MASTER EN SCIENCES INFORMATIQUES

Composition automatique de frottis numériques

Zuyderhoff, Louis

Award date:
2003

FACULTÉS UNIVERSITAIRES NOTRE-DAME DE LA PAIX, NAMUR
Institut d'Informatique.
Année Académique 2002-2003

Composition automatique de frottis numériques

par Louis Zuyderhoff

Mémoire présenté en vue de l'obtention
du grade de Maître en Informatique.

Résumé

Ce mémoire présente une solution de télémicroscopie mettant en oeuvre les concepts de "frottis numérique" et de "microscope virtuel". Ces deux concepts constituent les points clés d'une approche de la télémicroscopie encore peu développée de nos jours. L'argumentation se concentre dans un premier temps sur une description des avantages et sur une analyse des problèmes inhérents à cette approche.

Ce mémoire décrit ensuite l'élaboration de la solution qui s'organise en deux applications.

La première application est spécifique à la composition des frottis numériques. La réalisation de tels frottis comprend plusieurs étapes que l'on retrouve en télémicroscopie conventionnelle dont l'acquisition, l'assemblage et l'archivage d'un ensemble d'images. Mais elle nécessite aussi une phase supplémentaire de coregistration entre les images qui fait l'objet d'une étude approfondie dans ce mémoire.

La deuxième application est spécifique au microscope virtuel. Le microscope virtuel est une application distribuée qui exploite les possibilités spécifiques en matière de méga images du nouveau standard de compression d'images JPEG2000 afin de permettre une visualisation à la fois intuitive, efficace et à distance des images de frottis numériques.

Mots clés

coregistration, frottis numérique, JPG2000, télémicroscopie, méga image, microscope virtuel

Abstract

This thesis presents a solution of telemicroscopy implementing the concepts of "numerical smear" and "virtual microscope". These two concepts constitute the key points of an approach of telemicroscopy which is still little developed nowadays. The argumentation concentrates first on a description of the advantages and on an analysis of the problems inherent in this approach.

This thesis then describes the development of the solution, which is organized in two applications.

The first application is specific to the composition of the numerical smears. Realizing numerical smears includes several stages such as acquisition, assembly and storage of a set of images, which can be found in conventional telemicroscopy. But it requires also an additional phase of coregistration between the images which is the subject of an in-depth study in this thesis.

The second application is specific to the virtual microscope. The virtual microscope is a distributed application which exploits the specific mega images possibilities of the new standard JPEG2000. This allows an intuitive, effective and remote visualization of the images of numerical smear.

Keywords

coregistration, numerical smear, JPG2000, telemicroscopy, mega image, virtual microscope

Remerciements

Que serait un mémoire s'il n'était que le fruit du travail d'un seul homme ! Avant de commencer, je tiens à remercier toutes les personnes qui m'ont aidé, soutenu et guidé durant la réalisation et la rédaction de ce mémoire :

Je tiens tout d'abord à remercier mon promoteur, Monsieur le professeur Jean Paul Leclercq qui m'a donné l'opportunité de réaliser mon stage à La clinique universitaire de Mont-Godinne dans le domaine de l'imagerie. Je tiens aussi à le remercier pour ses judicieux conseils en ce qui concerne les parties les plus formelles de ce document.

Je tiens ensuite à remercier Monsieur Hubert Meurisse pour l'intérêt qu'il m'a apporté pendant toute la durée du stage. Je le remercie aussi pour m'avoir aidé à réaliser un premier plan de ce mémoire et pour les nombreuses critiques qu'il y a apportées par la suite.

Je tiens aussi à remercier Messieurs Bernard Chatelain et Yvan Cornet pour avoir mis à ma disposition la station d'acquisition du laboratoire de l'hôpital. Je remercie particulièrement Yvan pour les réponses à mes nombreuses questions sur le fonctionnement des appareils.

Je voudrais dire merci aussi à Benoit George et à Koen Waelput pour les échanges de mails que nous avons entretenus et qui m'ont permis d'avancer efficacement dans mon travail. Je remercie particulièrement Benoit pour sa disponibilité, son intérêt pour la télémicroscopie et pour m'avoir permis de découvrir JPEG2000.

Merci aussi à Guillaume, Pierre, Ravi et Frédéric pour les relectures et critiques de ce document qui ont contribuées à sa qualité.

Enfin, je remercie tout particulièrement mes parents pour leur présence à la fois discrète et chaleureuse, pour m'avoir soutenu et encouragé jusqu'au bout de mon mémoire et pour les relectures de celui-ci.

Table des matières

Introduction

La télémicroscopie est le nom donné à la discipline qui résulte de l'application des technologies de l'information et des communications à la microscopie. Le développement de cette discipline a été par le passé fortement freiné par une inadaptation des NTIC à la gestion de grandes quantités de données. Les approches poursuivies en télémicroscopie consistèrent donc dans un premier temps à fournir des solutions qui dépendaient fortement de ces limitations et qui n'étaient par conséquent pas toujours en accord avec les besoins réels des microbiologistes.

Cela dit, l'évolution constante des NTIC permet aujourd'hui de penser à des solutions nouvelles, beaucoup plus consommatrices en ressources mais aussi beaucoup plus adaptées aux besoins des microbiologistes. Dans ce contexte, **la composition de frottis numérique**[1] et **le microscope virtuel** constituent les points clés d'une approche relativement ambitieuse de la télémicroscopie par la quantité d'informations que constitue une image à numériser, à stocker et à manipuler.

Dans le cadre d'un stage effectué à la clinique universitaire de Mont-Godinne, j'ai pu travailler sur un projet ayant pour objectif le développement d'un système de télémicroscopie basé sur cette nouvelle approche. Cela fait plusieurs années déjà que des recherches dans le domaine de la télémicroscopie sont poursuivies à Mont-Godinne et notamment par des étudiants stagiaires. On pourra par exemple relever [Mer00], une étude sur la mise au point d'une station de télémédecine pour le laboratoire d'hématologie et plus récemment [Geo02], qui présente la mise au point d'une solution de télémicroscopie basée sur la constitution de galleries d'images. Mais aucun travaux relatifs à la composition de frottis numériques n'avait encore fait l'objet d'une étude approfondie.
Ce mémoire reprend un exposé des recherches et résultats engendrés durant la réalisation du projet poursuivi lors de ce stage.

La première partie présente le cadre de la télémicroscopie. Elle relate ensuite les deux approches classiques qui ont principalement fait l'objet de recherches jusqu'à présent en télémicroscopie en détaillant leurs qualités et leurs défauts et explique comment la "composition automatique de frottis numériques" permet de concilier leurs avantages respectifs. Cette première partie se termine en présentant les principaux problèmes qui ont freiné l'élaboration des solutions basées sur cette nouvelle approche et donnent des indications de solutions pour y remédier.

La deuxième partie est consacrée aux outils disponibles et aux technologies utilisées. Elle décrit les principaux appareils de la station d'acquisition de la clinique de Mont-Godinne, elle rappelle ensuite les concepts principaux liés à l'image numérique et se termine en présentant JPG2000, un nouveau standard de compression d'images sur lequel se base une grande partie de la solution développée.

La troisième partie est consacrée à la solution développée. Elle consiste en une analyse critique des choix effectués pendant le stage afin de résoudre les problèmes introduits dans la première partie. Elle commence par présenter les objectifs et les contraintes d'application liées principalement à la

[1] I.e une image numérique correspondant à un étalement (de sang ou de moelle par exemple)

qualité des images et décrit ensuite les deux applications développées : "la composition de frottis numérique" et le "microscope virtuel".

Enfin la quatrième partie est une discussion sur les limites du système développé. A chaque limite identifiée, elle présente des indications d'améliorations possibles.

Première partie

État de l'art

Chapitre 1

Télémédecine, télémicroscopie et frottis numériques

La composition automatique de frottis numériques constitue une approche particulièrement récente de la télémicroscopie. Ce premier chapitre a pour objectif de situer cette approche en définissant les domaines d'application dont elle fait partie à savoir, la télémédecine et la télémicroscopie.

1.1 De la télémédecine...

> La **télémédecine** est ***l'utilisation des télécommunications et des technologies de l'information dans le but de permettre la prestation de soins à distance ou de recueillir, organiser et partager des informations cliniques requises pour évaluer l'état d'un patient ou encore de poser un diagnostic et d'établir un traitement.[Montb]***

Partant de cette définition, on peut constater que le champ d'application que recouvre la télémédecine est très vaste. Dans la littérature, on retrouve une multitude de termes relatifs à la télémédecine et plusieurs classifications possibles.

Premièrement, la télémédecine intervient dans un grand nombre de disciplines médicales. On parle par exemple de la télépsychiatrie, de la téléradiologie, de la télécardiologie pour faire référence aux applications de la télémédecine dans ces différents domaines. Mais les services que proposent les NTIC à toutes ces branches de la médecine sont fort semblables et moins nombreux que les disciplines médicales. C'est pourquoi on convient généralement de classifier les différentes branches de la télémédecine par "types de services" plutôt que par "champs d'action".

Dans [Duvau], on retrouve une classification de la télémédecine en quatre classes de services différents :

- La **téléformation** ou la formation à distance : on parle aussi de **téléencadrement** pour faire référence à l'établissement d'une relation entre un spécialiste (jouant le rôle de mentor) et un médecin de soins primaires, un résident, une infirmière, etc.
- La **téléexpertise** ou expertise médicale à distance : on parle aussi souvent du **télédiagnostic** pour faire référence à un ensemble de services diagnostiques fournis par des groupes de spécialistes aux établissements éloignés qui n'ont pas d'expert sur place.
- La **télésurveillance** ou surveillance à distance.
- Le **système d'information à vocation régionale voire nationale**.

Il est évident que les avancées dans les différents domaines de la télémédecine sont étroitement liées aux avancées dans les technologies des télécommunications et en informatique. La transmission des données par exemple et plus spécifiquement des données d'images médicales est restée stagnante pendant au moins un demi-siècle. C'est en fait seulement depuis 10 ans que la télémédecine s'est véritablement développée de manière exponentielle. Cela peut être attribué à l'augmentation rapide de l'efficacité des technologies hardware de stockage, des réseaux, etc, combinée avec leurs coûts décroissants. Par exemple, les discussions d'il y a à peine quelques années sur le nombre de couleurs (28 ou 224) appropriées pour le codage d'images médicales sont devenues aujourd'hui non pertinentes alors que les technologies hardwares et softwares nécessaires pour compresser, stocker et transmettre ces fichiers images avec 224 couleurs ou plus, sont devenues omniprésentes.[Lam00]

1.2 ... à la télémicroscopie

Intuitivement, on peut dire de la télémicroscopie qu'elle est l'application de la télémédecine à la microscopie. Elle est définie plus formellement dans [Geo02] par :

> La **télémicroscopie** est ***la pratique de la visualisation et du diagnostic, à distance, d'images en provenance d'un microscope. Cette pratique nécessite la visualisation d'images numérisées sur écran vidéo plutôt qu'au travers des oculaires du microscope et également un moyen de communication entre le poste de visualisation et le microscope afin de permettre la transmission des images***

Bien que cette définition donne un sens relativement large à la télémicroscopie, on constate néanmoins que celle ci s'oriente principalement vers le télédiagnostic[1]. La microscopie est aujourd'hui une discipline essentielle dans l'élaboration d'une série de diagnostics[2]. Dans ce sens, l'analyse d'une lame au moyen d'un microscope par un pathologiste[3] permet l'identification de plusieurs symptômes chez un patient.

En microscopie, le rôle du pathologiste peut être résumé très brièvement par les étapes suivantes : suite à la demande d'une analyse, le spécialiste utilise le microscope afin de scanner avec différents grossissements une lame de verre à la recherche d'anomalies. Lorsqu'une anomalie est détectée, elle est confrontée avec une série d'autres informations cliniques disponibles telles que le dossier médical du patient dans le but d'identifier un ou plusieurs diagnostics possibles.

Parfois, la difficulté du cas nécessite un deuxième avis : il peut s'agir de l'avis d'un collègue travaillant dans le même département, mais bien souvent l'avis d'une personne externe et experte dans le type de cas analysé est requis. Dans cette dernière situation, le spécialiste qui a effectué l'analyse doit communiquer à l'expert le support nécessaire à sa consultation. Le but de la télémicroscopie est alors d'améliorer cette communication entre parties distantes.[Del99]

1.3 Des approches classiques de la télémicroscopie...

On distingue deux approches sensiblement différentes en télémicroscopie ou plus généralement en télépathologie :

[1] On parle aussi de la télépathologie qui est un terme plus adapté à la microscopie

[2] le **diagnostic** est défini comme *l'action de déterminer une maladie d'après ses symptômes.*

[3] Le spécialiste en pathologie.

- L'approche **Store and forward** (ou **statique**) : c'est l'accès à une galerie d'images en provenance d'un microscope (éventuellement transmis via un réseau) dans le but d'émettre un diagnostic.
- L'approche **real time** (ou **dynamique**) : c'est le pilotage à distance et en temps réel d'un microscope dans le but d'émettre un diagnostic.

1.3.1 La télémicroscopie store and forward

L'objectif principal des solutions "store-and-forward" est d'améliorer la communication entre un premier analyste situé sur le site où se trouve le microscope et un expert situé à distance. L'idée sous jacente à la solution est de sélectionner et de numériser des images représentatives à partir du microscope et d'en envoyer une copie au consultant distant grâce aux NTIC. On peut résumer la solution "Store and Forward" en la succession des quatre processus suivants [Del99] :

- La sélection d'une série d'images sur une lame de verre. La lame de verre étant le support de l'information à analysée, un étalement (ou frottis) de sang par exemple.
- L'acquisition des images sélectionnées grâce à une caméra ou un appareil photo.
- Le stockage des images acquises dans un format approprié.
- La transmission des images formatées.

L'avantage principal de cette approche est de permettre des analyses "asynchrones". En effet, la présence du premier analyste n'est plus nécessaire lors de la deuxième analyse par l'expert puisque ce dernier dispose à priori de tous les éléments nécessaires à son analyse[4]. Ce dernier effectue son analyse quand il le souhaite et peut même demander l'avis de plusieurs autres experts en leur transmettant simplement le document à analyser.
Un deuxième avantage est le coût relativement faible de cette solution : le transfert des données peut se faire via FTP ou via mail[5]. De plus l'acquisition des images peut se faire via une simple caméra analogique connectée à une carte d'acquisition vidéo ce qui permet d'obtenir des images avec une résolution suffisante pour un faible coût.[Lam00]
Enfin on peut encore citer une série d'avantages liés au stockage d'une information numérique, les galeries d'images plutôt qu'une information analogique, la lame de verre :

- Les frottis sont soumis à une coloration préalable qui peut s'estomper avec le temps.[Mer00]
- Des bulles d'air ou des souillures peuvent apparaître et fausser une analyse.
- Le verre est facilement cassable.
- Etc.

Cependant, l'approche "Store and forward" est aussi caractérisée par une analyse effectuée sur une partie restreinte du support d'origine : la galerie d'images. Selon [Lam00], le taux de diagnostics corrects pour les solutions "Store and forward" se trouve entre 85 et 90%. Bien que le taux de faux diagnostics positifs soit sensiblement plus faible que celui des faux diagnostics négatifs[6], ces résultats sont encore en dessous des taux atteins en microscopie conventionnelle.

La phase de sélection des images intervient de manière cruciale dans le résultat du diagnostic final puisque c'est lors de cette phase que sera déterminée l'information dont disposera l'analyste expert.

[4]Comme nous allons le voir dans la section suivante, cela n'est plus du tout vrai pour dans l'approche *real time*

[5]Dans ce dernier cas la taille des images sélectionnées et leur nombre sont alors restreints.

[6]Cette observation est expliquée dans [Lam00] par le fait que souvent, un faux diagnostic positif n'est pas la conséquence d'une information incomplète mais plus celle d'une information imprécise alors qu'un faux diagnostic négatif peut être du à un manque d'information lié aux choix des images sélectionnées.

Pour cette raison, dans plusieurs travaux ([Wei96], [Hal97]), cette phase de sélection est réalisée manuellement par un pathologiste. Cette phase souffre alors du caractère subjectif de son résultat car les images sélectionnées par deux pathologistes différents ne seront pas les mêmes.
Dans d'autres travaux ([Geo02]) cependant, la phase de sélection est automatique, ce qui limite[7] donc le caractère subjectif du processus.

1.3.2 La télémicroscopie real time

Une autre approche de la télémicroscopie est l'approche "real time" (ou temps réel). Son objectif est de fournir un maximum de liberté d'action à l'expert distant en lui fournissant les moyens de piloter le microscope et de visualiser le résultat à distance.
Le principal apport des applications "real time" est la liberté d'action de l'expert sur l'entièreté de la lame analysée et les sensations de proximité de celle-ci. Selon [Lam00], ce haut degré de liberté pour l'analyste a pour conséquence des taux de diagnostics corrects dépassant les 95% et ne semblant plus différer des résultats atteints en microscopie conventionnelle.

Cela dit, ce type de solutions perd une série d'avantages présents dans les solutions "store and forward". Tout d'abord, dans cette approche, il n'est plus possible d'effectuer des analyses "asynchrones". Une analyse à distance requiert en effet la présence d'une personne pour préparer les lames et la disponibilité de la station d'acquisition.
Ensuite, le coût de mise en place de solutions "real time" dépasse largement celui des solutions "Store and forward". Généralement, un service de télémicroscopie real time comprend au minimum :

- un microscope électronique pilotable à distance,
- une caméra vidéo haute résolution et
- un logiciel de vidéo conférence.[Nor99]

L'investissement le plus lourd revient sans conteste au microscope électronique dont les fonctionnalités principales doivent être automatisables : "changer d'objectifs", "déplacer la platine motorisée", etc.[8]. Notons qu'il existe des solutions alternatives à l'investissement d'un tel appareil mais qui sont alors plus consommatrices en ressource humaine. Le pathologiste reste alors dans la station d'acquisition et en contact avec l'expert durant toute l'analyse. Les images en provenance du microscope sont toujours vues en directe par l'expert mais c'est le pathologiste qui manipule le microscope sous la direction de l'expert.
Enfin, dans cette approche, les images ne sont plus stockées numériquement.

1.4 ... à la composition automatique de frottis numérique

Aujourd'hui, les évolutions en télépathologie permettent de penser à un nouveau type de solution réconciliant les avantages de la télémicroscopie statique et de la télémicroscopie dynamique[9]. On parle alors des concepts de "microscope virtuel" et de "composition automatique de frottis numériques". La solution consiste à stocker en information numérique un champ déterminé (position sur la lame et dimensions) dans un frottis et de représenter les fonctionnalités du microscope par une interface

[7]Remarquons tout de même que le caractère subjectif n'est pas totalement effacé : dans [Geo02] par exemple, le bon fonctionnement du programme dépend fortement de la paramétrisation de variables (dimensions et seuils de couleur pour les cellules de sang à détecter).

[8]Un tel microscope électronique est présenté dans le chapitre 3

[9]"Store and forward" et "real time"

adaptée. Si les deux premières approches se rapprochaient plus du domaine de l'aide à la décision, cette nouvelle approche se rapproche alors plus du domaine de la simulation.

Cette solution est attrayante car : tout comme dans les solutions statiques, elle permet de l'archivage numérique de l'information et une analyse asynchrone par l'expert et tout comme dans les solutions dynamiques, elle lui donne un grand niveau de liberté sur le support analysé et lui procure des sensations analogues à l'analyse directe sur un microscope.

1.4.1 Les trois principaux problèmes liés aux frottis numériques

Bien que séduisante par les avantages qu'elle promet, cette récente approche est ambitieuse et pose un certain nombre de problèmes :

La précision des appareils

Il est faux de croire que n'importe quel système d'acquisition est capable de réaliser directement la tâche de digitalisation d'une zone donnée sur une lame à n'importe quel niveau de grossissement. [Leo01] En effet, le théorème de Shannon-Niquist ou plutôt une version simplifiée du théorème nous dit que :

> Pour qu'un signal analogique, i.e. l'image du microscope, soit correctement échantillonné le taux d'échantillonnage doit être égal à au moins deux fois la plus haute fréquence composant le signal analogique.

Appliqué à la digitalisation, cela signifie qu'il faut s'assurer que pour un grossissement et une zone à scanner donnés, le nombre de pixels digitalisés par la caméra est au moins 2 fois le nombre de points pouvant être résolus séparément par le microscope optique sans quoi des pertes d'informations sont à prévoir. Autrement dit, pour chaque grossissement, la taille maximale du champ numérisé par une seule image dépend directement des propriétés du système (le microscope et la caméra).
Cela dit, digitaliser une zone importante n'est pas pour autant impossible, il est en effet possible de capturer plusieurs images contiguës qui seront assemblées par la suite. Cependant, même si cette étape semble évidente, le degré de précision[10] imposé aux appareils lors de l'acquisition des images à un fort grossissement engendre toute une série d'erreurs d'alignement et de variation dans les teintes des images capturées. Ces erreurs doivent alors être corrigées avant de pouvoir assembler les images. Le chapitre 2 présente les différentes techniques, appelées techniques de coregistration, permettant de détecter et de corriger ces erreurs.

La quantité d'information à stocker

Même si la capacité proposée par les technologies de stockage a augmenté de façon considérable ces dernières années, la digitalisation entière d'une lame à un niveau de grossissement élevé reste encore un problème majeur. Les deux facteurs qui influencent directement le volume mémoire d'un frottis numérique sont la "taille de la zone numérisée sur la lame" et "le grossissement auquel les images sont capturées". En effet, si l'on considère que le volume d'une image capturée est un paramètre fixé par la caméra, on admettra que le seul facteur déterminant la taille du "frottis numérique" est le nombre d'images numérisées. Or ce nombre dépend non seulement de la taille de la zone numérisée sur la lame, mais aussi, en vertu du théorème de Shannon-Niquist, du grossissement auquel les images sont capturées.

[10] de l'ordre du micron

Les résultats du tableau 1.1 donnent une estimation du volume en giga Bytes que représente un "frottis numérisé" à différents grossissements[11]. Ces chiffres ont été établis sur base de mesures effectuées sur le microscope du laboratoire de la clinique universitaire de Mont-Godinne et la taille de 20x25 mm correspond à une estimation de la zone entière d'étalement d'un frottis sanguin sur une lame.

TAB. 1.1 – Estimation du volume nécessaire à la digitalisation d'une zone de 20x25 mm sur une lame.

Grossissement	**Taille de 1 pixel numérisé (en μm)**	**Nombre de pixels nécessaires**	**Volume nécessaire pour des images RGB (en GB)**
40 x	0.204	98040 x 122550	33,6
60 x	0.13	153847 x 192307	82,6
100 x	0.08	250000 x 312500	218,3

Un premier choix qui devra être fait en accord avec les possibilités techniques et avec les utilisateurs sera donc de déterminer un rapport équitable entre la zone à numériser sur une lame et le volume engendré par les images capturées.

La manipulation de grandes quantités d'informations

Même si le problème du stockage des "frottis numériques" peut être partiellement résolu par des techniques de compression d'image, il ne reste pas moins un problème de représentation, de structuration et de manipulation de ces données images. Afin de représenter un ensemble d'images structurées tel qu'un frottis numérique, on peut assembler toutes les images dans une seule grande image appelée "méga image".

> Dans la littérature, lorsque l'on parle de **méga images**, on fait référence à ***des images dont les dimensions sont gigantesques, c'est-à-dire, des images qui nécessitent une grande quantité de ressources pour leur stockage et qui entraînent des délais d'accès importants.***

On peut remarquer que le qualificatif "*gigantesque*" rend cette définition fortement subjective. Généralement, on qualifie une image de méga lorsque les technologies classiques de compression et de décompression d'images ne sont plus adaptées à leur utilisation.
Il est évident que plus le nombre d'images capturées pour réaliser un frottis numérique est grand, plus ce dernier sera qualifié à juste titre de "méga image".

> Par la suite, *nous considérerons que n'importe quel frottis numérique est une méga image.*

Notons aussi que l'emploi des méga images n'est pas spécifique à la télémicroscopie. Dépassant le domaine de l'imagerie médicale, on peut retrouver aussi une utilité des méga images dans des domaines tels que l'astrologie ou la météorologie ou plus généralement dans la cartographie.

La représentation et le stockage d'un frottis numérique ou d'une carte macroscopique sous la forme d'une seule méga image n'est cependant pas la seule option possible. En effet, toutes ces applications sont caractérisées par un résultat correspondant à un ensemble d'images de base formant un tout structuré. Un autre choix de représentation pour ce résultat pourrait donc être une base de données

[11] On peut dériver de ce tableau le nombre d'images à capturer en divisant le "Nombre de pixels nécessaires" par le nombre de pixels d'une image. Une image capturée avec la caméra de la station d'acquisition du laboratoire de Mont-Godine fait 765x573 pixels.

d'images dans laquelle chaque image de base est stockée indépendamment avec une représentation de sa position dans l'ensemble structuré.[Leo01]

Si on compare les deux approches, on peut dire que l'approche méga image permet de faire abstraction de toute la structuration des images de bases. Cette remarque est surtout pertinente lorsque la position de chaque image est déterminée par une transformation résultante d'une technique de coregistration[12]. Cela dit, la deuxième approche semble cependant meilleure si on veut pouvoir accéder rapidement à une partie spécifique de l'ensemble structuré correspondant par exemple à une des images de base.

Cette dernière remarque n'est cependant plus tout à fait vraie aujourd'hui. En effet les standards de représentation d'images sont en évolution constante et on commence à voir apparaître des solutions spécifiques à la manipulation de méga images. Parmi ces solutions, on retrouve le nouveau standard JPEG2000 qui est présenté plus longuement dans le chapitre 4 et qui est utilisé dans la solution présentée dans la partie "Analyses et résultats".

[12]Cfr le chapitre 2

Chapitre 2

La coregistration d'images 2D

Dans le chapitre précédent, nous avons introduit l'approche de la télémicroscopie liées à la composition automatique de frottis numériques. Nous avons présenté plusieurs problèmes liés à cette approche. Nous avons notamment souligné que pour numériser un frottis de taille raisonnable, il est nécessaire de capturer une série d'images. Nous avons ensuite affirmé que des erreurs dans l'alignement de ces images étaient à prévoir. Ce chapitre présente un éventail des techniques permettant de corriger ces erreurs appelées techniques de coregistration.

2.1 La définition

> On entend par **coregistration** le ***processus d'alignement ou de mise en correspondance de deux images, appelées image maître et image esclave recouvertes entièrement ou partiellement par une même zone. Cette zone sous-entend donc que les images représentent entièrement ou partiellement une même information que l'on appellera la scène observée.***

Si l'on admet qu'une image "I" n'est rien d'autre qu'un tableau bidimensionnel de valeurs appelées intensités "$I(x, y)$", le résultat de ce processus peut être exprimé mathématiquement comme une double transformation à effectuer sur l'image esclave pour qu'elle corresponde à l'image maître [Bro92].
L'équation 2.1 représente cette double transformation. "f" est une transformation : $\mathbb{N}^2 \rightarrow \mathbb{N}^2$ des coordonnées spatiales de l'image et "g" est une transformation optionnelle : $\mathbb{N} \rightarrow \mathbb{N}$ des valeurs d'intensité de l'image.

$$I_2(x, y) = g(I_1(f(x, y)) \tag{2.1}$$

Le résultat d'une coregistration est donc la composée de deux transformations de types différents, les premières, dites "spatiales", modifient la position des valeurs d'intensité de l'image esclave et les deuxièmes, dites "photométriques", modifient directement les valeurs d'intensité de cette image. Cette observation permet alors de faire une première distinction entre deux types d'erreurs corrigeables par une technique de coregistration : les erreurs de type "spatial" et les erreurs de type "photométrique". Une classification plus détaillée de ces différentes erreurs et de leurs causes sera établie dans la section suivante.

En télémédecine, les applications de la coregistration sont multiples, on fait souvent la distinction entre :

- **La coregistration multimodale** : c'est la coregistration d'images provenant de différents appareils (PET, CT, MRI, etc). Elle permet la mise en relation d'images aux caractéristiques fort différentes (anatomiques ou fonctionnelles) prises sur un même patient.
- **La coregistration monomodale** : c'est la coregistration de plusieurs images provenant d'un même appareil. Elle permet par exemple de faire des comparaisons dans l'évolution d'une pathologie chez un même patient ou de comparer l'état de plusieurs patients différents. Cfr [Mar98]

2.2 La classification des déformations

Il n'existe pas de technique universelle de coregistration qui permette la mise en correspondance de n'importe quelle image dans n'importe quelle situation. En fait, il existe un large panel de transformations et de techniques applicables à la coregistration de deux images mais ayant chacune des avantages et des inconvénients différents. Le choix d'une technique plutôt qu'une autre pour une application donnée influencera les performances de la coregistration au niveau de sa précision et du temps de calcul.

> On appelle **déformation** entre deux images ***une erreur dans leur alignement par rapport à la zone observée ou une différence dans leurs valeurs d'intensité.***

Pour un problème donné, une analyse des déformations observables est utile afin de choisir la meilleure technique de coregistration à appliquer. En ce sens, [Bro92] fait la distinction entre plusieurs types de déformations :

- **statique Vs dynamique** : une déformation est *statique* lorsqu'elle est prévisible ou constante, c'est-à-dire que lors de l'acquisition de plusieurs images, on retrouvera toujours la même erreur.
- **interne Vs externe** : une déformation *interne* est une déformation liée à la précision des appareils[1]. Alors qu'une déformation *externe* est due à un changement d'appareil en coregistration multimodale ou à un changement de la scène observée. Par exemple, l'observation du déplacement d'une bulle d'air sur une lame porte objet en télémicroscopie.
- **spatiale Vs photométrique** : on distingue encore une déformation *spatiale* relative aux variations dans le positionnement des valeurs d'intensité de chaque image, d'une déformation *photométrique* relative aux variations des valeurs d'intensité elles-mêmes. Si la première rend toujours l'application d'une technique de coregistration nécessaire, la deuxième est souvent moins visible à l'oeil nu et n'est pas souvent appliquée. Cela dit, une déformation *photométrique* même non visible, peut rendre le résultat de la correction de déformations *spatiales* plus difficile à obtenir avec certaines techniques de coregistration. (cfr la section 2.4)

En télémicroscopie, dans le cadre de la réalisation de frottis numériques, les images proviennent d'un microscope et sont donc des images 2D[2]. La coregistration est utile pour trouver les décalages éventuels liés à l'acquisition de plusieurs images contiguës. Cfr le chapitre 6 pour une analyse des déformations observées.

[1] En télémicroscopie, il s'agit des objectifs et de la platine motorisée du microscope, de l'autofocus, de la caméra, etc.(voir le chapitre 3 pour plus de détails sur ces appareils).

[2] Par opposition aux images médicales 3D.

2.3 La pré-optimisation des techniques de coregistration

Afin d'augmenter le taux de réussite des techniques de coregistration spatiale qui seront présentés dans la section suivante, des transformations peuvent être effectuées sur les images ou directement sur la scène observée.

2.3.1 Le pré-marquage de la scène observée

Le pré-marquage consiste à marquer à différents endroits le corps ou la scène observée avant de capturer les images. Les marques, qui sont séparées par des distances connues, constituent des repères précis sur les images capturées. Typiquement, le marquage utilisé correspond à un quadrillage de lignes et colonnes sur l'entièreté de la zone à scanner[3]. Par exemple, dans [Cap02], les auteurs s'inspirent de cette technique pour l'assemblage d'images en provenance de rayon X.
Cela dit, le principal désavantage de cette méthode est que la trace laissée par le marquage sur le support se retrouve sur les images capturées.

2.3.2 Le pré-filtrage des images capturées

> Un **filtre** est ***une transformation globale appliquée sur les valeurs d'intensité des images maître et/ou esclave à comparer en vue de faciliter leur coregistration.***

Le pré-filtrage des images consiste à appliquer une ou plusieurs de ces transformations afin de limiter l'impact des déformations photométriques sur la recherche des déformations spatiales. Par exemple, un filtre fort connu est la réduction d'une image couleur à une image grise.[4]

Il existe plusieurs type de filtres et la plupart correspondent à des transformations irréversibles sur les valeurs d'intensité, c'est pourquoi l'application d'un filtre nécessitera souvent la duplication des images avant leur coregistration.

2.4 Les techniques de coregistration spatiale

2.4.1 La coregistration manuelle

La coregistration manuelle ou "calibration" est la technique de coregistration la plus simple qu'il soit puisqu'elle laisse le travail de déterminer les points de correspondance entre les images à un utilisateur humain. Typiquement, il s'agit de présenter côte à côte deux images à coregistrer à l'utilisateur et de lui fournir un ensemble d'outils lui permettant d'aligner manuellement l'image esclave par rapport à l'image maître. Une fois les deux images alignées, le système peut alors prendre les mesures des transformations effectuées par l'utilisateur sur l'image esclave.
Il est évident que cette technique est à éviter puisqu'elle représente un temps non négligeable de la part de l'utilisateur. Néanmoins, elle peut trouver son utilité dans les deux situations suivantes :

- Lorsqu'un grand nombre d'images doit être coregistré et que le décalage, bien que difficile à estimer automatiquement entre ces images, est *statique*.
- Suite à la coregistration par une autre technique, comme moyen de validation du résultat par un utilisateur.

[3] En microscopie plus particulièrement, il existe des lames sur lesquelles un tel quadrillage est gravé sur le verre.
[4] D'autres techniques se basent sur des propriétés de l'histogramme des images comme dans [Eus02].

2.4.2 La coregistration automatique

Il existe énormément de techniques de coregistration spatiale différentes, le but de ce mémoire n'est évidement pas d'en faire une étude exhaustive, de plus le problème de coregistration d'images en provenance d'un microscope est un problème particulier pour lequel la plupart de ces techniques n'a aucun intérêt. La suite de la section se base sur une classification en quatre pôles reprise dans [Sey02] pour donner un aperçu des techniques applicables à la coregistration d'images 2D propre au problème particulier des images en provenance d'un microscope.

L'espace des caractéristiques

Une première classification porte sur les différents types de "caractéristiques" qui peuvent être extraites des images maître et esclave et qui peuvent constituer une information comparable. On distingue les types de caractéristiques suivantes :

- **extrinsèques** : il s'agit de caractéristiques non implicites à la scène observée, c'est-à-dire une information liée au pré-marquage préalable de celle-ci.
- **iconiques** : Les seules informations utilisées sont les valeurs d'intensité (ou de pixel[5]) ainsi que leur position dans l'image.
- **de bas niveau** : il s'agit de propriétés géométriques de base (appelées points de contrôle) tels que : les coins, les intersections de ligne, les extrema locaux de courbes et les centres de gravité de zones particulières, etc.

Les techniques basées sur des caractéristiques "iconiques" sont évidemment les plus génériques dans le sens où elles ne dépendent pas des caractéristiques extrinsèques ou intrinsèques de la scène observée. Cela dit, elles sont aussi beaucoup plus sensibles au bruit et aux déformations photométriques d'où l'intérêt quasi obligatoire d'un pré-filtrage.
Les techniques basées sur des caractéristiques "de bas niveau" résistent beaucoup plus au bruit mais les performances de cette approche dépendent directement de la qualité des points de contrôle sélectionnés et de leur nombre. Pour être utilisable, un point de contrôle doit être rigide, stationnaire et facilement identifiable dans les deux images. De plus, leur nombre doit être suffisamment grand afin de déterminer une différence type entre les deux images mais pas trop grand pour que ces caractéristiques ne soient pas contradictoires.

Enfin, le lecteur attentif pourra aussi souligner un quatrième type de caractéristiques, dites de *haut niveau* mais qui ne s'appliquent pas bien à la coregistration 2D.[6]

L'espace de recherche

Une fois les caractéristiques définies, on peut caractériser la fonction de coregistration spatiale[7]. Cette fonction est une transformation caractérisable par une série de paramètres.

> L'**espace de recherche** est défini par ***l'ensemble des paramètres caractérisant la transformation à appliquer sur l'image esclave pour qu'elle soit alignée avec l'image maître.***

On distingue différents types de transformations applicables à l'image esclave pour qu'elle corresponde spatialement à l'image maître. On peut classifier ces transformations selon leur complexité.

[5]Le pixel est l'unité d'intensité d'une image numérique. Cfr le chapitre 4

[6]Il s'agit d'objets plus complexes identifiés dans l'image et comparables à un "atlas de formes". coeur, foi, etc. Cfr [Sey02]

[7]la fonction "f" dans la définition de la coregistration.

corrélation. Les paramètres de ces fonctions sont les matrices des valeurs d'intensité de l'image maître "M" et de l'image esclave "I", ainsi qu'une matrice de transformation "t". L'ensemble *Caract* désigne l'ensemble des valeurs d'intensité faisant partie d'une caractéristique et "I'" représente l'application de la transformation "t" à l'image esclave. I.e : $I' = t(I)$.

$$D(M,I,t) = \sum_{x,y \in Caract} |M(x,y) - I'(x,y)| \tag{2.3}$$

$$R^2(M,I,t) = \frac{\left(\sum_{x,y \in Caract} \left[M(x,y) - \bar{M}\right] \left[I(x,y) - \bar{I'}\right]\right)^2}{\sum_{x,y \in Caract} \left[M(x,y) - \bar{M}\right]^2 \sum_{x,y \in Caract} \left[I(x,y) - \bar{I'}\right]^2} \tag{2.4}$$

L'espace d'optimisation

Une dernière distinction peut encore être faite sur la manière dont la fonction explore les valeurs des paramètres de recherche en vue de trouver son optimum. On distingue deux extrêmes, le premier où aucune valeur de paramètres n'est explorée, c'est le cas des algorithmes déterministes et le deuxième pour lequel toutes les valeurs possibles de chaque paramètre sont évaluées par la fonctionnelle d'appariement. Pour des raison de limitation du temps de calcul, cette dernière solution est fortement déconseillée, surtout lorsque le nombre de paramètres est élevé. Entre ces deux extrêmes, on retrouve plusieurs techniques permettant de limiter le domaine de recherche pour chaque paramètre à explorer. Une classification grossière de celles-ci semble cependant beaucoup plus difficile à élaborer, c'est pourquoi le lecteur intéressé est invité à consulter [Sey02] pour une description détaillée de l'étendue des techniques possibles.
Remarquons cependant qu'une bonne connaissance du problème peut parfois permettre l'identification de bornes maximales et minimales pour chaque paramètre (translation, rotation, etc.) ce qui permet une première limitation du temps de calcul de la recherche.

Le tableau 2.1 reprend différents types de transformations de plus en plus complexe. Plus on descend dans le tableau, plus le nombre de paramètres augmente et donc forcément plus le nombre de possibilités augmente aussi. Il est évident que c'est la nature des déformations qui définit la meilleure

TAB. 2.1 – classification des différents types de transformations

transformation		**paramètres**	**propriétés après transformation**
rigide		vecteur de translation	- la distance entre deux points est conservée
		angle de rotation	
semi-rigide	affine	vecteur de translation	- une droite reste une droite - les rapports de longueur sont conservés - le parallélisme est conservé
		angle de rotation	
		matrice de cisaillement	
		facteur d'échelle	
	projective	matrice complexe[8]	- une droite reste une droite - le parallélisme n'est **plus** forcément conservé
élastique		matrice complexe[8]	aucune restriction[9]

transformation à appliquer. Cela dit, les transformations dites *affines* sont les plus généralement utilisées. Ce type de transformations est généralement suffisant pour les applications dans lesquelles une scène est observée à partir d'un même angle. Une transformation affine est le produit cartésien d'une translation, d'une rotation et d'un facteur d'échelle. La figure 2.1 montre comment une transformation affine peut permettre la fusion de deux images contiguës tandis que l'équation 2.2 exprime un transformation affine avec une translation de (t_x, t_y), un facteur d'échelle de "s" et une rotation d'un angle de "θ".

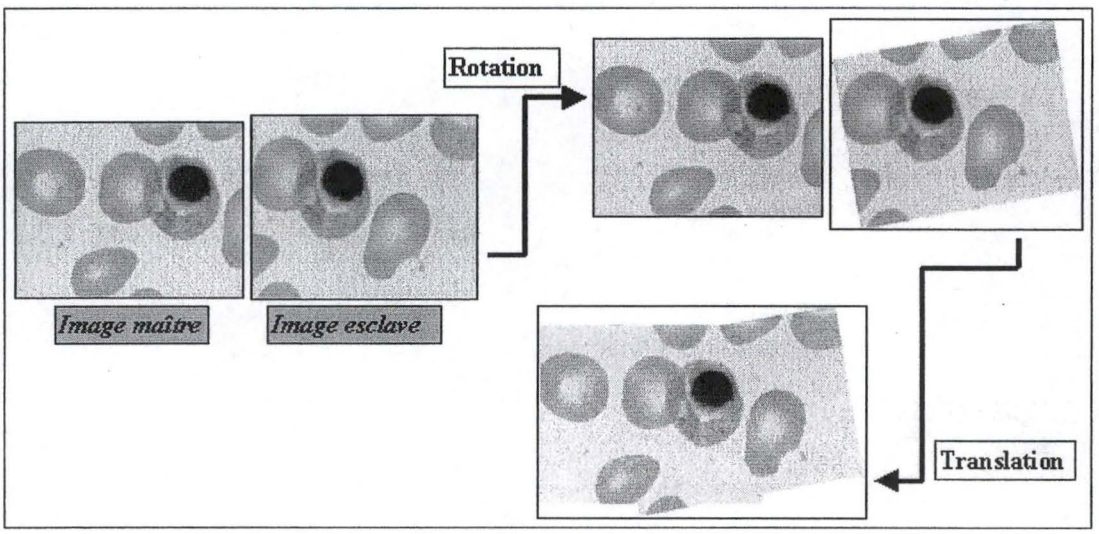


FIG. 2.1 – Exemple de la fusion de deux images.

[8]Pour une description mathématique de ces matrices, se référer à [Sey02]

[9]une droite peut devenir une courbe par exemple.

$$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} t_x \\ t_y \end{pmatrix} + s * \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} \quad (2.2)$$

L'espace algorithmique

Une fois les caractéristiques et le type de transformation identifiés, il faut définir une procédure qui, se basant sur les caractéristiques, permet de fixer les différents paramètres de l'espace de recherche. Encore une fois, il existe un large panel d'algorithmes mais que l'on peut diviser en deux groupes principaux :

- **Les algorithmes déterministes** sont des algorithmes qui calculent directement, après une simple analyse des caractéristiques, les paramètres de l'espace de recherche de la transformation à appliquer.
- **Les algorithmes itératifs** sont des algorithmes qui se basent sur le calcul d'un poids pour chaque ensemble de paramètres de l'espace de recherche et sur une recherche incrémentale de la transformation optimale selon ce poids.

Parmi les **algorithmes déterministes**, on pourra citer les méthodes basées sur la transformée de Fourier. L'idée de ces algorithmes est de passer d'un "domaine spatial" où chaque valeur d'intensité a une ou plusieurs positions dans l'image, vers un "domaine de fréquence" où chaque valeur d'intensité a une fréquence d'apparition dans l'image. Passer dans le domaine de fréquence[8] est intéressant car il existe plusieurs propriétés basées sur la comparaison de spectres de fréquence qui permettent de mettre directement en évidence des translations, des rotations et des dilatations entre deux spectres à comparer.

> La **transformée de Fourier** est ***une technique mathématique qui décompose une fonction du temps ou de l'espace en un spectre de fréquences, à la manière d'un prisme qui décompose la lumière en un spectre de couleurs.[Deb03]***

Le principe des méthodes peut être très brièvement résumé par les deux étapes suivantes :

- appliquer la transformée de Fourier aux zones correspondantes de l'images maître et esclave[9].
- utiliser les propriétés de celles-ci afin d'en extraire directement les paramètres des transformations.

Les **algorithmes itératifs** se basent sur la recherche incrémentale des paramètres. Pour ce faire, on utilise une fonction, appelée **fonctionnelle d'appariement**, qui prend en entrée les caractéristiques des deux images ainsi qu'une valeur pour chaque paramètre de recherche et qui détermine un poids à optimiser. On distingue deux classes de fonctionnelles d'appariement principales :

- les **fonctions à minimiser** qui représentent une mesure de la différence entre deux images recalées par la transformation définie par les valeurs des paramètres en entrée
- **fonctions à maximiser** qui représentent une mesure de la similitude entre deux images recalées par la transformation définie par les valeurs des paramètres en entrée.

Les équations 2.3 et 2.4 sont l'expression mathématique respectivement d'une fonction à minimiser basée sur la notion de distance et d'une fonction à maximiser basée sur la notion de coefficient de

[8] appelé aussi domaine de Fourier

[9] il existe un algorithme efficace ($o(n*log_2(n))$) permettant d'obtenir un spectre de fréquence appelé *FFT* (Fast Fourier Transform).

Deuxième partie

Matériel et méthode

Chapitre 3

L'infrastructure disponible

Ce court chapitre à pour but de présenter l'infrastructure disponible utilisée afin de réaliser les deux applications qui sont présentées dans les chapitres 5, 6 et 7 de la partie "Analyses et résultats". il décrit brièvement les appareils que l'on peut retrouver dans la station d'acquisition du laboratoire de la clinique de Mont-Godinne(la figure 3.1). Il décrit ensuite l'application d'analyse automatique de lame développée par Benoît George[1] qui a été un point de départ nécessaire à la compréhension du pilotage des différents appareils.

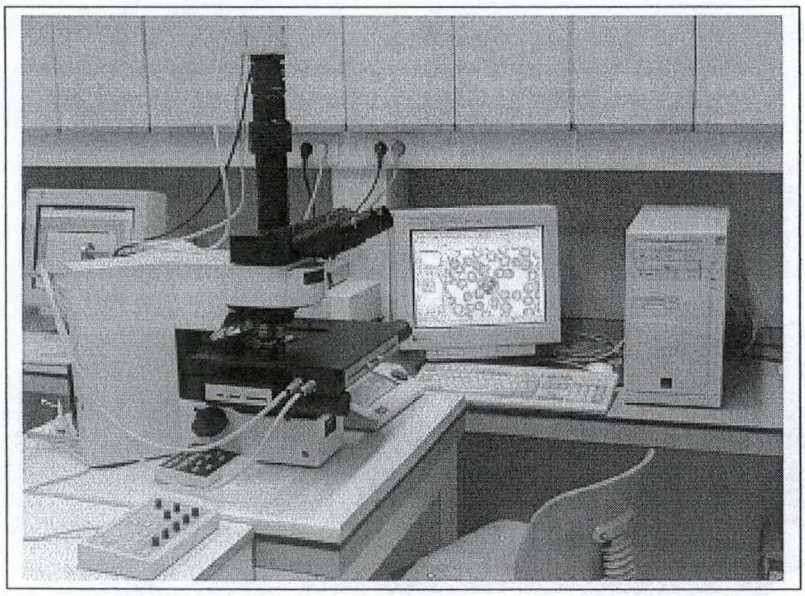


FIG. 3.1 – La station d'acquisition à Mont-Godinne.

3.1 Le microscope électronique

La station d'acquisition du laboratoire de la clinique de Mont-Godinne comprend principalement un microscope électronique AX70 d'Olympus.
Ce microscope a la particularité d'être entièrement motorisé. D'une part, un petit boîtier de commande

[1]Cette application a d'ailleurs fait l'objet en partie de son mémoire : [Geo02]

permet à la fois de diriger dans le plan vertical (axes "x" et "y") la platine sur laquelle repose la lame à observer et de changer le grossissement des objectifs. D'autre part, un panneau électronique U-MCB (*Unit-Main Control Board*) permet l'utilisation de fonctions plus avancées telles que le réglage du filtre de couleur, le réglage de la luminosité, etc.
De plus un ordinateur est relié au panneau U-MCB via un câble série, ce qui permet d'exécuter toutes les fonctions explicitées ci-dessus sur l'ordinateur via un jeu de commandes RS232.

Pour assurer l'acquisition des images, le système est doté une caméra analogique "Sony DXC-950" ainsi que d'un autofocus hardware "U-AF d'Olympus"[2] permettant le réglage de la netteté des images capturées par la caméra. La machine d'acquisition dispose d'un disque dur de 60 Go, de 640 Mo de mémoire RAM, d'un processeur de 730 Mhz. Le système d'exploitation installé sur cette machine est le système "Windows NT 4.0". La machine est connectée en permanence au réseau local et possède un accès à l'Internet. La station d'acquisition est illustrée par la figure 3.1[3].

3.2 Le logiciel Analysis

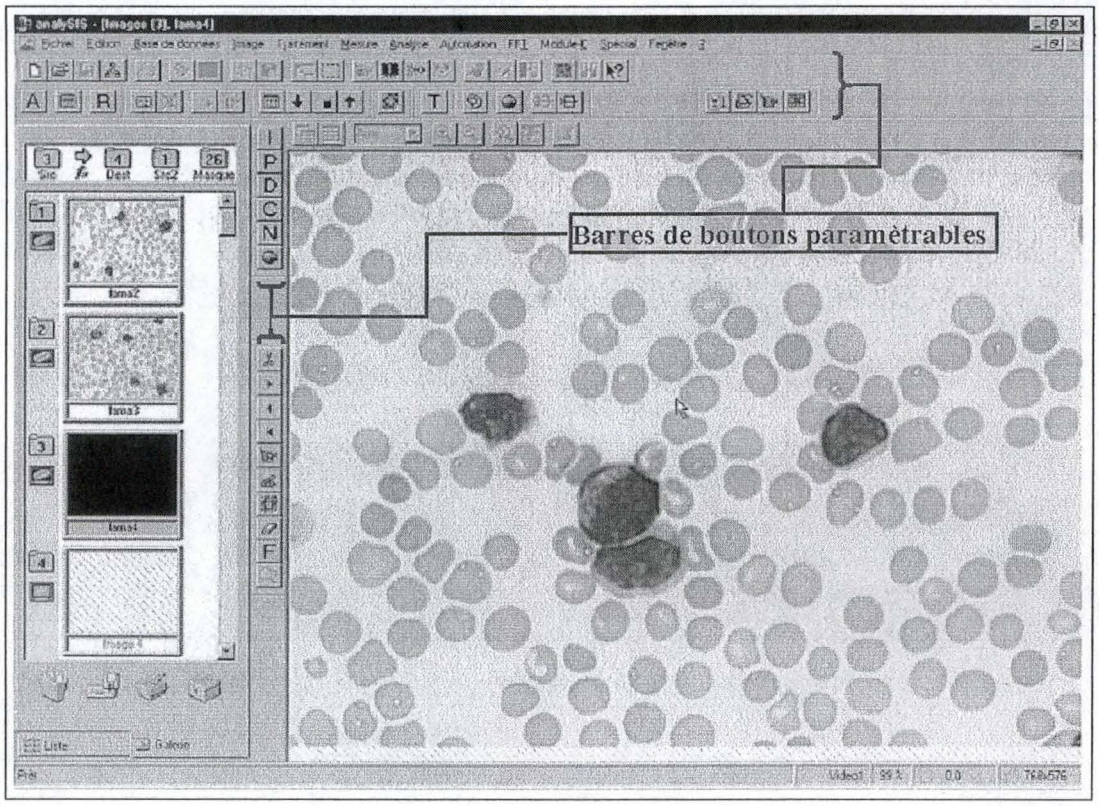


FIG. 3.2 – L'interface du logiciel Analysis.

Sur la machine de la station d'acquisition se trouve installé le logiciel "Analysis Pro 3.0". Ce logiciel est conçu pour l'acquisition, l'analyse, le traitement et l'archivage d'images.
Un avantage indéniable de ce logiciel est son haut niveau de paramétrisation. En effet, le logiciel permet le chargement dynamique de modules (librairies de fonctions) codés dans un langage proche du

[2]Cette autofocus est aussi relié à l'ordinateur mais via un autre cable série.
[3]Cette image est extraite de [Geo02].

standard ANSI-C appelé "Imaging-C". Le principal apport de cette variante du C par rapport au standard est la présence d'une série de fonctions supplémentaires liées au traitement d'images. De plus, Analysis comprend un éditeur de texte, un compilateur et un interpréteur permettant à l'utilisateur[4] de réaliser facilement ses propres modules et fonctions "Imaging-C".

La version d'Analysis présente sur la machine du laboratoire à Mont-Godinne possède en plus un module (*AX70*) très intéressant fournissant une interface permettant de lancer des commandes vers l'U-MCB du microscope électronique. Cette interface est en fait une librairie de fonctions permettant de créer des modules plus complexes faisant intervenir des actions sur le microscope.

Imaging-C est également compatible avec le "MS Windows Software Development Kit" (SDK) permettant la programmation de nouvelles interfaces graphiques dans l'environnement "Windows". Toutes les fonctions de base imaging-C ainsi que les fonctions développées dans les modules peuvent être intégrées facilement dans l'interface d'Analysis en les associant à des boutons ajoutés dans une barre de boutons ou à des items dans un menu. Comme le montre la figure 3.2.

3.3 L'autofocus

L'autofocus est un matériel extrêmement utile lorsque l'on fait de l'acquisition d'images puisqu'il permet de déterminer automatiquement la netteté des images.
Son fonctionnement est relativement simple : il consiste à faire une recherche incrémentale du plan focal. Ce dernier étant défini par la position de l'objectif par rapport à la platine (axe z) qui rend l'image la plus nette.

> Une **image nette** étant ***une image dont les contours des fins détails sont clairement distinguables. L'image doit donc être suffisamment contrastée que pour identifier les fins détails mais pas trop contrastée pour ne pas voir apparaître un effet de granulation. [Kow87]***

Pour réaliser cette recherche incrémentale de manière efficace, le calcul de la netteté est effectué non pas sur l'entièreté de l'image vue par l'objectif mais sur une zone restreinte et centrée par rapport à ce que voit l'objectif. Comme le montre la situation présentée dans la figure 3.3, il arrive que la zone de mise au point ne comprenne aucun élément contrasté. Dans cette situation, la recherche s'effectue dans un seul sens de l'axe "z" (axe de l'objectif) jusqu'à ce qu'un timer se termine et entraîne un bug système nécessitant le redémarrage de l'appareil. On peut cependant remarquer que si l'utilisation de l'autofocus à un fort grossissement et de manière automatique sur des zones aléatoires est déconseillée, elle ne pose en principe pas de problème pour des captures d'images résultant d'une analyse automatique de lame. En effet, dans ce cas, l'analyse automatique assure que les images à capturer comportent une cellule ou un globule en leur centre.

3.4 L'analyse automatique de lame

Les travaux réalisés par Benoit Georges dans son mémoire [Geo02] intitulé "*La télémicroscopie en cytologie hématologique*" et plus particulièrement la partie concernant l'analyse automatique de lame et la composition de galeries d'images ciblées constituent un bon exemple d'une application de télémicroscopie de type "store and forward"[5]. De plus, dans ces parties, on retrouve une utilisation

[4] par utilisateur, il faut évidemment comprendre ici un informaticien
[5] Cfr le chapitre 1.

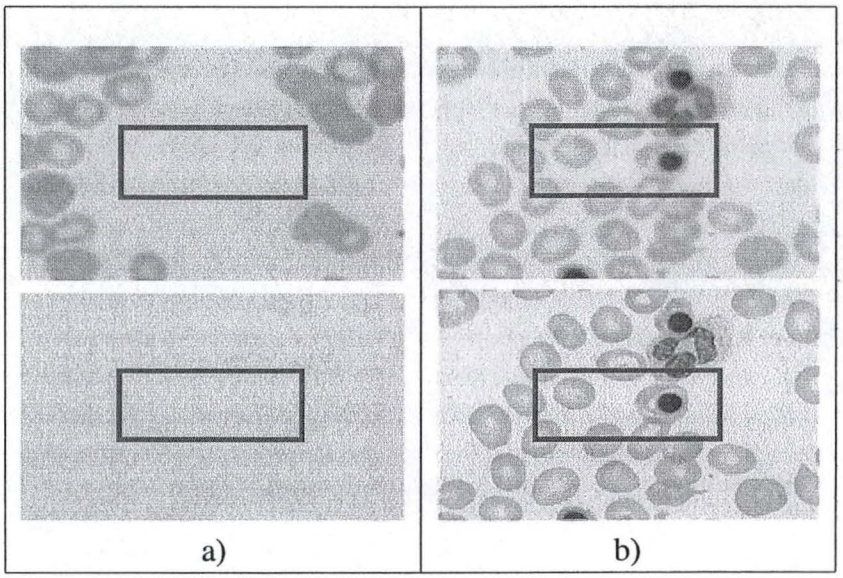


FIG. 3.3 – Lorsque la zone de mise au point ne comprend aucun élément contrasté, le focus est impossible.

assez poussée du logiciel Analysis et plus particulièrement du module AX70. Elles ont donc constitué une source d'information non négligeable sur le fonctionnement et les possibilités de ce logiciel.
On peut résumer le fonctionnement du programme d'analyse automatique de lame en trois phases :

- La sélection des images.
- L'acquisition des images.
- Le stockage et la composition de documents multimédia.

Cela dit, seules les deux premières phases sont représentatives d'un emploi des outils présents dans la station d'acquisition et seront donc discutées ici[6].

3.4.1 La sélection des images

La sélection des images est le processus qui va déterminer la position de chaque image à capturer sur base d'une analyse des propriétés de taille et de couleur des cellules visibles sur une zone déterminée sur la lame observée. Cette analyse est effectuée au grossissement 20X et la zone à scanner est déterminée par des paramètres introduits par l'utilisateur. L'utilisateur spécifie le nombre de "champs"[7] que le système doit parcourir ou le nombre minimal de zones d'intérêt que le système doit retenir pour une analyse automatique. Ensuite, le programme va parcourir tous les champs spécifiés et enregistrer les positions des images à capturer. La figure 3.4 extraite de [Geo02] illustre cette première partie du programme, on y voit une zone déterminée sur une lame et composée de "champs".

Cette première partie est intéressante car elle donne un moyen de parcourir une zone de la lame dont les dimensions sont spécifiées par un utilisateur.

[6]pour la troisième phase, se référer à [Geo02]

[7]Un champ correspond à une zone sur la lame dont les dimensions sont celles d'une image capturée avec la caméra.

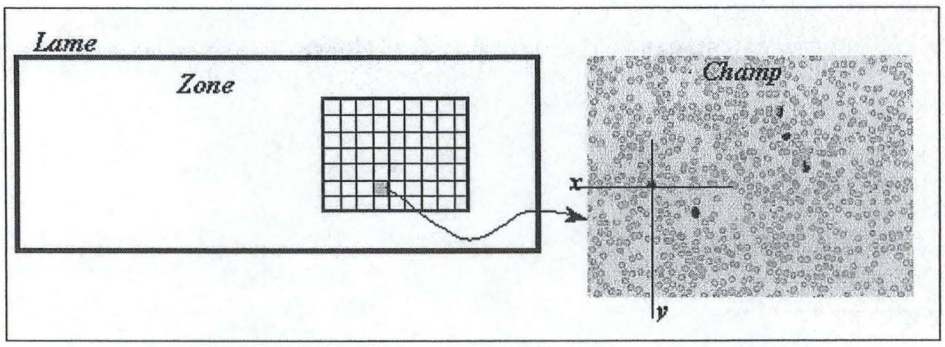


FIG. 3.4 – L'analyse automatique d'une lame : la sélection des zones d'intérêt.

3.4.2 L'acquisition des images

Une fois la sélection terminée, le système change d'objectif vers un grossissement plus fort, le 100X, et va, pour chaque position enregistrée dans la phase précédente, déplacer la platine à la bonne position, faire un focus et capturer une image. Les images sont alors stockées dans des buffers de Analysis jusqu'à ce que la capture de toutes les images soit terminée.

Cette deuxième partie est intéressante car elle montre comment capturer des images, comment changer le grossissement d'un objectif et elle illustre la gestion des buffers dans Analysis.

Chapitre 4

Les technologies utilisées

4.1 La représentation numérique d'une image

4.1.1 L'image numérique

L'image numérique est un des matériaux de base de la télémicroscopie. Dans un système de microscopie virtuel par exemple, l'image numérique est omniprésente : lors de la phase d'acquisition les images sont numérisées, ensuite pour chaque phase ultérieure (coregistration, stockage et lecture), des informations sont extraites de leur forme numérique. Il est donc essentiel d'en définir dès à présent les concepts principaux.

> La **numérisation** d'une image est ***la conversion de celle-ci de son état analogique (distribution continue d'intensités lumineuses dans un plan xOy) en une image numérique représentée par une matrice bidimensionnelle de valeurs numériques f(x,y) où x, y sont les coordonnées cartésiennes d'un point de l'image.[Kad99]***

4.1.2 Le pixel

Selon la définition précédente, l'image numérique est constituée d'un ensemble ou matrice d'éléments. Un tel élément est appelé **pixel**[1] et correspond à l'unité de couleur de l'image numérique. Chaque pixel est donc un chiffre représentant une couleur ou une nuance de gris. Il existe plusieurs formalismes pour coder[2] les couleurs, tous basés sur le fait que plus la taille en byte allouée à un pixel est grand, plus la palette des couleurs représentables est variée.
Par exemple, les pixels des images monochromes sont habituellement codées sur 1 octet ce qui permet la représentation de $2^8 = 256$ niveaux de gris différents[3], tandis que les pixels des images couleurs sont codées sur 3 octets dans le formalisme RGB (Red Green Blue), ce qui permet la représentation de $2^{24} = 16$ millions de couleurs différentes.

4.1.3 Les composantes d'un pixel

Cela dit, le pixel ne correspond pas à la plus petite information traitable de l'image numérique, celui-ci est en effet encore décomposable en plus petits éléments appelés les **composantes** du pixel.

[1] "**pixel**" est l'abréviation de "picture element".
[2] ou chiffrer
[3] En fait, même s'il existe plus de variante de gris dans la nature, l'oeil humain n'est pas capable d'en distinguer plus.

Le nombre de composantes d'un pixel dépend de la manière dont est il codé :

- En monochrome, le pixel n'a qu'une seule composante :"**gris**".
- Dans le formalisme RGB, le pixel a trois composantes : "**rouge**", "**vert**" et "**bleu**".
- Dans le formalisme RGBA, le pixel a quatre composantes : "**rouge**", "**vert**", "**bleu**" et "**transparence**".

Cela dit pour chaque modèle de représentation des couleurs, une composante est toujours codée sur un octet. De plus, si on peut parler d'une "**image RGB**" pour faire référence à une image dont les pixels sont codés dans le modèle RGB, on peut aussi parler de "**la composante rouge d'une image couleur**" pour faire référence à une matrice dont chaque élément (i, j) est la composante rouge du pixel correspondant à la position (i, j) de cette image.
La figure 4.1.3 représente "le cube RGB" habituellement utilisé pour définir les couleurs à partir des valeurs chiffrées des trois composante R, G et B. On peut y remarquer que les niveaux de gris se retrouvent sur la droite d'équation $R = G = B$, c'est-à-dire sur la droite transversale allant du point $(0, 0, 0)$ au point $(255, 255, 255)$ et comprenant 256 valeurs discrètes différentes.

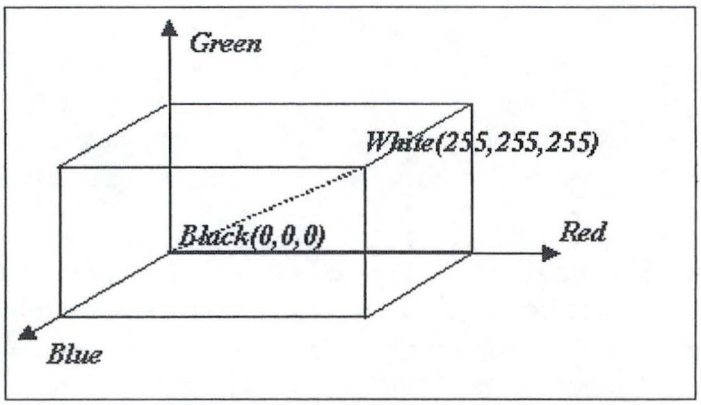


FIG. 4.1 – le modèle RGB

4.1.4 La luminance et le contraste d'une image

La luminance et le contraste sont deux propriétés essentielles d'une image. Ces deux propriétés sont définies à partir des niveaux de gris de l'image.

La luminance

> La **luminance** d'une image se définit comme étant ***la moyenne des valeurs de niveaux de gris de chaque pixel composant l'image.***

Donc, si l'on considère la représentation monochrome d'une image, on peut définir 256 niveaux de luminance différents. Une image ne contenant que des pixels noirs est l'image la moins lumineuse que l'on puisse créer et à l'inverse, une image ne contenant que des pixels blancs est l'image la plus lumineuse que l'on puisse créer. On peut donc définir une mesure de la luminance par l'équation suivante :

$$L = \frac{\sum_{i=0}^{N} G[i]}{N} \quad \text{Où } G[i] \text{ correspond au niveau de gris du pixel i.} \tag{4.1}$$

Le contraste

> Le **contraste** se définit comme ***l'opposition marquée entre deux régions d'une image, plus précisément entre une région sombre et une région claire de cette image.*** ***[Kad99]***

Si L1 et L2 sont les luminances respectivement de deux zones voisines A1 et A2 d'une image, le contraste C est défini par le rapport :

$$C = \frac{L2 - L1}{L1 + L2} \tag{4.2}$$

Cette définition définit formellement le contraste entre deux zones voisines d'une image, cependant on parle aussi du contraste pour qualifier une image entière. On retrouve une définition de cette notion plus subjective d'une "image fortement ou faiblement contrastée" dans [Fru95] :

> Une image **fortement contrastée** est une image qui ***présente une bonne dynamique de la distribution des valeurs de gris sur tout l'intervalle des valeurs possibles, avec des blancs bien clairs et des noirs profonds. Au contraire une image peu contrastée a une faible dynamique, la plupart des pixels ayant des valeurs de gris très proches.***

4.1.5 Les dimensions et le volume d'une image

Les **dimensions** d'une image numérique sont définies par deux chiffres représentant sa largeur et sa hauteur correspondant respectivement aux nombres de pixels constituant une ligne et une colonne de sa matrice. On parle aussi souvent de la **résolution** d'une image.

La résolution est aussi souvent désignée pour parler du "rendu d'une image par un média" (écran, imprimante, appareil photo, etc). Mais contrairement à la résolution d'une image, le rendu est exprimé en DPI (Dots Per Inch) et représente donc le nombre de points par unité de surface que peut représenter le média. Dans le cas d'un écran, par exemple, une image dont la résolution est de 100x100 pixels sera étalée sur une surface de 1 Inch carré si l'écran fait 100 DPI et sur une surface de 100x100 Inch carré si l'imprimante fait 1 DPI. On peut remarquer que sur un même média, plus la résolution d'une image est grande, plus son rendu est de bonne qualité.

Cela dit, plus la résolution d'une image est grande, plus son **volume**, i.e la place mémoire qu'elle occupe est grand. Si on connaît les dimensions d'une image et le formalisme de codage pour les pixels qu'elle utilise, on détermine aisément son volume.
Par exemple, une image dont la résolution est 5000 x 6000 et dont le codage est le modèle RGB a un volume de $5000 * 6000 * 3$ o $= 90$ Mo. Cela dit, même si le traitement d'une image entraîne souvent[4] la nécessité d'un tel volume en mémoire centrale, le volume occupé par l'image lorsqu'elle est stockée dans un disque dur, peut être considérablement diminuée par des techniques de compression.
La figure 4.1.5 résume les notions présentées dans cette section, elle représente une image numérique de dimension 4X3 pixels.

[4]La plupart des technologies obligent la lecture de l'image entière en mémoire centrale pour son traitement mais il existe des exceptions qui permettent la lecture d'une partie de l'image seulement. Voir la section 4.2 et la partie Résultat.

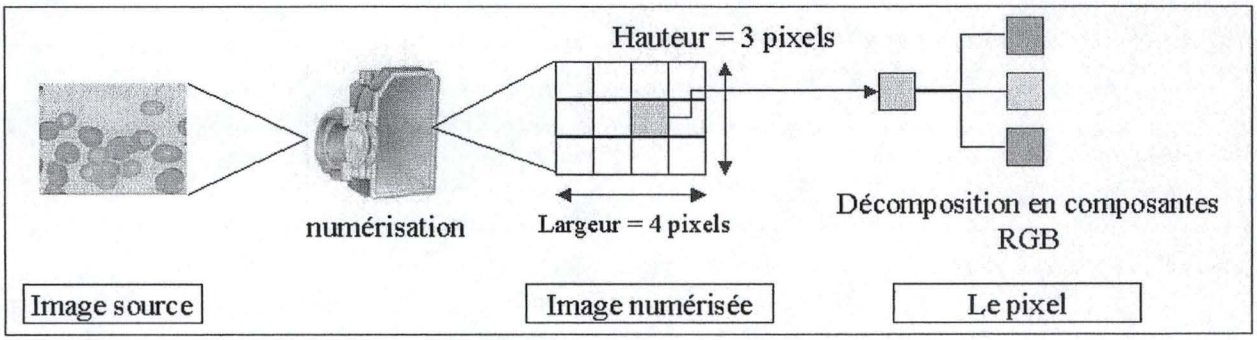


FIG. 4.2 – La numérisation d'une image et la décomposition en pixel.

4.2 JPEG2000

4.2.1 JPEG2000 et JJ2000

JPEG2000 est le dernier standard pour le codage des images fixes proposé par les groupes ISO (International Organization for Standardization) et IEC (International Electrotechnical commission) responsable du déjà très célèbre "JPEG" (Joint Photographic Experts Group). JPEG2000 est apparu pour faire face aux nouveaux besoins des applications d'imagerie numérique et au volume croissant des données images à manipuler.

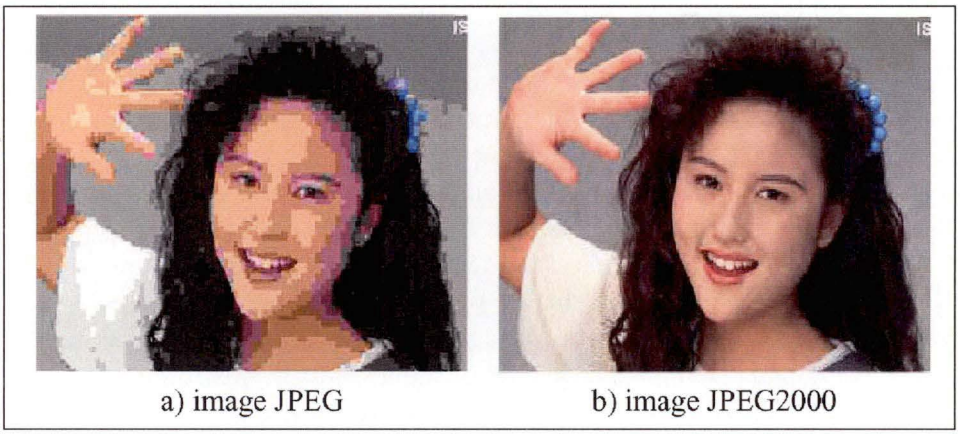
JJ2000 4.1 est une implémentation java du standard JPEG2000. Elle a été développée par l'EPFL (l'Ecole Polytechnique Fédérale de Lausanne), ERICSSON et Canon et terminée en septembre 2001. JJ2000 a fait l'objet de nombreux tests de validité et de concordance avec le standard et a été choisie comme l'une des deux implémentations officielles avec JASPER, une implémentation en C. [WebJpeg] [WebJJ2000] [WebJasper]

4.2.2 Les principaux avantages

- Une meilleure compression que le standard JPEG. voir figure 4.3
- Une compression avec et sans perte.
- Un algorithme optimisé pour la compression d'images naturelles.
- Une meilleure résistance aux erreurs de transmission.
- Une compression adaptée aux documents composés (texte et image par exemple).
- Des accès progressifs à l'image par précision ou par résolution.
- Des accès par R.O.I. (Region Of Interest) permettent par exemple de coder une partie de l'image de manière plus précise que le reste de l'image.

Curieusement, même en arborant de tels avantages par rapport à son prédécesseur, JPEG2000 n'a pas été conçu dans le but de remplacer JPEG déjà bien présent sur le Net[5] mais de le compléter en répondant à des besoins spécifiques.[WebJJ2000]

[5] 80% des images sur Internet sont au format JPEG.



a) image JPEG b) image JPEG2000

FIG. 4.3 – Reconstitution d'une image compressée en JPEG et en JPEG2000 avec taux de 0,125 bpp.

4.2.3 Les principes de l'encodeur

Le tiling

La première phase de l'encodeur JPEG2000 est d'effectuer de découper l'image source en éléments disjoints appelés "tiles". En fait, pour être plus précis, un "tiling" est effectué sur chaque composante de l'image source comme le montre la figure 4.4. Cette première étape définit donc les éléments de l'image (appelés "tile-components") qui doivent être codés indépendamment.

> Le terme **"tiling"** désigne ***la partition d'une image source en blocs rectangulaires disjoints et codés indépendamment (un peu comme s'ils étaient des images distinctes). [Chr00]***

Tous les éléments (ou "tiles") ainsi créés ont les mêmes dimensions sauf exception pour les bords de droite et du bas. Dans l'implémentation de l'encodeur JJ2000 par exemple, on retrouve un paramètre

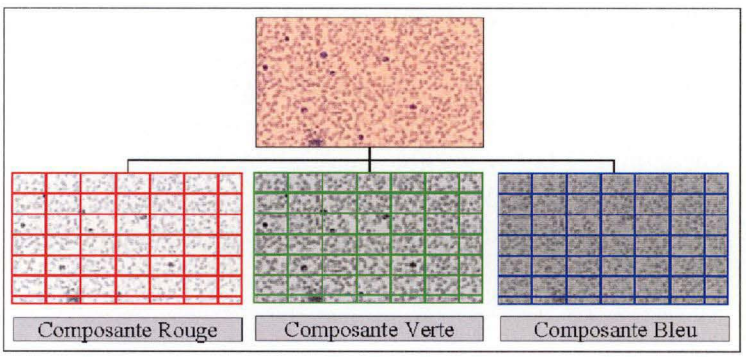


FIG. 4.4 – La découpe en tiles s'effectue selon les différentes composantes de l'image.

optionnel permettant de spécifier la longueur et la largeur des "tiles".
Le tiling permet une série d'avantages tels que le codage par R.O.I. ou la possibilité lors du décodage de décompresser une partie de l'image seulement.

La transformation en ondelettes discrète

Suite au "tiling", chaque élément "tile-component" est codé de manière indépendante. L'étape suivante de l'encodeur est d'effectuer une transformation appelée **"transformation en ondelettes discrète"** (DWT[6]) sur le flux de bits de chaque "tile-component". Le but de cette transformation est d'obtenir une structure plus facile à manipuler et notamment de pouvoir accéder plus facilement à l'image selon différents niveaux de résolution. Le but de cette section n'est pas d'expliquer comment fonctionne cette transformation mais de montrer son résultat et d'en expliquer sa portée.

Initialement, on a une représentation de l'image[7] sous la forme d'un tableau bidimensionnel dont les dimensions correspondent à la résolution de l'image. La DWT consiste à décomposer l'image en différents niveaux de résolution allant de 1, la résolution la plus forte, à N, la résolution la plus faible. Appelons "LL_1" l'image de départ. A chaque niveau "n" avec allant de 2 à N, correspond une version réduite et lissée de l'image initiale "LL_n" et trois images de détails (LH_n, HH_n, HL_n) représentant l'information nécessaire à la reconstruction de l'image du niveau inférieur. Ainsi, en composant "LL_n" avec LH_n, HH_n et HL_n on obtient l'image de résolution supérieure "LL_{n-1}".
Le résultat de la DWT sur une image est illustré par la figure 4.5. Dans cette figure, on peut constater

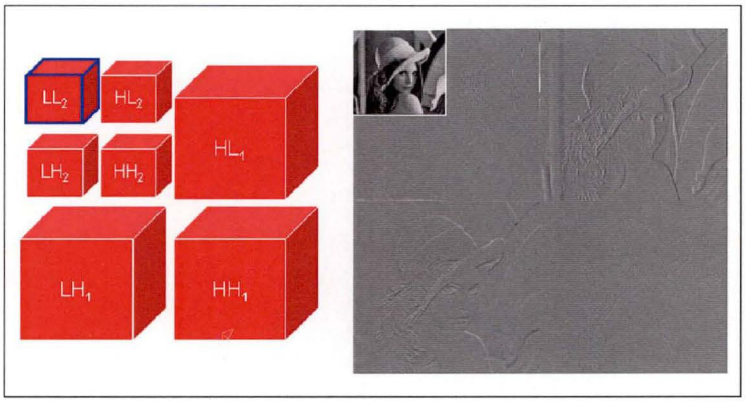


FIG. 4.5 – La décomposition en niveaux de résolution par DWT.

que seules les images de détails des différents niveaux sont représentées ainsi que l'image de résolution la plus faible, i.e celle du dernier niveau "N". Ceci est expliqué par le fait que les images réduites des niveaux intermédiaires "LL_n" peuvent toutes être dérivées par compositions successives de l'image "LL_N" avec les images de détail des niveaux supérieurs. L'avantage de cette représentation est évidemment un gain de place puisque cette représentation occupe exactement le même volume de l'image initiale. [Chr00][Sav01]

La compression JPEG2000

Après avoir été réorganisée en niveaux par la transformation en ondelettes, l'image doit encore être compressée. En fait, pour être tout à fait exact, la transformation en ondelettes commence déjà à réduire le volume de l'image. Mais des techniques supplémentaires sont appliquées sur l'output de celle-ci. Il s'agit principalement des techniques appelées "quantification" et "codage entropique" qui

[6]Discret Wavelet Transform
[7]Un tile-component pour être exacte.

ne seront pas expliquées ici[8] pour ne pas alourdir l'argumentation. Une étude qualitative nous semble meilleure pour apprécier une technique de compression. Dans [Kad99], l'auteur distingue trois critères d'évaluation pour toute méthode de compression :

- le taux de compression,
- la qualité de reconstitution de l'image,
- la rapidité du codeur et décodeur (codec).

Le critère de la qualité de reconstitution de l'image n'a évidemment de sens que pour les compressions avec perte. JPEG2000 comprend une seule technique de compression qui permet à la fois la compression avec ou sans perte. Si on spécifie une compression avec perte, on peut aussi définir le taux de compression de l'image. JPEG2000 propose des taux de compression très élevés. Cela dit, plus le taux est élevé, plus le risque que les images soient moins bien reconstituées augmente. Les deux premiers critères doivent donc être appréciés en parallèle. La figure 4.6 présente une image de frottis sanguin reconstituée après avoir été compressée à différents taux (ce taux est exprimé en bits par pixel). Mis à part le taux le plus fort, on ne remarque quasi pas de différence pour ce type d'images.

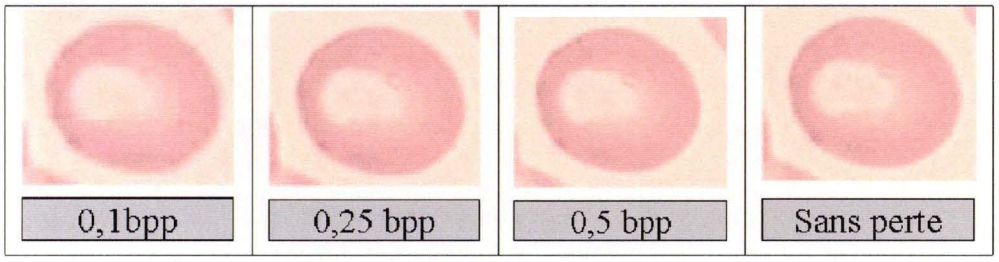


FIG. 4.6 – La reconstitution d'une image JPEG2000 compressée à différents taux.

La rapidité du codeur et du décodeur n'est certes pas une qualité de JPEG2000, c'est d'ailleurs sans doute une des raisons pour lesquelles ce standard n'a pas la prétention de vouloir remplacer JPEG. Cela dit, JPEG2000 a été conçu pour un traitement d'images spécifiques tel que les méga images par exemple.

4.2.4 Utilité pour les méga images

Comme déjà mentionné dans le chapitre 1, une image de frottis numérique représente un volume de mémoire important. Afin de faire face à ce problème, les problèmes principaux qui arrivent lors du stockage d'une image volumineuse sont de trouver un équilibre entre d'une part, un bon taux de compression et une qualité d'image acceptable après décompression et d'autre part, entre un bon taux de compression et la manipulation efficace des données compressées.

Un avantage indéniable de JPEG2000 est qu'il fournit des solutions aux deux problèmes. Tout d'abord, on peut constater une très bonne reconstitution des images après compression même pour des taux très fort (voir la figure 4.6). Et d'autre part, JPEG2000 fournit différents moyens de limiter le temps de visualisation d'une image grâce à différents points d'accès : on peut accéder à l'image selon les différents niveaux de résolution (ou de précision[9]) ou accéder à une région spécifique de l'image

[8] se référer à [Kad99] pour une définition simple de leurs principes ou à [Chr00] pour une description plus détaillée de leur implication dans JPEG2000.

[9] Les niveaux de précision correspondent aux niveaux de résolution. L'image de précision n correspond à l'image de résolution n redimensionnée.

(R.O.I.). Il est alors possible de définir un mode d'accès progressif (ou par "streaming") à l'image, c'est-à-dire suivant une progression, par l'un des différents modes énoncés ci-dessus. Ces trois modes progressifs sont illustrés par la figure 4.7.

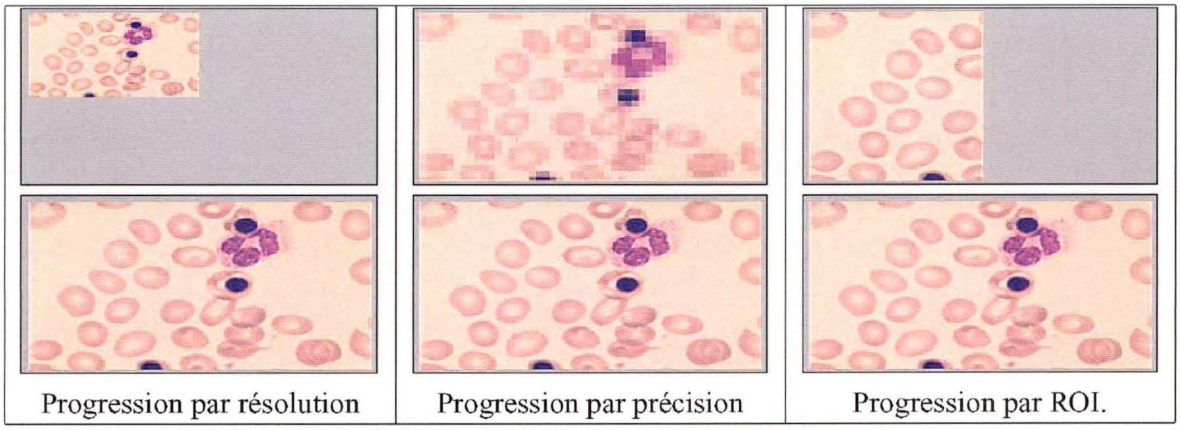


FIG. 4.7 – Les différentes façons de progresser dans une image JPEG2000.

Le **temps de visualisation** d'une image peut être défini comme ***le délai entre l'ouverture du fichier de l'image par un utilisateur et son affichage sur l'écran de l'utilisateur.***

Il comprend non seulement le temps de décompression, le temps d'affichage mais aussi dans le cas d'applications distribuées le temps de propagation de l'image sur un réseau. Bien que JPEG2000 offre différents modes d'accès progressif à l'image, il ne définit cependant pas d'architecture client-serveur ou de protocole pour le transport des portions d'image séléctionnées. Cela dit, il est toujours possible de définir une telle architecture et un tel protocole au-dessus de JPEG2000.[Des01][Mee03]

Troisième partie

Analyses et résultats

Chapitre 5

Les objectifs et les choix principaux

5.1 La présentation des objectifs

Le premier travail qu'il m'a été demandé de réaliser à l'hôpital universitaire de Mont-Godinne consistait en l'élaboration d'un système permettant la composition automatique et efficace de frottis numériques. La composition automatique de "frottis numériques" comprend généralement trois étapes auxquelles une attention particulière devait être apportée :

1. L'acquisition automatique d'un ensemble d'images qui vont composer le frottis numérique.
2. La détection d'éventuelles erreurs dans l'alignement de ces images et la correction de celles-ci.
3. L'organisation et le stockage de ces images en une méga image.

Un autre point requérant une attention particulière portait sur la qualité des images à produire. Le support constituant l'information à numériser était dans le cadre de mon travail un ***frottis sanguin*** (c'est-à-dire un étalement de sang sur une lame porte objet). La qualité d'une méga image de frottis sanguin est influencée par plusieurs facteurs que l'on peut regrouper en trois catégories :

- **catégorie 1** : Les facteurs liés au support et à l'objet observé : le frottis sanguin en l'occurence.
- **catégorie 2** : Les facteurs liés à l'acquisition des images.
- **catégorie 3** : Les facteurs liés à l'organisation des images assemblées.

Les premiers facteurs de qualité sont directement liés à la qualité des frottis. Par exemple, avant de pouvoir observer une lame au microscope, une coloration préalable qui met en évidence les structures cellulaires et permet leur identification est nécessaire.[Mer00] Le dosage de cette coloration constitue un premier facteur de qualité de la catégorie 1. Bien évidemment, cette première catégorie de facteurs est du ressort des spécialistes en cytologie et non des informaticiens.
Les facteurs qui constituent la deuxième catégorie sont les différents réglages effectués sur les appareils (microscope et autofocus) avant de capturer les images : il s'agit principalement des réglages de la luminosité et des filtres de couleur, du choix d'un objectif ainsi du réglage de la netteté des images. Ainsi, par exemple, les images de frottis sanguin doivent impérativement être capturées au **grossissement 100X** afin de fournir des images suffisamment précises que pour faire une analyse.
Enfin, les facteurs qui constituent la troisième catégorie sont : la technique de coregistration utilisée pour corriger les erreurs d'alignement ainsi que la technique de compression des images.

Ensuite, il m'a été demandé de réaliser un logiciel de visualisation permettant une navigation efficace mais aussi intuitive pour un cytologiste dans ces images particulièrement volumineuses.

Afin de répondre au premier objectif d'une "navigation efficace", la représentation des méga images au format JPEG2000 m'était imposée. Il m'a été demandé d'explorer les possibilités de ce nouveau standard en matière de gestion de méga images[1].
Le deuxième objectif, "une navigation intuitive", nécessitait la modélisation d'un minimum des fonctionnalités que l'on retrouve sur le microscope afin de reproduire l'environnement de travail du cytologiste. En ce sens, on parlera de "microscope virtuel". Les fonctionnalités principales suivantes ont été modélisées :

- le déplacement de la platine motorisée
- le changement d'objectifs
- l'application de filtres de couleur

La suite du chapitre présente l'architecture générale du système alors que les deux chapitres suivants explorent plus en détail successivement les applications "composition de frottis numérique" et "microscope virtuel".

5.2 Le choix d'une architecture distribuée

L'architecture générale du système, c'est-à-dire l'architecture commune aux deux applications présentées dans la section précédente, est une architecture basée sur le modèle client-serveur. Le choix de ce type d'architecture est en fait principalement poussé par une volonté d'intégrer directement les solutions des deux applications en un système de télémicroscopie.[2]

L'architecture distribuée, illustrée par le schéma de la figure 5.1, est composée de quatre modules distribués constituant deux applications client-serveur : le client-serveur de composition d'images et le client-serveur de visualisation d'images[3]. Comme le montre la figure 5.1, ces quatre composants sont distribués en trois stations localisées dans des endroits différents : la "station d'acquisition", la "station serveur" et "la station de visualisation". On peut remarquer sur le schéma que le système permet plusieurs stations de visualisation, c'est-à-dire que le serveur d'image peut servir plusieurs clients à la fois.

Afin de reproduire un système de télémicroscopie dont l'objectif principal est *de permettre une analyse à distance*[2], le développement d'un client-serveur pour la visualisation est indispensable. Par contre le choix d'un client-serveur pour la composition des méga images semble moins évident. Ce deuxième choix est en fait plus motivé par la restriction des ressources hardware de l'ordinateur de la station d'acquisition[4] que par un objectif de télémicroscopie. En effet, la détection et la compression d'une image est un processus coûteux en CPU et en RAM, de plus le stockage des images nécessite forcément un espace de stockage spécifique. Pour cette raison, il fut décidé de séparer la tâche d'acquisition des images des tâches de détection d'erreurs et de stockage en deux sites distants.

[1] Cfr le chapitre 4.

[2] Cfr le chapitre 1.

[3] le microscope virtuel

[4] Les caractéristiques de l'ordinateur de la station d'acquisition sont un disque dur de 60 Go, 640 Mo de mémoire RAM, un processeur de 730 Mhz.

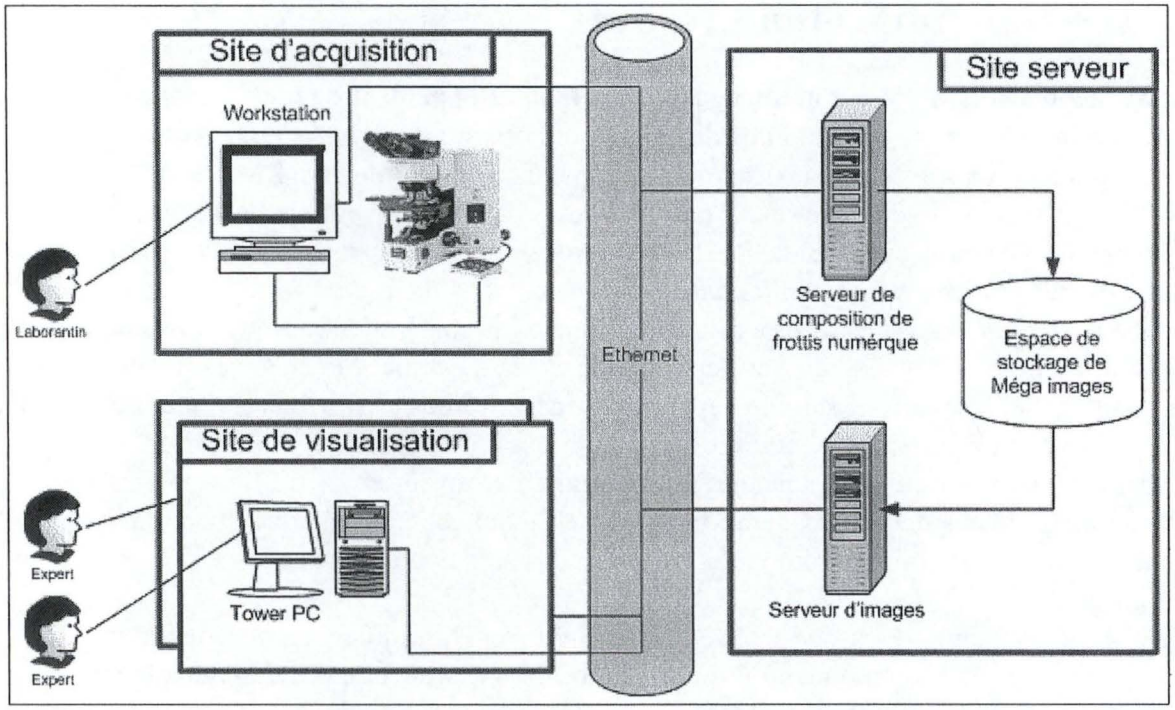


FIG. 5.1 – L'architecture du système.

5.3 Le choix du langage de programmation Java

Mis à part une partie du module de la station d'acquisition qui a été développée en C[5], le reste du système a été entièrement développé dans le langage Java. Outre une bonne connaissance de ma part de ce langage, Java possède plusieurs avantages intéressants :

Tout d'abord, le paradigme Orienté Objet se prête particulièrement bien à la conception de modules complexes permettant une découpe de ceux-ci en composants. Ce qui est fort utile pour le serveur de visualisation dont la structure relativement complexe est présentée dans le chapitre 7.

Ensuite, on pourra noter que java est un standard déjà bien établi. Il fournit un ensemble de classes réutilisables dignes d'intérêt et pour lesquelles une documentation claire et précise est disponible sur le site de Sun au format html[6]. Par exemple les classes ***Image, File, Thread, Vector*** se sont avérées particulièrement utiles.

Un autre avantage indéniable est évidemment sa portabilité. Il semble correct d'affirmer que les experts en cytologie qui sont familiers avec Unix sont peu nombreux et que par conséquent les modules clients sont en quelque sorte prédestinés à un environnement Microsoft. Par contre les modules serveurs pour lesquels la stabilité de l'environnement est une qualité indéniable sont probablement plus destinés à un environnement Unix.

Enfin, l'imposition d'utiliser JPEG2000 s'est quasi implicitement accompagnée du choix de sa version Java dont le code open source en facilitait sa compréhension et son utilisation.

[5]Cfr la section 5.5 pour une explication.

[6]La référence [WebJava1] est un lien vers cette documentation.

5.4 Les communications réseaux

Après avoir choisi Java comme langage pour le développement des différents modules il reste encore à définir comment ces modules distants vont communiquer entre eux. Afin de réaliser cette tâche simplement mais avec un maximum de transparence, le middleware RMI[7] a été choisi.
En effet, il est très facile de développer des applications client-serveur en Java grâce au mécanisme transparent d'invocation distante de RMI. Dit autrement, RMI permet "*de faire communiquer des objets java situés sur des machines distantes un peu comme s'ils étaient sur la même machine*".

Les seuls paramètres nécessaires à cette communication sont l'adresse IP de la machine du serveur ainsi qu'un numéro de port identifiant l'application sur la machine. Même si ces paramètres risquent de ne pas changer souvent, il semble cependant nécessaire de prévoir leur éventuelle modification afin de, par exemple, parer à une migration des serveurs. Il est cependant évident que si un utilisateur doit chaque fois introduire ces paramètres, il n'aura pas souvent envie d'utiliser l'application. Afin de tenir compte de ces deux considérations, chaque module client ainsi que chaque module serveur contient des fichiers de configuration comprenant ce type de paramètres qui ne doivent pas souvent être modifiés.[8]

Cela dit, le principal défaut de RMI est qu'il n'est pas un middleware interopérable. c'est-à-dire qu'il ne permet pas la communication directe entre modules développés dans des langages de programmation différents. Ce qui pose un problème pour le cas du module de la station d'acquisition qui est développé en C.

5.5 Le cas particulier de la station d'acquisition

Le chapitre 3 présente l'infrastructure disponible de la station d'acquisition. Dans ce chapitre, y sont mentionnés les avantages liés au logiciel Analysis et notamment en ce qui concerne la présence de la librairie de fonctions "Ax70" qui constitue un outil essentiel pour toute manipulation à distance du microscope. L'acquisition d'images rentre évidemment dans ce type de manipulation.

Le problème, c'est que cette librairie de fonctions n'est disponible que dans le langage C. Donc s'il est possible, et de manière relativement simple et efficace avec Analysis, de développer une application qui va acquérir les images souhaitées et les stocker dans des buffers, envoyer directement ces images avec RMI au serveur est impossible.
Cela dit, il existe plusieurs solutions à ce problème. Ecartant les solutions radicales qui consisteraient à coder l'ensemble de l'application soit dans le langage Java[9], soit dans le langage C[10], il reste encore deux solutions : La première est de laisser tomber RMI et passer à une solution middleware plus puissante telle que CORBA pour lequel l'interopérabilité est une des caractéristiques essentielles. La deuxième, beaucoup moins élégante, est d'utiliser un "*bidouillage informatique*". Malheureusement pour des raisons de limitation de temps, c'est la deuxième solution qui a été adoptée.

Cette solution consiste à joindre au module C, dont la tâche est restreinte à l'acquisition des images, un client java dont la tâche est restreinte à l'envoi des images vers le serveur. Reste encore le problème du transfert des images à partir du module C vers le client Java. Pour ce faire, les images sont préalablement sauvées sur disque dur dans un répertoire temporaire. Notons encore que c'est le module C qui initialise le client java et que ce dernier *meurt* lorsque les images ont été envoyées.

[7] Remote Method Invocation
[8] Ces fichiers sont présentés en annexe.
[9] Et donc perdre ce qui a déjà été développé de manière efficace dans les librairies de Analysis
[10] Et perdre la plupart des avantages énoncés ci-dessus.

Cette solution est présentée par le diagramme d'intéraction de la figure 5.2. Comme on peut le voir, les images sont envoyées par petits groupes au serveur, ce qui permet à celui-ci de commencer à travailler avant que l'acquisition des images soit terminée.

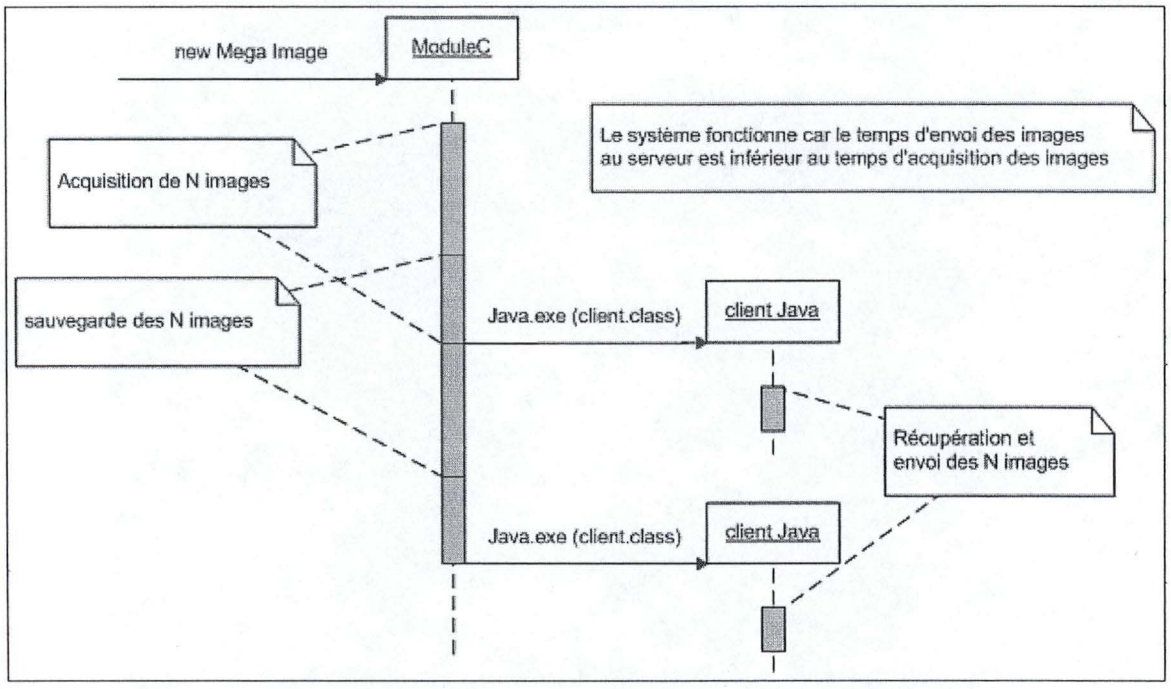


FIG. 5.2 – Le dialogue entre le module C et le client java.

Chapitre 6

La construction de frottis numériques

Ce chapitre présente les problèmes rencontrés et les solutions apportées durant l'élaboration de l'application "construction de frottis numérique" alors que le chapitre suivant présentera l'architecture du serveur et l'IHM du client de l'application "microscope virtuel". On peut résumer la construction de "frottis numériques" en trois étapes :

1. L'acquisition des images.
2. La coregistration des images.
3. L'organisation et le stockage des images.

Avant de détailler chacune de ces étapes, rappelons qu'il fut décidé de séparer la tâche d'acquisition des tâches de détection et de stockage en deux sites distants. Cfr le chapitre 5.
Notons aussi que, pour des raisons d'efficacité, ces trois étapes s'exécutent en parallèle dans la solution développée. Cela dit, pour faciliter l'argumentation, ce chapitre présentera ces trois étapes comme si elles s'exécutaient séquentiellement.

6.1 L'acquisition des images

L'étape **acquisition des images** consiste à capturer l'ensemble des images (en provenance du microscope) contiguës qui, une fois assemblées, vont constituer un frottis numérique. Cette première étape est la seule qui est réalisée dans la station d'acquisition et la seule qui nécessite l'introduction de paramètres par un utilisateur. L'utilisateur va manuellement centrer l'objectif sur un emplacement de la lame de son choix. Cet emplacement va alors définir la position centrale de son frottis numérique.[1]
Ensuite, l'utilisateur détermine les dimensions de son frottis en introduisant les deux valeurs suivantes : "nombre de champs en abscisse" et "nombre de champs en ordonnée".

> un **champ** est défini par ***une zone sur la lame ayant les dimensions d'une image capturée.***

Ces deux dimensions vont donc caractériser l'ensemble des images à capturer comme une matrice de champs ou plutôt, puisqu'un champ correspond à une image, comme une "matrice d'images". Ce résultat matricielle est illustré par la figure 6.1.

Après validation de ces paramètres par l'utilisateur, le logiciel va déterminer la position de la première image, c'est-à-dire l'image (1,1) dans la matrice, il va ensuite successivement capturer toutes

[1] On appelle cette première action une "calibration".

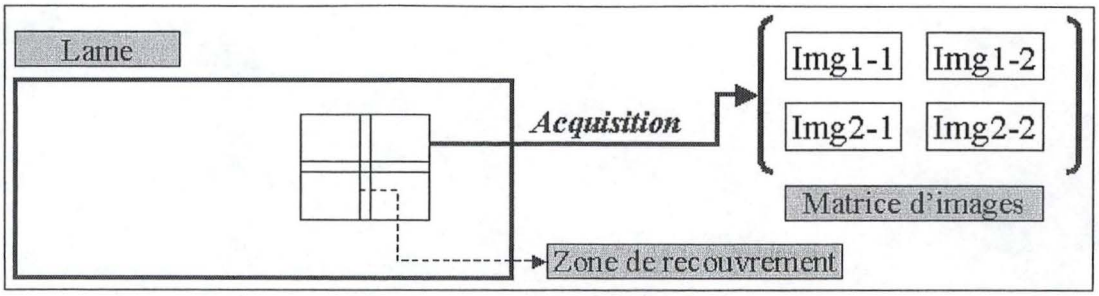


FIG. 6.1 – L'acquisition d'un frottis numérique composé de deux champs en abscisse et deux champs en ordonné.

les images en respectant un recouvrement de 100 pixels[2] et va les envoyer au serveur.

Notons encore qu'à Mont-Godinne, la caméra[3] permet de capturer des images avec une résolution de 768x573 pixels, ce qui donne un volume de $\approx 1,1MB$ pour chaque image capturée. De plus les images sont capturées avec le grossissement 100X.

Le résultat de la phase acquisition des images est une matrice d'images dans laquelle chaque image a une résolution de 768x573 pixels et recouvre une même zone d'observation de 100 pixels que ses images voisines dans la matrice.

6.2 La coregistration des images

Lors de l'étape d'acquisition, la position de chaque image est déterminée en utilisant le module AX70 de Analysis. Cela dit, on peut constater des erreurs dans l'alignement de chaque image par rapport aux positions demandées au module. La coregistration[4] va donc être nécessaire afin de corriger ces erreurs.

Cette section présente une analyse des erreurs spatiales observée et décrit l'observation d'erreurs photométriques. Elle présente ensuite la technique de coregistration utilisée afin de corriger ces erreurs.

6.2.1 Analyse des déformations spatiales

Rappelons qu'on entend par **déformation spatiale** d'une image esclave par rapport à une image maître, une erreur correspondant à un décalage spatial des valeurs des pixels de l'image esclave par rapport à celles de l'image maître. Cfr le chapitre2.

Afin de mesurer ce décalage entre chaque image capturée, considérons la construction d'un petit frottis numérique composé d'une matrice M de 4x4 images dans laquelle chaque image a une résolution de 768x573 pixels et est recouverte par ses voisines sur une zone de 100 pixels. On a donc 16 images $M(ij)$ avec $i, j \in [1, .., 4]$.

Sans erreur de décalage, on peut s'attendre à construire un frottis dont la résolution totale, c'est-à-dire après assemblage des images, est de exactement de 2772x1992 pixels[5]. Supposons maintenant

[2]La fonction d'un tel recouvrement est décrite dans le chapitre suivant.

[3]caméra SONY DXC950

[4]Cfr le chapitre 2.

[5]$2772 = ((4 * 768) - (3 * 100))$ et $1992 = ((4 * 573) - (3 * 100))$

que les images capturées ne sont pas parfaitement alignées et considérons les deux matrices suivantes qui ont la même dimension que M :

- P_{the} ou matrice des **positions théoriques** : qui reprend les positions des images de M telles qu'elles ont été capturées, c'est-à-dire sans correction des erreurs de décalage.
- P_{cor} ou matrice des **positions corrigées** : qui reprend les positions des images de M après avoir corrigé les erreurs[6] en effectuant seulement des translations.

On a donc, tenant compte des recouvrements et répertoriant les translations dans une matrice T 4x4 :

$$P_{the}(i,j) = ((i-1)*768-(i-1)*100,(j-1)*573-(i-1)*100) \quad (6.1)$$
$$P_{cor}(i,j) = P_{the}(i,j)-T(i,j) \quad (6.2)$$

La matrice de translation T représente alors **les décalages absolus** mesurables entre ces deux matrices et sera noté $DAbs$. Autrement dit, cette matrice représente les différences entre les positions théoriques et les positions réelles observées. Le tableau 6.1 reprend un exemple réel d'une matrice $DAbs$ relative à une matrice d'images de dimension 4x4 dans laquelle chaque image a une résolution de 768x573 pixels et est recouverte par ses voisines sur une zone de 100 pixels. Comme on peut le voir, il n'est pas facile d'en extraire une quelconque information.

TAB. 6.1 – Exemple d'une matrice $DAbs$ de dimension 4x4

(0,0)	(-5,30)	(-15,59)	(-35,87)
(-19,-10)	(-26,20)	(-35,49)	(-57,76)
(-39,-7)	(-45,23)	(-55,52)	(-78,79)
(-60,-18)	(-64,14)	(-77,43)	(-99,71)

Une manière de procéder pour en extraire de l'information utile est de ne considérer que des décalages entre deux images contiguës.

> On peut alors définir la matrice $DRel$ des **décalages relatifs** comme ***la matrice de décalages de chaque image par rapport à une image contiguë dans un sens donné.***

Si l'on considère toujours une matrice M d'images de dimension 4x4, $DRel$ est donc de dimension 3x4 si le sens est le sens horizontal ou 4x3 si le sens est vertical. Les matrices du tableau 6.2 correspondent aux décalages relatifs des images par rapport à leur image contiguë de gauche ($matriceA$) et par rapport à leur image contiguë du haut ($matriceB$). Elles sont obtenues à partir de la matrice $DAsb$ par soustraction des colonnes ou des lignes.

[6]par un recalage manuel par exemple.

TAB. 6.2 – Les deux matrices $DRel$ dérivées de la matrice $DAbs$ du tableau 6.1

matriceA			matriceB			
–	(-5,30)	(-10,29)	(-20,28)	–	–	–	–
–	(-7,30)	(-9,29)	(-22,27)	(-19,-10)	(-21,-10)	(-20,-10)	(-22,11)
–	(-6,30)	(-10,29)	(-23,27)	(-20,3)	(-19,3)	(-20,3)	(-21,3)
–	(-6,30)	(-11,29)	(-22,28)	(-21,-9)	(-21,-9)	(-22,-9)	(-21,-8)

Ces résultats bien que toujours difficiles à interpréter mettent néanmoins en évidence la présence de deux types d'erreur :

- **Les erreurs statiques** :
  La $matriceA$ nous dit que lorsqu'on se déplace en largeur de 768 pixels (c'est-à-dire la largeur d'une image) on observe une erreur de ± 29 pixels en hauteur. Inversement, la $matriceB$ nous dit que lorsqu'on se déplace en hauteur de 573 pixels (c'est-à-dire la hauteur d'une image) on observe une erreur de ± 21 pixels en largeur. Cela dit, un autre phénomène constant observé dans ces matrices est beaucoup plus difficile à interpréter : il s'agit du décalage observé en largeur lorsque l'on se déplace en largeur ($matriceA$) et du décalage observé en hauteur lorsque l'on se déplace en hauteur ($matriceB$). En effet, ceux-ci semblent dépendre de l'abscisse (resp. de l'ordonnée) du déplacement.
- **Les erreurs dynamiques** :
  Il correspond à une variation aléatoire de ± 3 pixels pour chaque position observée. Ce type d'erreur, bien que moins important que le premier (3 pixels par rapport à 30), est aléatoire et donc non prévisible.

Ces deux types d'erreurs sont difficiles à expliquer de manière rigoureuse, mais sont probablement dues à la précision de la platine motorisée, de la caméra et de la stabilité du microscope. En effet, les images sont capturées avec un grossissement 100x pour lequel un pixel correspond à $0,204\mu m$, il semble donc cohérent de penser que plusieurs faits prennent des proportions non négligeables à un tel niveau de précision :

- Une légère inclinaison la platine motorisée sur laquelle repose la lame par rapport au plan perpendiculaire à l'axe des objectifs.
- la précision avec laquelle la caméra va traduire deux points de lumière distincts en pixels, c'est-à-dire le phénomène de pixellisation.
- Les perturbations externes.
- La précision des mouvements de la platine.
- Etc.

Suite à ces observations, une détection automatique des positions de chaque image semble indispensable. Même s'il semble possible de limiter les erreurs observées en prévoyant deux variables de recalage pour faire face aux erreurs constantes, l'expérience a pu montrer que ces erreurs apparemment constantes pouvaient varier d'un jour à l'autre.

6.2.2 Observation de déformations photométriques

Rappelons qu'on entend par **déformation photométrique** d'une image esclave par rapport à une maître, une erreur correspondant à une variation dans l'intensité des valeurs des pixels correspondant dans les deux images. Cfr le chapitre2.
Les déformations photométriques bien que beaucoup moins visible à l'oeil nu que les erreurs spatiales rendent la détection de ces dernières plus difficile. Dans le cas des images en provenance du microscope, on peut observer, après avoir capturer plusieurs images, des résultats sensiblement différents en terme :

- de netteté,
- de teinte,
- de luminosité.

Ces différences se traduisent généralement par une variation uniforme dans les valeurs de pixels. Mais peut aussi correspondre à un phénomène de bruit qui se traduit par des taches de faible dimension dont la distribution sur les images est aléatoire.[Fru95]

6.2.3 La méthode de détection : deux images contiguës

Etant donné que les images sources capturées forment une matrice d'images, deux images peuvent être contiguës horizontalement ou verticalement, c'est pourquoi la méthode utilisée prévoit ces deux cas de figure. Cela dit, la méthode diffère peu d'un cas à l'autre[7] et l'exposé sera limité à l'élaboration d'une technique de détection dans le cas de deux images contiguës horizontalement.

La technique utilisée se base sur une analyse des valeurs de pixels[8] et détecte seulement les translations optimales à effectuer. Les observations ont en effet montré qu'aucunes rotation ou dilatation par un facteur d'échelle n'étaient nécessaires. Afin de trouver les translations optimales, la solution utilise une fonctionnelle d'appariement correspondant à une mesure de distance[9]. Pour rappel, cette méthode consiste à calculer un poids déterminant le degré de similitude entre deux zones d'images à comparer. Après avoir trouvé le candidat ayant le meilleur degré de similitude, les décalages sont calculés par une simple différence des positions relatives de la zone candidate et de la zone de référence dans la zone de recouvrement.

Afin de rendre possible la détection des décalages, un recouvrement entre chaque image contiguë est prévu lors de la capture des images. Celui-ci est illustré par les lignes noires dans la figure 6.2. Comme le recouvrement est aussi sujet aux erreurs, une zone de recouvrement suffisamment grande de 100 pixels est prévue.

La zone à comparer

Un premier choix, lié à toute technique de coregistration itérative et basée sur une analyse des valeurs de pixels, est de déterminer une zone de référence dans l'image maître à comparer avec des zones de même taille dans l'image esclave. Il est évident que si on compare la couleur du seul pixel, on n'est pas plus avancé puisque la couleur d'un pixel se retrouve plusieurs fois dans une image. Il est donc nécessaire de choisir une zone clairement identifiable dans les deux images. Un choix raisonnable pour cette zone semble être une ligne verticale (dans le cas d'une détection horizontale)

[7] il suffit de lire *ligne* à la place de *colonne* et *colonne* à la place de *ligne*
[8] caractéristiques iconiques par rapport à ce qui a été dit dans le chapitre 2.
[9] Cfr le chapitre 2.

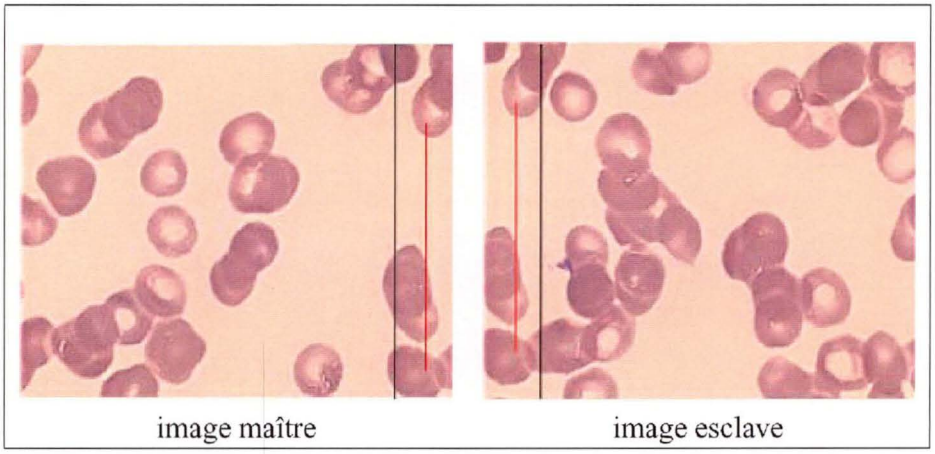


FIG. 6.2 – Lignes de détection (en rouge) pour deux images contiguës horizontalement.

centrée par rapport à la zone de recouvrement (voir les lignes rouges dans la figure 6.2).
Ce choix est en fait motivé par la prise en considération d'une série de facteurs :

- La taille de la zone de recouvrement : Les pixels comparables sont limités dans les deux images par les zones de recouvrement (en abscisse dans le cas d'un collage horizontal).[10]
- La nature des décalages : Ces erreurs ont aussi une influence sur le nombre de pixels réellement recouverts.
- Les caractéristiques de l'image : Les images de frottis sanguin sont des images qui peuvent être caractérisées par de grandes zones peu contrastées qui se prêtent peu à la comparaison et des zones plus petites et plus contrastées (cellules et globules) qui doivent faire partie de la zone de référence.

La fonctionnelle d'appariement

On a un tableau "M" de pixels de référence et un tableaux "I" de pixels candidats à comparer. La fonction $F(M, I)$ la plus simple consisterait à renvoyer la somme des différences entre les valeurs de pixels correspondant dans "M" et "I". Cela dit, la présence d'erreurs photométriques introduite ci-dessus, biaise les résultats engendrés par une technique aussi simple. Ce qui nous force donc à trouver une fonction de comparaison plus robuste qu'une simple fonction d'égalité.

La solution choisie est d'appliquer une série de "filtres"[11] sur les images maître et/ou esclaves avant de les comparer par une mesure de distance. La fonction "F" s'exprime alors de la manière suivante :

$$F(M, I) = \sum_{i=1}^{n} |FiltreA[T]_i - FiltreB[I]_i| \tag{6.3}$$

La mesure de la distance entre une référence et un candidat est égale à la somme des valeurs absolues des différences entre valeurs de pixels correspondantes de la zone de référence filtrée et de la zone candidate filtrée.

[10] On aurait pu augmenter encore la taille de la zone de recouvrement, mais alors le nombre d'images à capturer pour réaliser un frottis augmente.

[11] Cfr le chapitre 2

La solution candidate correspondant à la solution est celle qui minimise cette fonction. La suite présente les filtres choisis et appliqués aux images maître et esclave.

Filtre 1 : La réduction des couleurs en une échelle de gris

Un filtre couramment appliqué consiste à réduire les couleurs d'une image en échelle de gris. Pour ce faire, en se rappelant que dans le cube RGB les valeurs grises se situent sur la droite transversale allant du point noir $(0, 0, 0)$ au point blanc $(255, 255, 255)$, il suffit d'appliquer l'équation suivante à chaque pixel "i" de l'image :

$$gray(i) = \frac{R(i) + G(i) + B(i)}{3} \tag{6.4}$$

On pourra néanmoins remarquer que cette transformation comporte un certain risque puisqu'elle constitue une réduction d'information. En effet, on passe de 16 millions à seulement 256 valeurs différentes pour chaque pixel.

Cela dit, deux propriétés essentielles de l'image ne sont pas affectées par cette transformation, il s'agit de la luminance et du contraste. En effet, ces deux propriétés sont définies par rapport au niveau de gris des valeurs de pixel d'une image. Cfr le chapitre 4

Autrement dit, passer d'une image couleur à une image en niveau de gris sert à mettre en évidence le contraste et la luminance de l'image. De plus, on pourra remarquer que cette transformation permet de passer de valeurs tridimensionnelles (les trois composantes R,G et B) à des valeurs qui n'ont plus qu'une seule dimension (l'échelle de gris) et qui sont donc plus facilement manipulables. C'est pourquoi, la plupart des techniques de filtrage se basent d'abord sur une réduction de l'image en une échelle de gris qui n'est plus alors considérée comme un filtre proprement dit. *Filtrer une image c'est lui appliquer une transformation mathématique qui modifie les valeurs de gris de tout ou partie des pixels.*[Fru95]

Les images de frottis sanguin sont caractérisées par une grande zone peu contrastée (le fond) et de petites zones fortement contrastées (les cellules et les globules). Donc l'application de ce filtre aura comme effet de mettre en évidence les zones fortement contrastées et facilitera leur identification dans les deux images.

Notons encore qu'il est possible d'augmenter la différence entre ces zones et le fond en diminuant le nombre de valeurs de gris considérées. On parle alors de la segmentation d'une image. Cela dit, en diminuant le nombre de valeurs de gris, l'information constituant l'image est encore plus réduite. Après plusieurs essais, la segmentation s'est avérée inefficace dans le cas de la détection qui nous intéresse. La réduction en échelle de gris et la segmentation sont illustrées par la figure 6.3.

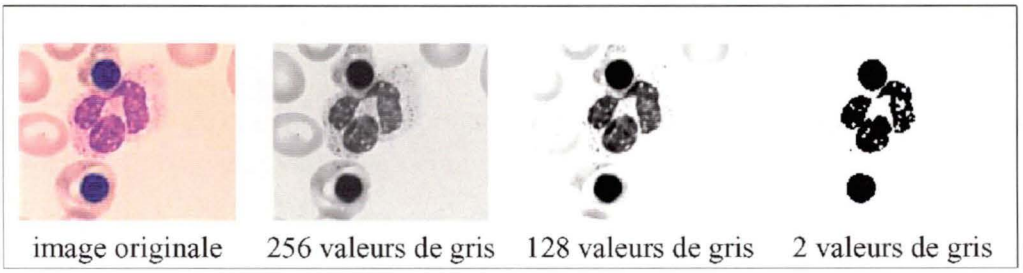


FIG. 6.3 – La segmentation par niveaux de gris.

Filtre 2 : La Normalisation

Un autre filtre couramment utilisé consiste à normaliser une image (l'image esclave) par rapport à une autre (l'image maître). Afin de comprendre l'utilité de ce filtre, rappelons la définition et quelques propriétés d'une norme :

La **norme d'un vecteur** X est ***le scalaire égal à sa longueur.***

$$||X|| = \sqrt{\sum_{i=1}^{n} X_i^2} \tag{6.5}$$

Normaliser un vecteur Y par rapport à un autre X consiste à ***le transformer en un vecteur*** $Ynorm$ ***dont la norme égale la norme de*** X ***et dont le rapport entre chaque partie scalaire est inchangé.***

$$Ynorm = \left(Y_1 * \frac{||X||}{||Y||} \quad \dots \quad Y_n * \frac{||X||}{||Y||} \right) \tag{6.6}$$

$$||Ynorm|| = ||X|| \tag{6.7}$$

$$\frac{Ynorm_i}{Ynorm_j} = \frac{Y_i}{Y_j} \tag{6.8}$$

Suite à ces définitions, on peut remarquer que les zones de pixels à comparer (la zone de référence et les zones candidates) peuvent être considérés comme des vecteurs de longueur n (où n est le nombre de pixels d'une zone). On peut donc écrire pour un candidat $Cand$ pris au hasard :

$$Ref = \left(Ref_1 \quad Ref_2 \quad \dots \quad Ref_n \right)^T \tag{6.9}$$

$$Cand = \left(Cand_1 \quad Cand_2 \quad \dots \quad Cand_n \right)^T \tag{6.10}$$

On a alors que pour tout i, $Cand_i$ et Ref_i sont des valeurs comprises entre 0 et 255 si l'on ne considère que l'échelle de gris.

Il est donc possible de normaliser chaque valeur "gris" des zones candidates par rapport à leur valeur respective dans la zone de référence. Les rapports entre les valeurs de chaque pixels d'une zone candidate ainsi modifiée ne seront pas changés. On peut remarquer que cette transformation appliquée sur une échelle de gris a pour effet d'adapter le contraste de l'image esclave au contraste de l'image maître.

Résumé de la technique de détection à 2 images

La technique choisie consiste en l'application de filtres "réduction à en niveau de gris" sur les images maître et esclave, suivie de l'application du filtre "normalisation" sur chaque vecteur candidat[12] de l'image esclave par rapport au vecteur de référence de l'image maître. Une fois l'application des filtres terminée, la mesure de similitude entre les images est déterminée par une simple soustraction des valeurs obtenues. Comme le montre le schéma de la figure 6.4, cette mesure détermine un poids à minimiser pour trouver la solution optimale.

Le résultat de la technique de détection entre une image maître et une image esclave contiguës est un l'indice de décalage en abscisse et en ordonnée à appliquer à l'image esclave pour qu'elle corresponde spatialement à l'image maître.

[12] pour rappel, il s'agit d'une sous colonne de la matrice image esclave

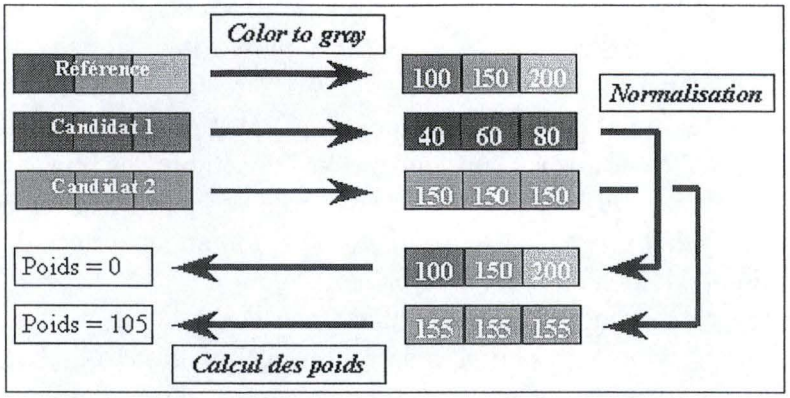


FIG. 6.4 – Application de la technique de détection pour deux vecteurs candidats.

6.2.4 La méthode de détection : généralisation à n images

Une méthode de coregistration considérant en entrée plus de deux images appelée aussi "mosaicking" consiste à déterminer la position de chaque image input dans un espace à deux dimensions dont l'origine est fixé par un point (le premier pixel en haut à gauche par exemple) d'une image. Autrement dit, si l'on considère que la première image se trouve à la position (0,0), par une suite de détections locales horizontales et verticales, on peut définir la position de chaque images.
On pourra noter que cette suite de détection n'est pas unique[13] puisqu'une image est généralement (sauf les images bords) contiguë à quatre autres.
La solution finale pourra donc être représentée par une matrice rectangulaire $(m * n)$ contenant les positions des différentes images.

Constat de la persistance d'erreurs après détection locales

Même si la solution proposée ci-dessus pour deux images fonctionne de manière satisfaisante dans la plupart des cas, certaines situations restent cependant problématiques. Il s'agit en fait de situations où la zone choisie comme référence dans la première image n'est pas assez contrastée par rapport aux zones adjacentes. On peut rajouter une procédure qui choisit dans la zone de recouvrement la meilleure zone de référence, mais il arrive que cette zone ne soit pas encore suffisamment contrastée. La situation la plus critique est en effet une situation où la zone de recouvrement ne contient pas de cellules comme dans la figure 6.5.

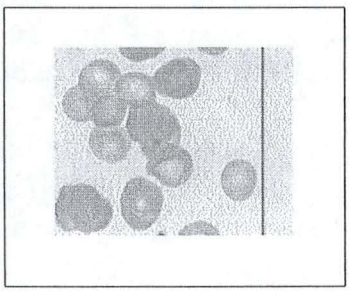


FIG. 6.5 – Exemple d'image dans laquelle la zone de recouvrement ne comprend pas de cellule.

[13] sauf dans les cas triviaux où la capture des images s'effectue en une ligne seulement ou en une colonne seulement.

Constat de la propagation des erreurs locales

Il faut quand même admettre que la situation de la figure 6.5 est extrêmement rare et que la plupart des erreurs observées peut paraître acceptable (quelques pixels de décalage) par rapport à la faible fréquence avec laquelle ces erreurs surviennent sur l'ensemble des images capturées.
Cela dit, lorsque l'on veut mettre bout à bout un ensemble de n images, les conséquences réelles d'un décalage local entre deux images ne se mesurent plus seulement sur ces deux images, mais sur plusieurs images par un phénomène de propagation des erreurs.

Pour s'en rendre compte reprenons l'exemple d'une matrice M 4x4 d'images $M(i,j)$ à coregistrer et définissons un premier algorithme pour la "détection des positions de chaque image". Afin de simplifier l'argumentation, considérons dans un premier temps que seules les abscisses de chaque image doivent être trouvées. Pour cette algorithme, nous disposons de :

- "$decAbs(A,B)$" : la méthode de détection des décalages entre deux images "A" et "B" énoncée ci-dessus mais restreinte aux décalages en abscisses.
- "L'" : la largeur "L" d'une image - le recouvrement de 100 pixel

Afin de trouver l'abscisse $Abs(i,j)$ de chaque images, on peut procéder de la manière suivante :

```
begin
Abs(1,1) := (0,0)
pour tout i tq 1<i<=4 :
        Abs(i,1) := Abs(i-1,1) - decAbs(M(i-1,1),M(i,1))
pour tout i tq 1<i<=4 :
        pour tout j tq 1<=j<=4 :
                Abs(i,j) := Abs(i,j-1) + L' - decAbs(M(i,j-1),M(i,j))
end
```

En fait, un algorithme analogue pourrait être appliqué pour déterminer les ordonnées $Ord(i,j)$ de chaque image donnant ainsi une première solution matricielle des positions de chaque image dans la méga image :

$$PSol1(i,j) = (Abs(i,j), Ord(i,j)) \tag{6.11}$$

Cette première solution est illustrée dans la figure 6.6[14]. Les images y sont représentées par des rectangles dans lesquels des numéros correspondent à l'ordre dans lequel leur position est déterminée dans la méga image. Les flèches vertes représentent les détections locales pointant vers les images qui ont servi de référence.

La figure 6.7 montre qu'avec cette solution, une seule erreur liée à une détection locale a des répercussions sur une ligne entière de la méga image. En fait, le sens de propagation des erreurs est déterminé par le choix de la suite de détections locales horizontales et/ou verticales dans l'algorithme de détection. Avec cette solution, les positions des images sont détectées en effectuant un maximum de détections locales horizontales. On aurait pu imaginer d'autres solutions plus farfelues, en augmentant le nombre de détections verticales par rapport aux détections horizontales par exemple, mais pour

[14]Pour ne pas alourdir le schéma, les zones de recouvrement n'ont pas été représentées.

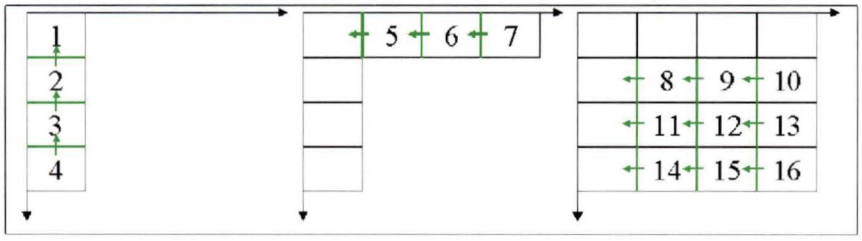


FIG. 6.6 – Les étapes du premier algorithme de détection des positions de n images.

lesquelles un phénomène de propagation, bien que moins facilement représentable sur un schéma, sera toujours présent.

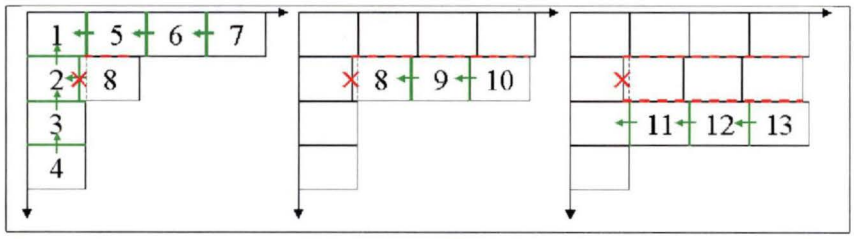


FIG. 6.7 – La propagation d'une erreur avec le premier algorithme de détection des positions de n images.

Dans la figure 6.6 on remarque que la position des images 8 à 16 peut être déterminée par une détection horizontale comme cela a été fait dans le premier algorithme mais aussi par une détection verticale. Cela dit, la détection locale est sujette à des erreurs et deux détections différentes pour la position d'une même image pourront donner lieu à des résultats contradictoires. La solution qui a été adoptée consiste à effectuer toutes les détections horizontales et verticales pour les images 8 à 16 et à appliquer ensuite un algorithme d'optimisation globale sur base de ces deux ensembles de résultats.

Solution : optimisation globale

Reprenons notre matrice M de 4x4 images et définissons les variables suivantes :

- Le vecteur ligne $L1$ qui reprend les positions des images de la première ligne de la matrice M.
- Le vecteur colonne $C1$ qui reprend les positions des images de la première colonne de la matrice M.
- La matrice $HDec$ de dimension 4x4 qui reprend les décalages déterminés par des détections horizontales uniquement tel que :

$$HDec(i,j) = \begin{cases} \text{le décalage donné par la détection entre } M(i,j) \text{ et } M(i,j-1) & \text{si } j > 1. \\ 0 & \text{sinon.} \end{cases} \tag{6.12}$$

- La matrice $VDec$ de dimension 4x4 reprenant les décalages déterminés par des détections ver-

ticales uniquement tel que :

$$VDec(i,j) = \begin{cases} \text{le décalage donné par la détection entre } M(i,j) \text{ et } M(i-1,j) & \text{si } i > 1. \\ 0 & \text{sinon.} \end{cases} \tag{6.13}$$

Les matrices $HDec$ et $VDec$ correspondent donc à deux critères de base différents et permettent chacun de prédire un résultat pour la position de chaque image. Afin de mettre en relation ces deux critères, définissons les deux matrices suivantes :

> $HPos$ et $VPos$ sont ***les matrices des positions de chaque image dans l'espace à 2 dimensions ayant comme origine le premier pixel de la première image de la matrice*** M ***et résultant de la correction des décalages prédits par l'un et l'autre de ces 2 critères.***

Afin de définir mathématiquement ces deux matrices, considérons encore les vecteurs de constantes suivants qui représentent les distances sans recalage entre chaque image d'une ligne (respectivement d'une colonne) de M :

- le vecteur $HCst$ défini par[15] : $((largeur - 100, 0), ..., (largeur - 100, 0))^T$
- le vecteur $VCst$ défini par : $((0, hauteur - 100), ..., (0, hauteur - 100))$

Les matrices $HPos$ et $VPos$ peuvent alors être définies de manière récursive par les équations suivantes :

$$HPos(-,j) = \begin{cases} C1 & \text{si } j = 1 \\ HPos(-,j-1) + HCst + HDec(-,j) & \text{si } j > 1 \end{cases} \tag{6.14}$$

$$VPos(i,-) = \begin{cases} L1 & \text{si } i = 1 \\ VPos(i-1,-) + VCst + VDec(i,-) & \text{si } i > 1 \end{cases} \tag{6.15}$$

Rappelons encore que les détections locales sont sources d'erreurs et que, par conséquent, ces deux résultats ne seront pas identiques. Une deuxième solution pourrait alors être de prendre comme résultat la moyenne des résultats de $HPos$ et de $VPos$. On aurait alors :

$$PSol2(i,j) = \frac{HPos(i,j) + VPos(i,j)}{2} \tag{6.16}$$

Cela dit, cette solution ne serait pas bien meilleure que la première. En effet, si on reprend l'exemple de la figure 6.7 avec une prédiction erronée de deux pixels pour la détection horizontale de l'image 8 et une prédiction correcte pour la détection verticale, la moyenne des deux résultats est une erreur de 1 pixels qui sera toujours propagées sur une ligne.
La solution finale consiste à résoudre un problème de minimisation en se basant non pas sur les matrices de positions prédites $HPos$ et $VPos$ mais sur les critères $HDec$ et $VDec$ directement. Intuitivement, il s'agit de trouver *la matrice de positions qui respecte au maximum les prédictions du critère de détection horizontale* $HDec$ *et du critère de détection verticale* $VDec$.

Soit donc $CPos$ une **matrice candidate de positions** pour cette solution. Il est évident que cette matrice comprend déjà $L1$ et $C1$ communes à $HPos$ et à $VPos$, de plus, comme $CPos$ est une

[15] 100, pour le recouvrement des images

matrice de positions elle peut être exprimée de la même manière récursive que dans les équations 6.14 et 6.15 :

$$CPos(-,j) = \begin{cases} C1 & \text{si } j = 1 \\ CPos(-,j-1) + HCst + CHDec(-,j) & \text{si } j > 1 \end{cases} \tag{6.17}$$

$$CPos(i,-) = \begin{cases} L1 & \text{si } i = 1 \\ CPos(i-1,-) + VCst + CVDec(i,-) & \text{si } i > 1 \end{cases} \tag{6.18}$$

Ce qui permet de définir des matrices de translation $CHDec$ et $CVDec$ directement comparables avec les matrices $HDec$ et $VDec$:

$$CHDec(-,j) = \begin{cases} 0 & \text{si } j = 1 \\ CPos(-,j) - (CPos(-,j-1) + HCst) & \text{si } j > 1 \end{cases} \tag{6.19}$$

$$CVDec(i,-) = \begin{cases} 0 & \text{si } j = 1 \\ CPos(i,-) - (CPos(i-1,-) + VCst) & \text{si } j > 1 \end{cases} \tag{6.20}$$

Intuitivement, $CHDec$ (resp. $CVDec$) représente la matrice **de décalage** correspondant à $CPos$ *si on considère que seules des détections horizontales (resp. verticales) ont été effectuées*. On peut alors déterminer un poids à minimiser pour chaque matrice de candidat de la manière suivante :

$$Poids(c) \stackrel{def}{=} |CHDec - HDec| + |CVDec - VDec| \stackrel{not}{=} |DH(c)| + |DV(c)| \tag{6.21}$$

Autrement dit, la solution finale est déterminée par :

$$PSol3 = CPos \text{ pour lequel le poids } Poids(c) \text{ est minimal} \tag{6.22}$$

Il est évident ce problème de minimisation est complexe tant le nombre de candidats possibles est grand. Cela dit, on dispose néanmoins de deux candidats potentiels $HPos$ et $VPos$ qui permettent de commencer les recherches avec une solution proche de la solution idéale. De plus, afin de minimiser les recherches de la solution, il est possible de définir des relations entre les trois matrices $CPos$, $DH(c)$ et $DV(c)$. Afin de comprendre les relations entre ces trois matrices, on peut observer l'effet que produit des perturbations dans la matrice $CPos$ sur les matrices $DH(c)$ et $DV(c)$. Dans ce but, définissons les perturbations suivantes :

$PLg(\alpha, i, j, nb)$	le décalage de "α" pixels sur les positions de la sous ligne de longueur "nb" et commençant à l'élément (i, j) dans la matrice CPos.
$PLg(\alpha, i)$	le décalage de "α" pixels sur les positions de la ligne "i" de la matrice CPos.
$PCo(\alpha, i, j, nb)$	le décalage de "α" pixels sur les positions de la sous colonne de longueur "nb" et commençant à l'élément (i, j) de la matrice CPos.
$PCo(\alpha, j)$	le décalage de "α" pixels sur les positions de la colonne "j" de la matrice CPos.

Ces relations sont illustrées par les matrices du tableau 6.3. En effet, même si leurs démonstrations sont assez simples[16], l'expression mathématique de ces relations est assez lourde et on préférera voir leur effet directement sur les matrices.

[16]Elles découlent des définitions récursives des matrices !

TAB. 6.3 – l'effet produit par des perturbations de la matrice $CPos$ sur les matrices $DH(c)$ et $DV(c)$

Relation	$CPos$	$DH(c)$	$DV(c)$
$PLg(\alpha,2,2,2)$	$\begin{vmatrix} * & * & * & * \\ * & *+\alpha & *+\alpha & * \\ * & * & * & * \\ * & * & * & * \end{vmatrix}$	$\begin{vmatrix} 0 & 0 & 0 & 0 \\ 0 & *+\alpha & * & *-\alpha \\ 0 & * & * & * \\ 0 & * & * & * \end{vmatrix}$	$\begin{vmatrix} 0 & 0 & 0 & 0 \\ 0 & *+\alpha & *+\alpha & * \\ 0 & *-\alpha & *-\alpha & * \\ 0 & * & * & * \end{vmatrix}$
$PCo(\alpha,2,2,2)$	$\begin{vmatrix} * & * & * & * \\ * & *+\alpha & * & * \\ * & *+\alpha & * & * \\ * & * & * & * \end{vmatrix}$	$\begin{vmatrix} 0 & 0 & 0 & 0 \\ 0 & *+\alpha & * & * \\ 0 & * & * & * \\ 0 & *-\alpha & * & * \end{vmatrix}$	$\begin{vmatrix} 0 & 0 & 0 & 0 \\ 0 & *+\alpha & *-\alpha & * \\ 0 & *+\alpha & *-\alpha & * \\ 0 & * & * & * \end{vmatrix}$
$PLg(\alpha,2)$	$\begin{vmatrix} * & * & * & * \\ *+\alpha & *+\alpha & *+\alpha & *+\alpha \\ * & * & * & * \\ * & * & * & * \end{vmatrix}$	$\begin{vmatrix} 0 & 0 & 0 & 0 \\ 0 & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \end{vmatrix}$	$\begin{vmatrix} 0 & 0 & 0 & 0 \\ 0 & *+\alpha & *+\alpha & *+\alpha \\ 0 & *-\alpha & *-\alpha & *-\alpha \\ 0 & * & * & * \end{vmatrix}$
$PCo(\alpha,2)$	$\begin{vmatrix} * & *+\alpha & * & * \\ * & *+\alpha & * & * \\ * & *+\alpha & * & * \\ * & *+\alpha & * & * \end{vmatrix}$	$\begin{vmatrix} 0 & 0 & 0 & 0 \\ 0 & *+\alpha & *-\alpha & * \\ 0 & *+\alpha & *-\alpha & * \\ 0 & *+\alpha & *-\alpha & * \end{vmatrix}$	$\begin{vmatrix} 0 & 0 & 0 & 0 \\ 0 & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \end{vmatrix}$

L'algorithme utilisé pour résoudre ce problème est présenté et illustré par un exemple en annexe sous l'intitulé **La procédure de réajustement**. Il choisit $HPos$ comme premier candidat pour la solution et consiste en l'application séquentielle de ces propriétés dans un ordre visant à minimiser $Poids(c)$.

Le résultat de la technique de coregistration appliquée à une matrice M d'images est une matrice de même dimension que M et qui reprend les positions (abscisses et ordonnées) de chaque image dans un espace à deux dimensions dont l'origine est le premier pixel de la première image de M (c'est-à-dire $M(1,1)$).

6.3 L'assemblage et le stockage des images

La dernière étape dans le processus de construction de frottis virtuels est le stockage des images sources "organisées". Suite aux étapes d'acquisition et de détection, on dispose d'un ensemble d'images liées sémantiquement par une matrice de positions absolues. On dispose alors de tous les éléments pour assembler les images input en une "méga image". Cette partie est assez triviale si l'on considère

les images sous leur forme numérique, en effet on a alors un ensemble de matrices de pixels représentables sous forme de tableaux bidimensionnels. L'assemblage ne revient donc plus qu'à placer dans un grand tableau les valeurs des pixels des différentes images aux positions déterminées dans l'étape précédente. L'assemblage de quatre images en une méga image est illustrée dans la figure 6.8.

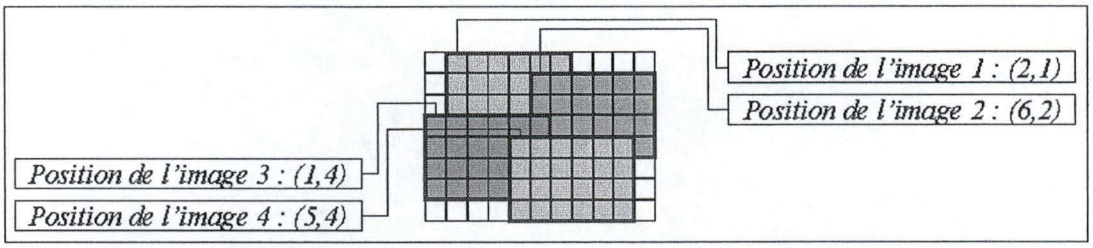


FIG. 6.8 – L'assemblage de quatre images en une méga image.

Une fois les images assemblées, il ne reste plus qu'à compresser et à stocker la méga image obtenue sur un disque dur. Afin de ne pas altérer les images, celles-ci sont compressées, par mesure de sécurité, avec la technique de compression sans perte de JPG2000.[17]

Rappelons aussi que l'assemblage des images sources en une méga image n'est pas une phase obligatoire (Cfr le chapitre 1) dans la construction d'un frottis virtuel mais il rend sa structure et sa navigation ultérieure beaucoup plus simple. En effet, l'autre solution qui consiste à sauver directement les images input sur le disque nécessite aussi la sauvegarde d'une représentation de la matrice des positions.

Enfin notons que, les avantages de JPEG2000 ne se limitent pas à fournir une représentation entière et déjà structurée de la méga image, il permet aussi, comme nous allons le voir dans le chapitre suivant, d'améliorer la navigation dans cette image volumineuse.

[17]Cfr le chapitre 4.

Chapitre 7

Le microscope virtuel

Alors que le chapitre précédent explorait les résultats liés à la modélisation de l'objet d'analyse en microscopie (le "frottis"), ce chapitre a pour but de présenter les résultats liés à la modélisation de l'outil permettant de faire des analyses sur cet objet (le "microscope").
Rappelons que le choix d'une application distribuée pour le microscope virtuel a été principalement motivé par la volonté de satisfaire un des objectifs principaux de la télémicroscopie. A savoir,*faciliter la possibilité pour un expert d'effectuer des analyses à distance.*

De plus, il est important de souligner que les choix effectués lors de la réalisation de cette application ont été motivés par l'hypothèse de la présence d'une machine puissante (disposant de beaucoup de ressources disque, mémoire et CPU) pour le serveur et des faibles ressources disponibles sur les pc clients c'est à dire les machines des experts analystes.

La première partie de ce chapitre présente le fonctionnement et l'architecture du serveur. L'objectif principal poursuivi lors de l'élaboration de ce serveur a été de *fournir un ensemble de requêtes visant à rendre la modélisation du microscope la plus fidèle possible.* Malheureusement, comme nous le verrons par la suite, en poursuivant cet objectif, ce sont surtout les problèmes liés à la minimisation des délais d'accès aux méga images qui furent attaqués au détriment des problèmes liés à la limitation des ressources mémoire des clients.

Ensuite ce chapitre se termine en présentant l'interface homme-machine du client en insistant sur les fonctionnalités du microscope qui ont été modélisées.

7.1 Les principaux désavantages des méga images

Les méga images, caractérisées par un volume important, sont difficiles à manipuler efficacement et cela pour deux raisons principales :

1. La nécessité d'une place en mémoire centrale importante pour l'image lue car pour être traitée, une image doit être sous son format brut :

   Une image RGB dont la résolution est de $10000 * 4000$ *pixels occupe un volume brut de* $10000 * 4000 * 3B = 120$ *MB.*

2. Un délai de visualisation important qui dépend aussi directement du nombre de pixels qui composent l'image.

   Le délai de visualisation correspond ***au temps de décompression + temps de chargement en mémoire + temps d'affichage.***

Cette difficulté est renforcée dans les applications distribuées. On peut en effet relever deux facteurs supplémentaires qui influencent l'efficacité avec laquelle les méga images peuvent être manipulées :

1. Le réseau reliant le client au serveur a un débit limité et est rarement réservé à l'application.
2. D'autre part, si l'on veut que les images soient accessibles simultanément par plusieurs utilisateurs, le risque de dépasser les ressources disponibles au niveau du serveur augmente.

Dans ce type d'application, soit les images sont d'abord décompressées, chargées en mémoire au niveau du serveur, puis envoyées au client, soit les images compressées sont directement envoyées au client. L'envoi d'images compressées au client est une solution abondamment utilisée dans le cas d'images de taille raisonnable. Cependant, dans le cas de méga images et sous l'hypothèse de machines non puissantes pour les clients cette solution perd de son utilité puisque les machines clientes ne permettent pas de décompresser ces images volumineuses dans des délais acceptables.
La solution utilisée est un compromis utilisant les moyens d'accéder de manière progressive à une image JPEG2000[1] qui permettent au serveur d'envoyer des parties d'images décompressées au client sans surcharger le réseau.

7.2 La répartition d'une image grâce à JPEG2000

Afin de comprendre le fonctionnement du serveur, introduisons un nouveau concept, "la répartition d'une image".

> L'idée de la **"répartition"** est ***de séparer l'image en éléments disjoints et de les répartir entre différents états.***

Evidemment, ce concept n'est réalisable que s'il est possible de décomposer l'image source en éléments disjoints. Cela dit, dans notre cas, les images utilisées sont codées[2] en respectant la norme JPEG2000 dont l'une des caractéristiques principales est le "tiling" qui permet l'accès à des zones spécifiques de l'image. (voir le chapitre 4)
Comme le montre la figure 7.1, une telle zone, appelée **bloc**, est la composition d'éléments unitaires,

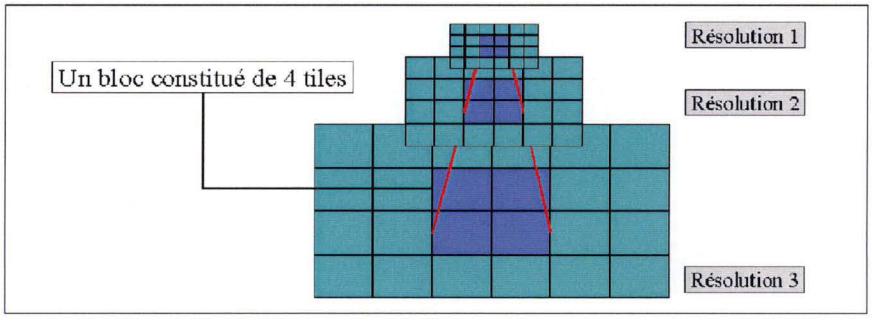


FIG. 7.1 – Le tiling de JPEG2000 permet l'accès à des blocs selon différents niveaux de résolution.

les **tiles**, un peu comme une image numérique est la composition de pixels. De plus, ces **blocs** sont accessibles dans tous les niveaux de résolution qui peuvent être définis dans une image JPEG2000.

> Par la suite, lorsque nous ferons mention d'un **bloc**, ***nous ferons toujours référence à un ensemble de tiles mais restreint à un niveau de résolution donné.***

[1] Cfr le chapitre 4.
[2] grâce au codec JJ2000

Autrement dit, dans la figure 7.1, nous considérerons trois blocs différents pour la partie de l'image représentée en bleu, c'est-à-dire un pour chaque niveau de résolution.

Après avoir observé qu'il était possible de séparer une image JPEG2000 en blocs, il nous reste à définir les différents états dans lesquels ils peuvent se retrouver. Pour ce faire, la section suivante présente une schématisation grossière des étapes par lesquelles passe habituellement une image durant le processus de sa visualisation : "Le cycle classique de visualisation d'une image", et montre ensuite que ce schéma n'est pas adapté à l'utilisation de méga images.

7.2.1 Le cycle classique de visualisation d'une image

La visualisation classique d'une image peut être grossièrement schématisée par un cycle de trois états par lesquels passe l'image concernée :

1. L'image entière n'est pas utilisée.
2. L'image entière est chargée en mémoire centrale.
3. L'image entière est "consommée".

Lors de l'état 1, on retrouve une seule instance de l'image stockée dans un disque dur, ensuite, lors de l'état 2, une copie de l'image se trouve également en mémoire centrale prête à être utilisée et enfin, lors de l'état 3, la mémoire centrale est de nouveau libérée. Les passages entre ces trois états sont modélisés dans le diagramme état-transition de la figure 7.2.

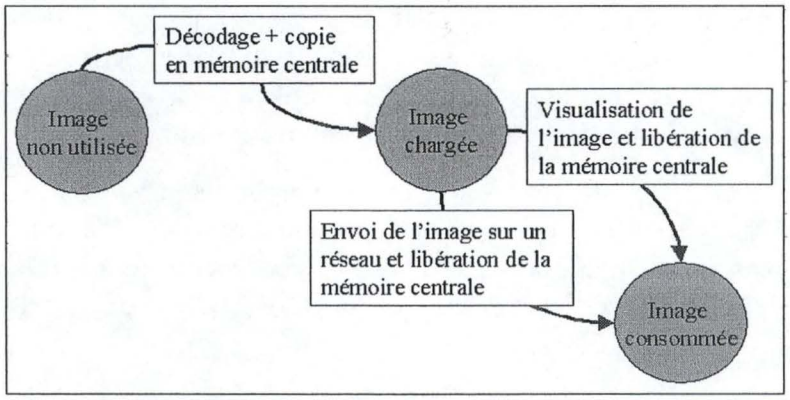


FIG. 7.2 – Le cycle classique de visualisation d'une image.

Les désavantages des méga images énoncés précédemment peuvent alors être réexpliqués dans les transitions entre ces états. Considérons le cas d'une application client-serveur :

- Le délai du passage entre l'état 1 et l'état 2 correspond au temps de décodage de l'image entière.
- La place mémoire du serveur utilisée augmente de façon considérable pendant la transition entre les deux premiers états jusqu'à atteindre le volume brut de la méga image lue. Elle n'est libérée totalement que dans l'état 3.
- L'état 2 se termine par un envoi massif de bytes sur le réseau.

Autrement dit, même si cette schématisation est assez grossière, elle permet d'un part, de mettre en évidence l'inadaptation du cycle classique de visualisation d'une image lorsque celle-ci atteint un certain volume (comme pour les méga images) et d'autre part, de définir trois états pour la répartition des blocs. En effet, si on considère un niveau de résolution donné, il est possible, comme le montre le schéma de la figure 7.3, de répartir une image JPEG2000 en plusieurs ensembles disjoints de blocs.

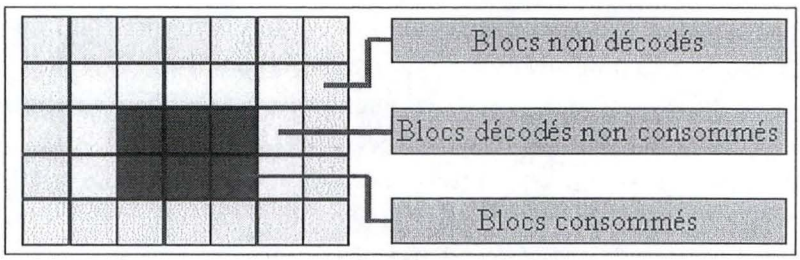


FIG. 7.3 – Exemple de la répartition des *blocs* selon les trois états du cycle de visualisation.

Si on considère l'image JPEG2000 sous son format pyramidal par niveaux, on peut affiner cette découpe en blocs aussi par niveaux de résolution, les blocs de résolution les plus faibles étant obtenus avant les blocs de résolution les plus forts.

Les avantages de cette technique sont multiples et sont basés sur le fait que la visualisation d'une image ne correspond plus à un seul cycle long et gourmand en mémoire pour le serveur mais en une suite séquentielle de cycles (par blocs) plus courts et requérant moins de mémoire :

- Cette technique permet d'obtenir des portions d'images dans des délais plus courts.
- Cette technique permet au serveur de pouvoir gérer plusieurs images en parallèle et donc servir plusieurs clients car le volume utilisé par une image en mémoire centrale est moindre.
- Cette technique permet aussi d'envoyer des portions plus petites sur le réseau.
- En outre, elle permet de limiter le décodage de l'image à des zones spécifiques appelées aussi *Regions Of Interest* dans la littérature. Lors d'une analyse, tout le frottis n'est jamais passé au peigne fin au grossissement le plus fort. En fait, seules certaines zones qui sont reconnues par le spécialiste doivent réellement être analysées en profondeur.

Cela dit, comme on peut le remarquer au niveau du client, ce sont surtout les problèmes liés à la minimisation des délais d'accès aux méga images qui sont attaqués au détriment des problèmes liés à la limitation des ressources mémoire. En effet selon cette découpe en trois états, les blocs sont consommés lorsqu'ils sont envoyés au client.

7.3 Le fonctionnement du serveur

Cette section présente le **fonctionnement du serveur** en présentant les différents accès effectués par un client lors de la visualisation d'un frottis numérique et en décrivant comment le concept d'image répartie a été utilisé.

7.3.1 Fonctionnement général

Les accès effectués par un client peuvent être résumés aux "changements d'objectif" et aux déplacements "latéraux dans le frottis".

Le changement d'objectif peut être facilement modélisé par un accès correspondant au changement de la résolution de l'image. Etant donné que les méga images utilisées ont été encodées avec le codec JJ2000 qui permet d'accéder aux images selon 5 niveaux de résolution, on peut modéliser 5 objectifs différents.

Pour ce qui est des déplacements latéraux, lorsque l'image entière est directement disponible au niveau du client, aucun accès supplémentaire au serveur n'est nécessaire. Cela dit, même si c'est le cas

pour les niveaux de résolution les plus faibles, à chaque changement d'objectif, le volume total de l'image à décoder par le serveur augmente et il arrive un objectif à partir duquel il devient nécessaire de *répartir l'image en blocs*. A partir de ce moment, l'utilisateur du client qui n'a effectué que des changements d'objectif ne dispose plus d'une version entière de l'image et s'il veut toujours pouvoir effectuer des déplacements latéraux sur l'entièreté du frottis, un autre type d'accès devient nécessaire : le "décodage de blocs adjacents" à la partie de l'image reçue.

Ces deux types d'accès sont représentés par les transitions du diagramme de la figure 7.4. Chaque état du diagramme représente un niveau de résolution allant de 1 à 5. On peut remarquer que le premier niveau de résolution ne permet pas de déplacements latéraux parce qu'une image entière est toujours renvoyée au client pour ce premier niveau. En effet, à ce premier niveau correspond une version de l'image de taille raisonnable : 2^5 fois plus petite que la taille d'origine.

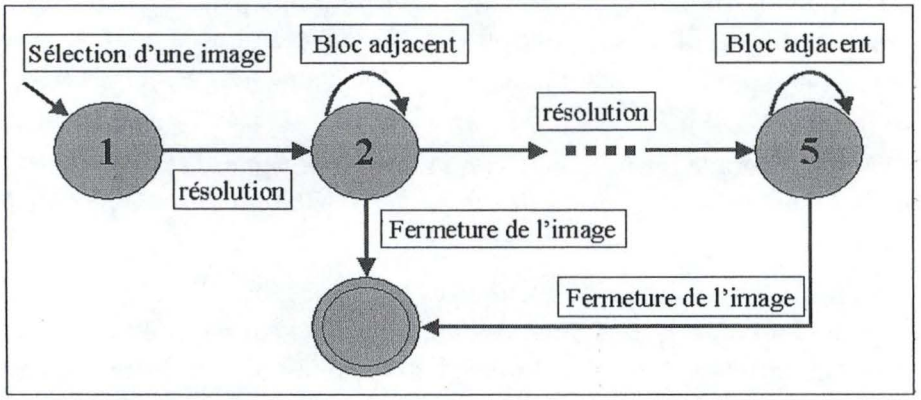


FIG. 7.4 – Les principaux accès d'un client au serveur.

En ce qui concerne la gestion des blocs par le serveur : les blocs envoyés aux clients sont considérés comme des blocs "*consommés*" et ne sont donc pas conservés en mémoire centrale. C'est à dire que le serveur ne décode un même bloc qu'une seule fois durant le processus de visualisation d'une image. Par contre les blocs qui n'ont pas encore été requis par le client sont décodés à l'avance et conservés jusqu'à leur demande ou jusqu'au passage au niveau de résolution supérieur.

7.3.2 L'accès "changement d'objectif"

L'accès **changement d'objectif** correspond à une requête visant à obtenir une version de l'image décodée à un niveau de résolution plus fort.[3] Comme nous l'avons dit aussi, il arrive un objectif ou un niveau de résolution à partir duquel l'image renvoyée au client n'est plus l'image entière mais une partie (ou bloc) de celle-ci. La première étape du serveur, suite à la réception de ce type de requêtes, est donc de déterminer les dimensions que doit avoir le bloc à renvoyer au client.

Afin de déterminer ces dimensions, il fut décidé de choisir un facteur minimal correspondant aux dimensions de la fenêtre d'affichage du client. En se rappelant qu'un bloc est composé de tiles, on peut définir les dimensions du bloc à renvoyer comme étant égales aux dimensions du plus petit bloc qu'il est possible de créer avec des dimensions supérieures à celles de la fenêtre d'affichage du client. De cette manière l'utilisateur qui vient de faire un changement d'objectif verra toujours une image complète.

[3]Comme nous allons le voir plus loin, pour modéliser un changement vers un objectif moins fort, un accès au serveur n'est pas nécessaire. Cfr la section 7.5.

L'accès "changement d'objectif" peut alors être représenté par une commande nécessitant deux paramètres dont le but est de permettre au serveur de déterminer la position et la taille minimale du bloc à envoyer au client :

```
Image Zoom(windowDimension, currentCenterPos)
```

- **Le bloc résultat** correspond à un objet de type Image.
- **windowDimension** correspond à la taille de la fenêtre du client qui affiche l'image.
- **CurrentCenterPos** correspond à la position du point sur lequel on veut zoomer.

Comme dit précédemment, un bloc qui a déjà été requis une fois par un client est considéré comme *consommé*, ce qui veut dire qu'il n'est pas nécessaire de le garder en mémoire au niveau du serveur. Seules sa position relative dans l'image totale et sa dimension (longueur et largeur en pixels) sont conservées.

> Par la suite, on utilisera le terme **image centrale** pour désigner ***la partie de l'image déjà "consommée" par le client. C'est à dire la partie de l'image formée par les blocs déjà reçus par le client.***

Cela dit, afin d'anticiper les accès du client aux blocs contigus à l'**image centrale** en limitant le délai entraîné par leur chargement, le serveur va suite à un accès ***zoom()***, décoder à l'aide d'un *Thread* tous les tiles du niveau de résolution courant et les stocker en mémoire centrale afin de pouvoir les redistribuer directement au client lorsqu'il les réclamera.

7.3.3 L'accès "récupération de blocs"

Comme déjà mentionné plus haut, le résultat du premier accès du client ne correspond à l'image entière que pour les faibles niveaux de résolution. Dans les autres cas, le client peut faire des accès supplémentaires pour récupérer les blocs manquants (c'est-à-dire contigus à l'**image centrale**).
Afin de procurer une certaine transparence au client vis à vis de la structure interne du serveur, les accès du client sont seulement identifiés par leur direction par rapport à l'**image centrale**. Un accès pourra donc être représenté par la commande :

```
Image getBloc(direction).
```

- **Le bloc résultat** correspond à un objet de type Image.
- **direction** appartient à Up, Down, Left, Right

Lorsque le client fait un accès au serveur, les trois situations suivantes peuvent se produire :

- **Situation 1 :** le serveur a déjà décodé et mis en cache le bloc demandé.
- **Situation 2 :** le serveur n'a pas encore décodé le bloc demandé.
- **Situation 3 :** le serveur est en train de décoder le bloc demandé.

Les situations 1 et 2 sont représentées par des diagramme d'interaction dans la figure 7.5. Chaque situation, se termine cependant de la même manière par une suppression en mémoire du bloc concerné et une mise à jour de la position et des dimensions de l'**image centrale**.

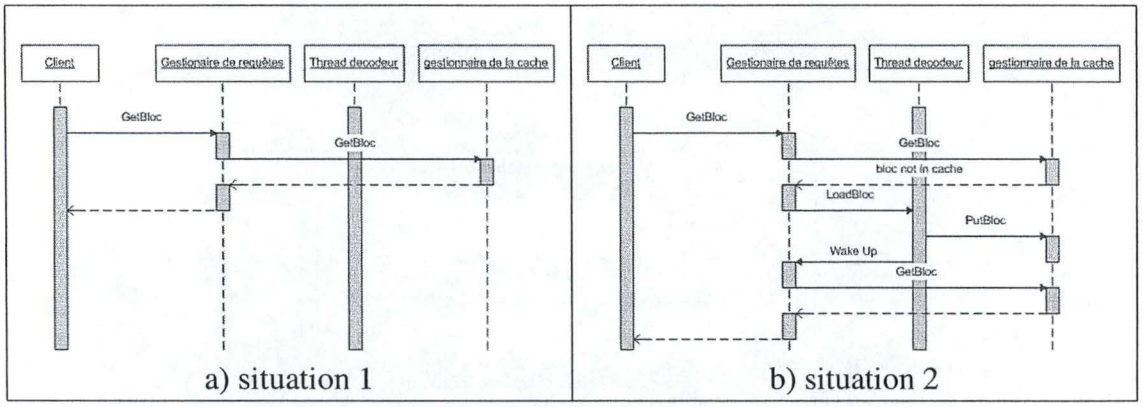


FIG. 7.5 – Diagrammes d'interaction représentant un accès ***getBloc()***.

7.3.4 Le système de cache

Afin de permettre au client de récupérer efficacement les blocs adjacents à l'**image centrale** déjà décodés par le serveur, ceux-ci sont organisés dans une structure de données relativement complexe. Rappelons que seuls sont décodés les blocs du niveau de résolution courant. Cette structure est organisée en quatre listes correspondant à une sous structure de données pour les blocs décodés dans chacune des directions (Up, Down, Left et Right) par rapport à l'**image centrale**. Notons que le serveur ne sachant pas à l'avance la direction correspondant au prochain ***getBloc()***, les blocs sont décodés en spirale autour de l'**image centrale** et stockés progressivement dans chacune de ces listes.
Chaque élément d'une liste est composé des trois sous éléments suivants :

- Un premier bloc (**imgCenter**) identifié par les mêmes données largeur et coordonnée abscisse (resp. longueur et coordonnée ordonnée) que celles de l'**image centrale** si la liste est Up ou Down (resp. Left ou Right).
- Deux sous listes (**ListImg1** et **ListImg1**) de blocs adjacents à **imgCenter**.

La figure 7.6 présente cette structure, le schéma a) représente une image JPEG2000 découpée en tiles et y représente les éléments de la liste Down tandis que le schéma b) représente la sous structure d'un élément d'une liste.

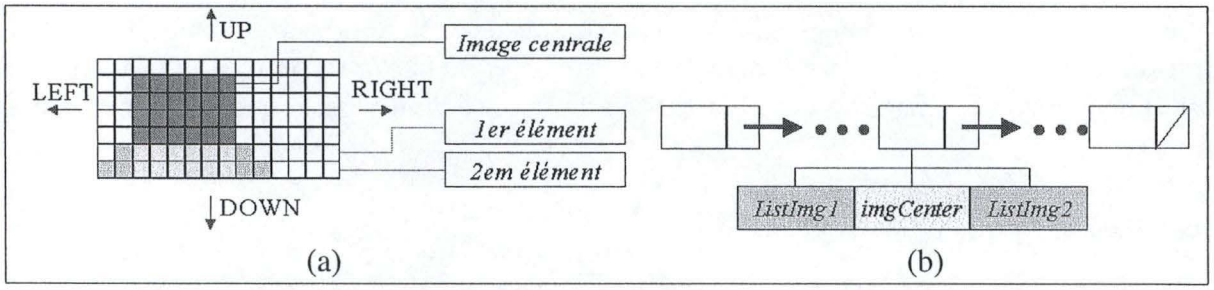


FIG. 7.6 – La structure de stockage des blocs du serveur.

Lorsqu'un client fait un accès ***getBloc(direction)***, le bloc qui lui est renvoyé correspond au premier sous élément **imgCenter** de la liste correspondant à son accès. De cette manière, on est assuré d'avoir toujours un bloc rectangulaire pour l'**image centrale**. L'utilité des listes de blocs **Listimg1** et **Listimg2** vient du fait que si un accès ***getBloc(direction)*** nécessite une mise à jour des données relatives à l'**image centrale**, alors des mises à jour doivent aussi être effectuées sur tous les **imgCenter**

de la cache. Ces blocs supplémentaires permettent d'effectuer cette mise à jour sans devoir charger forcément de nouveaux blocs, il suffit en effet de les fusionner avec les **imgCenter** comme dans la figure 7.7.

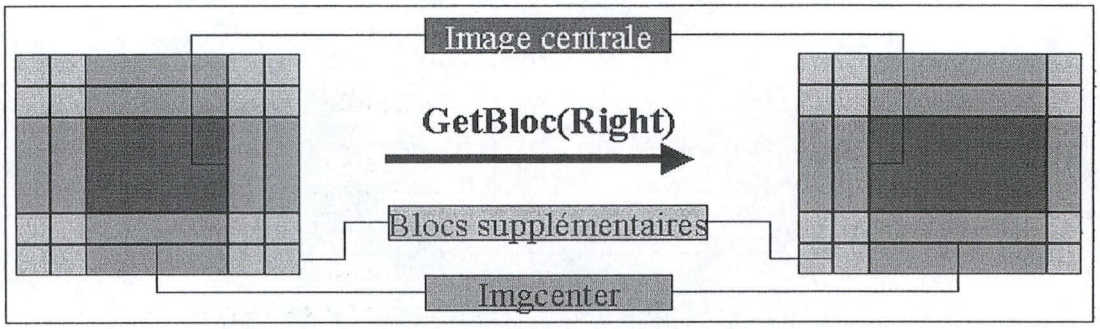


FIG. 7.7 – L'effet d'un getBloc() sur la structure de stockage.

7.4 L'architecture du serveur

Le schéma de l'architecture logique est présenté dans la figure 7.8. Comme on peut le constater sur le schéma, le serveur est composé de deux composants principaux : le "Gestionnaire d'Images" et le "Gestionnaire de clients". De plus le serveur a un accès direct à un "espace de stockage" qui contient les méga images JPEG2000 de frottis numériques.

L'espace de stockage JPEG2000

Une image JPEG2000 est stockée dans un répertoire portant le nom de l'utilisateur par lequel elle a été construite. Tous ces répertoires sont rangés dans un répertoire (*"users"*) accessible par les deux serveurs : le serveur de composition de frottis numériques et le serveur d'image.

Le composant *Gestionnaire d'images*

Ce composant est directement lié à l'espace de stockage. Il comprend le codec JPEG2000 qui permet de décoder les images par niveaux de résolution et/ou par *tiles*. De plus, il offre au client la possibilité de lister les images d'un utilisateur ou d'effacer définitivement une image de l'espace de stockage.

Le composant *Gestionnaire de clients*

Ce composant s'occupe de servir les accès spécifiques liés à l'ouverture d'une image par un client. Il s'agit donc principalement des accès ***zoom()*** et ***getbloc()*** décris ci-dessus. Comme le serveur permet de gérer plusieurs clients, à un moment donné, il y a autant d'instances de ce composant qu'il y a de clients qui ont ouvert une image sans la fermer.

Notons encore qu'avec un même client connecté, un utilisateur ne peut ouvrir qu'une seule image à la fois. Cependant chaque application cliente possède un identifiant *"clientID"* qui permet à la fois à un utilisateur de lancer plusieurs clients sur la même machine et à plusieurs utilisateurs de lancer plusieurs clients sur des machines différentes sans problème de concurrence. Cet identifiant est défini par

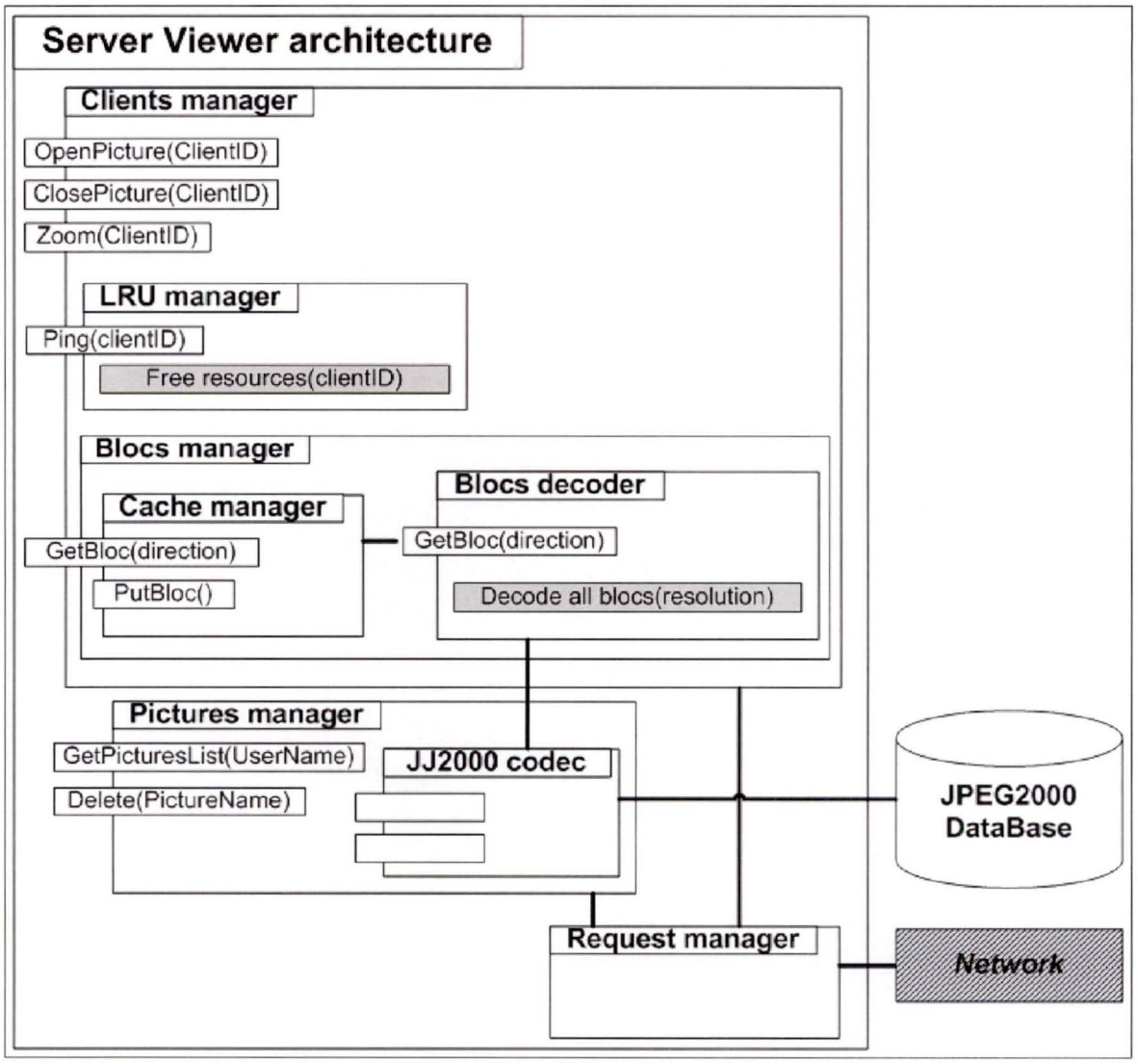


FIG. 7.8 – L'architecture logique du serveur d'image.

l'adresse IP de la machine du client + l'instant (en "ms") où le programme client a été lancé.

Le composant *Gestionnaire LRU*

Le serveur a une durée de vie théorique illimitée or il alloue des ressources en mémoire centrale pour chaque instance du composant **Gestionnaire de clients**, c'est pourquoi un système de *"ping"* est présent afin de permettre au serveur de détecter les problèmes éventuels liés à la connexion de chaque client. Lorsqu'un problème est détecté pour un client donné, le serveur ferme automatiquement l'image que ce dernier avait ouverte et libère les ressources mémoires [4].

[4]Le système ping fonctionne évidement aussi en sens inverse : si le serveur ne répond pas à la requête *ping()* le client sait qu'il y a un problème de connexion.

7.5 L'IHM du client

Cette section termine ce chapitre en présentant l'interface homme-machine de l'application cliente et en y décrivant les différentes fonctionnalités modélisées qui justifient son appellation de microscope virtuel.

La figure 7.9 représente la fenêtre principale du client lorsqu'un utilisateur démarre le programme. On peut remarquer que, mis à part les onglets, toutes les fonctionnalités sont grisées. Ces fonctionnalités ne seront en effet disponibles que lorsque l'utilisateur aura choisi et ouvert une image.
La figure 7.10 représente l'état de la fenêtre principale de l'application lorsque que l'utilisateur a déjà ouvert une image.

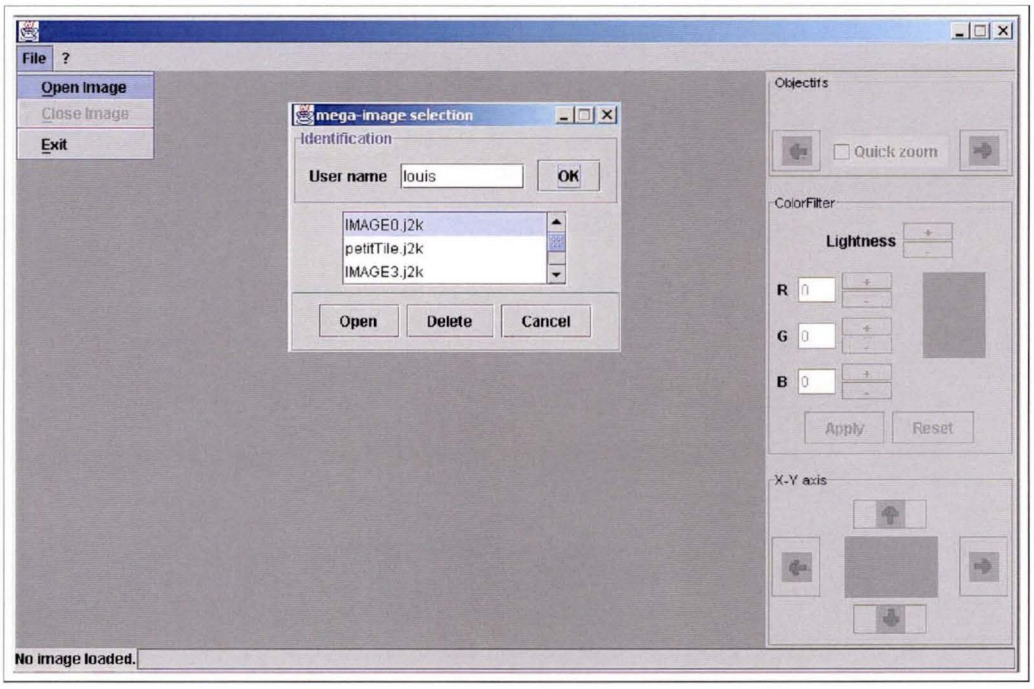


FIG. 7.9 – La fenêtre de démarrage et d'identification du microscope virtuel.

7.5.1 La description des fonctionnalités

Les fonctionnalités disponibles dans cette application sont séparées en trois parties (voir figure 7.10) toutes correspondant à des fonctions que l'on peut retrouver dans le microscope de la station d'acquisition :

- La modélisation des objectifs : *"Objectifs"*
- La modélisation des filtres de couleur : *"Color filter"*
- La modélisation des mouvements de la platine du microscope : *"X-Y axis"*

La modélisation des objectifs

Elle offre l'opportunité de zoomer et de dézoomer dans l'image selon différents niveaux de résolution(de 1 à 5). Le niveau de résolution maximale (5) correspond au grossissement 100x sur le

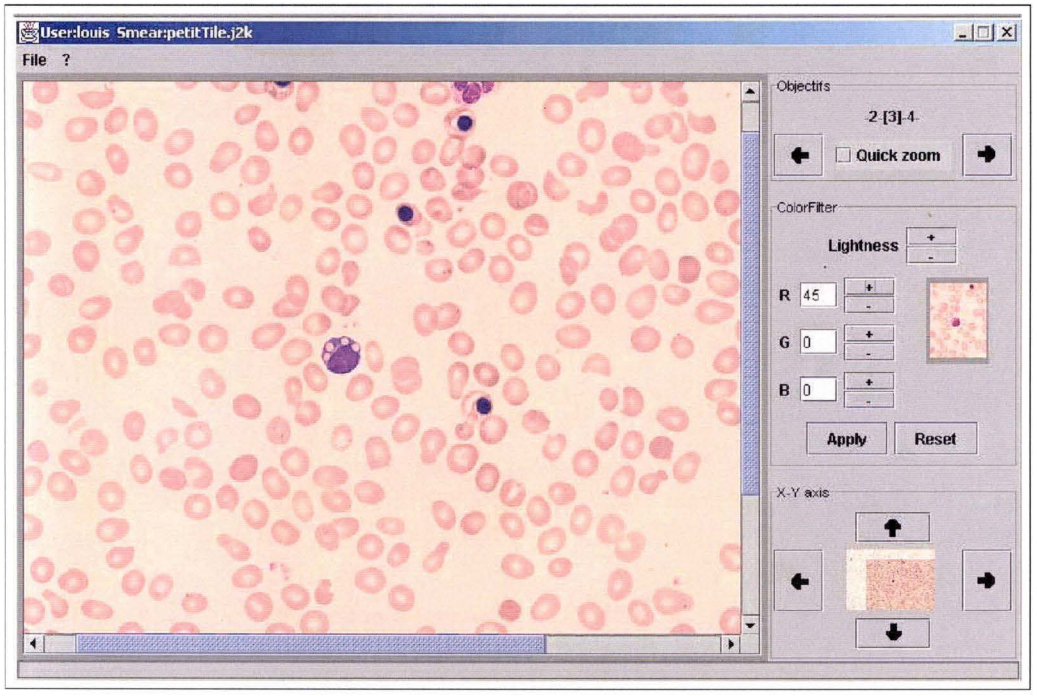


FIG. 7.10 – La fenêtre générale du microscope virtuel.

microscope électronique, les grossissements des niveaux précédents sont définis à partir de ce dernier par un facteur $\frac{1}{2}$. C'est en fait au grossissement 100x que les images sources qui composent la méga image sont capturées.

Une option *"Quickzoom"* permet aussi d'effectuer un zoom rapide sans accéder au serveur[5]. On ne décode donc plus ici le niveau de résolution suivant mais on extrapole les valeurs des pixels pour ce nouveau grossissement. Evidemment, en extrapolant les valeurs des pixels, on obtient un effet de pixellisation. Cette technique n'a donc pas la prétention de remplacer les accès au serveur par résolution qui fournissent des images nettes, mais doit être vue comme "un plus" pour passer d'un grossissement à un autre plus rapidement. Son utilité est d'ailleurs beaucoup plus marquée pour le passage entre les deux derniers grossissements pour lequel un accès au serveur entraîne un délai important alors que le *"Quickzoom"* produit un effet de pixellisation pratiquement invisible à l'oeil nu.

Notons aussi que redimensionner une image ne produit un effet de pixellisation que lorsque celle-ci est agrandie. C'est pourquoi tous les "dézoom", qui correspondent aux passages à des objectifs moins puissants, sont effectués de la même manière qu'un *"Quickzoom"* sans accès au serveur.

La modélisation des filtres de couleur

Tout comme dans la station d'acquisition, l'interface du client permet à l'utilisateur de pouvoir modifier les couleurs de l'image qu'il voit. Les modifications de couleurs sont définies sur base du modèle RGB, c'est à dire que l'utilisateur peut modifier les teintes rouge, verte et bleue de l'image.

> Par exemple, un coefficient égale à "45" pour la composante *R* signifie que la composante rouge de chaque pixel de l'image est égale à la valeur initiale de la composante + 45

[5]La méthode Java utilisée pour redimensionner une image est présentée en annexe

modulo(256).

De plus tout comme on peut jouer sur l'obturateur de la lampe du microscope afin de rendre le frottis plus clair ou plus sombre, l'option *"lightness"* permet de jouer sur les trois composantes de couleur à la fois et donc de produire un effet analogue.

Remarquons tout de même que ces options peuvent paraître inutiles si l'utilisateur professionnel du microscope a fait les bons réglages avant l'acquisition et la construction des méga images. Cela dit, un facteur non négligeable lorsque l'on fait de la télémicroscopie est la qualité et le réglage des écrans. Dans cette optique, l'utilisateur a à sa disposition des outils qui d'une part lui sont familiers et d'autre part lui évite de devoir modifier les paramètres de son écran à chaque fois qu'il utilise (ou qu'il arrête d'utiliser) l'application.

La modélisation des mouvements de la platine du microscope

Cette partie a pour but de modéliser le boiter de commandes de la station d'acquisition qui permet de faire déplacer la platine du microscope. Cette partie a été la plus difficile à réaliser non pas tant dans la modélisation du boîtier (en effet 4 flèches directionnelles suffisent) mais plutôt dans la fluidité des déplacements dans l'image. En effet, comme expliqué plus haut, l'utilisateur ne voit pas directement toute l'image lorsque sa résolution dépasse une certaine taille. Il peut cependant par la suite charger les parties manquantes. Cette notion n'apparaît évidement pas dans l'utilisation traditionnelle d'un microscope, il a donc fallu la rendre la plus transparente possible.

Il n'y a pas de bouton supplémentaire *"charger une partie adjacente"*, les flèches directionnelles permettant dans un premier temps de se déplacer dans l'image visible[6] et ensuite lorsque qu'un bord est atteint dans une direction, charge automatiquement la partie contiguë à l'image visible dans la bonne direction.

De plus, une image de prévisualisation a été rajoutée pour permettre à l'utilisateur de voir la quantité d'image visible (cfr la partie foncée de la petite image en bas à droite dans la figure 7.10) et la quantité d'image qu'il reste à charger (cfr la partie claire de la petite image en bas à droite dans la figure 7.10).

[6] des *Scrollbars* ont été rajoutés pour permettre la même action.

Quatrième partie

Discussion

Chapitre 8

Les limites du système et leurs solutions

Dans les chapitres précédents, nous avons présenté le développement et les résultats de notre solution. Nous avons présenté une application permettant la composition de méga images de frottis numériques et une application permettant la visualisation de ces méga images. Ces applications ne sont évidemment pas parfaites et l'on peut constater plusieurs limitations. Ce chapitre a pour objectif d'identifier les limites principales de ces deux applications et de proposer des éléments de solution.

8.1 Les limites de la composition des frottis

8.1.1 La taille des frottis numériques

L'étape de construction de frottis est une étape relativement longue c'est pourquoi durant les phases de test, les plus gros frottis ont été réalisés sur base d'une matrice d'au maximum 10x10 images sources. Bien que ces images ne soient pas représentatives de la taille maximale d'un frottis numérique, une limitation peut cependant être décelée au niveau de la mémoire centrale du serveur. En effet, dans la solution proposée, les images sources s'accumulent dans les buffers de celle-ci avant d'être assemblées en une méga image qui sera entièrement compressée et stockée ensuite dans l'espace de stockage. Si l'on se rappelle les résultats exposés dans le chapitre 1 sur la place mémoire nécessaire à la digitalisation d'un frottis[1], on comprend aisément que la solution n'est pas encore adaptée à la digitalisation entière d'une lame. Selon Monsieur Yvan Cornet, spécialiste en hématologie à la clinique de Mont-Godinne, la taille de la zone à numériser sur une lame pour qu'un frottis numérique soit caractéristique est de environ 1 cm^2, ce qui donne encore un volume de 44 GB pour le frottis numérique si les images sont capturées au grossissement 100x.

Deux solutions visant à contourner la limitation de la mémoire centrale du serveur peuvent cependant être apportées : l'une basée sur le codage par ROI (region of interest) et l'autre basée sur la constitution d'une base de données d'images plutôt que sur la constitution d'une seule méga image.

La première solution se base sur le fait que puisqu'on peut accéder à une image JPEG2000 par régions en lecture[2], il est aussi possible d'y accéder par régions en écriture. Il devrait donc être possible d'insérer progressivement les images arrivant au serveur dans une méga image JPEG2000 sur le disque dur. La faisabilité d'une telle solution est cependant liée à la possibilité de créer une méga image JPEG2000 vide aux dimensions gigantesques sur le disque dur dans laquelle viendraient s'insérer les différentes images.

[1] A un grossissement 100X, le volume est estimé à 218 GB pour une zone de 20*25 mm sur une lame.

[2] ce qui est fait dans le microscope virtuel

Un autre problème lié à cette solution est que si les images sont insérées et fixées progressivement (ou par petits groupes) dans la méga image vide, on doit alors connaître leurs positions à chaque instant. Or la solution qui a été présentée dans le chapitre 6 se base sur une procédure d'optimisation globale qui ne détermine la position de chaque image qu'une fois effectuées toutes les détections entre chaque couple d'images contiguës.

La deuxième solution consisterait à ne pas assembler les images sources, à les compresser et à les stocker séparément dans une structure de données organisée. Le nom de chaque image pourrait par exemple être suffixé par sa position dans le frottis. Encore une fois les images vont pouvoir être sauvées progressivement sur le disque dur, mais dans cette solution, comme les images sont stockées indépendamment, leurs positions peuvent n'être déterminées qu'une fois toutes les images capturées et donc la procédure de coregistration globale est toujours applicable.
Cette solution perd cependant tous les avantages liés à JPEG2000 relatifs à la manipulation d'images volumineuses.[3] D'autres artefacts doivent alors être pensés afin de réduire les délais de visualisation. Une telle solution est présentée dans [Leo01].

8.1.2 La netteté des images

A cause de l'observation du plantage de l'autofocus en présence de zones peu contrastées[4], la solution proposée n'utilise pas l'autofocus. Elle laisse le soin à l'utilisateur de régler le focus une fois pour toutes avant la capture des images. En ce qui concerne la netteté des images, bien que les résultats observés lors des phases de test pour des frottis numériques de tailles raisonnables semblent corrects, on est en droit de se demander si, lors de la digitalisation de plus gros frottis, il en sera toujours de même pour l'entièreté des images capturées.

A dire vrai, des observations pratiques semblent même apporter une réponse négative à cette question. En effet, une légère inclinaison de l'axe "z" des objectifs par rapport au plan "x-y" de la platine motorisée semble indiquer un dérèglement linéaire du plan focal[5] suivant cette même inclinaison. Cela dit, si ce dérèglement est réellement linaire sur tout le plan "x-y", une mesure du plan focal en trois positions du plan x-y devrait suffire à déterminer le plan focal de chaque autre position du plan par extrapolation linéaire. Cette situation est présentée avec une platine motorisé n'ayant qu'une seule dimension "x" dans la figure 8.1.

Cela dit, l'autofocus n'est pas fiable dans n'importe quelle situation : une condition nécessaire pour que l'autofocus fonctionne est "la présence un élément contrasté dans sa zone de focus"[6]. Donc, si on a besoin de trois points de repère sur la lame, il faut que ces trois points de repère correspondent à des éléments contrastés. Afin de garantir cela, on pourrait par exemple effectuer en trois points une recherche de la cellule la plus proche basée sur la détection de couleurs comme dans [Geo02].

8.1.3 L'interface du client

L'interface du client consiste en une fenêtre intégrée au logiciel Analysis. Elle comporte cependant plusieurs lacunes.

Tout d'abord, elle ne donne pas de limitation à l'utilisateur quant à la taille maximale des frottis réalisables. Ensuite, cette taille est exprimée en nombre de "champs à capturer"[7], c'est-à-dire le

[3] Cfr le chapitre 4.
[4] Voir le chapitre 3
[5] La position de l'objectif dans l'axe "z" pour laquelle une image est nette.
[6] La zone dans le plan "x-y" utilisée par l'autofocus pour déterminer le plan focal. Cfr le chapitre 3
[7] nombre de champs en abscisse et nombre de champ en ordonnée.

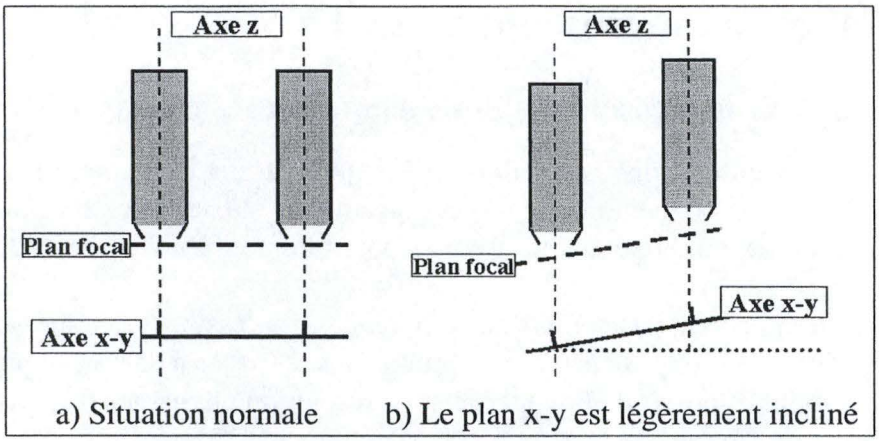


FIG. 8.1 – Si le plan x-y a une inclinaison d'un angle α, le plan focal a une même incinaison.

nombre d'images à capturer. Une mesure en mm des dimensions sur la lame serait sans doute plus représentative, il conviendrait alors aussi de donner une estimation du volume de l'image compressée et une estimation du temps nécessaire à la réalisation du frottis. Notons aussi que dans le cas de frottis sanguin, la taille caractéristique de la zone à numériser est une variable qui peut être estimée par une mesure de la leucocytose du prélèvement de sang.

La **leucocytose** est ***une mesure du nombre de globules blancs par*** mm^3**.**

Avant d'effectuer une analyse au microscope, la leucocytose est toujours calculée. Autrement dit, il doit être possible de déterminer une fonction qui prend en paramètre la leucocytose et qui renvoie la taille à numériser.

Enfin, on peut encore regretter l'impossibilité de stopper proprement le programme une fois qu'il est lancé. En fait, Analysis fonctionnant en monothread, il est impossible d'y développer une application qui à la fois exécute une fonctionnalité déclenchée à partir d'une interface et qui continue d'écouter les actions de l'utilisateur sur cette interface. La seule solution pour arrêter le programme est de tuer le processus lié à Analysis, ce qui n'est évidemment pas une solution élégante et pratique.

8.1.4 La diversification des frottis

Bien que la solution ai été développée dans le but de réaliser des frottis numériques sur base de prélèvements sanguins, on peut imaginer qu'elle puisse aussi être utilisée avec d'autres types de supports. On pourra citer les *étalements de moelle* ou les *empreintes ganglionnaires* qui font aussi l'objet d'analyse en microscopie.[8] L'avantage de la solution proposée est que la technique de détection est basée sur une analyse de pixels qui ne dépend pas des caractéristiques intrinsèques de l'objet observé.

[8]On estime la zone à numériser pour un étalement de moelle entre 2x2 mm et 10x10 mm et pour une empreinte ganglionaire à 5x5 mm à 10x10 mm.

8.2 Les limites du microscope virtuel

8.2.1 La limitation des ressources dont disposent les clients

Le microscope virtuel a été conçu avec l'objectif de permettre à n'importe quel utilisateur[9] possédant un ordinateur et une connexion, de pouvoir installer le client et de se connecter au serveur. Bien que seules des connexions en réseau local soient possibles pour l'instant, cet objectif semble être atteint.

Cela dit, La solution ne tient pas compte des différences des ressources en mémoire dont dispose chaque client. En effet, elle n'impose pas de restriction dans les accès à une méga image, l'utilisateur peut par exemple : zoomer jusqu'au dernier niveau de résolution et charger (progressivement) toute l'image de résolution maximale. De plus, le serveur non plus n'est pas à l'abri d'un dépassement de capacité puisque suite à un accès ***zoom()*** d'un client, il va décoder petit à petit tous les blocs du niveau de résolution demandé.

Il est évident que si on tient compte des remarques précédentes sur le volume que peut atteindre une méga image de frottis numérique, l'application est prédestinée à des problèmes récurrents de dépassement de la capacité mémoire.

Une amélioration de l'application qui tiendrait compte des limitations en mémoire de chaque client et du serveur semble donc indispensable. Cette amélioration pourrait consister à déterminer un seuil maximal pour le volume des blocs que peut décoder le serveur et des seuils maximaux pour les blocs qui peuvent être gardés en mémoire par les clients. Cela dit, cette amélioration n'est pas directement applicable à la solution proposée dans le chapitre 7, elle serait en effet incompatible avec le fait que le serveur ne décode qu'une et une seule fois chaque bloc d'une méga image lors de sa visualisation par un client. En effet, si on émet des limitations sur les blocs du client tout en permettant toujours à l'utilisateur de se déplacer où il le désire dans un frottis numérique, il arrivera très fréquemment que le client demande plusieurs fois les mêmes blocs au serveur.

Autrement dit, en plus des deux seuils maximaux, cette amélioration devrait comprendre un moyen de permettre au serveur de connaître à tout moment les blocs détenus par le client. Ce moyen pourrait par exemple être un nouveau message que le client enverra au serveur à chaque fois qu'il aura libéré de la mémoire en supprimant un ou plusieurs blocs.

8.2.2 L'interface du microscope

Les fonctionnalités du microscope virtuel correspondent aux fonctionnalités de base d'un microscope électronique : changer d'objectifs, déplacer la platine motorisée et changer les filtres de couleur. Cela dit, les possibilités liées au traitement d'images numériques dépassent largement ces quelques fonctionnalités. On pourrait par exemple imaginer un outil permettant la sélection d'images dans le but de créer une galerie d'images ou encore un ensemble d'outils de dessin permettant d'annoter le frottis à différents endroits.

[9] Il faut penser ici à l'expert distant qui doit faire une analyse sans se déplacer.

8.3 Les limites du système globales

8.3.1 L'intégration de données cliniques

Dans un souci d'améliorer le système par rapport à l'un des objectifs principaux de la télémicroscopie et plus précisément du télédiagnostic, à savoir *l'amélioration du taux de diagnostic correct à distance*, les deux questions suivantes pourrait encore être approfondies :

- Quelles données cliniques supplémentaires et nécessaires à l'établissement d'un diagnostic pourraient être intégrées au frottis numérique et comment les représenter de manière à faciliter le travail de l'analyste ?
- Quel rôle doit jouer le frottis numérique dans le dossier médical d'un patient suite à l'établissement d'un diagnostic ?

Autrement dit, dépassant l'objectif premier de simulation du microscope, il serait intéressant de poursuivre en identifiant la place d'un frottis numérique dans le processus d'élaboration d'un diagnostic afin d'en améliorer son utilisation.

Conclusion

Pour conclure, nous allons faire le point sur la composition automatique de frottis numériques, sur les avantages et les enjeux de cette approche de la télémicroscopie et nous allons résumer et critiquer les résultats s'y référant qui ont été apportés dans ce mémoire.

La "composition automatique de frottis numériques" et le "microscope virtuel" constituent les points clés d'une approche particulièrement intéressante de la télémicroscopie. Nous avons montré que cette nouvelle approche était motivée par la volonté de concilier les avantages des approches précédentes de cette discipline : fournir un moyen d'effectuer des analyses par des experts éloignés qui, d'une part est asynchrone et qui, d'autre part lui laisse un haut degré de liberté sur l'objet analysé.

Nous avons souligné trois problèmes qui rendent cette approche particulièrement ambitieuse. Un frottis à digitaliser nécessite la capture d'un ensemble d'images pour lesquelles des erreurs d'alignement doivent être corrigées. De plus, une fois numérisé, le frottis constitue un volume de données considérable. Enfin une fois stocké, le frottis forme une image gigantesque appelée méga image qu'il est difficile de manipuler.

Tout développement de télémicroscopie qui poursuit cette approche doit donc faire face à ces trois problèmes en y apportant des solutions spécifiques. La solution développée que nous avons présentée est constituée de deux applications. La première est relative à la composition des méga images de frottis numériques et la deuxième à la visualisation de celles-ci.

Afin de répondre au premier problème des erreurs d'alignement entre les images, nous avons présenté diverses techniques de coregistration permettant de réaligner deux images contiguës et nous en avons choisi une pour notre solution. Nous avons ensuite montré que la correction des erreurs en utilisant seulement une technique de coregistration pour chaque couple d'images n'était pas suffisante dans le cas d'un ensemble d'images car une petite erreur non corrigée entre deux images pouvait avoir des répercussions sur plusieurs autres. Nous avons alors présenté un algorithme de post-optimisation globale permettant de répondre efficacement à ce problème.

Parallèlement, nous avons présenté JPEG2000, un nouveau standard pour le codage d'images. Nous avons montré comment les spécificités de ce standard pouvaient répondre au troisième problème de la manipulation des méga images. JPEG2000 permet, en effet, d'accéder à l'image de manière progressive par précision, par résolution ou par région. Nous avons alors présenté une application distribuée, qualifiée de microscope virtuel, qui d'une part, permet un accès à distance aux méga images de frottis numériques et qui d'autre part, utilise les types d'accès progressifs de JPEG2000 ainsi qu'une interface graphique adaptée afin de reproduire les fonctionnalités d'un microscope.

Le deuxième problème, c'est-à-dire le volume considérable de données que constitue un frottis numérique, a somme toute été le plus difficile à attaquer. Le volume des méga images requiert non seulement un espace disque suffisant pour leur stockage mais aussi des ressources importantes en mémoire centrale pour leur composition. Pour ce qui est du stockage des méga images, JPEG2000 s'est encore une fois révélé fort utile en fournissant des taux de compression très avantageux. Par contre la solution développée ne permet pas encore de construire un frottis numérique dont le volume

excède la capacité de la mémoire centrale de la machine utilisée pour assembler et compresser les images qui le constituent.

En guise de perspectives, nous avons présenté deux directions possibles pour contourner la limitation de cette ressource. La première se base sur des propriétés de l'encodeur JPEG2000 et la deuxième sur la composition d'une base de données d'images. Le choix de l'une ou l'autre de ces directions nécessitant de repenser certains éléments de la solution présentée dans ce mémoire pourrait faire l'objet d'une étude approfondie.

D'autre part, nous avons aussi souligné le fait que l'application microscope virtuel telle qu'elle a été développée ne tient pas compte des limitations variables en mémoire des différentes machines. Si une solution au problème de la taille des frottis réalisables est apportée, une amélioration tenant compte des limitations en mémoire devra aussi être apportée pour le microscope virtuel.

Enfin, replaçant la composition de frottis numériques et le microscope virtuel dans leur contexte de la télémicroscopie, il pourrait être intéressant d'étudier de manière plus approfondie le rôle d'un frottis numérique dans le processus d'élaboration d'un diagnostic. Cette étude permettrait alors d'intégrer en une même solution divers éléments input et output de l'analyse au microscope en vue de faciliter ou de simplifier la tâche d'élaboration du diagnostic.

Bibliographie

Bibliographie

Ouvrages, articles et publications

[Bro92] L.G. Brown. "A Survey of Image Registration Techniques", *ACM Computing Surveys Vol.24*, No. 4, pp. 325-376, December 1992.

[Cap02] M. Capek, R. Wegenkittl, and P. Felkel. "A Fully Automatic Stitching of 2D Medical Datasets", *The 16th international EURASIP conference*, vol. 16, pp. 326-328, ISBN 80-214-2120-7, Brno, Czech Republic, June 26 - 28, 2002.

[Chr00] C. Christopoulos, A. Skodras and T. Ebrahimi. "The JPEG2000 still image coding system : An Overview", *IEEE Transactions on Consumer Electronics*, Vol. 46, No. 4, pp. 1103-1127, November 2000.

[Deb03] J. Debord. "La transformée de Fourier rapide", http://www.unilim.fr/pages_perso/jean.debord/math/fourier/fft.htm, (last revised May 01, 2003) (Date of access 16/07/03).

[Del99] V. Della Mea. "Store-and-Forward telepathology", *european telemedicine 1998-1999*, pp. 74-76, 1999.

[Montb] F. de Montbel & al, "Télémédecine et Imagerie Médicale", http://www.ese-metz.fr/metz/eleves/themes/imagerie/tele_part1.html. (Date of acces 15/07/03)

[Des01] S. Deshpande W. Zeng. "Scalable streaming of JPEG2000 images using hypertext transfer protocol", *ACM Multimedia*, pp. 372-281, October 2001.

[Duvau] C. Duvaux. "La télémédecine état des lieux en France en 1996", http://ordmed.org/c3.html, (Date of access 15/07/03).

[Eus02] R. Eustice, O. Pizarro, H. Singh, and J. Howland. "Underwater Image Toolbox for optical image processing and mosaicking in MATLAB", *IEEE Int. Sympos. on Underwater Technology*, pp. 141-145, 2002.

[Fru95] J. Fruitet. "Outils et méthodes pour le traitement des images par ordinateur", *Image numérique dans l'enseignements des sciences*, june 1995.

[Hal97] B. Halliday, A. Bhattacharyya, A. Graham, J. Davis, S. Leavitt, R. Nagle, W. McLaughlin, R. Rivas, R. Martinez, E. Krupinski, R. Weinstein. "Diagnostic accuracy of an international static-imaging telepathology consultation service", *Human Pathology*, vol. 28, pp. 17-21, 1997.

[Kow87] P. Kowaliski. "Photographie", *Encyclopaedia Universalis*, corp. 14, pp. 538, november 1987.

[Lam00] K. Lam. "Telepathology : Making Diagnoses is Overcoming Confinements in Place and Time", *South West Cancer News*, Issue 4, 2000.

[Leo01] F. Leong, J. McGee. "Automated complete slide digitization : a medium for simultaneous viewing by multiple pathologists", *journal of pathology* Vol. 195, pp. 508-514, 2001.

[Mar98] B. Marchese. "Image Fusion : A marriage of convenience", *Eye on technology*, pp. 122-128, november 1998

[Mee03] J. Meessen, T. Suenaga, M. Guerrero, C. De Vleeschouwer and B. Macq. "Layered architecture for navigation in JPEG2000 Mega-Images", *4th European Workshop in Image Analysis for Multimedia Interactive Services*, pp. 92-95, april 2003.

[Nor99] I. Nordrum. "History and present status of real-time telepathology", *european telemedicine*, pp. 72-73, 1999.

[Sav01] G. Savaton, E. Casseau, E. Martin, C. Lambert-Nebout. "Composants Virtuels Comportementaux pour Applications de Compression d'Images", http://lester.univ-ubs.fr:8080/publications/IP/gretsi2001_paper.pdf, september 2001.

[Wei96] R. Weinstein. "Static Image Telepathology in Perspective", *Human Pathology* vol. 27, pages 99-101, february 1996.

Mémoires

[Geo02] B. Georges. "La télémicroscopie en cytologie hématologique", *Mémoire en informatique*, FUNDP Namur, 2002.

[Kad99] C. Kaddour, "Compression des images fixes par fractales basée sur la triangulation de Delannay et la quantification vectorielle", *Mémoire en informatique*, université des sciences et de la technologie Houari Boumediene, Algérie, 1999.

[Mer00] B. Mercier, "Mise au point d'une station de télémédecine pour le laboratoire d'hématologie", Mémoire en biologie médicale, Institut Paul Lambin, UCL, 1999-2000.

[Sey02] S. Seynave, "Coregistration d'image médicales : contexte, modèlisation du problème et sa traduction en une première conception Orientée Objets", *Mémoire en informatique*, FUNDP Namur, 2002.

Sites Internet

[WebAX70] "Research System Microscope AX70" :
http://www.olympus.co.jp/LineUp/Microscope/ax70E.html

[WebIPT] "Image Processing Toolbox" :
http://www.mathworks.com/access/helpdesk/help/toolbox/images/images.shtml

[WebJava1] "The API Java proposed by Sun" :
http://java.sun.com/j2se/1.3/docs/api/

[WebJava2] "Providing a scalable Image Icon in Java" :
http://developer.java.sun.com/developer/JDCTechTips/2003/tt0220.html#2

[WebJasper] "The JasPer Project Home Page" :
http://www.ece.uvic.ca/~mdadams/jasper/

[WebJJ2000] "The jj2000 Project home page" :
http://jj2000.epfl.ch/

[WebJpeg] "The official Jpeg homepage" :
http://www.jpeg.org/

Annexes

La procédure de réajustement

Cette annexe est un complément à la discussion sur la procédure d'optimisation globale du chapitre 6. Elle présente par un exemple la "procédure de réajustement" introduite dans ce chapitre.

L'idée de la procédure est de choisir $HPOS$ comme candidat initial, (de ce fait DH est initialement nulle) et d'appliquer de manière itérative les relations définies dans le chapitre 6 afin de réduire le poids de DV sans augmenter le poids de DH dans l'équation $\min Poids_c = |DH| + |DV|$.

Afin de ne pas augmenter le temps de calcul, le choix des opérations à appliquer à chaque itération est basé sur l'observation que si $CPOS = HPOS$, alors DV prédit principalement des erreurs propagées en ligne et que généralement, ces erreurs sont propagées jusqu'au bout de la matrice. Comme c'est le cas pour la colonne DV de l'exemple du tableau 1. Afin de modifier une sous ligne de la matrice DV, on dispose de l'opération *PLg(a,i,k,nbr)*. Cette opération à l'effet de reporter le décalage d'une sous ligne i à une ligne $i+1$ (comme le montre la deuxième colonne du tableau 1). On peut alors définir l'opération *shiftLn(a,i,j)* comme la composition d'une suite d'opération *PLg(a,i,k,nbr)* visant à reporter le décalage "en dehors de la matrice" :

```
for (k = j ; k <= nbrLigne ; k++)
begin
   PLg(a,i,k,nbr) //avec nbr égal nbrcolonne - i
end
```

TAB. 1 – L'impact de l'opération *shiftLn()* sur la matrice DV.

DV	après un *PLg(2,2,3,1)*	après un *shiftLn(2,2,3)*
0 0 0 0 0	0 0 0 0 0	0 0 0 0 0
0 0 -1 -1 -1	0 0 0 0 0	0 0 0 0 0
0 0 0 0 0	0 0 -1 -1 -1	0 0 0 0 0
0 0 0 0 0	0 0 0 0 0	0 0 0 0 0
0 0 0 0 0	0 0 0 0 0	0 0 0 0 0

La procédure de réajustement consiste à appliquer des opérations *shiftLn()* en partant des lignes du bas vers le haut. Elle est illustrée par un exemple dans le tableau 2. Notons que dans cette exemple, tous les chiffres sont réels sauf ceux de la matrice $CPOS$ initiale qui ont été arrondis afin de mettre en évidence les transformations effectuées par la procédure. Notons que, comme le calcule du poids n'est déterminer qu'à partir de DH et DV, cette modification n'a pas d'incidence sur la procédure.

TAB. 2 – Exemple de la "procédure de réajustement".

DV	*DH*	*CPOS*	*Détail*
$\begin{vmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & -2 & -2 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & -2 & -1 & -1 \\ 0 & 2 & 2 & 2 & 2 \end{vmatrix}$	$\begin{vmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{vmatrix}$	$\begin{vmatrix} 0 & 10 & 20 & 30 & 40 \\ 0 & 10 & 20 & 30 & 40 \\ 0 & 10 & 20 & 30 & 40 \\ 0 & 10 & 20 & 30 & 40 \\ 0 & 10 & 20 & 30 & 40 \end{vmatrix}$	Itération : 0 $poids = 18$
$\begin{vmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & -2 & -2 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & -2 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 \end{vmatrix}$	$\begin{vmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & -2 & 0 & 0 & 0 \end{vmatrix}$	$\begin{vmatrix} 0 & 10 & 20 & 30 & 40 \\ 0 & 10 & 20 & 30 & 40 \\ 0 & 10 & 20 & 30 & 40 \\ 0 & 10 & 20 & 30 & 40 \\ 0 & 8 & 18 & 28 & 38 \end{vmatrix}$	Itération : 1 $poids = 12$ *shiftLn(-2,5,2)*
$\begin{vmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & -2 & -2 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{vmatrix}$	$\begin{vmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \end{vmatrix}$	$\begin{vmatrix} 0 & 10 & 20 & 30 & 40 \\ 0 & 10 & 20 & 30 & 40 \\ 0 & 10 & 20 & 30 & 40 \\ 0 & 11 & 21 & 31 & 41 \\ 0 & 9 & 19 & 29 & 39 \end{vmatrix}$	Itération : 2 $poids = 8$ *shiftLn(1,4,2)*
$\begin{vmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{vmatrix}$	$\begin{vmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \end{vmatrix}$	$\begin{vmatrix} 0 & 10 & 20 & 30 & 40 \\ 0 & 10 & 21 & 31 & 41 \\ 0 & 10 & 21 & 31 & 41 \\ 0 & 11 & 22 & 32 & 42 \\ 0 & 9 & 20 & 30 & 40 \end{vmatrix}$	Itération : 3 $poids = 9$ *shiftLn(1,2,4)*
$\begin{vmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{vmatrix}$	$\begin{vmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & -1 & 1 & 1 & 0 \end{vmatrix}$	$\begin{vmatrix} 0 & 10 & 20 & 30 & 40 \\ 0 & 10 & 21 & 32 & 42 \\ 0 & 10 & 21 & 32 & 42 \\ 0 & 11 & 22 & 33 & 43 \\ 0 & 9 & 20 & 31 & 41 \end{vmatrix}$	Itération : 4 $poids = 11$ *shiftLn(1,2,5)*

Dans cette exemple, on remarque que le poids de la solution finale n'est pas à son minimum, ce qui s'explique par le fait que l'on a simplement travaillé en optimisant la matrice DV en négligeant l'impact des opérations *shift()* sur la matrice DH. On pourrait recommencer la même procédure à l'envers, c'est à dire en optimisant la deuxième matrice mais on risquerait alors de revenir à une situation précédente et boucler indéfiniment. Il y a cependant un moyen de limiter les dégats : utiliser les opérations qui n'ont d'incidence que sur DH, c'est-à-dire les opérations *PCo(α,j)*[10]. Tout comme pour l'opération *shiftLn()*, on peut définir l'opération *totshiftCo(a,j))* par :

```
for (k = i ; k <= nbrColonne ; k++)
begin
   PCo(a,k)
end
```

Le tableau 3 montre le résultat après application d'opérations "totshiftCo()".

TAB. 3 – Optimisation du résultat par application des opérations PCo(α,j).

$\begin{vmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{vmatrix}$	$\begin{vmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & -1 & 1 & 1 & 0 \end{vmatrix}$	$\begin{vmatrix} 0 & 10 & 20 & 30 & 40 \\ 0 & 10 & 21 & 32 & 42 \\ 0 & 10 & 21 & 32 & 42 \\ 0 & 11 & 22 & 33 & 43 \\ 0 & 9 & 20 & 31 & 41 \end{vmatrix}$	Itération : 4 $poids = 11$
$\begin{vmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{vmatrix}$	$\begin{vmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 & 0 \end{vmatrix}$	$\begin{vmatrix} 0 & 10 & 19 & 29 & 39 \\ 0 & 10 & 20 & 31 & 41 \\ 0 & 10 & 20 & 31 & 41 \\ 0 & 11 & 21 & 32 & 42 \\ 0 & 9 & 19 & 30 & 40 \end{vmatrix}$	Itération : 5 $poids = 7$ totshiftCo(-1,3)
$\begin{vmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{vmatrix}$	$\begin{vmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \end{vmatrix}$	$\begin{vmatrix} 0 & 10 & 20 & 28 & 38 \\ 0 & 10 & 21 & 30 & 40 \\ 0 & 10 & 21 & 30 & 40 \\ 0 & 11 & 22 & 31 & 41 \\ 0 & 9 & 20 & 29 & 39 \end{vmatrix}$	Itération : 6 $poids = 3$ totshiftCo(-1,4)

[10]Cfr le chapitre 6

Voici le pseudocode de la procédure de réajustement :

```
Reajustement(int[][] HABS, int[][] VABS, int[][] HDEC, int[][] VDEC)
begin
   int[][] PC := HABS                  /**La matrice candidat pour la solution**/
   int[][] DH := Hdec(PC)-HDEC         /**La différence entre les décalages hori.
                                          de PC et de HABS**/
   int[][] DV := Vdec(PC)-VDEC         /**La différence entre les décalages vert.
                                          de PC et de VABS**/

   for each row "i" in PC              /**from below to up**/
   begin
     while (shiftLn(n,j) exist in PC)
       begin
         apply shift(n,j) to (PC,DH,DV)
       end
   end
   for each column "j" in DH
   begin
     if (totalshiftCo() is better)  /**i.e donne un meilleur poids**/
       begin
         apply totalshift() to (PC,DH,DV)
       end
   end
end.
```

Les fichiers de configuration des deux applications client-serveur

Cette annexe décrit le contenu des fichiers de configuration des clients et serveurs des deux applications réalisées. Chaque fichier de configuration est un fichier texte qui respecte le formalisme que l'on retrouve dans les fichier d'initialisation ("**.ini**") :

```
[Section Name]
Entry Name=Entry description
```

De plus, dans chaque fichier, on peut retrouver une **Section** particulière intitulée "*comments*" qui reprend une définition textuelle de chaque **entrée** du fichier.

Les fichiers de l'application "composition de frottis numériques"

Le fichier du client

```
[objetDistant]
IP=10.0.0.3
port=3000

[comments]
IP=l'adresse IP de la machine du serveur distant.
port=le numero de port du serveur distant.
```

Le fichier du serveur

```
[parameter]
overlapAbcisse=100
overlapOrdonnee=100
jj2000EncoderOptions=-debug -tiles 600 400
debug=YES
userspath=../serveurViewer/users/
[reseau]
port=3000

[comments]
overlapAbcisse= la longueur en pixel de la zone de recouvrement en abcisse.
overlapOrdonnee= la longueur en pixel de la zone de recouvrement en ordonnée.
jj2000EncoderOptions= reprend les options d'encodage proposées par jj2000.
debug= si YES affiche des messages sur la progression du serveur.
port= le numero de port de l'application serveur.
userspath= le chemin du repertoire comprenant les sous-répertoires de méga images.
```

Les fichiers de l'application "microscope virtuel"

Le fichier du client

```
[objetDistant]
IP=10.0.0.3
port=3001

[comments]
IP=l'adresse IP de la machine du serveur distant.
port=le numero de port du serveur distant.
```

Le fichier du serveur

```
[general]
delay=2
[reseau]
port=3001

[comments]
delay=le délai (en min) avant que le serveur ne supprime les ressources
    allouees à un client qui ne répond pas.
port=le numéro de port du serveur.
```

Redimensionner une image en Java

La façon la plus simple d'afficher une image en Java est d'implémenter les 3 méthodes de l'interface ***Icon***, on obtient alors un objet qui peut être inséré dans un ***Panel*** via la méthode ***addComponent()***. Afin de redimensionner cette image à afficher, en Java on peut procéder de deux manières différentes :
La première est d'utiliser la méthode ***getScaledInstance(int width, int height, int hints)*** de la classe ***Image*** dans laquelle on spécifie les nouvelles dimensions de l'image ainsi que le type d'algorithme à appliquer. Cette méthode a cependant le gros désavantage de consommer beaucoup de mémoire puisque l'image est dupliquée lors de son redimensionnement.
La deuxième technique consiste à redimensionner l'image "à la volée" au moment de son affichage via la méthode ***drawImage(Image image, int x, int y, int width, int height, ImageObserver observer)***. Cette méthode est beaucoup plus efficace car elle ne nécessite pas une duplication de l'image.

La solution utilisée dans le client du microscope virtuel se base sur les conseils proposés par Sun dans [WebJava2], elle consiste étendre la classe ***ImageIcon*** en utilisant la méthode ***drawImage()*** :

```
import java.awt.*;
import javax.swing.*;
import java.net.*;

public class ImageIconScalable extends ImageIcon {
  int width = -1;
  int height = -1;

  public ImageIconScalable() {
    super();
  }

  public ImageIconScalable(byte imageData[]) {
    super(imageData);
  }

  public ImageIconScalable(byte imageData[], String description) {
    super(imageData, description);
  }

  public ImageIconScalable(Image image) {
    super(image);
  }

  public ImageIconScalable(Image image, String description) {
    super(image, description);
  }

  public ImageIconScalable(String filename) {
    super(filename);
  }
```

```
  public ImageIconScalable(String filename, String description) {
    super(filename, description);
  }

  public ImageIconScalable(URL location) {
    super(location);
  }

  public ImageIconScalable(URL location, String description) {
    super(location, description);
  }

  public int getIconHeight() {
    int returnValue;
    if (height == -1) { returnValue = super.getIconHeight(); }
    else { returnValue = height; }
    return returnValue;
  }

  public int getIconWidth() {
    int returnValue;
    if (width == -1) { returnValue = super.getIconWidth(); }
    else { returnValue = width; }
    return returnValue;
  }

  public void paintIcon(Component c, Graphics g, int x, int y) {
    if ((width == -1) && (height == -1)) { g.drawImage(getImage(), x, y, c); }
    else { g.drawImage(getImage(), x, y, width, height, c); }
  }

  public void setScaledSize(int width, int height) {
    this.width = width;
    this.height = height;
  }
}
```