apparente dans son principe à l'algorithme d'accroissement de région que je viens de présenter. La différence principale provient de la nature des régions élémentaires agrégées. Dans l'algorithme split and merge, les régions agrégées proviennent d'une première phase de traitement de l'image qui construit de manière récursive des régions carrées de tailles variables mais homogènes.

A Principe de fonctionnement :

Première étape : Le découpage (split)

La méthode de découpage de l'image utilisée dans cet algorithme est basée sur la notion de *quadtree*. Cette structure de donnée est un arbre quaternaire qui permet de stocker l'image sur plusieurs niveaux. On part d'une région initiale qui est l'image tout entière. Si cette image vérifie un critère d'homogénéité (couleur, déviation standard...), l'algorithme s'arrête. Sinon, on découpe cette région en quatre parties de même taille et on lance la procédure récursivement dans ces quatre parties. La région initiale va être stockée comme un noeud dans un graphe et les sous parties comme les fils de ce noeud.

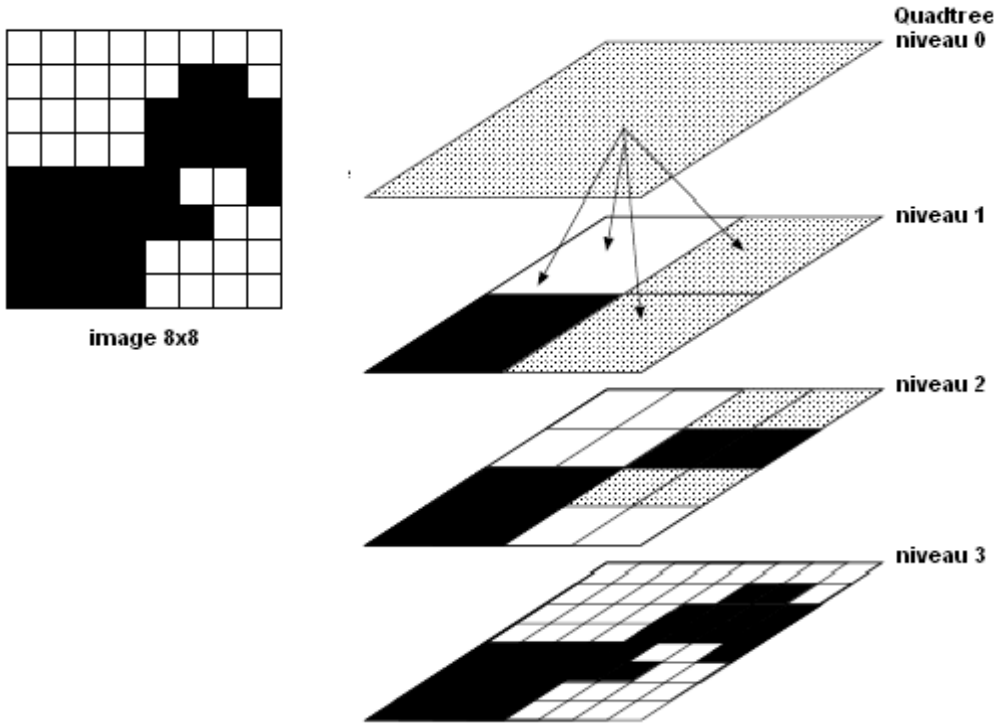


Illustration 14: Découpage d'une image en quadtree

Dans cet exemple, le critère d'homogénéité est absolu : une zone est dite homogène si elle ne contient que des pixels de même couleur (le seuil est de 100%). On peut être plus tolérant et accepter qu'une zone soit déclarée homogène dès que plus de 75% d'une couleur domine.

De manière plus générale, on va appliquer ce principe de réduction à des images colorées. Le critère d'homogénéité est fixé par un seuil sur la variance de la couleur dans la zone en cours d'examen. Au dessus de ce seuil, la zone est découpée en quatre, en dessous, elle est conservée et constitu un noeud terminal de l'arbre (une feuille). On lui attribue alors la couleur de la moyenne des pixels la constituant.

Deuxième étape : La fusion (merge)

La procédure de découpage décrite précédemment aboutit à un nombre de régions trop élevé. La cause fondamentale de cette sur-segmentation est que l'algorithme découpe les régions de manière arbitraire. Il se peut que l'algorithme coupe de cette façon une zone homogène en quatre parties.

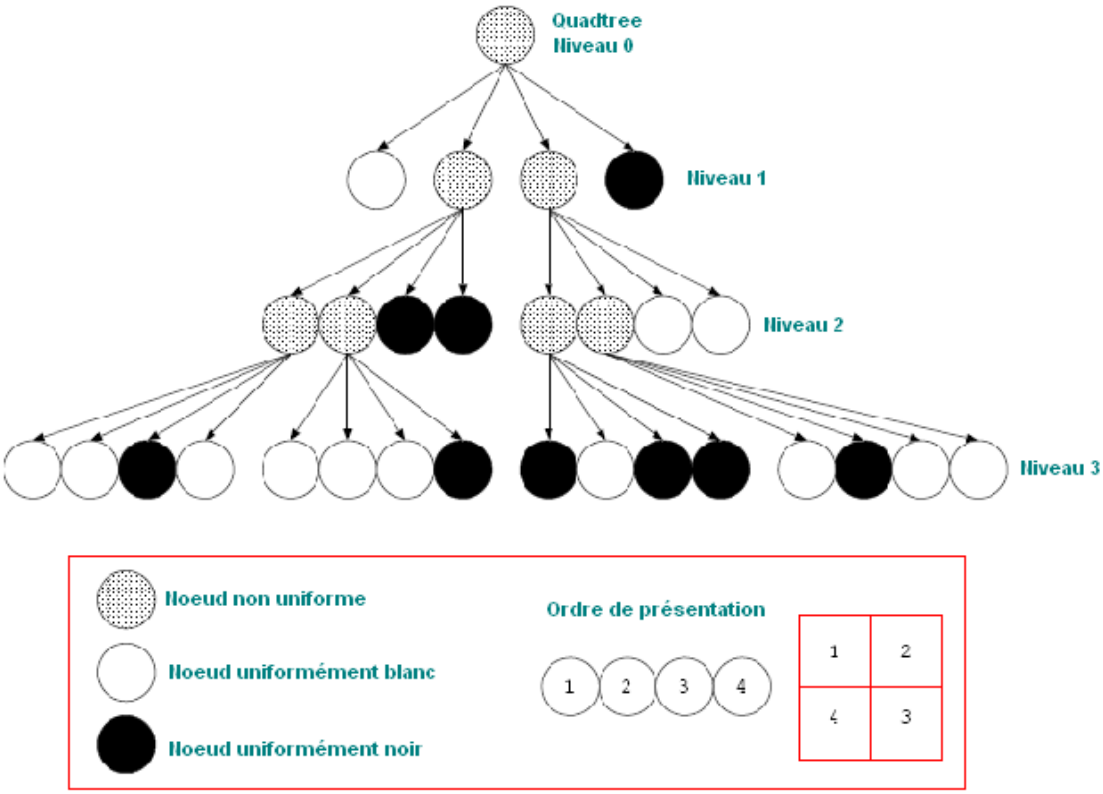


Illustration 15: Graphique des niveaux de quadtree

La solution, qui correspond à la phase « merge » de l'algorithme, est de procéder à une fusion de régions après le découpage. L'implémentation la plus simple de cette fusion cherche tous les couples de régions adjacentes dans l'arbre issu du découpage et cherche à les fusionner si elles respectent un ou plusieurs critères prédéfinis.

Pour réaliser cette fusion, il faut d'abord tenir à jour une liste des contacts entre régions. On obtient ainsi un graphe d'adjacence de régions ou *Region Adjacency Graph*.

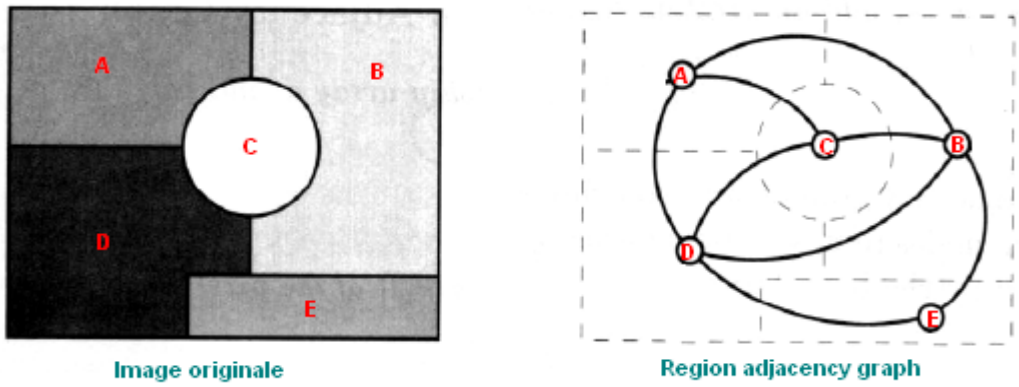


Illustration 16: Region Adjacency graph

- A a comme voisins : B C D
- B a comme voisins : A C D E
- C a comme voisins : A B D
- D a comme voisins : A C B E
- E a comme voisins : B D

Ensuite, l'algorithme va chercher deux régions, parmi la liste complète, qui respectent les critères et les fusionner. La première région absorbe la deuxième, et cette deuxième est supprimée de la liste. L' algorithme continue jusqu'à ce qu'il ne trouve plus de régions qui respectent les critères.

A chaque fois que des régions sont fusionnées, ses attributs (aire, intensité...) sont recalculés.

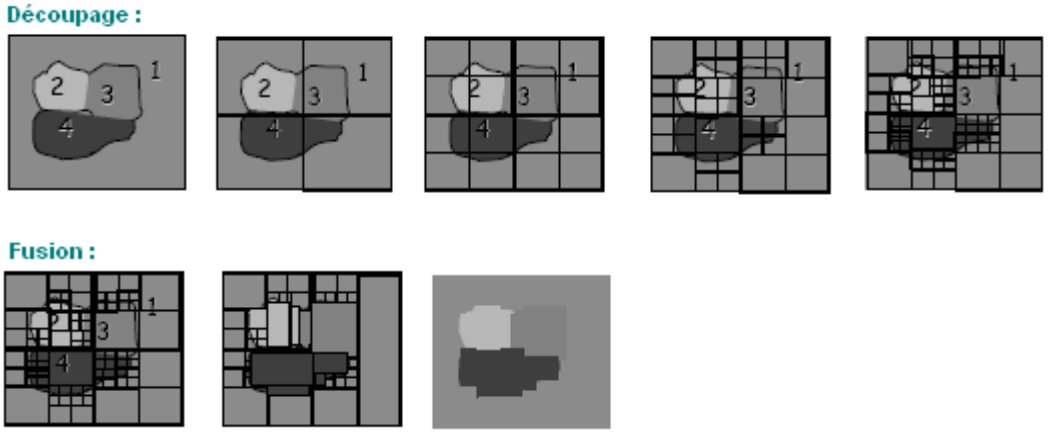
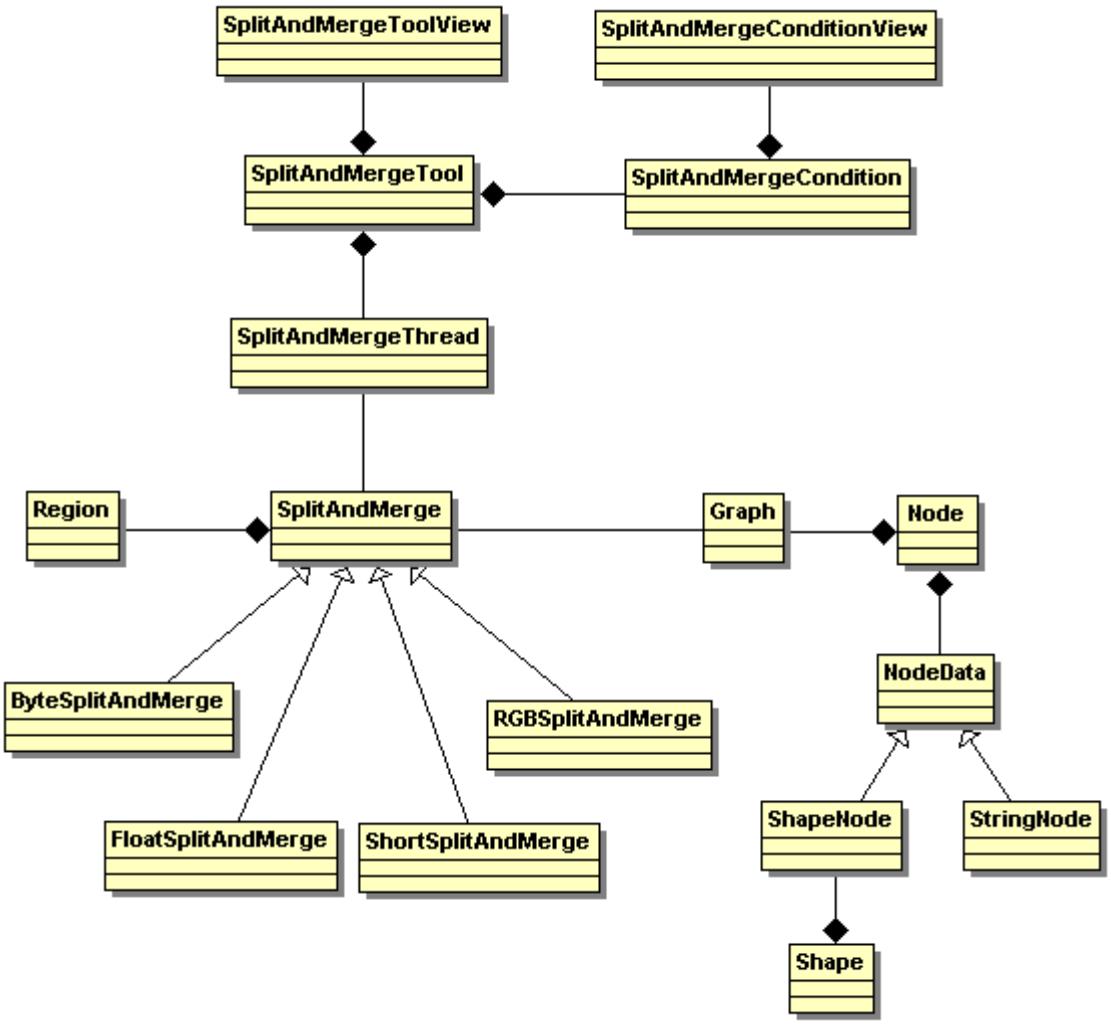


Illustration 17: Etapes du split and merge

B Analyse du sujet :

Le diagramme de classe de l'algorithme que j'ai développé se présente de la manière suivante :

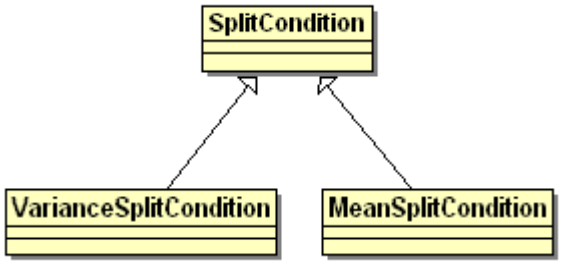


Dans mon analyse, les classes d'interface : SplitAndMergeConditionView et SplitAndMergeToolView sont complètement indépendantes du reste du programme. On peut l'utiliser sans passer par l'interface grâce aux 2 classes SplitAndMergeTool et SplitAndMergeCondition. La partie de droite du diagramme à partir de la classe Graph ont été utilisées pour faire une totale abstraction des données afin de réaliser plus facilement la partie Merge de l'algorithme.

La classe SplitAndMerge a été polymorphée afin de pouvoir traiter tous les différents types d'images : Byte, RGB, Float, Short.

Conditions de découpage :

Deux conditions ont été implémenté, la variance et la moyenne de l'intensité.

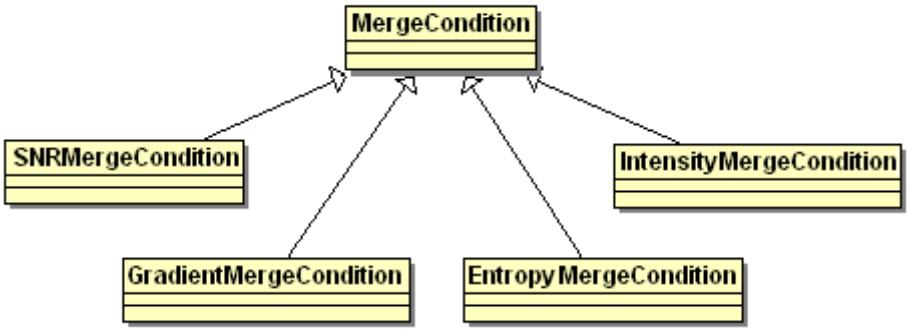


Conditions de fusion :

Comme pour les conditions de découpage, un polymorphisme s'est imposé pour pouvoir rajouter de nouvelles conditions. Ici plusieurs conditions de fusion :

- Intensité : si deux régions ont une intensité moyenne proche.
- Entropie : si deux régions ont une entropie moyenne proche (désordre).
- Gradient : si le gradient est élevé alors c'est qu'on est sur un bord ou à une séparation. A l'inverse si le gradient est faible, on fusionne. Le gradient correspond aux zones de fort changement d'intensité :
- Ratio signal bruit (SNR)

Illustration 18: Gradient sur une cellule



C Le développement :

Le développement du plug-in s'est étalé sur 2 mois environ, durant lequel j'ai pris beaucoup d'initiatives concernant les méthodes de programmation, mais toujours en vérifiant auprès de mon tuteur que mes choix étaient appropriés.

Tout au long de cette période de développement, j'ai réalisé des tests qui me permettait de vérifier l'intégrité de mes méthodes les unes avec les autres. A chaque nouvelle fonctionnalité, je lançais mes séquences de tests pour vérifier que la nouvelle fonction ou méthode ne générait pas d'erreur avec ce qui avait déjà été développé.

D Les problèmes rencontrés :

Les problèmes rencontrés tout au long du développement ont été multiple :

- Travailler dans un projet d'une telle taille. Le projet MRI Cell Analizer étant le projet de mon tuteur depuis plusieurs années, il a été difficile de comprendre tout le programme et de savoir réutiliser tout le code déjà développé.

- L'apprentissage de la méthode de segmentation « split and merge ». Il m'a fallut au moins 1 semaine de recherche pour bien tout structurer et commencer l'analyse du sujet. L'analyse et le traitement d'image étant un domaine complexe.

- Par la suite j'ai rencontré des problèmes lors de la construction de mon graphique d'adjacence et de la fusion des zones. J'ai alors rajouté la partie de droite du diagramme de classe avec les classes graph, node... pour faire totalement

abstraction des données et ne travailler qu'avec des noeuds sur un graphique. Ceci m'a permis de réaliser plus facilement la partie fusion de l'algorithme.

- Un autre problème a été l'absence de mon tuteur. Celui-ci m'avait prévenu des le début du stage et même lors de mon entretien qu'il serait beaucoup absent pendant la période de mon stage. D'une part avec les vacances et aussi pour donner des congrès et s'occuper de sa promotion au sein de MRI. J'ai donc du prendre des initiatives concernant le développement du logiciel. Mais je suis resté bloqué sur certain point et plutôt que de perdre une à deux semaines à chercher et attendre le retour de mon tuteur, j'ai pris rendez-vous avec le secteur informatique de l'IGMM (bâtiment en face du CRBM) afin qu'il m'explique et me donne des idées pour continuer. J'ai aussi beaucoup travaillé avec des forums spécialisés dans la programmation et le traitement d'image ou je posais de multiples questions.

- Le choix des conditions m'a aussi posé un réel problème. En effet, trouver des conditions n'était pas une chose aisée. Des calculs d'entropie, de ratio signal/bruit, de gradient, de variance etc. étant un domaine totalement inconnu pour moi. J'ai donc lu plusieurs publications sur internet et avec l'aide de mon tuteur j'ai pu trouver les méthodes de calculs de ces conditions.

- Lorsque l’on développe, on néglige bien souvent l’intérêt des tests. Il est vrai que l’on n’a pas l’habitude de tester toutes fonctionnalités d’une application avant son implémentation car cela est parfois fastidieux et représente (à tort) une perte de temps. Mais la mise en place de test, en particulier lorsque l’on développe par module comme en objet s’avère fort utile. Cela rallonge au départ la durée du développement mais limite très vite la maintenance et la recherche de « bugs » au fur et à mesure du développement. En effet à chaque nouvelle fonctionnalité il est possible de tester si cela n’a aucune répercussions négatives sur les autres modules de l’application. Il m’a été difficile de comprendre l’intérêt de tel test, je n’en avais jamais conçu et cela me paraissait très fastidieux et abstrait. Pourtant au fur et à mesure du développement j’ai appris à apprécier la mise en place de ces tests qui fournissent une certaine sécurité pour la suite du développement.

E RÉSULTATS OBTENUS :

La première étape de l'algorithme : le split, utilise la variance avec un seuil de 9. L'image est segmentée en zones homogènes (ici représenté par chaque rectangle sur l'image).

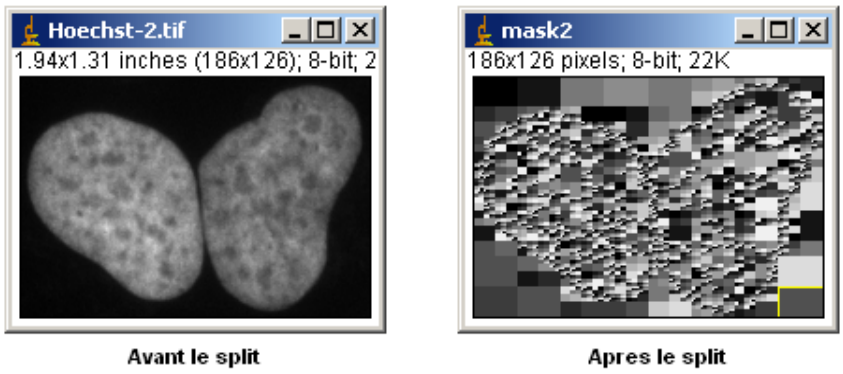


Illustration 19: Résultat du split

Ces régions vont ensuite être passées une par une afin de les fusionner avec leurs voisins si elles respectent les conditions définies par l'utilisateur via l'interface du plug-in.

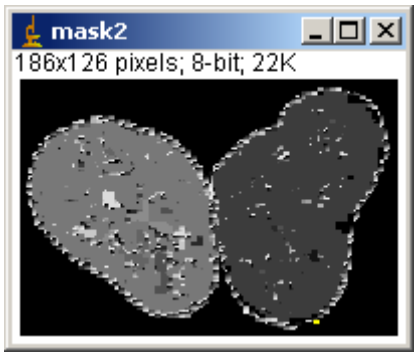


Illustration 20: Premiere lissage de l'image

On voit que les zones aux frontières ne sont pas fusionnées car elles ont un gradient élevé et des intensités différentes. Pour enlever ce « bruit » sur l'image, un lissage est effectué qui va fusionner les zones de petites tailles avec son voisin le plus grand tout en excluant le fond de l'image définie dans l'interface.

Sur l'image ci-dessous, en combinant un critère d'intensité et le calcul du gradient, on voit que les 2 cellules sont bien séparées. La légère bande noire entre les 2 cellules permet grâce au calcul du gradient de séparer correctement les cellules.

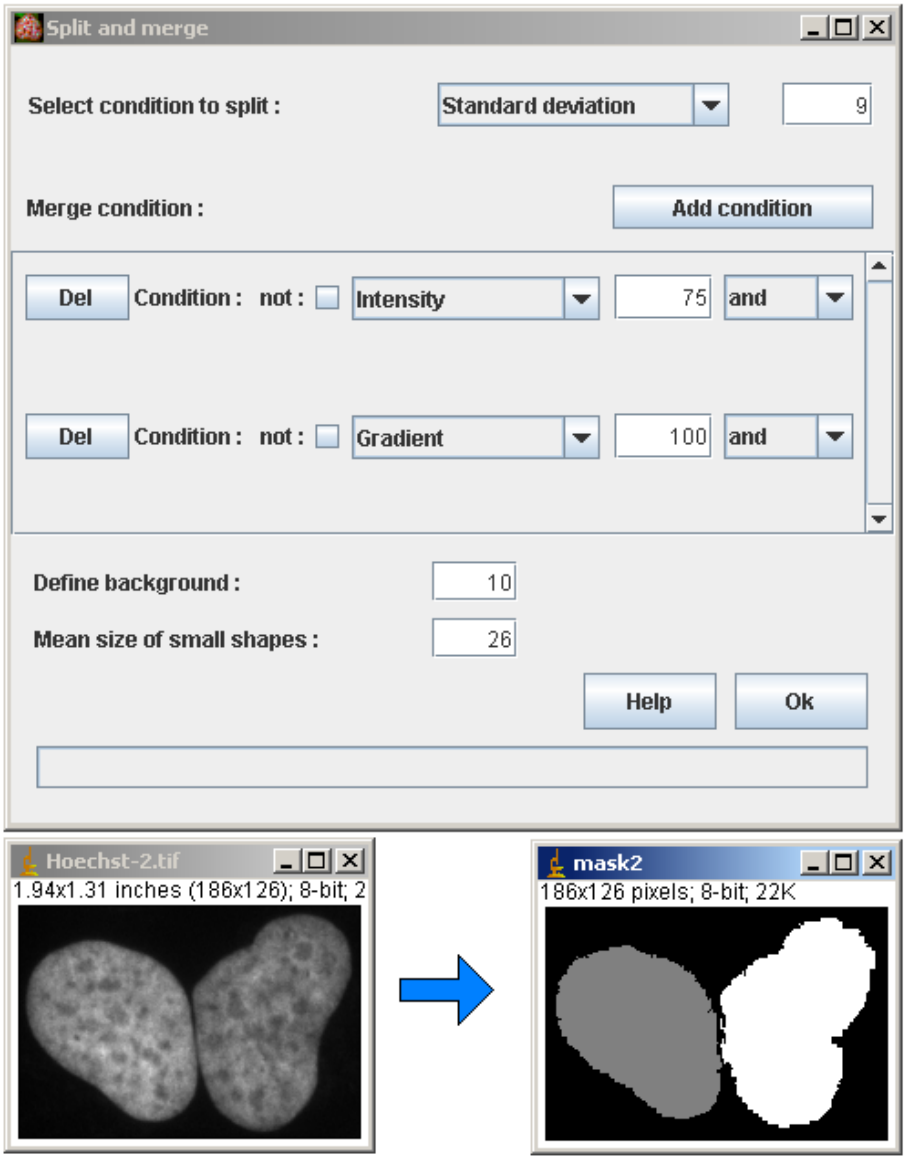
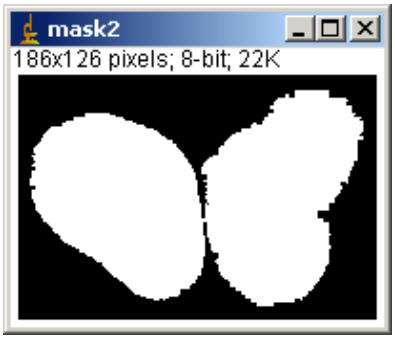


Illustration 21: Résultat obtenu apres utilisation du plug-in

Sans appliquer la condition du gradient, les deux cellules ne sont pas séparées :



12 Ajout de fonctionnalités au logiciel ImageJ

A Calcule d'entropy

Présentation

Au cours du développement du plug-in « split and merge », j'ai rajouté des fonctionnalités au logiciel ImageJ, comme le calcul d'entropie (utilisé comme condition dans l'algorithme) que j'ai rajouté dans les opérations de mesure via une interface en drag and drop. Il suffit à l'utilisateur de faire une sélection sur l'image (par default l'image entière est sélectionnée) et de cliquer sur le bouton « measure entropy », le résultat est affiché dans une petite fenêtre.

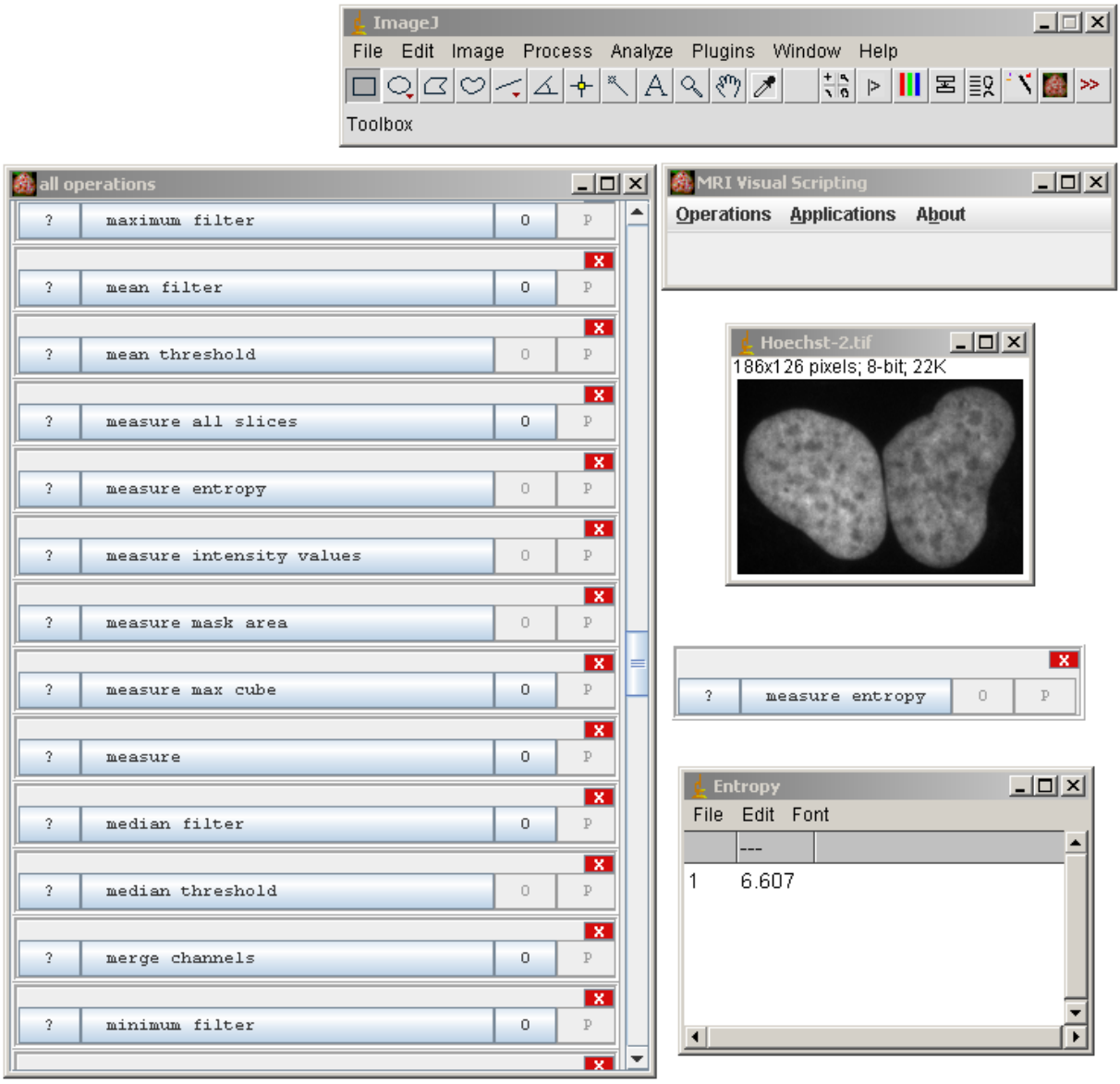


Illustration 22: Ajout d'une nouvelle fonctionnalité : calcul d'entropie

Fonction

Le calcul de l'entropie se fait à partir de l'histogramme d'une zone ou de l'image de la manière suivante :

```
public void calculateEntropy() {
	double sum = 0;
	int i = 0;
	for(i=0;i<histogram.length;i++) {
		if(histogram[i]==0) continue;
		double ratio = histogram[i]/((double)area);
		sum += (ratio*(Math.log(ratio)/Math.log(2)));
	}
	entropy = -sum;
}
```

13 Discussion

L'élaboration de ce projet, même si elle a parfois été très difficile, a été bénéfique en tout point et je suis assez content du résultat obtenu malgré tous les problèmes rencontrés.

Ce stage m'a demandé une longue période d'adaptation pour arriver à m'immerger totalement dans le logiciel, dans le monde de la biologie, du traitement et et de l'analyse d'image numérique. L'apprentissage des techniques de segmentation n'a pas été une chose aisée. Les documentations scientifiques étant toutes rédigées en anglais et sur le thème de la biologie, la compréhension n'a pas été de tout repos.

L' algorithme que j'ai choisi de développer, n'est pas encore terminé. Il faut maintenant rajouter des conditions pour qu'il soit plus efficace sur de nombreux types d'images et de cellules. Il va servir de base à plusieurs futurs projets de mon tuteur.

C'est pour cela que j'ai développé de façon à ce que mon code soit totalement réutilisable et que l'ajout de fonctionnalité soit le plus simple possible.

J'ai énormément appris lors de ce stage, tant en développement (Java, méthode de travail et organisation) qu'en traitement et analyse d'image. Même si le coté biologie reste encore assez obscure !

Ce stage m'a aussi permit de m'améliorer en anglais. D'une part avec la présentation que j'ai donné et d'autre part, la majeur partie des chercheurs n'étant pas français, il m'a fallut communiquer en anglais y compris avec mon tuteur.

Le travail réalisé s’est avéré très enrichissant pour mon expérience professionnelle aussi bien en ce qui concerne le domaine technique que l’aspect humain. Dans les travaux réalisés, j’ai pu apporter mes connaissances théoriques et approfondir certains domaines que je ne connaissais pas encore. J'ai pu découvrir Eclipse, qui est un logiciel de développement très utilisé dans la plus plupart des entreprises.

Enfin le simple fait de sentir l’utilité de son travail est une très grande satisfaction en particulier lorsque les obstacles ont été nombreux.

14 Conclusion

Les objectifs d'un stage professionnel sont multiples : le stagiaire doit tirer une double expérience (immersion dans le monde du travail et acquisition de nouvelles connaissances sur les activités en entreprise) et pouvoir apporter à l'entreprise un bénéfice aussi bien sous forme de nouvelles compétences liées à sa formation qu'à sa personnalité.

Ce stage a été enrichissant, aussi bien au niveau humain que professionnel et sera un atout pour mon entrée dans la vie active.

Le développement de ce plug-in sous ImageJ à travers ce stage en entreprise m'a tout d'abord permis de réaliser un projet informatique assez complexe dans une structure professionnelle. J'ai également pu approfondir mes connaissances en Java acquises en 2e année à l'EPSI. De plus, les nombreux problèmes rencontrés m'ont permis d'apprendre énormément.

Ce stage s'est avéré être très formateur, il m'a apporté de nouvelles connaissances tant organisationnelles que techniques et m'a permis d'approfondir les compétences que j'ai acquises tout au long de ma scolarité.

15 BIBLIOGRAPHIE

- http://www.developpez.com/

Site et forum spécialisé dans la programmation avec une section de traitement et d'analyse d'image.

- http://www.cs.ru.nl/~ths/rt2/

Explications de plusieurs algorithmes de segmentation d'image.

- http://www.ph.tn.tudelft.nl/Courses/FIP/noframes/fip-Segmenta.html

Beaucoup de méthodes de calculs pour différents algorithmes.

- Les bibliothèques de revues scientifiques (IEEE....) ou mon tuteur m'a donné les accés du CNRS.

16 TABLE DES ILLUSTRATIONS

Index des illustrations

17 GLOSSAIRE

Algorithme :

Assemblage d'instructions à suivre pour obtenir l'exécution d'une tâche donnée. C'est l'algorithme qui est à la base de la structure du programme ou d'une de ses fonctions.

Baguette magique :

Outil utilisé en retouche d'image et permettant de réaliser une sélection dans l'image selon une tolérance de couleur.

Cytométrie :

La cytométrie en flux est une technique permettant de faire défiler des particules, molécules ou cellules à grande vitesse dans le faisceau d'un laser.

Drag and drop :

Littéralement Glisser-Déposer en langue anglaise, désigne une manipulation effectuée à l'aide de la souris, consistant à déplacer un objet virtuel (icône, élément graphique, widget, etc.) d'un point vers un autre sur le plan de l'écran, ou d'une zone contenue dans ce dernier.

Entropie :

L'entropie d'un système caractérise son degré de désordre.

Framework :

Un Cadre d'applications (en anglais, Application Framework) est un ensemble de bibliothèques permettant le développement rapide d'applications. Ils fournissent suffisamment de briques logicielles pour pouvoir produire une application abouti. Ces composants sont organisés pour être utilisés en interaction les uns avec les autres. Ils sont en principe spécialisés pour un type d'application.

Gradient :

Le gradient est une grandeur vectorielle qui indique comment une grandeur physique varie en fonction de ses différents paramètres.

Histogramme :

Représentation graphique de la répartition de la luminosité d'une image.

J2EE :

Java 2 Platform, Enterprise Edition est un framework pour le langage de programmation Java de Sun plus particulièrement destiné aux applications d'entreprise.

Ligne de partage des eaux :

Algorithme de segmentation d'images, ce qui revient à décomposer une image en régions homogènes.

Logiciel libre :

L'expression "logiciel libre" correspond à la liberté donnée aux utilisateurs d'adapter le logiciel à ses besoins, de distribuer des copies, d'améliorer le programme, de publier des modifications.

Morphologie mathématique :

Fonction associant à chaque valeur possible d'une variable une probabilité.

Open source :

L'expression Open Source caractérise les logiciels dont le code source est visible, modifiable et librement redistribuable sous certaines conditions, ces conditions peuvent être plus ou moins strictes. La formulation de ces conditions constitue d'ailleurs le critère principal qui différencie le logiciel open source du Logiciel libre.

Photoshop :

La référence "pro" en matière de retouche d'images et de création graphique signée Adobe. Intègre une large bibliothèque d'effets, filtres et calques.

Plug-in :

Programme complémentaire ajouté à un programme principal afin d'offrir des fonctionnalités supplémentaires.

Ratio signal bruit (SNR) :

rapport signal sur bruit désigne la qualité d'une transmission d'information par rapport aux parasites.

Seuillage :

C'est la technique la plus basique, mais aussi la plus rapide, qui consiste à ne garder que les pixels dont l'intensité est supérieure (ou inférieure) à une valeur de référence, le seuil.

Squelettisation :

La squelettisation est une classe d'algorithmes utilisée en analyse de formes. Elle consiste à réduire une forme en un ensemble de courbes, appelées squelettes, centrées dans la forme d'origine. La squelettisation est un outil d'analyse de forme non-scalaire, qui conserve les propriétés topologiques de la forme d'origine ainsi que les propriétés géométriques, selon la méthode employée.

Systèmes d'exploitation :

Un système d'exploitation (SE ou OS en anglais pour Operating System) est un ensemble cohérent de logiciels permettant d'utiliser un ordinateur et tous ses éléments (ou périphériques). Il assure le démarrage de celui-ci et fournit aux programmes applicatifs les interfaces pour contrôler les éléments de l'ordinateur.

Transformation de Fourier :

Calcul (qui porte le nom de son inventeur Jean-Baptiste Joseph Fourier, 1768-1830) qui permet la représentation graphique d'un son et de son déroulement temporel. Elle se présente sous forme d'un graphique à trois dimensions.

UML :

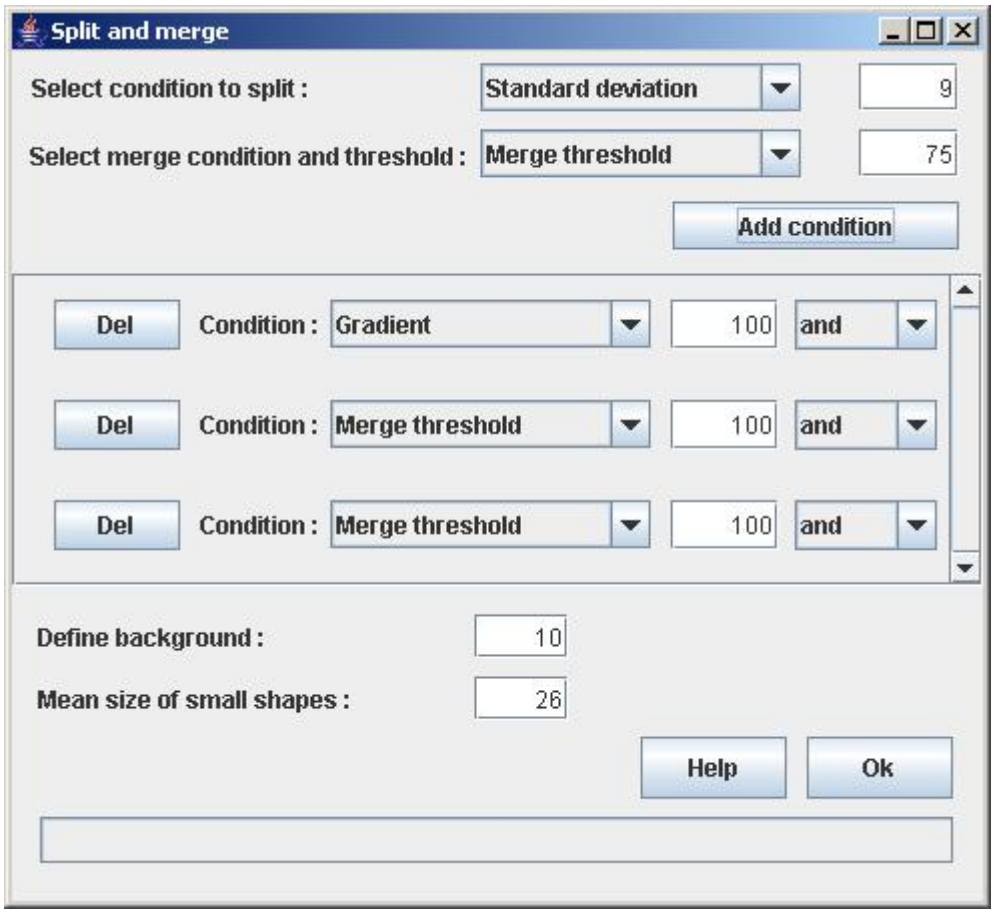
Unified Modeling Language ("langage de modélisation unifié") Langage de modélisation de troisième génération, permettant de décrire une application en fonction des méthodes objet avec lesquelles elle a été construite. Douze types de diagrammes standard ont été définis. C'est une fusion des idées des méthodes Booch, OMT et OOSE.

18 Annexes

Fichier d'aide de mon plug-in :

1. Split and merge plugin
    1. Help
        1. What is split and merge ?
        2. Select condition to split
        3. Select merge condition and threshold
        4. Add merge condition
        5. Mean size of small shapes
        6. Define background

Split and merge



What is split and merge ?

The basic idea of region splitting is to break the image into a set of disjoint regions which are coherent within themselves :

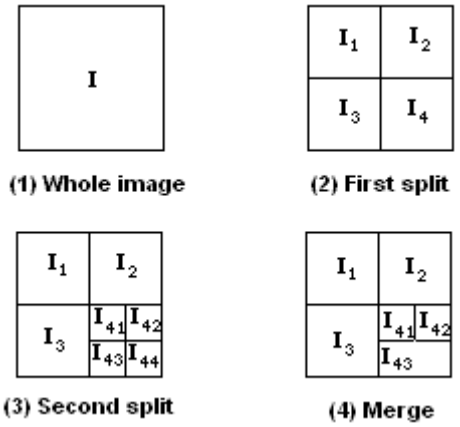
- Initially take the image as a whole to be the area of interest.
- Look at the area of interest and decide if all pixels contained in the region satisfy some similarity constraint.
- If TRUE then the area of interest corresponds to a region in the image.
- If FALSE split the area of interest (usually into four equal sub-areas) and consider each of the sub-areas as the area of interest in turn.
- This process continues until no further splitting occurs. In the worst case this happens when the areas are just one pixel in size.
- This is a divide and conquer or top down method.

If only a splitting schedule is used then the final segmentation would probably contain many neighbouring regions that have identical or similar properties.

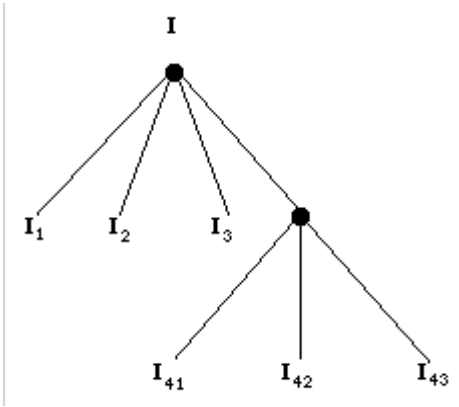
Thus, a merging process is used after each split which compares adjacent regions and merges them if necessary. Algorithms of this nature are called split and merge algorithms.

To illustrate the basic principle of these methods let us consider an imaginary image.

- Let **I** denote the whole image shown in the figure
- Not all the pixels in **I** are similar so the region is split.
- Assume that all pixels within regions **I1**, **I2** and **I**3 respectively are similar but those in **I4** are not.
- Therefore **I4** is split next as in the figure.
- Now assume that all pixels within each region are similar with respect to that region, and that after comparing the split regions, regions **I43** and **I44** are found to be identical.
- These are thus merged together.



We can describe the splitting of the image using a tree structure, using a modified quadtree. Each non-terminal node in the tree has at most four descendants, although it may have less due to merging.



Select condition to split

- Standard deviation : Standard deviation of the gray values used to generate the mean gray value. For example, in an image composed of 50% black (intensity 0) and 50% white (intensity 255), the standard deviation is 127,5.
- Mean : Average gray value within the selection. This is the sum of the gray values of all the pixels in the selection divided by the number of pixels.

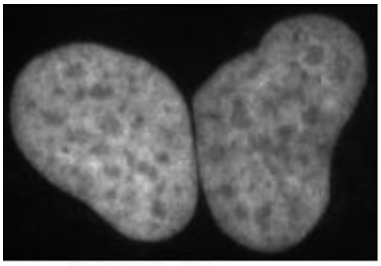
Select merge condition and threshold

Fill the input with the threshold value of intensity to merge two regions.

- Merge threshold : It's the standard threshold used in merging step. It compares

mean intensity of two regions to decide if they have need to be merged.

- Gradient : The goal of gradient is to mark the points in a digital image at which the luminous intensity changes sharply. In the exemple the gradient threshold is set to 100. We can see in the image the border between the two nucleus.



(1) Original Image

(2) Image with gradient's condition

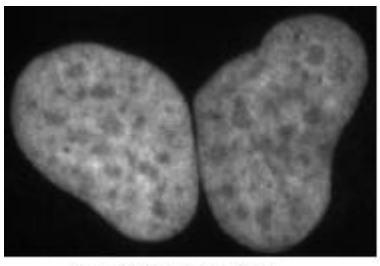
(3) Image without gradient's condition

Add merge condition

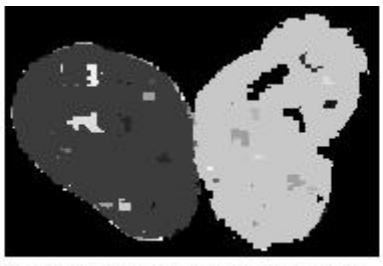
If you want to add one other condition in the merge step, you can click on the button "add condition".
A new line of merge condition is add and you can select a new condition in the drop down menu and fill the input on the right.
If you decide finally not to use the condition you can click on the "del" button to remove the condition's line.

Mean size of small shapes

The goal of this field is to merge small shape after the merging processus with his biggest neighbor.



(1) Original image



(2) Without small shape merging

(3) With small shape merging

You can define the mean area (in pixels) of small shapes in the image. In the exemple, the threshold is set to 26 pixels.

Define background

To improve circularity of cells, we need to merge areas with an other after the merging processus. To do this, we need to know the background of the image.
This value define the maximum mean of intensity of the background (generally near to 10 if the background is black).