Ministère de l'Education Nationale

UNIVERSITE MONTPELLIER II
- Faculté des Sciences -

Master Intégration de Compétences
SPECIALITE BIOINFORMATIQUE

RAPPORT DE STAGE

effectué à

L'Unité Mixte de Recherche « bot**A**nique et bioinfor**M**atique de l'**A**rchitecture des **P**lantes » **AMAP**.

du 15/03/2009 au 15/09/2009

par

Pol Kennel

Identification automatique des **fi**les **cel**lulair**e**s chez les **r**ésineux appliquée à l'Epicéa et au Pin Noir.

Projet **FICELER**

Directeur du stage de l'entreprise : Philippe Borianne et Gérard Subsol

Directeur du stage de l'Université : Sèverine Bérard

Remerciements

Je tiens tout d'abord à remercier mes tuteurs Philippe Borianne de l'AMAP et Gérard Subsol du LIRMM qui m'ont encadré tout au long de ce stage. Ils m'ont permis d'une part de découvrir la discipline du traitement d'image et d'autre part de me guider intuitivement et avec patience sur une bonne démarche de recherche.

Je voudrais aussi remercier Christine Heinz et Yves Caraglio pour m'avoir transmis les connaissances en architecture des plantes et en anatomie, nécessaires à la bonne compréhension du contexte biologique de ma mission.

Un grand merci aussi à Michaël Gueroult pour le temps passé sur l'acquisition des images nécessaires à mon travail.

Mes remerciements vont aussi à Sèverine Bérard, ma tutrice à l'université, et Laurent Bréhélin chercheur CNRS au LIRMM, pour leurs judicieux conseils lors de certains choix cruciaux à la bonne réussite de mon stage. Mais aussi à Olivier Taugourdeau en thèse à l'AMAP pour ses conseils avisés en statistique.

Je tiens enfin à remercier l'ensemble de l'UMR AMAP pour leur accueil dans la bonne humeur durant ces six mois.

Sommaire

I. Introduction

Ce rapport est consacré à la présentation du travail effectué dans le cadre de mon stage de fin d'étude en Master BioInformatique (Université Montpellier II) au sein de l'UMR bot**A**nique et bioinfor**M**atique de l'**A**rchitecture des **P**lantes (AMAP) du 15 mars au 15 septembre 2009.

Dans un premier temps on posera explicitement le contexte d'étude proposé, aussi bien à un niveau organisationnel que scientifique, nécessaire à la bonne compréhension du travail à réaliser.

Le sujet traite d'un travail de reconnaissance automatique de file cellulaire sur des images microscopiques de bois. Ce travail est à l'heure actuelle réalisé de façon manuelle par les botanistes et anatomistes, ce qui est aussi bien long et fastidieux, que limitant quant à la quantité de données récupérées.

L'automatisation et la standardisation de ce protocole seraient donc un gain important de temps mais aussi une perspective nouvelle quant à l'étude statistique de l'organisation des files cellulaires. Cette problématique a déjà fait l'étude de travaux de chercheurs sans obtenir de résultats réellement satisfaisants, ce qui démontre la réelle difficulté du problème. De plus d'autres laboratoires se trouvent intéressés par cette étude, tel le LERFOB (Laboratoire d'Etudes des Ressources FOrêt-Bois) de l'INRA de Nancy.

On peut diviser le sujet en plusieurs étapes successives : la segmentation de l'image, la caractérisation des régions extraites, et la reconnaissance des files.

On aura orienté volontairement, en accord avec les tuteurs encadrant le stage, une méthode de travail prospective. En effet le sujet étant large et très peu étudié dans la littérature, le critère de temps rentre considérablement en jeu. On préféra donc obtenir à l'issue du stage une solution non optimale mais couvrant tous les aspects du travail, plutôt qu'une solution partielle plus robuste.

Toutes les méthodes existantes à la résolution de chaque étape ne seront pas testées, mais plutôt discutées. La solution proposée pourra être reprise et sera donc largement améliorable, mais proposera d'ores et déjà une trame complète et opérationnelle.

La méthode proposée présente une façon originale de procéder et fait appel à des domaines très divers tels que le traitement d'images, la théorie des graphes, ou la classification supervisée utilisée en fouille de données.

I.1. Présentation de l'entreprise

Mon stage s'est déroulé au sein de l'UMR (Unité Mixte de Recherche) AMAP (bot**A**nique et bioinfor**M**atique de l'**A**rchitecture des **P**lantes), installée dans les locaux montpelliérains du CIRAD (Centre de Coopération Internationale de Recherche Agronomique pour le Développement).

Le CIRAD est l'institut français de recherche agronomique au service du développement des pays du Sud et de l'outre-mer français. Il privilégie la recherche en partenariat et a choisi le développement durable comme ligne de force de son action à travers le monde. Cette démarche prend en compte les conséquences écologiques, économiques et sociales, à long terme, des processus de transformation des sociétés et des territoires du Sud. Il intervient par des recherches et expérimentations, des actions de formation, d'information et d'innovation, et des expertises. Ses compétences relèvent des sciences du vivant, des sciences humaines et des sciences de l'ingénieur, appliquées à l'agriculture et l'alimentation, à la gestion des ressources naturelles et aux sociétés.

L'UMR AMAP est une unité multidisciplinaire : elle regroupe des disciplines aussi variées que la biologie, la botanique, l'écologie, les mathématiques appliquées et l'informatique, dont l'association constitue aujourd'hui des enjeux scientifiques et techniques majeurs pour l'agronomie et la foresterie modernes.

Elle est composée de 52 agents permanents (dont 40 chercheurs ou enseignants chercheurs et 2 chercheurs associés) dont 21 agents CIRAD répartis en Chine, en Guyane Française, au Laos, à l'INRIA de Rocquencourt et à Montpellier, 8 agents CNRS, 8 agents INRA, 10 agents IRD (dont 3 en Guyane française), 3 enseignants chercheurs de l'Université Montpellier II et 2 chercheurs détachés au MAE en Inde, plus une quinzaine de doctorants et une quarantaine de stagiaires par an.

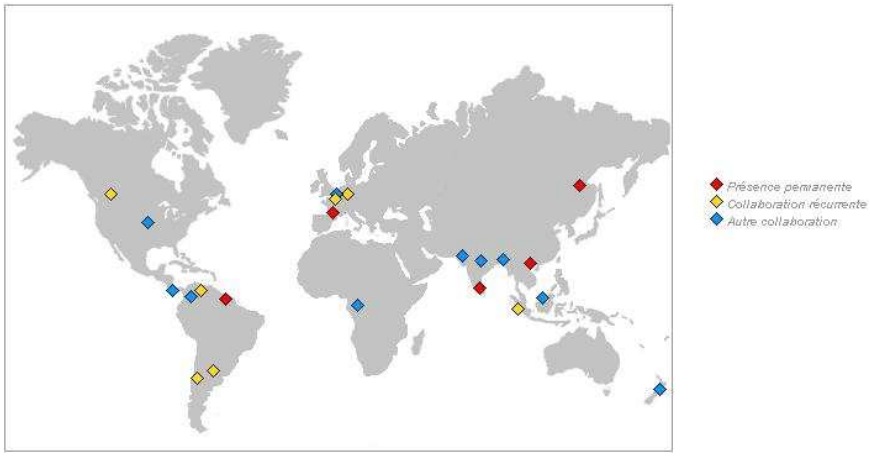


Figure 1 : *Localisation de l'AMAP dans le monde.*

L'UMR est constituée de trois équipes scientifiques :

- Equipe « évolution des formes végétales, fonctions, systématique, floristique » : Évolution morphologique ; biomécanique, évolution, écologie ; systématique, taxonomie, floristique.

- Equipe « architecture et développement des plantes » :
  diversité et plasticité de l'architecture des plantes ; analyse et modélisation intégrées de l'architecture et du fonctionnement des plantes ; imagerie volumique ; nous détaillerons ces domaines.
- Equipe « organisation et dynamique des peuplements et des paysages » :
  diversité et dynamique des peuplements forestiers hétérogènes ; structure et dynamique des peuplements ; dynamique spatiale des paysages et ingénierie écologique.

Les activités de l'UMR dans le domaine des mathématiques et de l'informatique appliquées s'articulent autour des trois axes :

- la modélisation et la simulation de la croissance et de l'architecture des plantes ;
- l'analyse et le traitement statistique des informations ;
- l'ingénierie logicielle.

Les recherches accordent une place centrale à la description, l'analyse et la modélisation de la structure, de la dynamique de développement et de la diversité des plantes et des peuplements végétaux. Elles privilégient les approches génériques, communes à l'ensemble des plantes, annuelles ou pérennes, actuelles ou fossiles, sauvages ou cultivées, tempérées, méditerranéennes ou tropicales. Elles concernent la mesure, la représentation, l'analyse et le traitement des données, la simulation informatique, l'organisation et la gestion des connaissances.

L'UMR contribue aussi :

- à la formation initiale (cours et encadrement dans les Universités et écoles d'ingénieur) et professionnelle ;
- à la diffusion des connaissances et recherches sous forme de logiciels le plus souvent libres mais parfois commercialisés ;
- à la coopération scientifique internationale, notamment avec des pays du sud.

L'UMR entretient des liens étroits avec des équipes de plusieurs organismes nationaux (notamment organismes de tutelle mais aussi CEMAGREF[1], ECP[2], ENGREF[3], INRIA[4],…) et développe un réseau diversifié de partenariats, académiques, scientifiques, technologiques et commerciaux, non seulement en Europe mais aussi en Amérique du Nord, en Argentine, au Chili, en Chine, en Inde, au Laos et en Nouvelle-Zélande.

Les problématiques de recherche abordées dans les différentes équipes proviennent de questions biologiques. Afin de répondre à ces questions, il est parfois nécessaire de créer de nouveaux outils mathématiques et/ou informatiques. Dans certains cas, ces développements sont eux-mêmes porteurs d'une problématique scientifique à part entière. Les activités dans le cadre du domaine MIA[5] concernent la formalisation mathématique, la modélisation ou la production logicielle. C'est dans ce même cadre que s'inscrit le projet Ficeler en réponse à une problématique concernant l'équipe d'architecture des plantes.

[1] CEMAGREF : L'institut de recherche finalisée de référence pour la gestion durable des eaux et des territoires.
[2] ECP : Ecole Centrale de Paris.
[3] ENGREF : Ecole nationale du génie rural, des eaux et des forêts.
[4] INRIA : Institut National de Recherche en Informatique et Automatique.
[5] MIA : Mathématiques Informatique Appliqué.

I.2. Contexte biologique

I.2.1. L'architecture et le développement des plantes

Le but de cette discipline est d'analyser, de comprendre et de prédire les modes d'édification et de développement architectural des plantes, leur plasticité, leur production et d'identifier le rôle respectif des facteurs génétiques, environnementaux et culturaux au cours de leur croissance et de leur ontogénie. A cette fin, différents axes d'études sont posés :

- Définir et mettre en ouvre, à partir de descripteurs essentiellement morphologiques et anatomiques, les concepts et les méthodes nécessaires à la description, à la mesure, à l'analyse et à la compréhension de l'architecture des plantes.
- Evaluer les composantes de l'expression phénotypique et génétique de l'architecture des plantes en analysant et en caractérisant leur diversité structurale et leur plasticité phénotypique.
- Analyser l'édification de l'architecture des plantes en fonction de processus physiologiques pour comprendre les mécanismes d'élaboration de leur production.
- Développer des outils de modélisation et de simulation intégrant des processus physiques, physiologiques et morphogénétiques pour prévoir le développement architectural des plantes dans des conditions agronomiques, écologiques et climatiques variées.

L'originalité de ces recherches résulte d'une approche spatio-temporelle (i.e. globale et dynamique), multi-échelles et multidisciplinaire de la structure végétale. Les méthodes développées et les résultats obtenus fournissent, aux autres domaines de la biologie végétale, des connaissances sur la plante entière, des protocoles d'observation et d'échantillonnage de leur structure et des modèles d'analyse et de représentation de l'édification de l'architecture des plantes.

I.2.2. Le bois

C'est dans ce cadre que s'inscrit l'étude du bois. Celui-ci ayant plusieurs rôles dans l'architecture des plantes :

- Assure le grossissement en diamètre des tiges
- Permet l'alimentation et le stockage dans la plante
- Permet la rigidité des tiges
- Son élaboration conditionne :
  - La production de biomasse
  - La production de matière première
  - Les caractéristiques biomécaniques
  - Les propriétés physiques du matériau (technologie du bois)

Plusieurs approches sont utilisées quant à son observation :

L'étude cambiale (du cambium) consiste à connaître son accroissement en périmètre (analyse circulaire), à connaître la nature de la régularité des types cellulaires, à identifier comment s'insère la nature des nouvelles cellules (initiales fusiformes : vaisseaux, fibres, parenchyme, initiales de rayon : parenchyme). Mais aussi de savoir comment son fonctionnement change au cours des saisons, de moments de l'année. Lors de la production d'une initiale fusiforme, on réalise une analyse radiale pour chercher comment se succèdent les différents types

cellulaires. On s'intéresse aussi à la proportion de tissu vasculaire (bois et liber) entre les fibres (soutien) et les rayons (réserves), mais aussi entre la taille des éléments vaisseaux (lumière) et fibres (parois). Etudier les cernes permet de caractériser la régularité d'accroissement en diamètre du tronc. On cherche aussi à caractériser le lien allongement/épaississement.

Plus généralement, l'étude de l'architecture bois cherche à modéliser le maillage cellulaire :

- Détecter les lignées cellulaires
- Détourer les tissus
- Caractériser (dimension)
- Reconstruction 3D

Dans le cadre de la réflexion sur le fonctionnement cambial, la production et la différenciation des cellules issues des initiales radiales et fusiformes sont le reflet des variations du contexte de croissance tant du point de vue interne (régulation des flux et échange liber/bois) qu'externe (modification de l'environnement lumineux et hydrique). Pouvoir détecter et dénombrer les cellules permet d'accéder à la mise en évidence de régularités dans la succession des types cellulaires (tant pour les initiales -agrandissement du cambium-, que pour les cellules produites -différenciation cellulaire-). Ce type de questionnement nécessite de pouvoir accéder à de l'échantillonnage suffisant pour manipuler des outils et modèles statistiques. Le côté fastidieux du comptage et typage cellulaires pourrait être allégé par une délimitation et identification (semi-)automatique des cellules.

Dans le cadre précédemment posé sur l'architecture du bois, les spécialistes en biologie végétale analysent la taille et la disposition des cellules du bois à partir d'images microscopiques de sections de troncs ou de branches. Les files cellulaires sont définies par les alignements « naturels » de cellules, facilement identifiables à l'œil nu sur l'image numérique. Il s'agit en fait de retrouver l'organisation spatiale des cellules, c'est-à-dire les motifs dominants qui se dessinent à partir des seules informations géométriques.

Pour bien interpréter le contenu des images sur lesquelles on travaillera ainsi que le rôle des files cellulaires, il est nécessaire de bien comprendre la formation et la structuration du bois ainsi que le protocole d'acquisition de ces images.

I.2.3. Formation du bois

Le développement de l'arbre résulte d'une élongation et d'une croissance en diamètre, correspondant chacune à une organisation différente des tissus. Les croissances apicale (en hauteur) et radiale (en diamètre) se traduisent par une accumulation géométrique des cernes annuels d'accroissement, plutôt analogue à un empilement de cônes. Chez les espèces ligneuses et pérennes (arbres, arbustes, buissons), le fonctionnement du cambium suit un cycle saisonnier. En région tempérée, le fonctionnement s'interrompt à l'automne et reprend au printemps. Chaque année, un nouveau cylindre de bois est formé à l'extérieur du précédent.

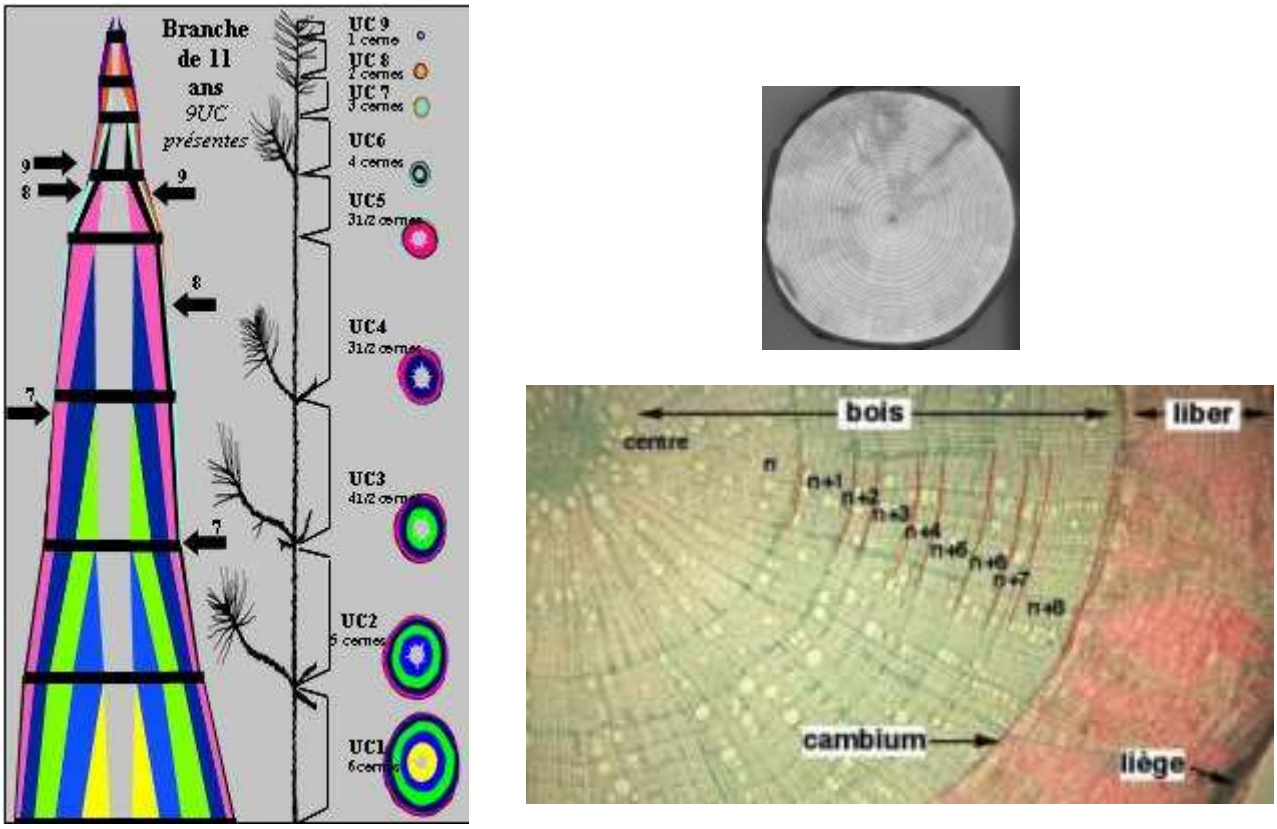


Figure 2 : *A gauche : Visualisation des unités de croissance, à droite : vue macro et micro d'une coupe transverse de tronc.*

Définitions :

- Cambium (jaune) :

Zone de cellules peu différenciées à divisions actives qui participent à l'accroissement en diamètre de la tige en se différenciant en xylème secondaire (bois) vers le centre (proportion 2x) ou en phloème secondaire, ou liber, vers l'extérieur (proportion 1x). Il est composé d'initiales fusiformes (qui se différencient en vaisseaux, fibres, parenchyme) et d'initiales de rayon (qui se différencient parenchyme).

- Liber ou phloème (rouge) :

Conduit la sève élaborée (eau et sucres formés dans les feuilles par la photosynthèse) vers toutes les régions de la plante.

- Bois ou xylème (vert) :

Conduit verticalement la sève brute (eaux et éléments minéraux en provenance des racines) du sol vers les feuilles.

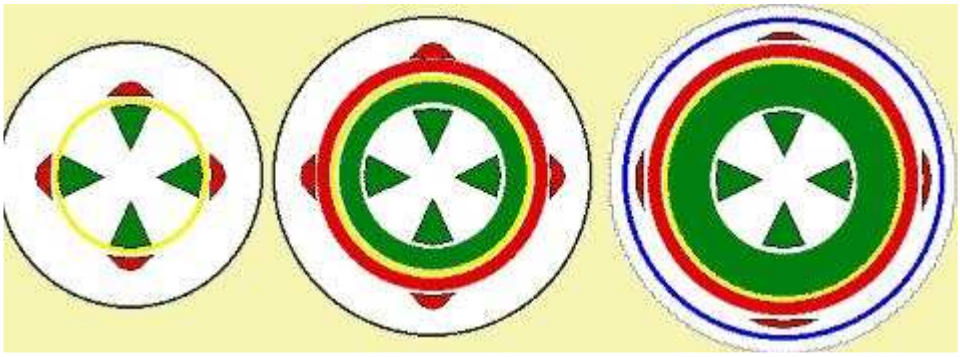


Figure 3 : *Schéma des éléments de croissance du bois.*

I.2.4. Structure du bois

Voici les éléments constitutifs du bois et leur agencement spatial :

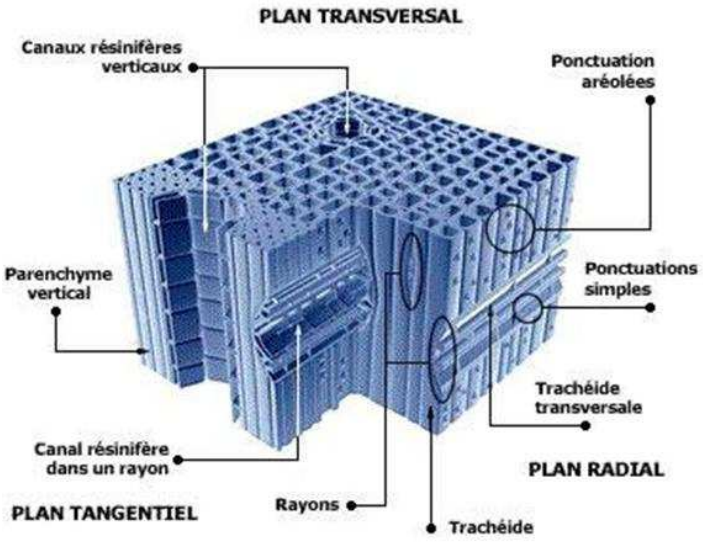


Figure 4 : *Eléments structurants chez le résineux.*

Définitions :

- Les fibres :

Les fibres simples ou cloisonnées jouent un rôle de soutien. On désigne sous le nom de fibres, des paquets des faisceaux de fibres. Les faisceaux sont déviés par les rayons médullaires si ces rayons sont épais. Il en résulte des madrures, des ondulations, particulièrement marquées dans les coupes. Les fibres ont des sections polygonales. Leurs parois sont plus ou moins épaissies suivant l'espèce et suivant l'âge de la partie considérée. Elles sont lignifiées et leur cavité peut être simple ou cloisonnée.

- Les vaisseaux :

Les parois des vaisseaux à quelques exceptions près (le frêne par exemple) sont plus minces que les parois des fibres. Elles sont encore plus faibles au niveau des ponctuations par lesquelles s'effectuent les passages osmotiques. Les vaisseaux influent par leurs dimensions et leurs distributions sur les propriétés du bois.

- Les parenchymes :

Court ou long le parenchyme est formé de cellules aux parois peu épaisses. Le parenchyme est formé de cellules polyédriques à peu près de même diamètre. Dans le bois jeune, il sert de magasin de réserve à l'amidon. Dans le bois âgé, ses cellules s'épaississent et, comme les fibres, il devient un tissu de soutien. On le trouve autour des vaisseaux. Ils constituent également avec une orientation radiale, les rayons médullaires.

- Les rayons médullaires :

Les rayons médullaires sont formés de cellules parenchymateuses, il existe :
- de grands rayons allant de la périphérie à la moelle existant dès la structure primaire (rayons complets).
- des rayons de second ordre (rayons incomplets) n'aboutissant pas jusqu'à la moelle, formés par le jeu du cambium.

Les rayons sont un peu des "chevilles" naturelles assurant une cohésion entre les cernes annuels. Ils se caractérisent par leurs 3 dimensions :
- Longueur; on trouve plus de rayons incomplets que de rayons complets.
- Épaisseur ; faible chez les résineux (une seule rangée de cellules, mais beaucoup plus importante chez les feuillus où elle varie de 0,02 à 2 mm.
- Hauteur: de 7 à 15 rangs de cellules chez les résineux elle atteint jusqu'à 5 cm dans certaines variétés de chêne.

- Les canaux sécréteurs :

Dans les bois à sécrétion, on remarque des canaux de fort diamètre bordés de cellules à parois minces. Les cellules parenchymateuses, très actives tout autour, mobilisent les déchets métaboliques et les déversent dans le canal résinifère. La transformation de ces produits de nutrition non utilisés par la plante, devenant par conséquent des matières d'excrétion, s'effectue dans des cellules ou des canaux, rares dans certains bois, abondants par contre chez les conifères.

- Les trachéides :

Les trachéides sont des cellules allongées du xylème des plantes vasculaires servant au transport de la sève brute (composée d'eau et de sels minéraux). La formation de trachéides varie en fonction de l'endroit où ils se produisent. Ce sont de longues cellules étroites, dont les extrémités sont en biseau. Elles possèdent une paroi primaire et secondaire lignifiées et sont mortes à maturité.
On distingue sur la figure 5 le bois d'été d'une année *n* à gauche (trachéides de faible diamètre à parois épaisses) et le bois de printemps de l'année *n+1* à droite (trachéides de fort diamètre et à parois plus fines). Les files de cellules qui traversent la coupe de gauche à droite sont les rayons formés de cellules non allongées dans le sens longitudinal.

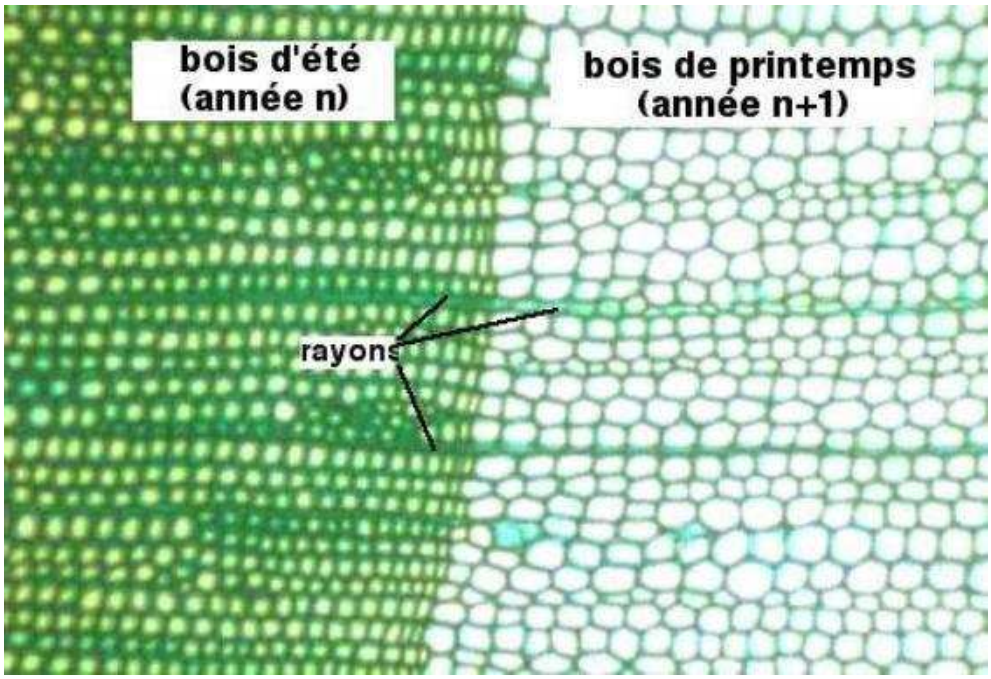


Figure 5 : *Coupe à la limite de deux cernes.*

I.2.5. Protocole d'acquisition

L'étude des lignées cellulaires s'effectue sur des images microscopiques tirées de troncs ou branches coupés sur le plan transverse, ainsi le suivi radial du cambium à la moelle est possible.

Il est important de savoir comment de telles images sont obtenues. En effet, les traitements qu'elles subiront seront directement affectés par la variabilité d'acquisition qu'il existe à ce niveau.

On peut découper en trois étapes successives l'obtention d'images :

- L'échantillonnage.
- La préparation
- L'acquisition

A chaque niveau des particularités apparaissent et dépendent de l'opérateur et de ses motivations, de ses moyens. En l'absence de protocoles très rigoureux (i.e. très automatisés), chaque étape doit être menée en connaissance de la finalité : on ne produit pas avec la même « application » des images destinées à un comptage manuel de cellules et des images utilisées pour une estimation automatique de parois cellulaires. Tout opérateur de la chaîne doit donc directement être impliqué dans le processus d'analyse et de quantification, c'est-à-dire savoir exactement comment seront exploitées les données qu'il produit ; dans le cas contraire, les données risquent de ne pouvoir être exploitées de manière optimale. A cette contrainte forte s'ajoute l'effet opérateur : la majorité des étapes du protocole d'acquisition sont laissées à l'appréciation « subjective » des opérateurs : qu'il s'agisse du prélèvement du « bon échantillon », de l'appréciation de la planéité d'une coupe après ponçage, de la mesure des temps d'immersion, de la pression exercée sur les lames ou le choix des grossissements ou des zones d'observation, tout est sujet à caution.

I.2.5.1. L'échantillonnage

Il s'agit de rondelles obtenues par section transversale de branches ou tronc. Avant d'en extraire une coupe à observer, l'angle de section est ajusté par ponçage perpendiculairement à l'axe vertical de la branche/tronc. Les rondelles sont ensuite échantillonnées de deux façons différentes : par découpe ou par ponçage et ne suivent pas le même conditionnement.

I.2.5.2. Préparation

- En ce qui concerne la découpe :

La rondelle trempe au préalable dans l'eau. S'il s'agit d'une rondelle de tronc, le préparateur sectionne des cubes au ciseau. S'il s'agit d'une tige (quelques centimètres au maximum) on garde la section entière. La découpe se fait grâce à un microtome[6] (model *slide 2003*) de calibrage 15-30µm. Les contenus cellulaires sont tués à l'eau de javel (rayons vidés). Le bois est déshydraté à l'alcool. Puis vient la coloration : plusieurs types de colorants sont appliqués selon ce que l'on souhaite observer (par exemple : safranine pour les parois, carmino-vert de Mirande pour, en vert la lignine et en rouge la cellulose ; il existe aussi des colorants pour identifier contenu cellulaire... ect). Une combinaison de ceux-ci est possible. Dans le cas de notre étude la safranine a été utilisée. Les échantillons obtenus sont fixés sur une lame de verre à la glycérine, placée sous microscope et nécessitant un éclairage par-dessous.

- En ce qui concerne le ponçage :

Une rondelle d'environ 5mm est prélevée, placée sur de la patte à modeler, puis poncée jusqu'à l'obtention d'une section de quelques µm. Directement placée sous microscope, cette préparation ne comporte pas de coloration et nécessite un éclairage du dessus.

I.2.5.3. Acquisition

Les échantillons sont finalement photographiés par l'intermédiaire d'une caméra numérique (Olympus DP71) montée sur microscope. Plusieurs grossissements sont utilisés, pour observer un cycle annuel de la croissance, ou plus en détail dans un cycle, ou un ensemble de cycles. L'image est capturée par un logiciel d'acquisition (*Olympus cell^A*) où l'opérateur peut y ajuster la couleur, le contraste, la balance couleur… ect, mais aussi choisir la résolution, le format d'export.

On se rend bien compte que selon les choix faits par le laboratoire au niveau du matériel, de l'opérateur selon ses affinités ou de sa minutie, les résultats obtenus auront sans surprise une hétérogénéité certaine comme nous allons le voir.

[6] Microtome : appareil utilisé pour produire des rubans de coupe de très faible épaisseur.

I.3. Présentation du sujet de stage

Le projet Ficeler (pour **Fi**les **cel**lulair**e**s dans les **r**ésineux), nom de mon sujet de stage, a donc pour but de travailler sur des images de coupes transverses de troncs ou branches chez les résineux uniquement (Epicéa et Pin noir), leur structure étant la plus simple parmi les arbres (en opposition au feuillus). L'objectif est d'identifier automatiquement les files de cellules (trachéides) en détournant au mieux tout autre type de tissus et éléments structurants (vaisseaux, rayons, parois…) puis d'en extraire des caractéristiques exploitables pour le biologiste (taille de la file, taille des cellules …).
Actuellement, la délimitation et l'identification des cellules et des files cellulaires s'effectuent manuellement. Or, pour bien comprendre le développement d'une espèce d'arbre, il faut recueillir des observations sur un très grand nombre d'échantillon. Il devient donc indispensable d'automatiser cette reconnaissance par des méthodes de traitement d'images.

Du fait de l'hétérogénéité multiple des images, la segmentation des cellules des fibres, des vaisseaux pose un réel problème. L'idéal serait d'avoir un processus de segmentation auto paramétrable, donnant un résultat satisfaisant sur l'intégralité des images.

Le sujet permettra :

- De recenser les différents problèmes et solutions potentiellement intéressants.
- De mettre en place une solution opérationnelle en Java, sous forme de plugin dans le logiciel libre *ImageJ*.

Exemple d'image à traiter sur les deux espèces citées provenant de deux laboratoires différents et n'ayant pas subi le même protocole :

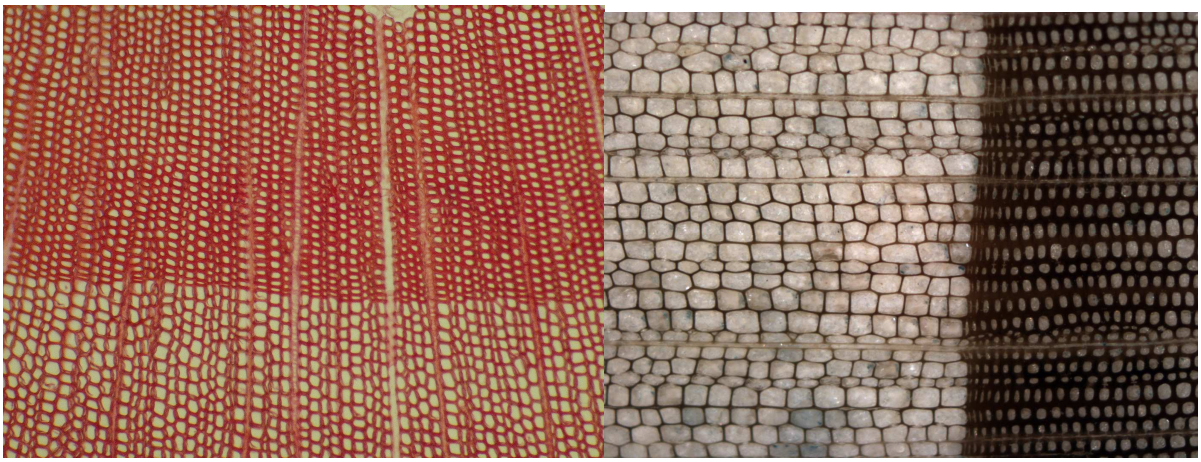


Figure 6 : *à gauche une image de Pin Noir coloré (Montpellier), à droite une image d'Epicéa poncé (Nancy).*

I.4. Planning prévisionnel

Durant le déroulement du stage, mes tuteurs et moi-même avons régulièrement mis en place des objectifs à atteindre durant une période de temps donnée. Le suivi de l'avancement du projet a pu être fait lors de réunions régulières hebdomadaires voire mensuelles avec les tuteurs pour discuter des avancés significatives du travail et des problèmes rencontrés, pour évaluer ensemble les options possibles, fixer les priorités ou acter des choix forts.

J'ai aussi eu l'occasion de faire une présentation orale de ma problématique en début de stage, et une pré-soutenance de fin de stage, au LIRMM devant l'équipe d'imagerie du laboratoire (équipe ICAR[7]),

Pour résumer les périodes de travail :

- Mois 1 : découverte du sujet et appréhension du contexte biologique, comprenant une étude bibliographique, des rencontres et des discussions avec les acteurs principalement concernés par le sujet, en particulier deux botanistes et un assistant de laboratoire d'Amap ; j'ai également consacré cette période à approfondir mes connaissances en traitement d'images, à maîtriser mon environnement de travail et à mener quelques tests,
- Mois 2 et 3 : exclusivement consacrés au prétraitement de l'image (et à son implémentation), à la recherche de méthodes de segmentation puis au choix d'une d'entre elle (et à son implémentation) et à la mise en place d'un graphe supportant les données extraites.
- Mois 4 : a porté sur l'étude des méthodes de classification pouvant déterminer la nature des régions extraites de la segmentation, au choix d'une d'entre elle, et son intégration dans le projet.
- Mois 5 : axé sur la reconnaissance des files en s'appuyant sur la structure de graphe et de la caractérisation des régions.
- Mois 6 : à partir du 15 Août, formalisation du rapport de stage en vue d'un dépôt début septembre ; les quinze derniers jours seront consacrés à la préparation de la présentation orale, la finalisation de certains détails algorithmiques, la préparation de l'après stage.

Il est à noter que le planning initialement prévu ne comprenait que quatre phases : la nécessité d'introduire une étape de caractérisation des tissus est apparue lors de l'étude réalisée sur le prétraitement de l'image.

[7] ICAR : descriptif de l'équipe et de leurs recherches, http://www.lirmm.fr/icar/

II. Matériel et Méthodes

II.1. Matériel

II.1.1. Environnement technique

Le cadre du stage impose le développement d'une solution de reconnaissance de files sous la forme de plugin à intégrer au logiciel libre de traitement d'image ImageJ (http://rsbweb.nih.gov/ij/) le tout devant être implémenté en java.

> Extrait de l'article consacré à ImageJ sur wikipedia :
>
> « *ImageJ est un logiciel de traitement et d'analyse d'images écrit en Java, ce qui en fait un logiciel utilisable sur différents systèmes d'exploitation. […] La plupart des opérations courantes de traitement d'images sont réalisables avec ImageJ. […] L'ajout personnalisé de fonctions est possible grâce aux plugins à écrire en java. […] ImageJ a été initialement développé pour des applications biomédicales. Il permet par exemple de faire des analyses de gels d'électrophorèse, ou de la détection et analyse de tumeurs. Son usage s'est depuis étendu à d'autres domaines, comme la science des matériaux (détermination de tailles de grains, traitement d'images obtenues par microtomographie X par exemple).* »

De plus, mes tuteurs ont souhaité utiliser un gestionnaire de projet collaboratif nommé Redmine[8] open source permettant à tous de partager documents et codes sources centralisés sur les serveurs de l'AMAP. Ainsi le projet *Ficeler* est accessible à l'adresse suivant : http://amap-dev.cirad.fr/projects/show/ficeler. On y trouve un wiki, un gestionnaire de document, un SVN supportant les sources du plugin régulièrement mis à jour… etc.

Le développement du plugin à proprement parler s'est réalisé sous l'environnement de développement Java Eclipse[9] et à l'aide de nombreuses librairies permettant le déploiement du plugin automatiquement (*Ant*[10]) ou de l'utilisation du SVN (*Subeclipse*[11])… etc.

II.1.2. Supports numériques

Le travail à réaliser a pour support un ensemble d'images numériques acquises comme expliqué en I.2.5**.** La plupart provient de mon laboratoire d'accueil, l'AMAP, et subit le protocole par découpe et coloration. D'autres ont été envoyées par le LERFOB de Nancy et obtenue après un protocole par ponçage.

---

[8] Redmine est un système Open Source de gestion complète de projet en mode web, développé en Ruby sur la base du framework Ruby on Rails : http://www.redmine.org/

[9] Eclipse IDE est un environnement de développement intégré libre extensible, universel et polyvalent, permettant de créer des projets de développement mettant en œuvre n'importe quel langage de programmation : http://www.eclipse.org/

[10] « Ant est un projet open source de la fondation Apache écrit en Java qui vise le développement d'un logiciel d'automatisation des opérations répétitives tout au long du cycle de développement logiciel, à l'instar des logiciels Make. » (Wikipédia) ; http://ant.apache.org/

[11] Système de gestion de versions : http://subversion.tigris.org/

Rappelons tout d'abord la notion d'image numérique. C'est une matrice rectangle de points, chaque point correspondant classiquement au centre géométrique d'un élément de surface appelé Pixel (Picture element) ; par abus de langage, on appelle Pixel indifféremment le point ou l'élément de surface qui lui est associé. L'image est aussi caractérisée par sa résolution.

La plupart des images ont une résolution entre 1024*768 et 2048*1536 pixels en RGB. On préféra un format non destructeur comme le TIF[12] pour le traitement de ces images, bien que cela ne soit pas toujours dans les habitudes d'acquisition. Le niveau de zoom lors de l'acquisition est lui aussi différent selon les images ; on aura admis qu'un zoom variant de 40x à 100x cadrait les besoins d'identification des files.

Toutes les images sont faites sur des résineux, mon jeu d'essai comporte neuf espèces :

- Epicéa
- Pin Noir
- Chamaecypavis obtusa
- Chamaecypavis thyoides
- Pinus Banksiana
- Pinus Khasya
- Pinus longifolia
- Pinus monticola
- Pinus radiata

II.2. Pré étude

II.2.1. Les a priori

Le sujet a déjà été abordé par des chercheurs mais n'a pu aboutir à cause de certains problèmes rencontrés. Voila quelques éléments qui en sont ressortis.

L'identification des cellules est rendue difficile par l'hétérogénéité de l'image :

- qualité localement variable (du fait de défaut de ponçage, de la coloration,…)
- contraste jour/tissu cellulaire localement non suffisant
- déchirement local, induit lors de la préparation de l'objet d'étude (ponçage, découpe, mise sous lame…)

L'identification des alignements cellulaires est rendue difficile du fait de différents artefacts biologiques : dédoublement local des files, cellules d'interférences provenant de plans différents.

Voici un essai de reconnaissance de files sur une image d'Acajou grâce à un partitionnement par un diagramme de Voronoï[13] réalisé sur une image segmentée et localement retouchée notamment aux abords du vaisseau. On y voit le parcours des files (en jaune) perturbé aux alentours du vaisseau central, celles-ci s'arrêtent ou prennent un mauvais chemin. De plus certaines cellules ne sont par reconnues.

[12] Le format **TIF** ou *TIFF* (*Tagged Image File Format*) est un format de fichier graphique bitmap (raster). Il a été mis au point en 1987 par la société Aldus (appartenant désormais à Adobe).

[13] Un diagramme de Voronoï représente une décomposition particulière d'un espace métrique déterminée par les distances à un ensemble discret d'objets de l'espace, en général un ensemble discret de points. Il doit son nom au mathématicien russe Georgi Fedoseevich Voronoï (1868 - 1908).

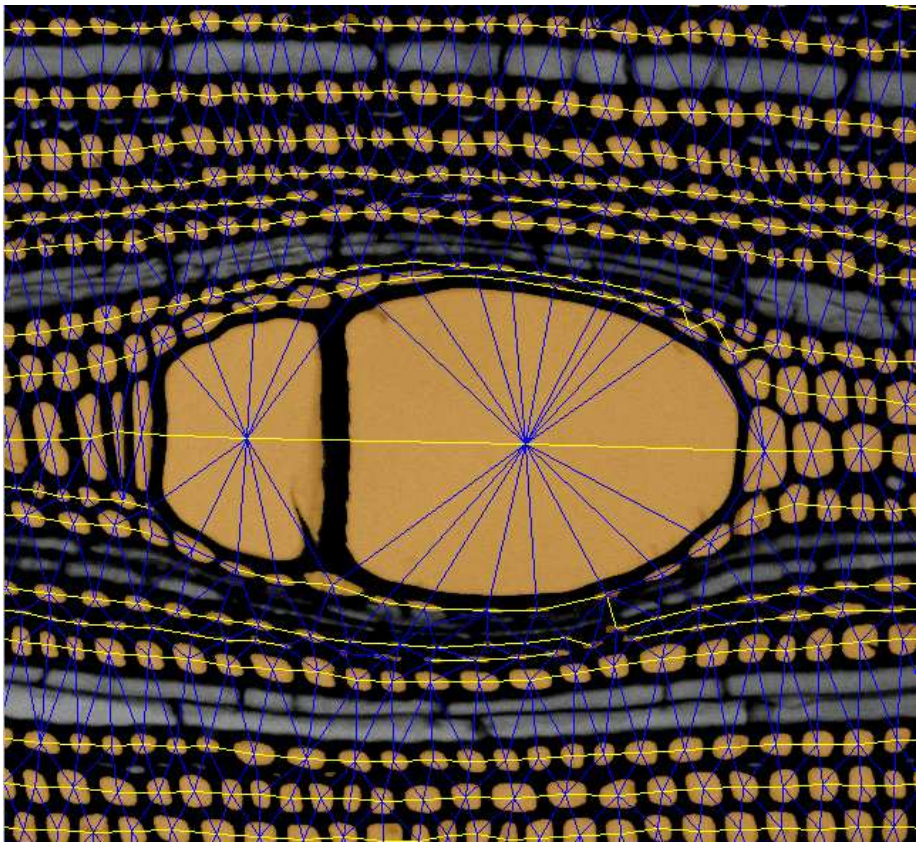
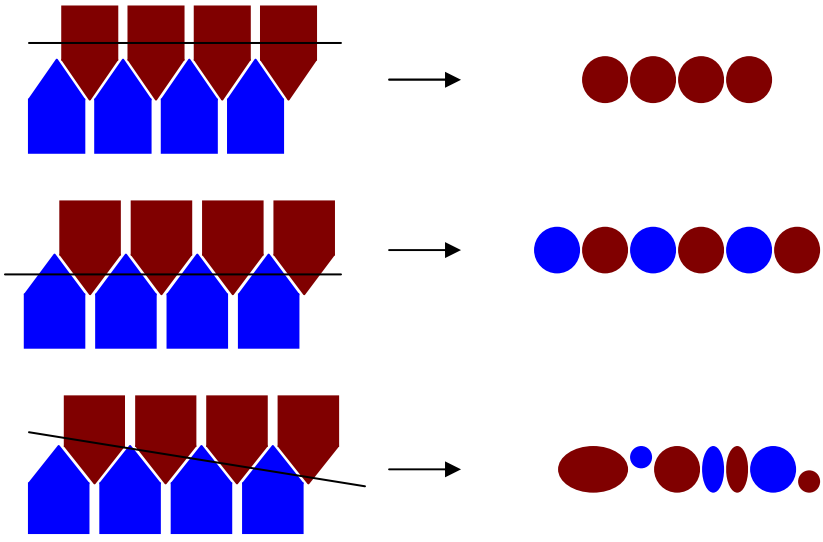


Figure 7 : *Reconnaissance de files sur l'Acajou.*

Un problème majeur expliquant l'intrusion de cellules n'appartenant pas au même plan est le plan de coupe comme l'illustre le schéma ci-contre. Les cellules ont une forme fusiforme ; elles sont généralement disposées en quinconce pour former des emboîtements globalement réguliers, présentant toutefois des perturbations locales : *à gauche, schématisation des empilements cellulaires ; à droite, représentation de la ligne cellulaire correspondant au trait de coupe noir.*



II.2.2 Nouvelle étude prospective sur la segmentation

Quant à ma pré-étude, elle a consisté :

- à tester plusieurs méthodes de segmentation automatique en parcourant la bibliographie tout en cherchant un prétraitement améliorant celle-ci à chaque fois.
- à d'identifier aussi pourquoi telle ou telle méthode n'était pas suffisante, et de cibler les zones à difficulté récurrentes.

II.2.2.1 Seuillage

A partir d'une image en niveau de gris, le seuillage d'image peut être utilisé pour créer une image binaire, généralement noir et blanc (monochrome).

Les méthodes les plus basiques déjà employées sur la segmentation de cellule (exemple en [1]) comme le seuillage simple, double (seuil haut et seuil bas), par entropie[14] n'on pas été suffisantes pour obtenir un résultat satisfaisant sur les différentes images fournies. La variabilité (variation de la dynamique du signal, et/ou variabilité dans la précision des structures anatomiques) au sein d'une image étant importante et le contraste pas toujours suffisant, le résultat de telle méthode entraîne une mauvaise distinction des cellules et des rayons. De plus d'une espèce à l'autre, le seuillage peut complètement faire disparaître une partie des cellules dans le bois d'automne quand le lumen est trop faible.

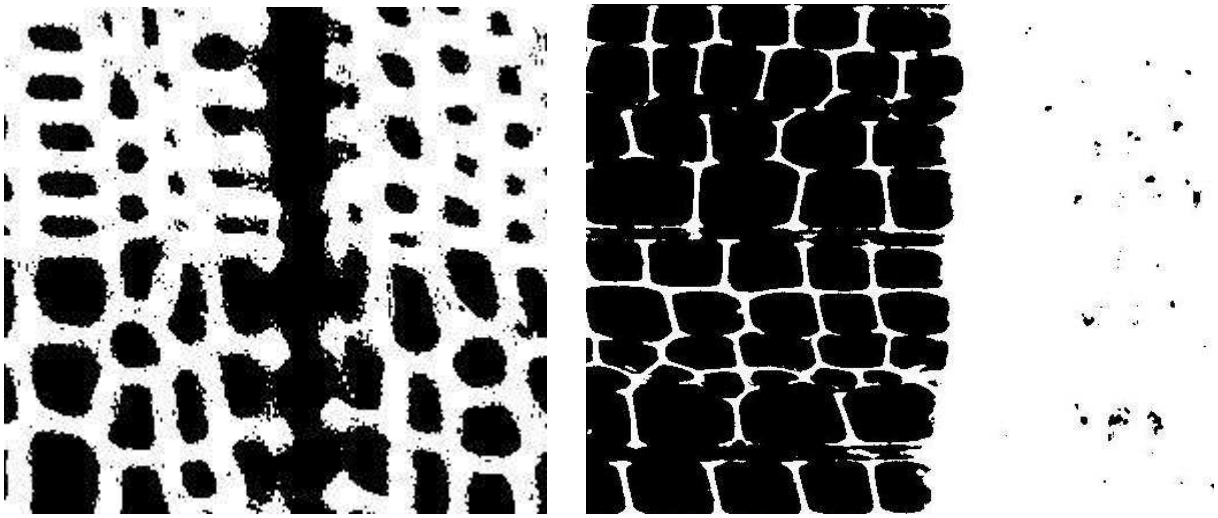


Figure 8 : *seuillage par entropie sur Pin Noir à gauche et l'Epicéa à droite.*

Nous avons pu obtenir une amélioration notable avec ces méthodes en réalisant une soustraction de fond, mais sans résoudre les problèmes cités précédemment.

Une approche plus séduisante, testée, a consisté à réaliser un seuillage multiple sur une image RGB, apportant ainsi plus d'informations qu'une image en niveaux de gris. Quelques résultats ont été satisfaisants en ajustant au mieux les seuils manuellement et en choisissant correctement l'espace des couleurs sur lesquels appliquer ces seuils. Malheureusement, cette méthode n'apporte pas de résultats automatisables simplement. Même après l'étude des différents histogrammes des espaces couleur essayés (RGB, HSV, LUT), aucune caractéristique mathématique n'est apparue clairement pour l'identification de ces seuils.

[14] Seuillage minimisant l'entropie des deux régions à identifier (l'entropie mesure le degré de désordre d'un système).

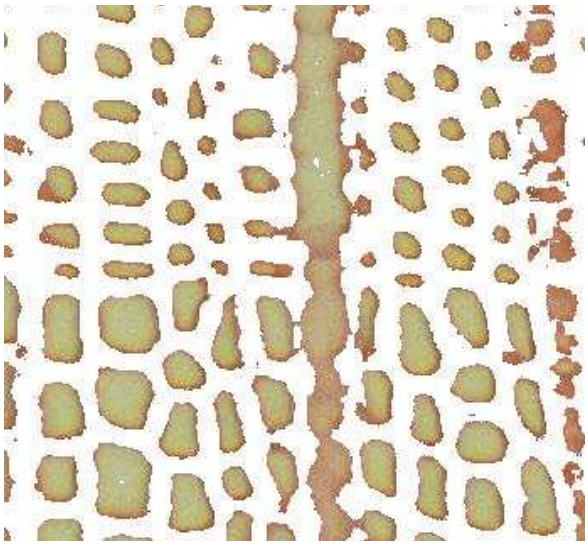


Figure 9 : *seuillage couleur (RGB) sur Pin Noir.*

II.2.2.2. Détection de contours

Une autre approche pour segmenter les cellules aurait pu être la détection de contour. En effet la variabilité d'intensité au contour des cellules est généralement importante du fait du contraste paroi/lumen et donc potentiellement détectable, le but de la détection de contours étant de repérer les points d'une image numérique qui correspondent à un changement brutal de l'intensité lumineuse. Ces changements de propriétés de l'image traduisent en général des événements importants ou des changements dans les propriétés du monde. La détection des contours d'une image réduit de manière significative la quantité de données et élimine les informations qu'on peut juger moins pertinentes, tout en préservant les propriétés structurelles importantes de l'image. Il existe un grand nombre de méthodes de détection de l'image mais la plupart d'entre elles peuvent être regroupées en deux catégories :

- La première recherche les extremums de la dérivée première, en général les maximums locaux de l'intensité du gradient[15].
- La seconde recherche les zéros de la dérivée seconde, en général les annulations du laplacien[16] ou d'une expression différentielle non-linéaire.

Les méthodes se basant sur la dérivée première étant plus simples à implémenter et plus compétitives que celles du second ordre d'après la littérature, mes essais se sont focalisés sur les filtres de Prewitt, Sobel et Canny. En sortie de ces filtres, on récupère une image en niveaux de gris comportant les contours détectés, il faut seuiller cette image d'une manière ou d'une autre.

Cette approche paraît satisfaisante globalement sur les images, sans pour autant apporter une solution directe à notre problème.

[15] Le gradient est un vecteur représentant la variation d'une fonction par rapport à la variation de ses différents paramètres.

[16] Opérateur différentiel défini par l'application de l'opérateur gradient suivie de l'application de l'opérateur divergence (opérateur différentiel linéaire aux dérivées partielles premières qui transforme un champ vectoriel en un champ scalaire).

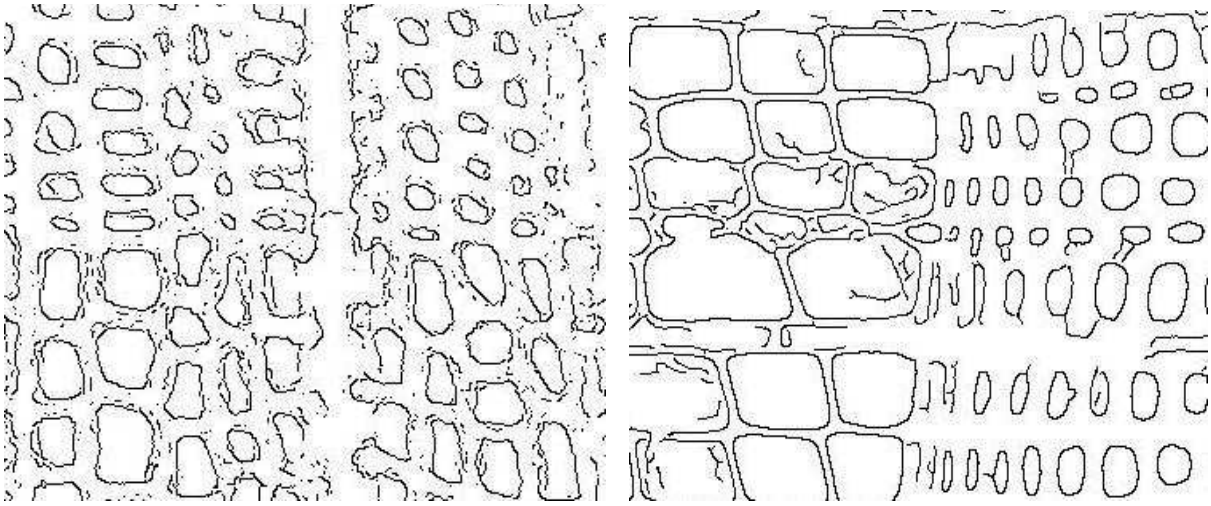


Figure 10 : *Résultat de la détection de contours par Canny, à gauche sur Pin Noir, à droite sur l'Epicéa.*

Plusieurs problèmes ont été rencontrés au sein de ces différents essais :

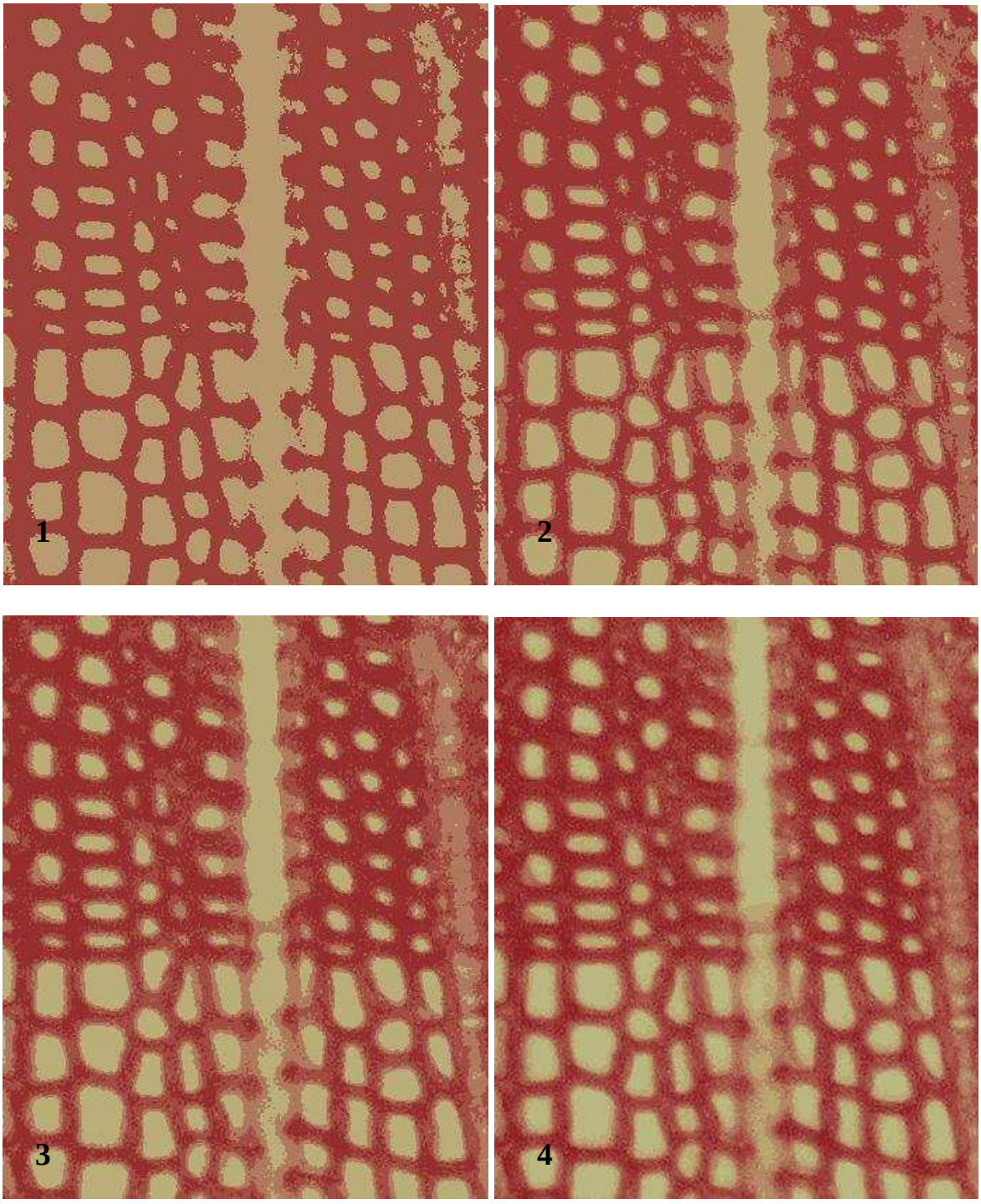
- La détection de faux contours (qui ne sont pas des contours de cellules).
- Les contours sont ouverts, voir partiels.
- Il arrive que l'on n'ait qu'un seul contour pour plusieurs cellules : fusion de cellules.
- Des régions entières, mal contrastées, floues, ne font ressortir aucun contour de cellules.

Au delà du seuillage (ici le seuillage par entropie), qu'il faut correctement choisir et automatiser sur les images de contours, car il influe directement sur une sous-détection ou sur-détection des contours détectés, il faudrait un travail supplémentaire de fermeture de contours pour obtenir de ces images ce qu'on l'on souhaite. Après quelques essais de caractérisation des différents cas à gérer dans une image, et quelques recherches bibliographiques dans ce domaine, il nous a semblé bon de ne pas s'attarder sur des méthodes complexes nécessaires à la résolution de ce problème ô combien de fois traités abondamment dans la littérature [2] sans toutefois conduire à un résultat satisfaisant.

II.2.2.3. Clustering

Une autre approche intéressante de segmentation, du point de vue région, est le clustering, méthode largement employée dans le domaine du partitionnement de données quelles qu'elles soient. Le clustering est une méthode statistique d'analyse de données qui a pour but de regrouper un ensemble de données en différents paquets homogènes, en ce sens que les données de chaque sous-ensemble partagent des caractéristiques communes, qui correspondent le plus souvent à des critères de proximité que l'on définit en introduisant des mesures de distance. Dans le domaine du traitement d'images, les caractéristiques communes sont l'intensité lumineuse que porte chacun des pixels.
Plusieurs types d'algorithmes de clustering existent ; nous avons retenu pour nos tests celui des k-means, car simple à comprendre et d'une efficacité éprouvée : il permet de répartir un ensemble de données en k classes.

Voici les résultats avec un k variant de 2, 3, 4 et 8 :



Figures 11 : *Application du K-means avec k=2 (1), k=3 (2), k=4 (3), k=8 (4) sur Pin Noir.*

Cette méthode semble bien différencier les différents types structuraux de l'image comme le lumen et la paroi. Cependant on remarque assez rapidement que deux clusters ne suffiront pas à identifier correctement ces deux régions. On se doit alors d'augmenter le nombre de partitions et de choisir correctement les clusters d'intérêt dans notre cas ; et c'est bien là que réside le problème de cette approche. En effet comment définir le nombre de cluster optimal permettant à chaque groupe de classes de bien séparer le lumen de la paroi ? De plus, même si cette question pouvait trouver réponse dans des méthodes de convergence (sans garantir le

résultat), un autre problème majeur concerne le groupe de cluster représentant le lumen de la cellule. Celui-ci n'est pas toujours localement correct (notamment au contact des cellules au rayon). Il faudrait donc faire une sélection cas par cas en partant des clusters représentants le groupe ayant une intensité moyenne la plus élevée. Il faudrait ensuite sélectionner le cluster supérieur en accord avec des conditions qui nous sont restées dures à définir clairement pour fonctionner dans la totalité d'une image (facteurs de forme, taille, distance, travail sur les masque des clusters… etc). Il est à noter que cette méthode reste tout de même très performante sur des portions d'image, contrairement à une image entière.

II.2.2.4 Ligne de partage des eaux (LPE)

La ligne de partage des eaux (ou watershed en anglais) utilise la description des images en termes géographiques. Une image peut en effet être perçue comme un relief si l'on associe le niveau de gris de chaque point à une altitude. Il est alors possible de définir la ligne de partage des eaux comme étant la crête formant la limite entre deux bassins versants. Cet algorithme est détaillé au chapitre II.3.1.2.1.

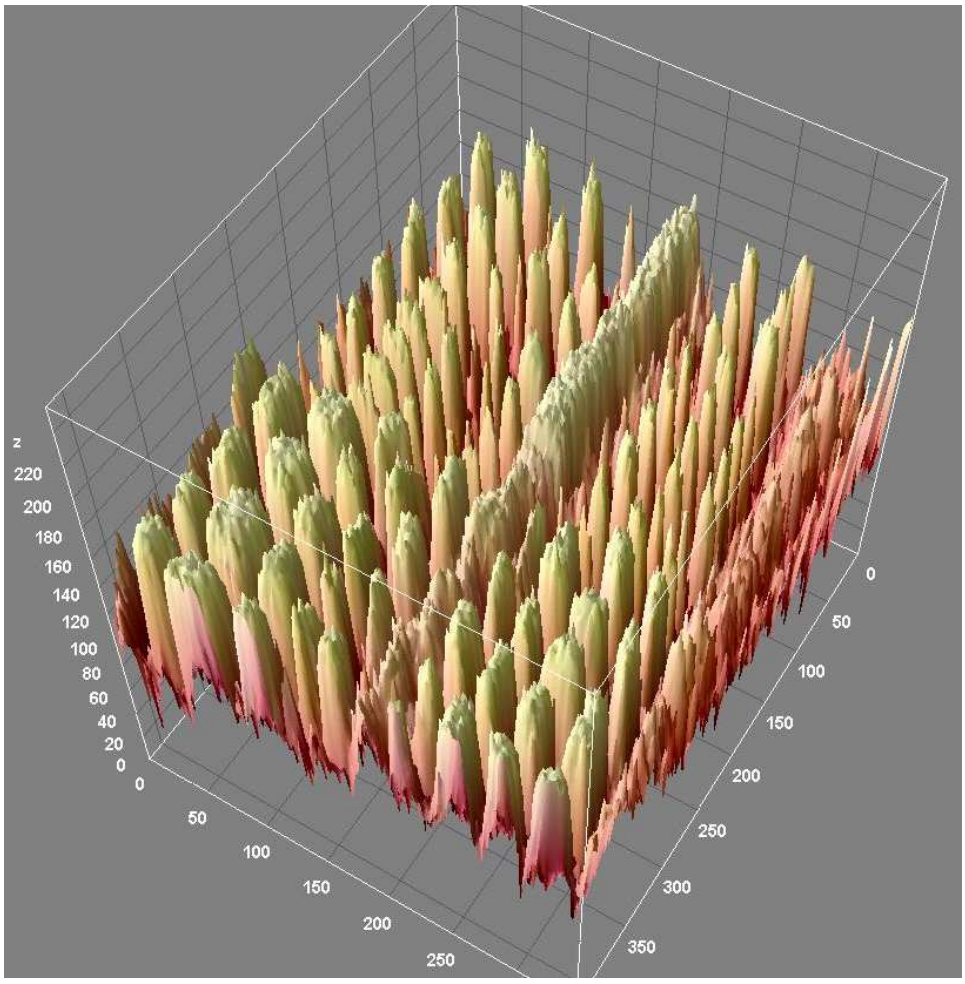


Figure 12 : *Plot 3D de Pin Noir.*

Cette méthode nous fournit des résultats intéressants, mais certains points restent à préciser. En effet le défaut majeur de cette méthode est la sur-segmentation qui est due à beaucoup de minimaux locaux présents dans une image. Nous avons donc dû remédier à ce problème en

« lissant » l'histogramme de notre image grâce à plusieurs filtres qui dans l'ordre sont le Mean Shift, le filtre Médian, puis le filtre de flou gaussien. Nous détaillerons ces filtres dans le chapitre II.3.1. Ainsi les bruits, et détails de l'image sont atténués et un nombre moins important de minimums locaux donnent naissance à un nombre réduit de bassins versants. Nous verrons aussi que d'autres techniques prenant part d'informations géométriques comme une carte des distances[17] ou un gradient morphologique de l'image[18] permettent de diminuer considérablement la sur-segmentation mais n'ont pas été employées dans notre cas.

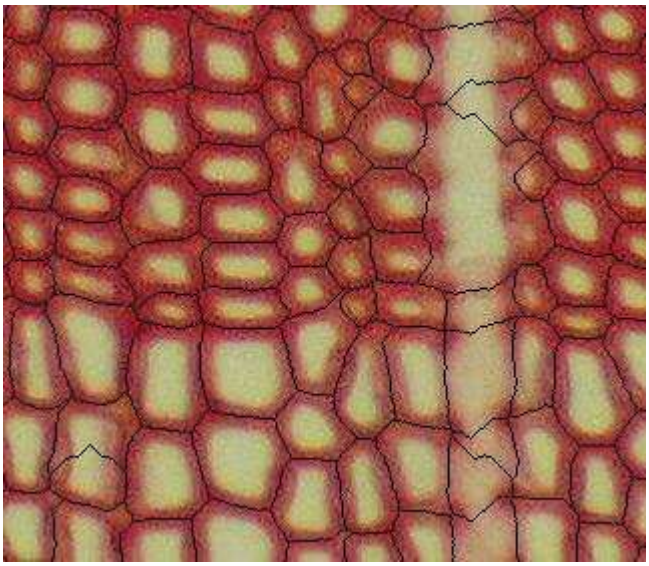


Figure 13 : *Test du LPE sur Pin Noir.*

Malgré le prétraitement réalisé sur l'image, il semble encore y avoir de fausses lignes de partage, notamment et principalement certaines qui traversent le lumen des cellules en plein milieu. Apres une étude prospective statistique sur le masque de ces arêtes, on peut définir quel profil d'arêtes devrait disparaître et ainsi remédier à notre sur-segmentation. Seule l'information du maximum d'intensité peut nous aider dans ce cas.

La réduction du nombre de bassins de la LPE peut se réaliser sous la forme d'une réduction de graphe. Comme nous le verrons plus tard la structure de graphe nous apportera d'autres informations primordiales à la résolution de notre problématique [5].

Nous détaillerons cette réduction de graphe dans le chapitre II.3.1.2.2.

[17] Représentation d'une image numérique qui associe à chaque pixel de l'image la distance au *point obstacle* le plus proche. Ces points obstacles peuvent être les points du contour de forme dans une image binaire.

[18] Mesure en chaque point de l'image la différence entre le maximum et le minimum des niveaux de gris sur le voisinage défini par un élément structurant (rectangle, disque…).

II.2.2.5. Espace de couleur

Nous nous sommes posé la question de l'espace de couleurs le plus adapté lors des traitements réalisés précédemment. En effet certains espaces apportent d'autres informations (contraste, saturation…) et permettent parfois un meilleur traitement (la segmentation notamment). De même les différents canaux R, G, B ont de la même façon été testés.

Nous nous sommes notamment penchés sur le HSV, reconnu pour faire ressortir certains aspects colorimétriques. HSV est l'acronyme pour *Hue Saturation Value* en anglais ou TSV en français *pour Teinte Saturation Valeur* où :

- teinte : le type de couleur (comme rouge, bleu, jaune…),
  La valeur varie entre 0 et 360, mais est parfois normalisée en 0–100 % ;
- saturation : l'« intensité » de la couleur :
  Varie entre 0 et 100 % ;
  Est parfois appelé « pureté » ;
  Plus la saturation d'une couleur est faible, plus l'image sera « grisée » et plus elle apparaîtra fade, il est courant de définir la « désaturation » comme l'inverse de la saturation ;
- valeur : la « brillance » de la couleur :
  Elle varie entre 0 et 100%.

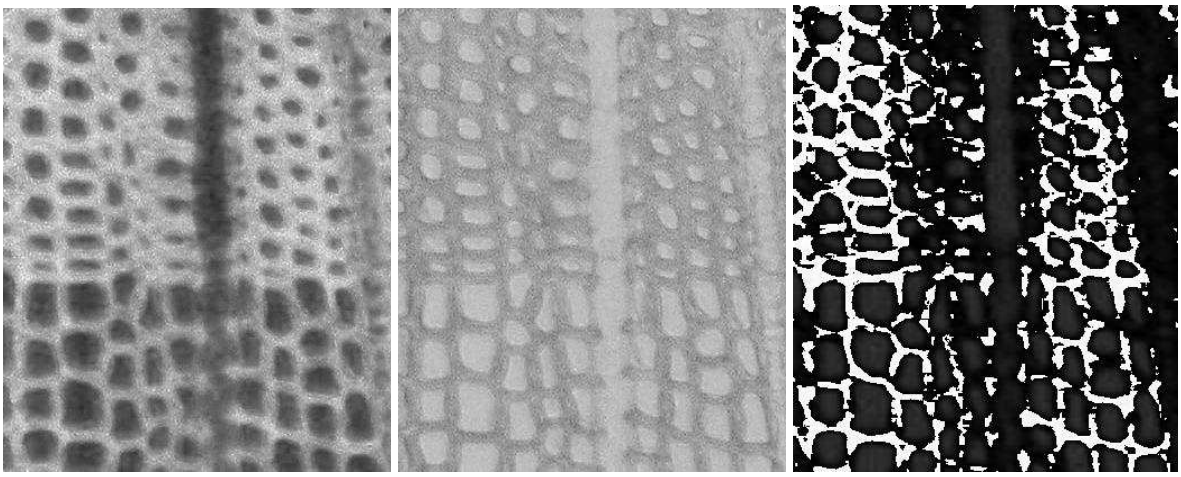


Figure 14 : *de gauche à droite ; teinte, saturation, valeur ; sur Pin Noir.*

Bien que l'image de la teinte semble faire ressortir le contraste lumen/paroi, les algorithmes de segmentation précédemment vus ne semblent pas mieux fonctionner. Nous avons donc choisi de rester dans un espace classique RGB.

II.2.3. Conclusion

Toutes les méthodes classiques existantes pour la segmentation des images cellulaires ne sont pas satisfaisantes car pas assez robustes et pas entièrement automatiques.

Tout au long de cette pré étude, il est apparu clairement qu'un traitement générique pouvant traiter aussi bien les images produites par le protocole avec coloration (Pin Noir) et sans coloration (Epicéa) n'était pas possible.
Le problème majeur reste la variabilité existant entre toutes les images aussi bien d'un point de vue général que d'un point de vue local, une variabilité aussi bien protocolaire qu'humaine.

Les différences sont apparues lors de la segmentation au niveau du changement de cerne principalement mais aussi : aux zones périphériques aux rayons, sur les cellules portant des points de ponctuation, sur des cellules déchirées pendant la préparation.

Du fait du manque d'images concernant le protocole sans coloration, nous n'avons pu faire de plus amples essais sur ce type d'image

Les problèmes rencontrés n'étant pas aisément formalisables, et les anatomistes, architectes, préparateurs n'étant pas sûrs de la pérennité de telle ou telle forme d'acquisition, un choix a été fait en accord avec les encadrants : ne travailler que sur les images produites suivant le protocole avec coloration, qui étant lui-même réalisé à l'AMAP, permettait de fournir davantage de clichés et interagir avec les préparateurs.

L'approche qui donne les meilleurs résultats est la segmentation par LPE après avoir réalisé un travail de débruitage de l'image grâce aux filtres Mean Shift et Médian, puis de l'avoir floutée grâce à une floue gaussien. Cette méthode a en particulier pu résoudre le problème de la séparation entre les trachéides et parenchymes.

II.3. Méthode mise en œuvre

Nous décriront ici la chaîne de traitement adoptée pour l'identification automatique des files cellulaires. Chaque étape serra détaillée, brièvement discutée notamment sur l'aspect paramétrage, après avoir présenté synthétiquement l'enchaînement général des algorithmes.

La chaîne de traitement adoptée peut se découper en trois phases :

- La segmentation
- La caractérisation des régions
- La reconnaissance des files

Suite à l'étude prospective décrite précédemment, nous avons orienté notre segmentation sur l'algorithme de lignes de partage des eaux (LPE) suivi d'une réduction du nombre de bassins.

La chaîne de traitement complète prenant en entrée une image prise par le biologiste et produisant en sortie une image résultat identifiant les files et un fichier de données avec les caractéristiques de celles-ci est présentée ci-dessous.

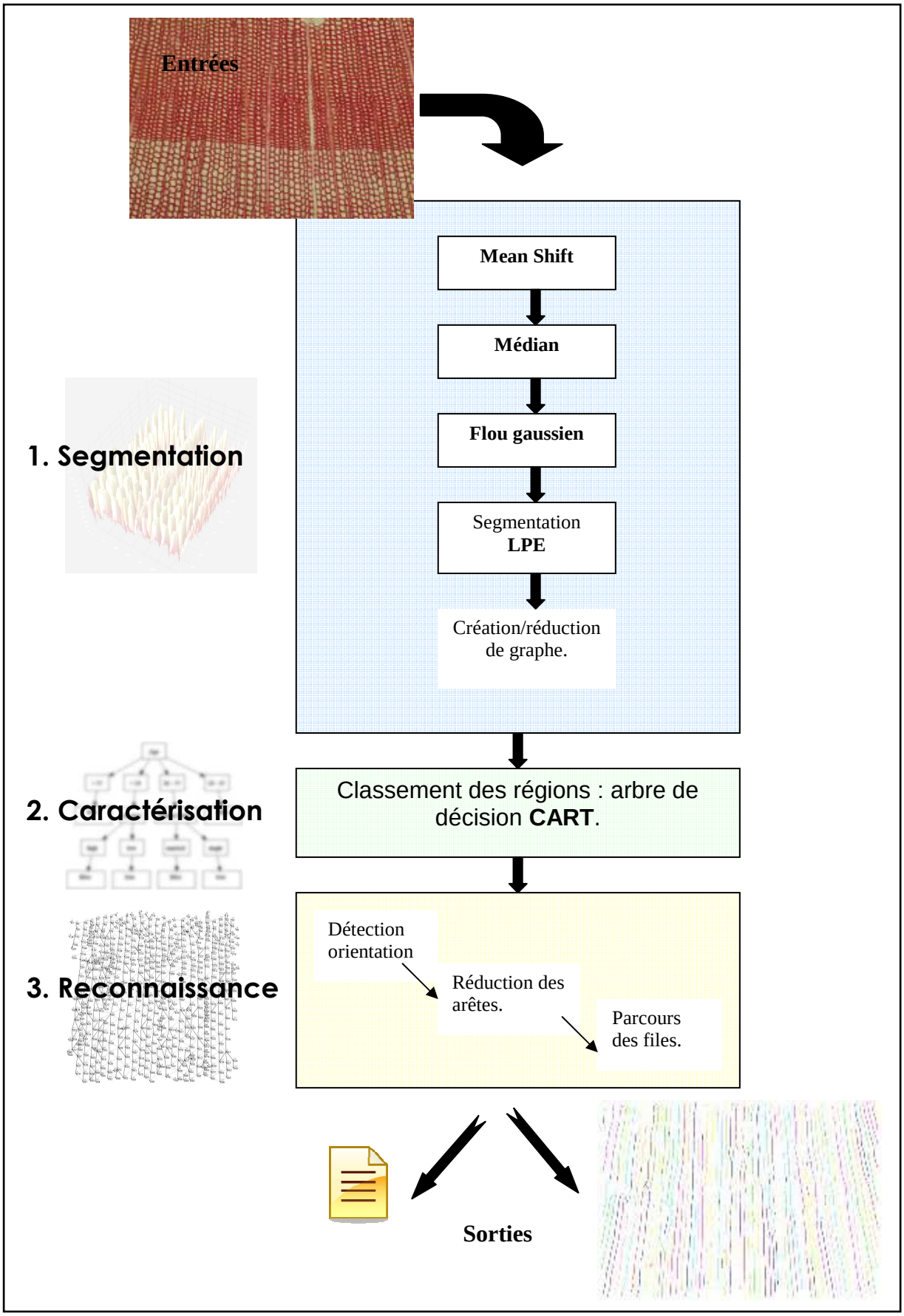


Figure 15 : *Chaîne de traitement établie pour le plugin Ficeler.*

Nous illustrerons le résultat de chaque traitement sur un fragment de l'image teste Pin Noir :

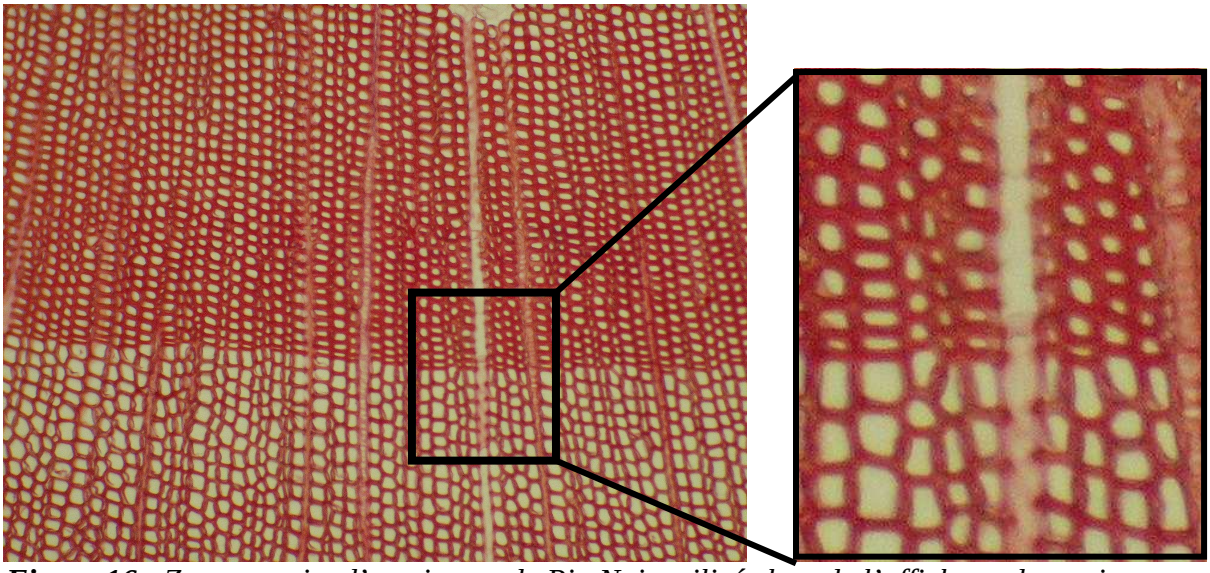


Figure 16 : *Zone extraite d'une image de Pin Noir utilisée lors de l'affichage des traitements.*

II.3.1. Segmentation

La segmentation automatique des images fournies se décompose en trois sous parties qui sont :

- le prétraitement.
- le découpage en régions.
- la mise en forme des données et réduction du nombre de régions.

II.3.1.1. Prétraitements

Les traitements appliqués dans cette partie ont non seulement pour but d'améliorer la lisibilité de l'image en supprimant au maximum le bruit et en atténuant les variations locales, mais aussi de soustraire un maximum de détails pouvant perturber la segmentation. Un bon travail à ce niveau évite une trop grande sur-segmentation car élimine les minimums locaux identifiés par l'algorithme de partage des eaux comme nous le verrons dans la partie suivante.

II.3.1.1.1. Le filtre Mean Shift :

Le premier filtre appliqué est le filtre Mean Shift qui à l'origine a pour but de réduire le bruit gaussien. On appelle bruit gaussien une perturbation dont la densité de probabilité de l'amplitude suit une loi gaussienne centrée.

Le Mean Shift est un algorithme itératif qui a pour objectif de faire converger un point vers le maximum local le plus proche. Celui-ci fonctionne aussi bien sur une image en niveau de gris que sur une image couleur : dans le premier cas, cela revient à chercher le maximum d'une gaussienne sur l'histogramme local défini par une fenêtre, dans le second cas, cela revient à rechercher le maximum d'une hypersurface gaussienne (l'histogramme comportant 3 dimensions : Rouge, Vert, Bleu, toujours selon une fenêtre fixée).

Voici comment l'on procède :

1. On commence par choisir un point de départ P.
2. On cherche l'ensemble E des points qui sont dans le voisinage de P. (définit par la taille d'un masque fixé)
3. On déplace P vers l'isobarycentre de E.
4. On réitère depuis l'étape 2 jusqu'à convergence.

Les déplacements successifs vers l'isobarycentre font converger le point P vers les zones de fortes densités.

Exemple du déroulement de l'algorithme pour des points en dimension 2 :

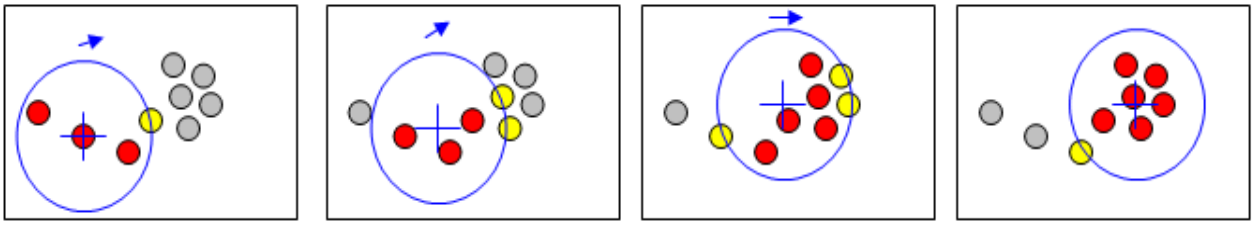


Figure 17 : *Convergence de l'algorithme Mean Shift sur la recherche du voisinage de plus forte densité.*

L'espace des caractéristiques représente l'espace des points sur lesquels nous allons appliquer l'algorithme Mean Shift.

Dans notre cas, cet espace devra contenir l'histogramme RVB de l'image bruitée, c'est donc un espace à 3 dimensions :

E = { ensemble des points P, avec P=(rouge, vert, bleu) }

Chaque point P aura une valeur qui représente le nombre d'occurrences de la couleur (rouge, vert, bleu) dans le voisinage considéré.

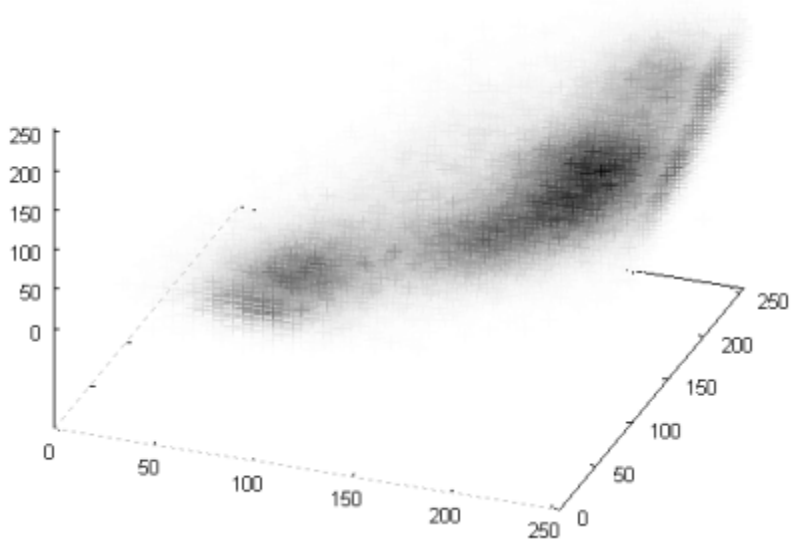


Figure 18 : *Espace des caractéristiques (occurrences RVB).*

Les résultats en sortie de ce filtre sont très satisfaisants ; la majorité du bruit de l'image a été supprimé. Le bruit restant peut alors être considéré comme un bruit impulsionnel et être traité avec des filtres spécifiques.

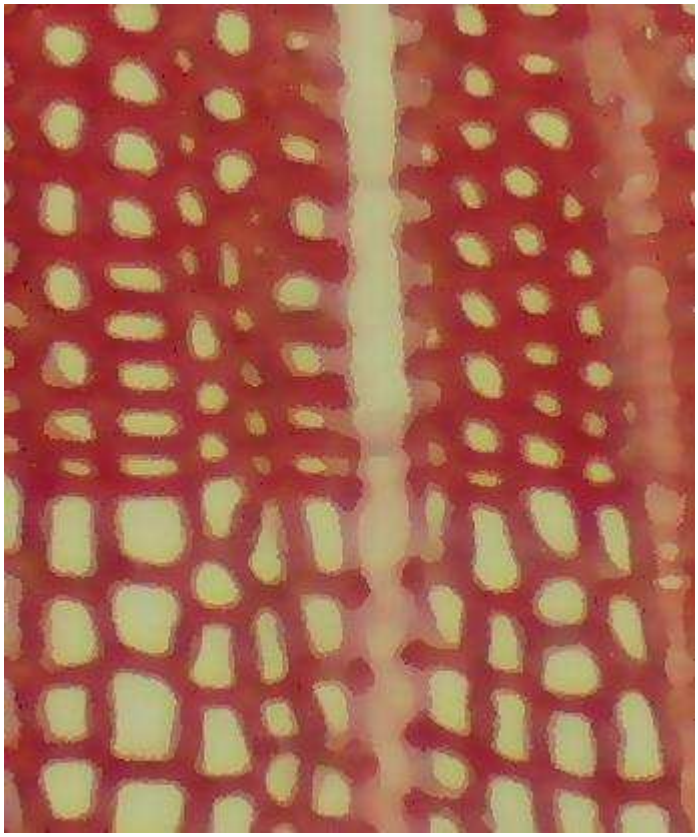


Figure 19 : *résultat du filtre Mean Shift sur Pin.*

Remarque sur les paramètres :

Ce filtre utilise deux paramètres :

- la taille du masque correspondant au voisinage d'un pixel P à traiter lors du calcul de l'espace des caractéristiques (2D).
- La taille du voisinage (3D) à visiter dans l'espace des caractéristiques lors de la recherche de l'isobarycentre de P

Comme très souvent en traitement d'images, la difficulté pour automatiser une tâche est souvent liée aux choix des paramètres. Ici le résultat du filtre dépend principalement de la taille du voisinage RGB. Une taille trop petite va laisser des pixels bruités (effet de grain). Une taille trop grande va uniformiser les couleurs (effet de flou).

Après une étude expérimentale, nous avons pu déterminer une règle simple déterminant ces paramètres en fonction de la résolution de l'image et de son niveau de zoom, même si l'on remarque que plus l'image est zoomée et en grande résolution, plus on va chercher à uniformiser les couleurs. Cependant, augmenter ces paramètres est coûteux en temps de calcul, et il est souvent bien plus simple et économique de tout simplement baisser la résolution de telles images.

II.3.1.1.2. Filtre médian

Comme nous venons de le voir le bruit restant sur l'image peut être considéré comme bruit impulsionel, c'est-à-dire très fort et ponctuel.
Ce type de bruit peut être réduit par l'utilisation d'un filtre Médian. Son fonctionnement consiste à remplacer la valeur d'un pixel par la valeur médiane de l'ensemble des pixels de son voisinage suivant la taille d'un masque donnée.

Les résultats montrent bien que les derniers pixels bruités sont corrigés :

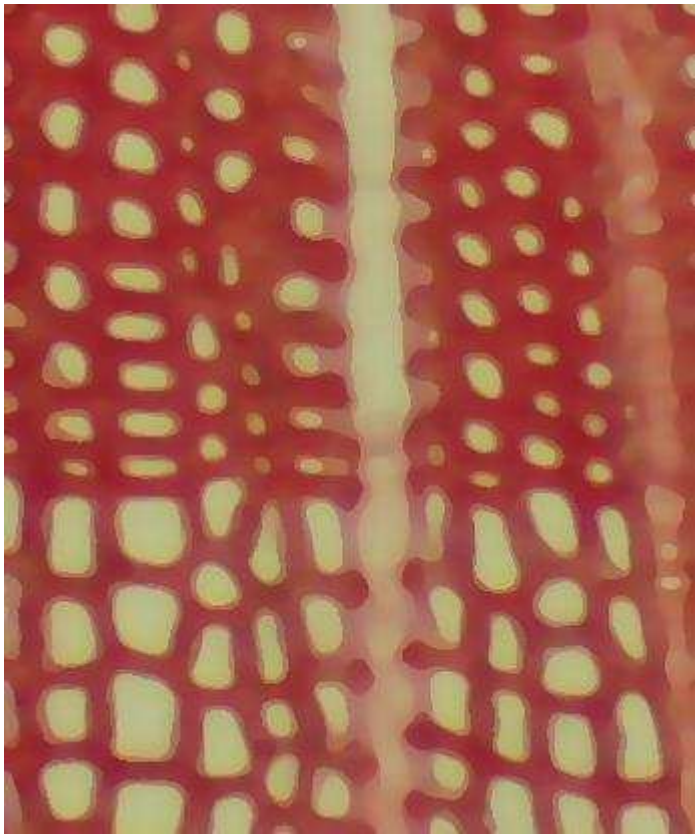


Figure 20 : *Résultat du filtre Médian sur Pin Noir.*

On peut encore se pencher sur la taille du masque à utiliser dans le cas d'un traitement tout automatique. Après plusieurs essais il semble qu'un voisinage de taille égale à trois fonctionne sur toutes les images.

II.3.1.1.3. Flou gaussien

Enfin on prépare l'image pour l'algorithme de Ligne de Partage des Eaux. Pour ce faire, on veut diminuer le détail de l'image ; l'utilisation d'un filtre de floutage reste simple et efficace dans ce cas.
On choisira un filtre à noyau gaussien dont les valeurs son calculées selon le modèle suivant pour une fenêtre de 3*3 pixels :

$$\begin{pmatrix} g(-1,-1) & g(0,-1) & g(1,-1) \\ g(-1,0) & g(0,0) & g(1,0) \\ g(-1,1) & g(0,1) & g(1,1) \end{pmatrix}$$

Où $g(x,y) = e^{-\frac{(x^2 y^2)}{2\acute{o}^2}}$

Ici on aura σ = 2, comme l'utilise les fonctionnalités d'*imageJ* par défaut.
Par contre la taille du masque a une importance significative sur l'effet de sur/sous-segmentation. De manière générale plus l'image est zoomée et de grande résolution, plus on augmente le masque pour éliminer plus de détails.

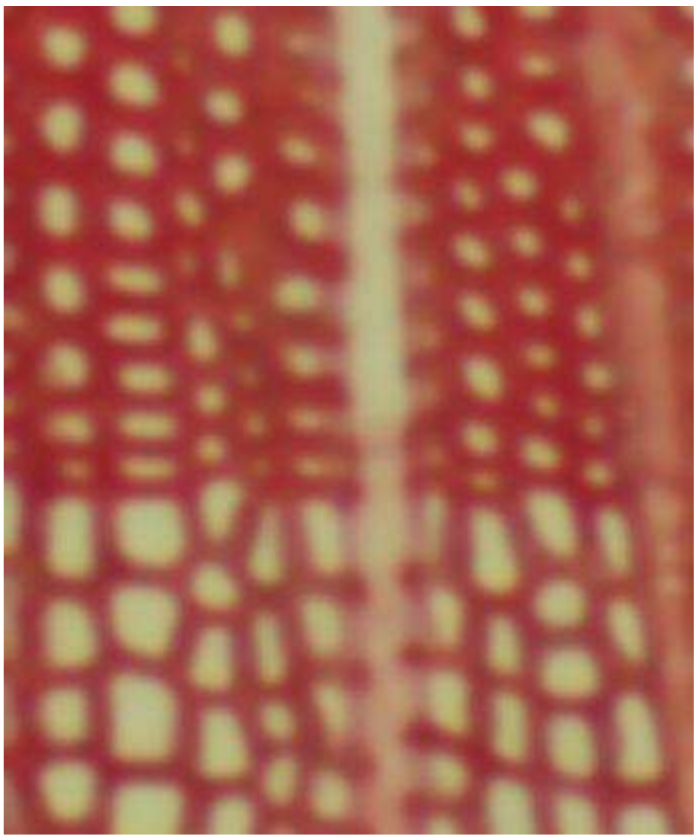


Figure 21 : *Résultat du filtre de Flou gaussien, taille du masque = 3.*

II.3.1.2. Segmentation en région

Comme nous l'avons déjà vu, cette étape cruciale est réalisée grâce à l'algorithme de Ligne de Partage des Eaux (LPE) ou WaterShed. Nous allons décrire les grands principes de cet algorithme dans cette partie comme on le trouve dans la littérature [3].

II.3.1.2.1. L'algorithme de Lignes de Partage des Eaux (LPE)

L'idée de base derrière toutes les méthodes de segmentation reposant sur la ligne de partage des eaux est de considérer une image en niveaux de gris comme un relief topographique. Il s'agit alors de calculer la ligne partage des eaux du dit relief. Les bassins versants ainsi obtenus correspondent aux régions de la partition.

Si on considère l'image à niveaux de gris comme une fonction de $\mathbb{R}^2$dans $\mathbb{R}$ suffisamment régulière, on peut alors définir proprement la ligne de partage des eaux. Cependant, dans le cas discret (fonctions de $\mathbb{Z}^2$dans$\mathbb{Z}$), qui correspond mieux aux images considérées, il n'existe pas à ce jour de définition consensuelle de cette transformation. Plusieurs algorithmes ont été proposés pour la calculer, donnant autant de définitions, pas toujours équivalentes.

On peut classer les algorithmes de construction de la ligne de partage des eaux en trois catégories. Les algorithmes par inondation simulent une montée progressive du niveau d'eau à partir des minima du relief. Les algorithmes par ruissellement suivent, à partir de chaque pixel de l'image, la ligne de plus grande pente jusqu'à atteindre un minimum. Finalement, les algorithmes topologiques proposent une déformation progressive du relief, préservant certaines caractéristiques topologiques, jusqu'à ce qu'il soit réduit à une structure fine correspondant à la ligne de partage des eaux.

Nous nous intéresserons à la version par inondation dont voici le principe :
La montée des eaux consiste à immerger la surface topographique dans de l'eau. A chaque fois que la hauteur des eaux atteint la hauteur d'un label, un nouveau bassin versant est créé dont la couleur est celle du label. A chaque fois que deux bassins de couleurs différentes se rencontrent, on empêche leur fusion en créant une digue, nommée ligne de partage des eaux. En revanche si les deux bassins sont de même couleur, ils fusionnent.

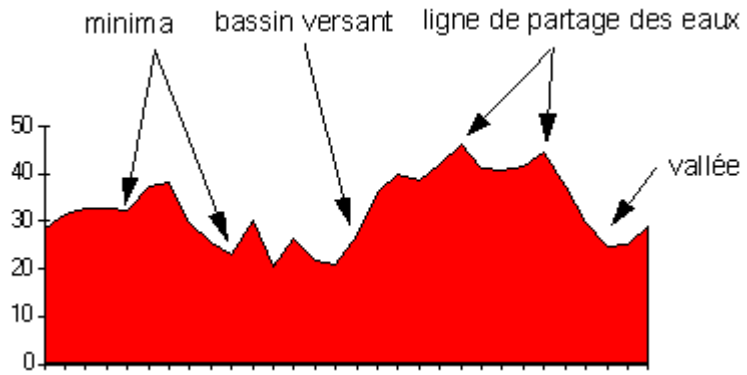


Figure 22 : *Illustration d'une surface topographique.*

Pour construire la ligne de partage des eaux, il suffit d'imaginer le procédé suivant :

- On perce un trou dans chaque minimum local du relief.

- On procède à une inondation du relief à partir des minima régionaux, de manière à ce que le niveau des eaux monte à vitesse constante et soit uniforme dans tous les bassins versants, en précisant que l'eau ne peut pénétrer dans les vallées que par ses minima.
- Chaque fois que les eaux issues de deux minima régionaux se rencontrent (les premiers points où cela se produit sont des points col ou selle du relief), on construit une digue le long de la ligne de crête, de manière à ce que les eaux issues de deux bassins versants différents ne se mélangent pas.
- Une fois le relief totalement immergé, les digues formées sont les lignes de partages des eaux (watersheds).

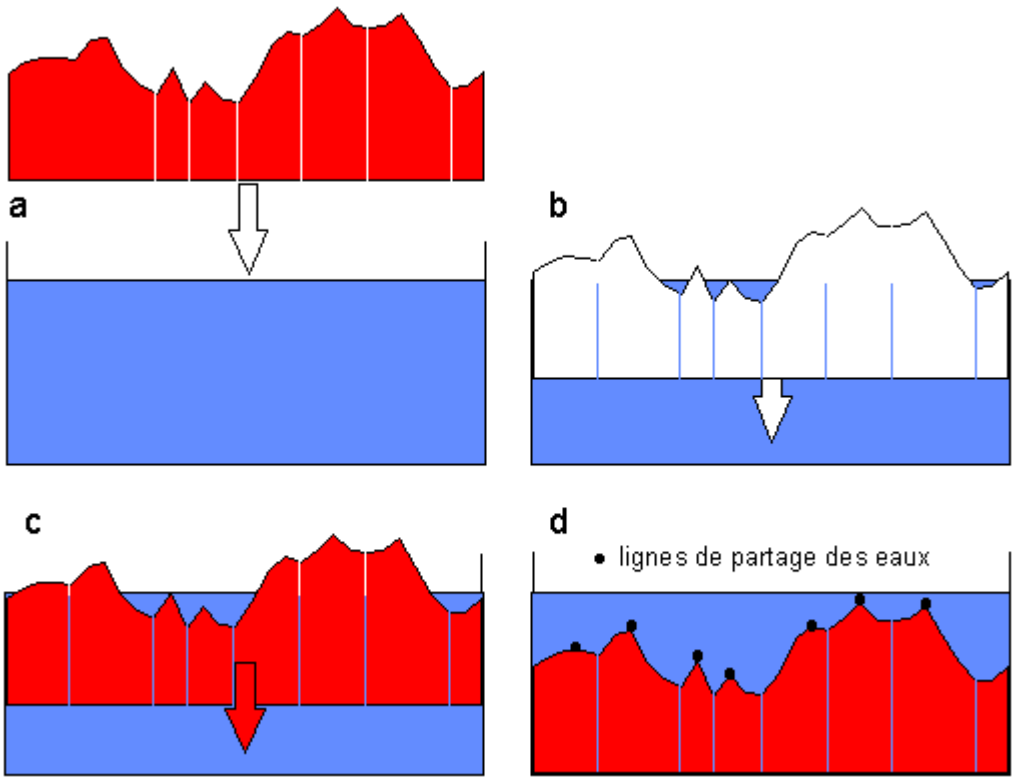


Figure 23 : *illustration de l'immersion d'une surface topographique.*

Une des difficultés de la mise en œuvre de cette analogie intuitive est qu'elle laisse beaucoup de liberté quant à sa formalisation. Il existe deux classes principales d'implémentation : l'une est basée sur un algorithme récursif d'immersion [3] et une autre basée sur l'utilisation de fonctions de distances géodésiques [4].

Nous avons repris une implémentation de Vincent and Soille (1991) [3] disponible librement par la communauté de développeur autour d'ImageJ. Nous l'avons adapté à nos besoins, notamment inséré dans notre plugin. On trouvera en annexe 1 un pseudo code de l'algorithme.

Les résultats restent satisfaisants hormis quelques zones sur-segmentées. En réalité la sur-segmentation dépend principalement du paramétrage des filtres utilisés durant le prétraitement. Comme nous l'avons vu dans la pré-étude, nous n'avons aucun moyen de remédier à la sur-segmentation restante.

Il est à noter que les lignes fournies par l'algorithme n'étant pas toujours en 4-connexité[19], nous avons appliqué l'algorithme de la squelettisation[20] sur cette image binaire. Ainsi les lignes de partage seront exploitables plus facilement lors de l'extraction des régions.

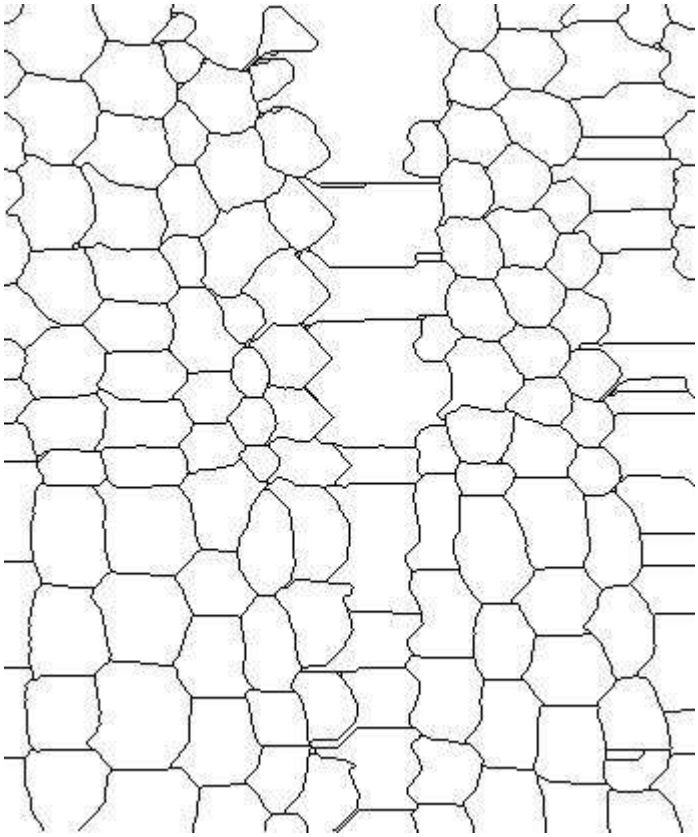


Figure 24 : *Résultat de l'algorithme LPE.*

II.3.1.2.2. Réduction des régions

La réduction de régions suite à l'application d'un Watershed est une chose courante dans la littérature en traitement d'image. Elle se fait principalement en créant un graphe représentant la topologie de l'image en identifiant les bassins comme nœuds et en reliant les nœuds dont les bassins sont adjacents. Ainsi le travail de réduction de régions revient à faire une réduction de graphe en fusionnant certains nœuds selon les conditions recherchées. Nous verrons que la structure de graphe sera très adaptée aux étapes ultérieures du traitement, comme suggéré en [5].

II.3.1.2.2.1. Création d'un graphe d'adjacence

Au préalable, nous avons extrait de l'image résultante de l'algorithme de Lignes de Partage des Eaux une structure de données comportant une liste de bassins et leur relation d'adjacence entre eux (matérialisée par une ligne de séparation). On trouvera en détail l'algorithme de détection de régions en annexe 2.

[19] Système de voisinage entre pixel, ici une grille carrée (nord, sud, est, ouest).
[20] La squelettisation fait partie du domaine de la morphologie mathématique : il fournit un squelette (trait d'épaisseur égale à 1 pixel) en partant d'une image binaire.

De ces données, nous allons donc créer un graphe d'adjacence. Le choix de la structure de graphe étant classique, nous nous sommes orientés sur une librairie performante portant bon nombre de variantes de structures et beaucoup de fonctionnalités optimisées. Nous utiliserons la librairie JUNG[21].

Ici, l'intérêt de l'utilisation d'un graphe réside d'abord dans le fait que nous allons faire porter des informations sur les nœuds tout comme sur les arêtes. En l'occurrence chaque nœud porte un objet des bassins versants de la LPE, alors que chaque arête porte un objet arête stockée dans notre structure. Le graphe créé est de type non orienté.

Les objets portés par le graphe sont donc les objets *FBasin* et *FEdge* identifiés dans notre classe *FData*. La description des ces classes est fournie en annexe 3, le diagramme des dépendances en annexe 8. Nous pouvons cependant citer quelques informations portés par chacun de ces objets :

- Un bassin est défini par : un identifiant, sa liste de pixels positionnés dans l'espace, sa taille, mais aussi une liste de masques portées sur différents états de l'image traitée (par exemple sur l'image d'origine, sur l'image prétraitée… etc.)
- Une arête est définie par : les identifiants des bassins qu'elle relie, la liste de pixels de la ligne d'adjacence[22] qui existe entre les centres de ces deux bassins ainsi que celle de la ligne de séparation[23] qui existe entre ces deux bassins, des informations de taille, une liste de masques des lignes d'adjacence et de séparation portés sur différents états de l'image, etc..

Nous avons éliminé volontairement les bassins bordant l'image lors de la création du graphe pour éviter tout effet de bord.

Voici une image présentant le graphe d'adjacence :

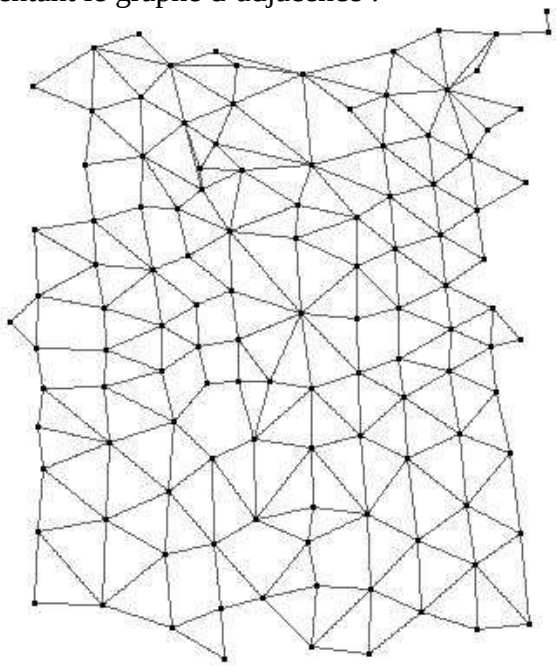


Figure 25 : *Graphe d'adjacence obtenu de la segmentation.*

[21] Java Universal Network/Graph Framework : http://jung.sourceforge.net/
[22] Ligne d'adjacence : ligne reliant les centres de deux bassins voisins.
[23] Ligne de séparation : ligne séparant deux bassins voisins (équivalent à la ligne de partage).

II.3.1.2.2.1. Réduction du graphe

Nous allons maintenant réduire le nombre de régions afin de remédier à la sur-segmentation grâce aux données portées par le graphe.

Nous avons identifié lors de la pré-étude que le maximum d'intensité de la ligne de séparation portée par les arêtes nous permet d'identifier les lignes de partage à supprimer. Cependant, dans certains cas, cette seule information n'est pas suffisante, en particulier pour les lignes de partage ne retombant pas exactement sur la paroi intercellulaire lors de la superposition de ces lignes sur l'image originale (figure 25).

Voici une illustration du problème rencontré :

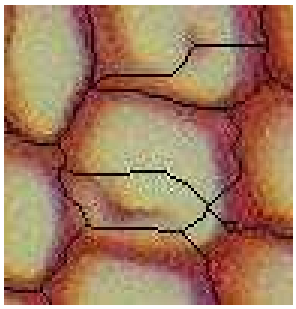


Figure 26 : *Décalage des lignes de partage par rapport aux parois.*

Pour remédier à ce problème, nous avons couplé l'information du maximum d'intensité de la ligne de séparation d'une arête au minimum d'intensité de la ligne d'adjacence. En effet, ce minimum se doit d'être faible si cette ligne ne passe pas par une paroi cellulaire (ce qui est le cas lors de sur-segmentation des cellules).

Ces deux seuils sont automatiquement évalués, ce qui permet de rendre l'approche stable et robuste.

Explication du calcul des seuils :

Nous utilisons pour ces seuils les valeurs d'intensité que portent principalement la paroi et le lumen. A l'observation de l'histogramme en niveaux de gris de l'image prétraitée, on peut aisément identifier deux pics d'intensité dans le sombre (paroi vers 80) et dans le lumineux (lumen vers 180).

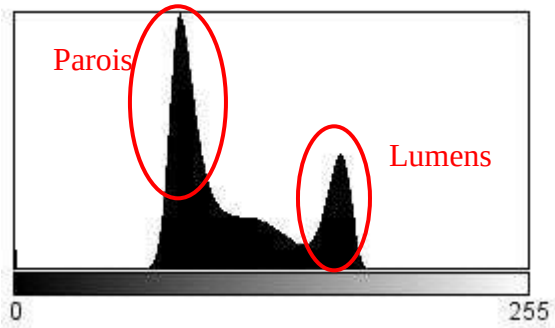


Figure 27 : *histogramme en niveaux de gris de l'image.*

Nous avons donc identifié les deux plus grands pics de cet histogramme pour définir quels seraient les seuils utilisés pour la fusion de basin :

maximum d'intensité de la ligne de séparation = valeur(pic lumineux) – 10
minimum d'intensité de la ligne d'adjacence = valeur(pic sombre) + 10

Le choix de s'écarter des pics de 10 est largement critiquable car cet écart dépend en fait de l'aplatissement de chaque pic ; une meilleure approche serait d'approximer l'histogramme par deux gaussiennes et d'en utiliser leurs sigmas (voir chapitre III.3.1). Cependant par manque de temps et soucis de parvenir jusqu'au bout du traitement des files, nous nous sommes satisfaits de ce choix qui semblait dans un premier temps remplir son rôle sur la majorité des images.

Les critères de fusion mis en place, il ne reste qu'à les appliquer au graphe. Dans un premier temps, tous les couples de bassins portant sur leur arête ces critères sont identifiés, puis fusionnés entre eux.
La fusion de deux basins V1 et V2 reliés par l'arête E, doit assurer :

- la suppression de V2
- la suppression de l'arête E
- la mise à jour de V1 : la concaténation des listes de pixels décrivant le nouveau bassin.
- La mise à jour des masques correspondant.
- La mise à jour des voisins de V1 en y ajoutant ceux de V2 ; c'est-à-dire ajouter une arête si elle n'existe pas, ou la fusionner si elle existe déjà (ne pas perdre l'information de la ligne de séparation) et mettre sa ligne d'adjacence à jour.

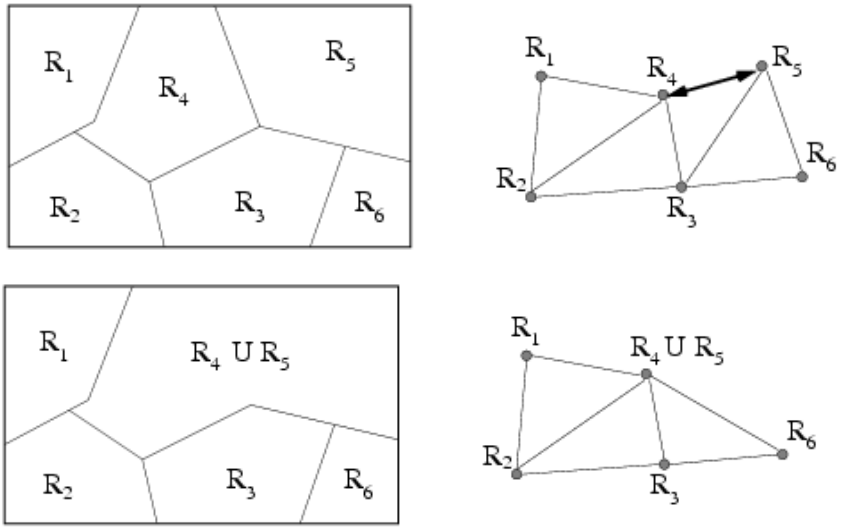


Figure 28 : *Exemple de fusion de deux régions dans le graphe (R4 et R5).*

Il est à noter qu'une table de correspondance est nécessaire pour retrouver un basin qui aurait déjà été fusionné et ne portant donc plus le même identifiant.

Nous pouvons remarquer que seule la sur-segmentation des cellules a été étudiée. Cependant les autres structures telles les vaisseaux, canaux, rayons, ou parois subissent aussi une sur-segmentation. Les critères de fusion énoncés précédemment peuvent servir sur les régions comportant un lumen, en particulier les vaisseaux, canaux et parfois rayons (cela dépend de la

préparation). Comme nous l'évoquerons dans le chapitre III.3.1, il serrait intéressant de définir d'autres critères de fusion spécifiques à la fusion de chaque type de structure pour ainsi obtenir une image la moins segmentée possible. Pour l'instant, nous préférons travailler avec une image sur-segmentée plutôt que sous segmentée pour la suite des traitements.

II.3.2. Caractérisation des régions

Nous avons obtenu des étapes précédentes un ensemble de régions dont nous ne connaissons pas, a priori, la nature. Il serrait pourtant particulièrement intéressant de connaitre de quelle nature sont chacune de ces régions pour la reconnaissance des files cellulaires (toutes les régions n'étant pas des cellules !). Cependant nous pouvons largement décrire ces régions avec bon nombre de paramètres aussi bien au niveau colorimétrique que topologique voir géométrique.

Nous décrirons donc, dans ce chapitre, comment nous avons tenté de caractériser la nature de nos régions, en nous restreignant à savoir s'il s'agissait de cellule ou non.

Il s'agit de déterminer deux classes de régions en fonction de leurs paramètres. De nombreux outils statistiques et probabilistes sont disponibles pour résoudre ce type de problème appartenant au domaine de l'analyse de données.

Quelques essais non concluants ont été faits avant de trouver une solution. Il est intéressant de noter l'approche menée pour déterminer le type et le nombre de paramètres pertinents.

Nous avons dans un tout premier temps fait une Analyse en Composante Principale[24] (ACP) sous R[25] pour déterminer la corrélation entre les *n* variables décrivant les régions. Il est apparu rapidement que notre jeu de données ne comportait soit pas assez de variables, soit pas assez d'individu, car nous obtenions des corrélations différentes en fonction des jeux utilisés. De même une analyse factorielle discriminante[26] (AFD) ne nous a pas permis de classer à 100% le peu de régions identifiées dans chacun des jeux. Une tentative de clustering a aussi été faite sur ces jeux de données, mais la superposition des résultats avec ceux de notre annotation manuelle nous a révélé que l'on ne pouvait pas délimiter correctement les deux classes recherchées, même en rajoutant une classe d'incertitude (essais avec deux, puis trois clusters).

Nous avons donc fait part de nos besoins d'images, mais avons aussi finit l'implémentation des sorties de données des images traitées (après réduction du graphe).

Entre temps une réunion à été réalisée avec ma tutrice de stage Sèverine Bérard affiliée à l'Université Montpellier 2 ainsi que M. Laurent Bréhélin enseignant l'analyse de données à l'Université. Plusieurs points ont été soulevés et un état de l'art a été dressé sur les méthodes de fouilles de données supervisées ou non supervisées pouvant aider à résoudre ce problème de classification. Il est apparu clairement que, ayant la possibilité de vérifier les résultats

[24] Méthode mathématique d'analyse des données qui consiste à rechercher les directions de l'espace qui représentent le mieux les corrélations entre *n* variables aléatoires.
[25] Langage de programmation et un environnement mathématique utilisés pour le traitement de données et l'analyse statistique.
[26] Technique statistique qui vise à décrire, expliquer et prédire l'appartenance à des groupes prédéfinis (classes, modalités de la variable à prédire, ...) d'un ensemble d'observations (individus, exemples, ...) à partir d'une série de variables prédictives (descripteurs, variables exogènes, ...).

d'une méthode quelconque en les comparant à une annotation manuelle, il était plus judicieux d'utiliser une méthode supervisée pouvant prendre en compte un apprentissage. En effet, celles-ci fourniraient dans tous les cas des résultats plus justes. Beaucoup de méthodes existent telles que les méthodes Bayesiennes, les K plus proches voisins, les arbres de décision, les réseaux de neurones… etc.
Ces différentes méthodes ont pu être testées grâce à un logiciel de fouille de données nommé Weka[27] sous licence GNU[28] réalisé en Java. Il n'a pas été aisé de choisir une méthode et d'en justifier son emploi. Nous discuterons d'ailleurs ce point dans le chapitre III.3.1. Cependant quelques éléments ont pu nous guider : ne connaissant pas a priori l'ensemble des variables utiles à notre classification, mais en disposant d'un grand nombre, certaines méthodes ont trouvées un franc avantage en n'utilisant d'elles-mêmes que les seules variables utiles à la classification. Par soucis d'avancer sur la reconnaissance de files, nous avons voulu nous affranchir d'une étude détaillée des corrélations entre variables nécessaire à leur choix préalable (utilisation de méthodes bayesiennes par exemple, contrairement aux arbres de décision). De plus, les recherches bibliographiques réalisées sur la segmentation des trachéides nous ont permis de choisir une méthode déjà éprouvée dans ce domaine, l'article de R. Jones [5] différencie en effet les cellules parmi les autres structures grâce à la méthode Classification And Regression Tree (CART). L'obtention d'un arbre de décision fournies des règles de décision intelligibles (par exemple : taille des cellules < 200), et une représentation hiérarchisée intuitive. Celui-ci peut être validé, modifié par un expert si nécessaire.

II.3.2.1. La méthode CART (Classification And Regression Trees)

L'algorithme CART, dont l'acronyme signifie « Classification And Regression Trees », s'attelle à construire un arbre de décision en classifiant un ensemble d'enregistrements. Cet arbre fournit ensuite un modèle pour classer de nouveaux échantillons. Il a été publié par Leo Breiman en 1984. Techniquement, l'arbre de décision obtenu est binaire et base ses critères de choix de variables sur l'indice de Gini.

Il a l'avantage :

- d'être peu perturbé par des individus « extrêmes »
- d'être peu sensible au bruit des variables non discriminantes
- de permettre l'utilisation de variables de tout types (continues, discrètes, catégoriques)
- de travailler sans hypothèses préalables,

Mais pour défaut :

- l'utilisation de règles heuristiques
- de nécessiter un grand nombre d'individus.

II.3.2.1.1. Théorie

[27] Logiciel libre de fouille de données Weka : http://www.cs.waikato.ac.nz/ml/weka/
[28] GNU est un projet de système d'exploitation composé exclusivement de logiciels libres.

Voici les étapes principales de la construction d'arbre. Nous ne rentrerons pas dans les détails mais préciserons les spécificités de l'algorithme CART.

- Trouver les variables qui séparent le mieux les individus de chaque classe.
- Déroulement de la construction :

  - Recherche de la variable et du seuil qui sépare le mieux
  - Applique la séparation à la population
  - Obtention de nouveaux nœuds

- Arrêt de l'approfondissement de l'arbre lorsque les conditions d'arrêts sont rencontrées
- « élagage » de l'arbre

Conditions d'arrêts existantes :

- La profondeur de l'arbre atteint une limite fixée (= nombre de variables utilisées)
- Le nombre de feuilles atteint un maximum fixé
- L'effectif de chaque nœud est inférieur à un seuil fixé
- La qualité de l'arbre est suffisante
- La qualité de l'arbre n'augmente plus de façon sensible

L'algorithme CART utilise l'indice de Gini pour définir les critères de séparation, il est défini par :

$$I = 1 - \sum_{i}^{n} f_i^{\,2}$$

Où :

n = nombre de classes à prédire
f_i = fréquence de la classe i dans le nœud

Plus l'indice de Gini est petit, plus le nœud est pur. En séparant un nœud en deux nœuds fils, on cherche la plus grande hausse de la pureté. Une fois l'arbre construit, les critères de divisions sont connus. On affecte donc chaque individu de l'apprentissage selon les règles obtenues, on remplit les feuilles. A chaque feuille, plusieurs classes d'individus sont présentes, on affecte donc à la feuille la classe pour laquelle la proportion d'individus à cette classe est la plus grande.

Pour éviter les problèmes que posent les arbres trop étoffés (trop de règles, trop spécifiques aux données ~ règles non reproductibles, trop peu d'individu dans les feuilles ~ aucune signification), on réalise un élagage.

Règles d'élagage :

- Création de l'arbre maximum (toutes les feuilles des extrémités sont pures)
- Elagages successifs de l'arbre
- Retenir l'arbre élagué pour lequel le taux d'erreur mesuré sur un échantillon test est le plus bas possible.

II.3.2.1.2. Construction de notre Arbre

Nous avons construit notre jeu d'apprentissage à partir de sept espèces du jeu d'images décrits dans le paragraphe II.1.2. : *Chamaecypavis obtusa, Chamaecypavis thyoides, Pinus Banksiana, Pinus Khasya, Pinus longifolia, Pinus monticola, Pinus radiata.* Nous avons pris deux images à un niveau de grossissement différent par espèce, soit quatorze images ; le jeu d'apprentissage a été défini avec l'expert biologiste ayant réalisé l'acquisition ; ont été retenues des zones structurellement hétérogènes ou non et de qualité de préparation variables. Nous avons manuellement réparti en trois classes environ 300 régions par image en trois classes : cellule, non cellule, indéterminé. Les variables caractérisant chaque région sont présentées en annexe 4. Celles-ci ont été normalisées (où centrées-réduites, c'est à dire. soustraites par la moyenne puis divisées par l'écart type) pour pouvoir les comparer entre elles sans corrompre le résultat du fait de l'échelle par exemple. Les données de toutes les régions annotées ont ensuite été placées dans un même fichier servant à l'apprentissage par l'algorithme CART.

Le logiciel Weka nous permet facilement d'importer nos données en format CSV[29] et de lancer la création de l'arbre. En plus de l'arbre de décision, Weka nous fournit les résultats de la validation croisée[30] (cross validation) nous permettant d'apprécier la fiabilité des résultats. Un exemple d'utilisation de Weka est disponible en annexe 5 ainsi qu'une sortie des résultats en annexe 6.

L'idée est de réaliser un premier arbre de décision avec une moitié du jeu de données (en éliminant la population «indéterminé »), et de l'appliquer à l'autre moitié pour vérifier sa validité. On renforce l'approche en prenant ensuite la deuxième moitié du jeu pour réaliser l'arbre et la première moitié comme jeu test. C'est en combinant ces étapes préliminaires que l'on peut faire un jeu d'apprentissage optimal.

Un tableau récapitulatif des résultats obtenus avec la construction de plusieurs jeux d'apprentissage, de plusieurs arbres de décision, est présenté ci-dessous (figure 29). Pour chaque arbre réalisé on a précisé les images comprises dans son apprentissage, les pourcentages de régions correctement classées pas l'arbre sur les images étalonnées, ainsi que les pourcentages de faux positif/faux négatifs.

[29] Comma-separated values (CSV) est un format informatique ouvert représentant des données tabulaires sous forme de « valeurs séparées par des virgules ».Ce format n'a jamais vraiment fait l'objet d'une spécification formelle. Toutefois, la RFC 4180 décrit la forme la plus courante et établit son type MIME "text/csv", enregistré auprès de l'IANA.

[30] Il s'agit d'une technique permettant d'évaluer comment les résultats d'une analyse statistique vont se généraliser sur un jeu de donnée indépendant. Il est surtout utilisé quand l'on fait de la prédiction, et que l'on veut estimer exactement comment un modèle prédictif va opérer en pratique. Une partie de la méthode implique de partitionner d'un jeu de donnée en sous ensembles complémentaires, de réaliser l'analyse d'un sous ensemble (entrainement), et de valider l'analyse sur l'autre sous ensemble (validation). Dans le but de réduire la variabilité des résultats, plusieurs passes de validation croisée sont réalisées sur des partitions différentes et le résultat de la validation est moyenné au fur et a mesure des passes.

Espèce	Image	etalonnée	CART1	% réussite	% faux positif	% faux negatif	CART2	% réussite	% faux positif	% faux negatif	CART3	% réussite	% faux positif	% faux negatif	CART5	% réussite	% faux positif	% faux negatif	CART6	% réussite	% faux positif	% faux negatif
Chamaecypavis_obtusa	Chamaecypavis_obtusa_x4_03	oui	+	**97%**	22%	78%		**95%**	35%	65%	+	**98%**	66%	34%	+	**98%**	50%	50%		**97%**	27%	73%
	Chamaecypavis_obtusa_x10_01	oui	+	**98%**	40%	60%		**94%**	0%	100%	+	**99%**	0%	100%		**96%**	70%	30,00%	+	**99%**	0%	100%
Chamaecypavis_thyoides	Chamaecypavis_thyoides_x4_03	oui	+	**96%**	13%	87%		**41%**	0%	100%	+	**98%**	85%	15%	+	**98%**	83%	17,00%		**94%**	41%	59%
	Chamaecypavis_thyoides_x10_01	oui	+	**94%**	18%	82%		**82%**	0%	100%	+	**97%**	37%	63%		**91%**	14%	86,00%	+	**95%**	23%	77%
Pinus_banksiana	Pinus_banksiana_x4_03	oui	+	**95%**	4%	96%		**32%**	1%	99%	+	**98%**	25%	75%	+	**96%**	25%	75,00%		**92%**	38%	62%
	Pinus_banksiana_x10_02	oui	+	**98%**	80%	20%		**97%**	83%	17%	+	**97%**	85%	15%		**96%**	83%	17,00%	+	**98%**	80%	20%
Pinus_Khasya	Pinus_Khasya_x4_03	oui	+	**94%**	55%	45%	+	**77%**	3%	97%		**87%**	100%	0%	+	**96%**	100%	0,00%		**89%**	41%	59%
	Pinus_Khasya_x10_02	oui	+	**99%**	0%	100%	+	**97%**	20%	80%		**97%**	100%	0%		**98%**	100%	0,00%	+	**99%**	100%	0%
Pinus_longifolia	Pinus_longifolia_x4_03	oui	+	**96%**	50%	50%	+	**94%**	52%	48%		**92%**	96%	4%	+	**95%**	81%	19,00%		**95%**	33%	67%
	Pinus_longifolia_x10_01	oui	+	**97%**	33%	67%	+	**95%**	54%	46%		**92%**	84%	16%		**93%**	63%	37,00%	+	**98%**	0%	100%
Pinus_monticola	Pinus_monticola_x4_01	oui	+	**97%**	50%	50%		**47%**	5%	95%		**96%**	70%	30%	+	**96%**	75%	25,00%		**95%**	60%	40%
	Pinus_monticola_x10_01	oui	+	**90%**	90%	10%		**87%**	87%	13%		**76%**	97%	3%		**82%**	96%	4,00%	+	**90%**	83%	17%
Pinus_radiata	Pinus_radiata_x4_03	oui	+	**96%**	37%	63%	+	**97%**	77%	23%		**95%**	90%	10%	+	**98%**	71%	29,00%		**94%**	39%	61%
	Pinus_radiata_x10_01	oui	+	**95%**	85%	15%	+	**95%**	66%	34%		**90%**	100%	0%		**93%**	95%	5,00%	+	**97%**	66%	34%
Images ayant servi à l'etude de la segmentation																						
Pin noir	Pinoir14	oui		**93%**	13%	87%		**92%**	4%	96%		**94%**	71%	29%		**94%**	51%	49,00%		**90%**	18%	82%
Pinus caribensis	Pinus_caribensis02	oui		**88%**	100%	0%		**84%**	91%	9%		**77%**	98%	2%		**82%**	97%	3,00%		**87%**	93%	7%

Figure 29 : *Comparaison d'arbres de décision construits par la méthode CART.*

L'arbre **cart1** contient toutes les données du jeu d'apprentissage :

Les résultats sont bons dans tout les cas (en accord avec la cross-validation =~ 95%).

L'arbre **cart2** est réalisé avec 6 images de 3 espèces différentes :

L'arbre semble marcher sur les autres images hormis 3 images qui descendent en dessous des 50% de réussite.

L'arbre **cart3** est réalisé avec 6 autres images de 3 autres espèces :

L'arbre marche mieux dans ce sens, la totalité des autres images sont bien gérées.

L'arbre **cart5** est réalisé avec toutes les images au même niveau de zoom *4 et l'arbre **cart6** avec toutes les images au niveau *10 :

Ces tests mettent en évidence que l'apprentissage sur un niveau de zoom n'influe pas sur la classification d'une image acquise à un autre niveau (ou quasiment pas).

On a précisé dans chaque résultat de classification le pourcentage de faux positif et de faux négatif quant aux régions mal classées. Il ne semble pas que l'on puisse repérer une tendance. Par contre, il est peut être intéressant qu'il y en ait une comme nous le discuterons dans le chapitre III.2.1.

→ Les cas les plus réalistes sont donc les résultats obtenus avec cart3 et cart4, ainsi que les résultats obtenus sur les deux dernières images qui ne font parti d'aucun jeu d'apprentissage (Pinoir14 et Pinus_caribensis02). La méthode semble satisfaisante même s'il faudra choisir astucieusement les images du jeu d'apprentissage pour couvrir un maximum de variations possibles : espèces, échelle, éléments structurants, qualité, résolution… afin d'éviter un sous apprentissage et/ou un sur apprentissage.

Enfin, il faut noter que le travail de segmentation préalable influe beaucoup sur la qualité de la classification des régions : une sur-segmentation entraînera beaucoup de petites régions homogènes correctement classées comme non cellules, mais aussi beaucoup de petites régions ne représentant qu'un morceau de cellule qui ne sera pas forcément classé comme cellule.
Il est à noter qu'une simplification de l'arbre définitif pourrait être réalisée manuellement, en observant les autres arbres et en réalisant une sorte d'arbre consensus. Celui réalisé dans notre cas sera modifié manuellement pour ajuster quelques valeurs. L'Arbre de décision obtenue est présenté ci-dessous.

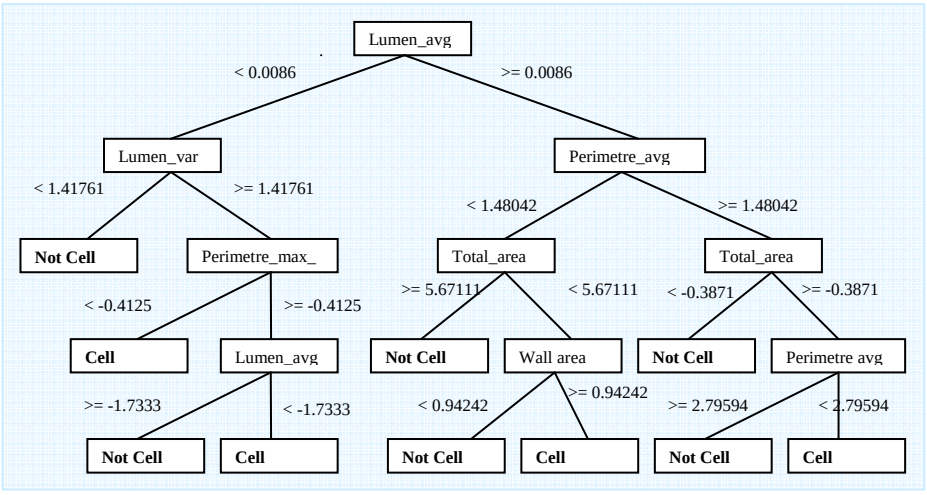


Figure 30 : *Arbre de décision obtenu de l'algorithme CART.*

Ici une coloration des régions classifiées sur une image résultat :

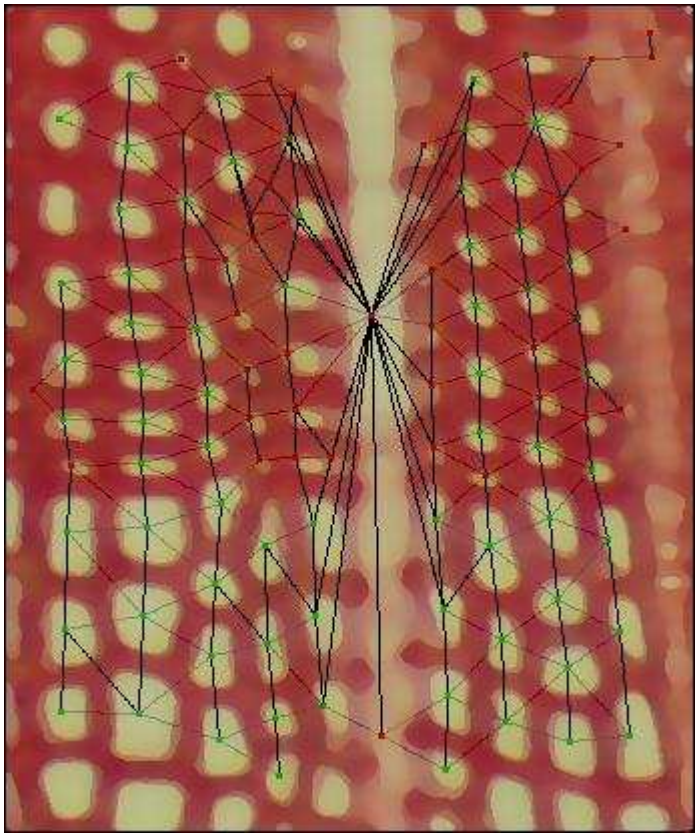


Figure 31 : *Résultat de la classification, en vert les « cellules » en rouges les « non cellules » (présenté sur la superposition du graphe à l'image prétraitée).*

II.3.3. Reconnaissance des files

Nous expliquerons dans ce chapitre comment nous avons identifié les files cellulaires suite aux traitements que l'image a déjà subit. Nous utiliserons largement le graphe réalisé dans le chapitre II.2.1.2.2, car il nous suffit, dans l'absolu, de le parcourir « correctement » pour obtenir nos files. Nous utiliserons aussi la caractérisation des régions pour : d'une part, amorcer l'identification d'une file et d'autre part, caractériser la fiabilité d'une file reconnue.

La première question qu'il faut se poser avant même de tracer les files est : dans quelle direction sont elles alignées ? Autrement dit, savoir comment est orientée l'image.

II.3.3.1. Calcul de la direction principale des files

Pour réaliser cette tâche, il faut s'aider du graphe et plus particulièrement des arêtes du graphe. D'après l'article de Ronald J. [5] dont nous nous sommes largement inspiré, il suffit d'observer la distribution des angles que forme chaque arête par rapport à un axe fixé, et d'en extraire le pic présent. En effet, le graphe d'adjacence, vu la topologie de l'arrangement cellulaire du bois, porte une majorité d'arêtes qui suivent la direction des files.

Nous avons donc dans un premier temps calculé l'angle qui existe entre chaque arête et l'axe des abscisses afin d'obtenir une distribution angulaire.

Les arêtes obtenues sont comprises entre -89° et 90°. Ainsi taguées, on peut observer l'histogramme de distribution des angles de l'ensemble des arêtes (un angle de -90° est assigné à 90° pour ne pas disperser un éventuel pic).

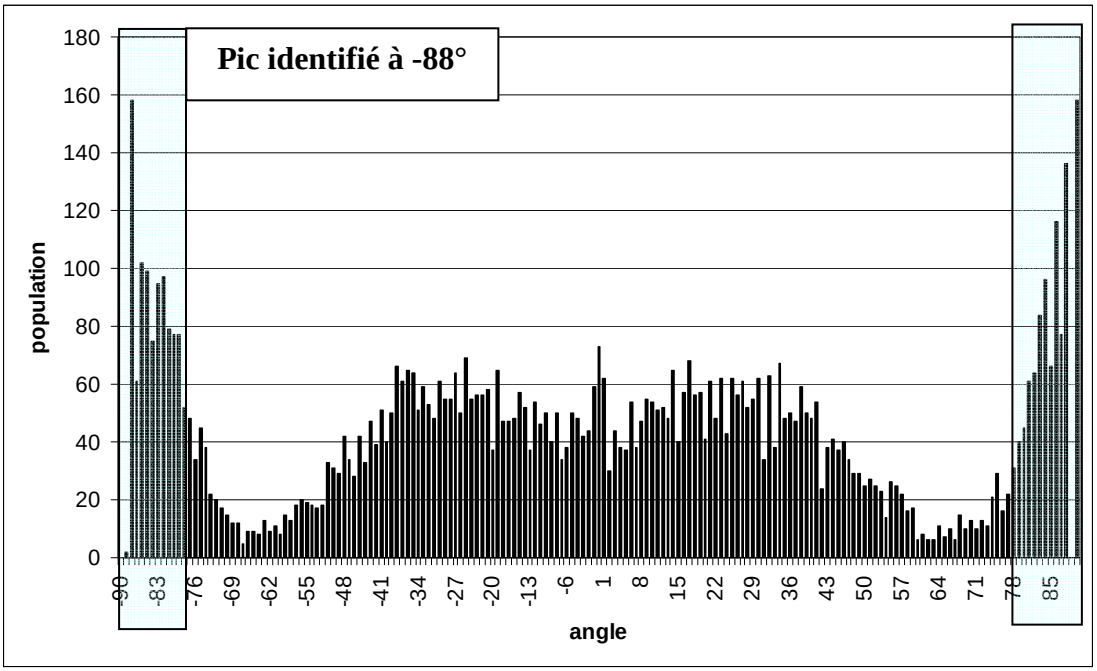


Figure 32 : *Distribution des angles formés par les arêtes avec l'axe des abscisses, sur Pin Noir.*

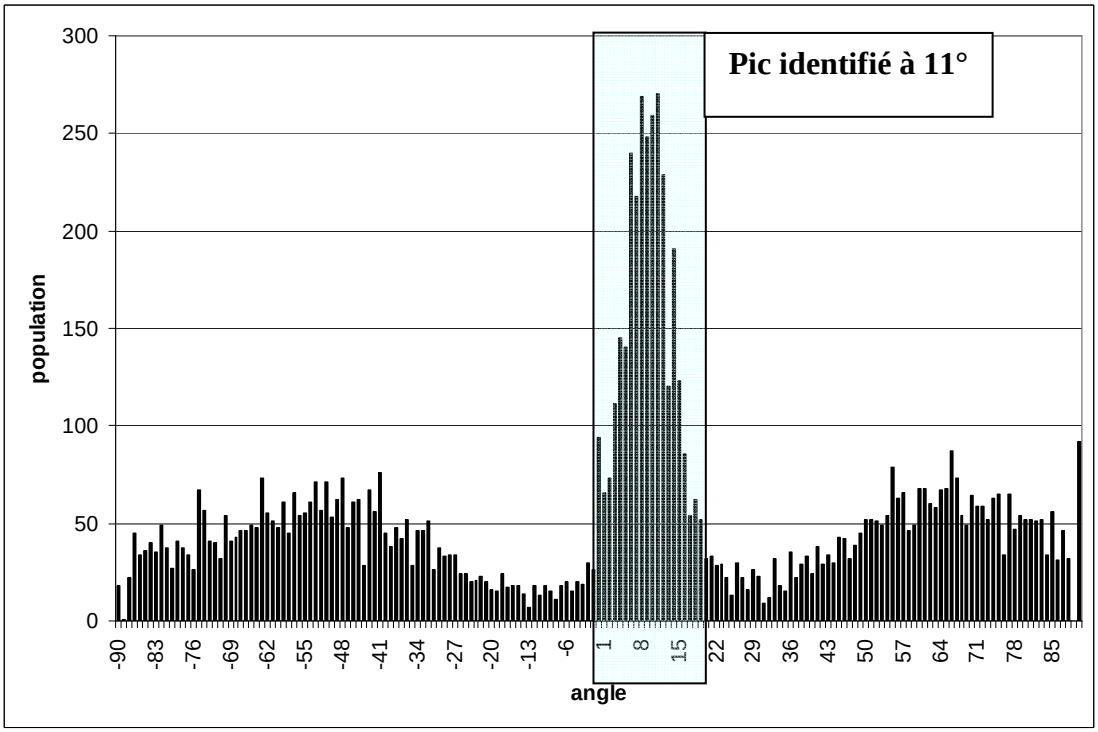


Figure 33 : *Distribution des angles formés par les arêtes avec l'axe des abscisses, sur Pinus Khasya 03.*

De ces histogrammes, il nous suffit d'extraire le pic maximum pour connaître l'orientation des files. Nous pouvons vérifier que cette direction est en accord avec l'estimation qu'en fait l'œil sur les images originales correspondantes, ou encore en faisant pivoter volontairement l'image de *n* vérifiant le décalage du pic sur l'histogramme. Encore une fois l'extraction de cette valeur peut être critiquable : une approximation de l'histogramme par une gaussienne devrait améliorer cette étape, nous en discuterons dans le chapitre III.3.1.

Cette information va nous servir dans un premier temps à réduire le nombre d'arêtes du graphe utilisables pour le parcours des files. Du fait de la justesse de la direction extraite (et de la variabilité de l'image), nous prendrons une différence maximale de 40° entre l'angle de chaque arête et la direction. Cette marge est complètement arbitraire et semble marcher pour la poursuite de nos traitements. Cependant cette valeur pourrait être égale au sigma de la gaussienne énoncée plus haut (les valeurs hypothétiques à +/- sigma sont encadrées sur les figures 30 et 31). Les arêtes sont au final taguées « orientée » ou « non orientée ».

Voici les résultats de la réduction d'arêtes (arête « orientée » en noir, et « non orientée » en gris) :

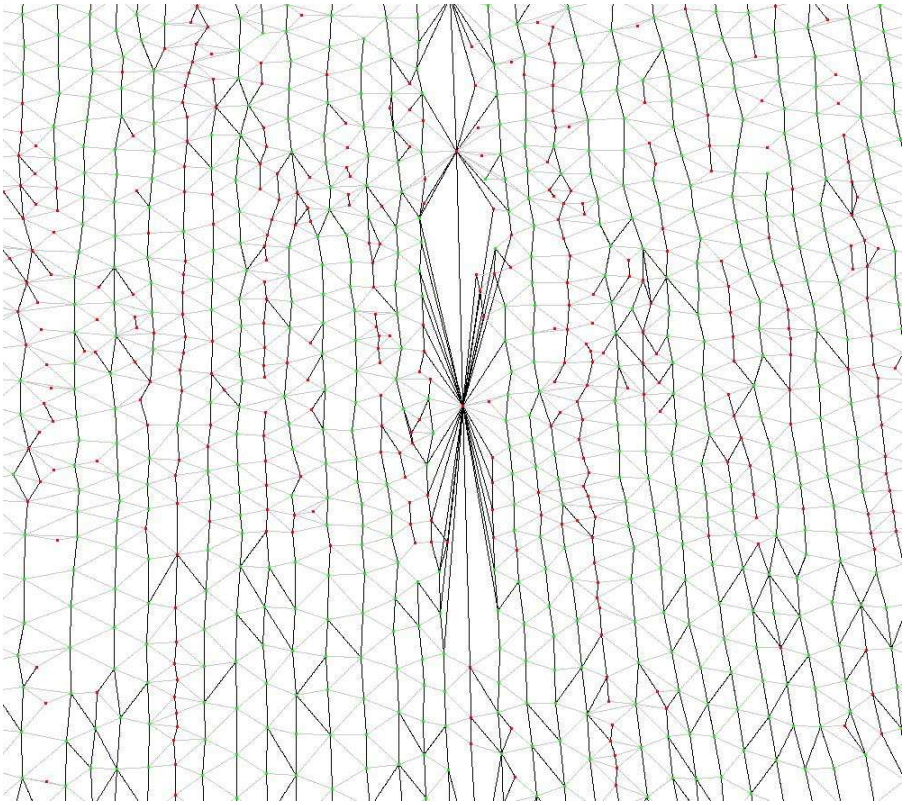


Figure 34 : *Identification des arêtes « orientée ».*

II.3.3.2. Parcours des files

Le parcours du graphe nécessaire à l'indentification des files à été fait en deux temps. Un premier passage à été fait pour identifier les tronçons « faciles » à suivre et dont la fiabilité est bonne. Ensuite un deuxième passage a permis le raccord des tronçons susceptibles d'appartenir à la même file.

II.3.3.2.1. Identification de tronçons fiables

Nous avons implémenté un algorithme récursif parcourant le graphe selon certaines conditions relativement strictes dans l'optique de n'obtenir que des files partielles dont le parcours est d'une grande fiabilité.

Voici comment nous avons procédé :

- Récupérer la liste des bassins tagués « cellule » lors de la classification par la méthode CART.
- Pour chacun de ces bassins, s'ils n'appartiennent pas déjà à une file :
  - Créer une nouvelle file.
  - Ajouter cette file à la liste de file.
  - Lancer le parcours récursif du graphe sur le bassin courant dans le sens de la direction principale identifiée.
  - Relancer le parcours récursif du graphe sur ce bassin dans le sens opposé de la direction principale.

La fonction récursive de parcours du graphe fonctionne de la façon suivante :

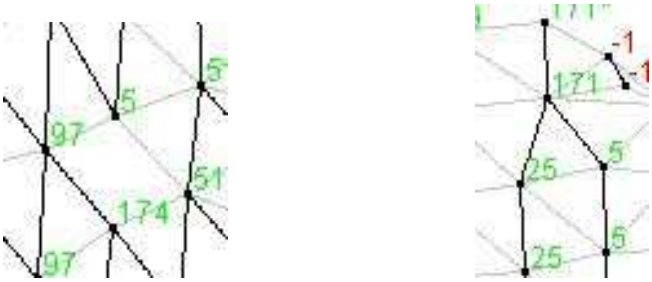
- Ajouter la cellule à la file courante traitée.
- Récupèrer les voisins de la cellule qui sont reliés par une arête orientée et n'appartenant à aucune file.
- Pour chaque voisin :
  - Récupérer l'arête reliant la cellule au voisin traité
  - Calculer l'angle existant entre la direction majeure et l'arête.
- Sélectionner le voisin minimisant cet angle.
- Relancer la fonction sur ce voisin si l'angle formé par la nouvelle arête à ajouter dans la file et la dernière de la file est supérieur à 160° (un alignement parfait donnant un angle de 180°).

En réalité quelques autres conditions sont respectées pour gérer des cas particuliers, mais ne sont pas indispensables à la compréhension de ce parcours. A terme nous disposons d'une liste de files plus ou moins courtes et relativement droites. En fait, nous avons évité les cas perturbateurs où le parcours du graphe nécessiterait plus de réflexions. C'est le rôle de la seconde étape.

II.3.3.2.2. Raccord des tronçons

Nous avons isolé les difficultés grâce à l'étape précédente. Nous proposons maintenant une nouvelle approche quant à la finalisation des files. Pour ce faire, nous avons à répondre à un problème simple : à chaque extrémité des tronçons de file, quel autre tronçon est le meilleur candidat à un raccord s'il y a besoin ? Il nous faut donc réfléchir en termes de proximité directe de voisinage, cette fois à un ordre plus important dans le parcours du graphe, en termes de d'alignement entre les tronçons et en termes de nature de raccord à réaliser. Plusieurs cas de figure ont été identifiés et une solution fournie pour tenter de répondre au mieux à tous.

Voici deux cas différents identifiés, où un raccord est nécessaire, présentant quelques problèmes rencontrés :



Cas 1 Cas 2

Figure 35 : *Deux cas ou un raccord est nécessaire entre tronçons de files.*

Dans le cas 1, la file 174 doit être reliée avec la file 5. On remarque que l'arête reliant ces deux cellules n'existe pas.
Dans le cas 2, la file 25 doit être reliée avec la file 171. On remarque dans ce cas que deux candidats s'offrent au raccord de la file 171.

II.3.3.2.2.1. Calcul de l'alignement entre deux tronçons

Nous avons répondu à cette question par un procédé mathématique simple, la régression linéaire[31]. En effet si l'on calcule une droite de régression linéaire (dont nous expliquerons le calcul) sur chacun des tronçons en utilisant les centres des bassins les composant comme coordonnées, on obtient une droite représentative de l'orientation de chacun de ces tronçons. L'idée est, afin de pouvoir calculer les distances inter droites et ainsi déterminer l'alignement des tronçons (plus la distance est faible plus les tronçons sont alignés), de fixer leur orientation ; cette orientation sera donnée par la direction principale des files identifiées au chapitre II.3.3.1. Pour être plus précis, nous garderons la pente du vecteur de cette direction pour chacune des droites calculées : ainsi toutes les droites seront parallèles deux à deux et la distance entre celles-ci calculable

Soit le vecteur de la direction principale de coordonnées $\vec{v}\begin{pmatrix} a \\ b \end{pmatrix}$ ayant pour équation de droite :

$$-bx + ay + c = 0$$

Ainsi seul le coefficient c reste à déterminer pour chacune des droites : il représente le décalage vertical ou horizontal selon la valeur de la pente, c'est-à-dire l'endroit où la droite coupe les axes du repère.

Classiquement, la régression linéaire consiste à minimiser l'erreur d'un ensemble de point à une droite en utilisant une différence d'abscisses ou d'ordonnées ; ici, nous minimiserons la somme des distances des points du tronçon à cette droite.

La distance d d'un point $p(x_p, y_p)$ par rapport à la droite D d'équation : $ax + by + c = 0$ est donnée par :

[31] Méthode d'approximation de droites, courbes, sur un ensemble de point mesurés en minimisant les erreurs entre ces points et la droite/courbe.

$$d(p,(D)) = \frac{\left|ax_p + by_p + c\right|}{\sqrt{a^2 + b^2}}$$

On cherche donc à minimiser la somme des distances des points du tronçon à la droite. Minimiser une fonction consiste à annuler sa dérivée première, or la valeur absolue est une fonction non dérivable sur $\mathbb{R}$. La valeur absolue impose de raisonner sur $\mathbb{R}^-$ puis sur $\mathbb{R}^+$. Pour éviter ce problème, nous travaillerons avec le carré des distances ; à noter que l'on retrouve cette astuce dans la régression linéaire classique dans laquelle on cherche à minimiser la somme des erreurs élevées au carré, dans ce cas, l'élévation sert à éviter que les erreurs négatives annulent les erreurs positives.

Soit la fonction *f* à minimiser :

$$f = \sum_{i=1}^{i=n} \frac{(ax_i + by_i + c)^2}{a^2 + b^2}$$

Où :

- *n* est le nombre de point (de bassins) de la file.
- x_i et y_i les coordonnées des points (des centres de chaque bassins).
- *a* et *b* les coordonnées déduites du vecteur de la direction principale.

Minimiser *f* revient à trouver les zéros de sa dérivée ; *a* et *b* étant fixés (direction de la droite de régression), la seule variable à considérer est le paramètre *c*.
On doit trouver les conditions sur *c* qui annulent la dérivée partielle de *f* par rapport à la variable *c*, c'est à dire résoudre l'équation suivante :

$$\frac{a}{a^2 + b^2} \sum_{i=1}^{i=n} x_i + \frac{b}{a^2 + b^2} \sum_{i=1}^{i=n} y_i + nc = 0$$

Ainsi *c* est facilement calculable :

$$c = \frac{-\frac{a}{a^2 + b^2} \sum_{i=1}^{i=n} x_i - \frac{b}{a^2 + b^2} \sum_{i=1}^{i=n} y_i}{n}$$

Nous avons dans un premier temps calculé ces droites sur l'ensemble des points composant les files partielles, mais sommes vite tombés sur des cas où les files étant légèrement courbes, les droites s'éloignaient réellement trop des files. Les distances entre ces droites qui aident au choix des raccords dans la partie suivante, étaient donc totalement faussées. Nous nous sommes donc restreints à calculer deux droites de régression aux extrémités de chaque file sur quelques cellules seulement. Les résultats s'en sont vus nettement améliorés.

II.3.3.2.2 2. Algorithme de raccord

Voici comment nous avons procédé pour le raccord des tronçons :

- Trier la liste de tronçon par ordre décroissant en fonction de leur longueur (on cherche d'abord à maximiser la longueur des tronçons de grande taille).
- Créer une pile avec ces tronçons de type LIFO (Last In First Out).

Puis :

```
Tant que pile non vide

        change = faux
        file ← pile.first()

        // Tente de raccorder la tête
        candidat =  Fit(file,head)
        Si candidat != null
                candidant_retour = Fit(candidat)
                Si candidat_retour != null
                        Si file = candidat_retour
                                Fusion(file,candidat)
                                Change = vrai
                                pile.remove(file,candidat)
                        Fin
                Fin
        Fin

        // Tente de raccorder la queue
        [Idem… avec le paramètre queue]

        Si change
                pile.add(file)
        Fin

Fin
```

On remarque que lors de la recherche d'un tronçon candidat, on vérifie que celui-ci ait bien le tronçon de départ comme meilleur candidat. Cela nous permet d'éviter le *cas 2* illustré en II.3.3.2.2 :

Le tronçon traité est le 5 car il est le plus grand, il ne trouve que le tronçon 171 comme candidat, il se raccorde donc au 171. Alors qu'en réalité le raccord à réaliser est entre les tronçons 25 et 171. Dans le cas d'une retro vérification, le tronçon 171 ne trouvera pas le tronçon 5 comme meilleur candidat mais le 25, la fusion n'a alors pas lieu. Par contre, lors du traitement du tronçon 25, la fusion entre celui-ci et le 171 aura bien lieu.

Nous n'expliquerons pas la méthode Fit() citée dans le pseudo code car trop longue, mais citerons les critères de choix utilisés pour choisir le meilleur tronçon candidat si il y en a un.

Cette fonction cherche d'abord les files partielles au voisinage d'ordre 2 de l'extrémité de la file courante traitée dans le graphe (cela nous permet de traiter le *cas 1* illustré en II.3.3.2.2). On sélectionne ensuite celle étant la plus alignée possible (distance entre les droites de

régression concernées la plus petite possible). On caractérise ensuite le raccord à faire entre le tronçon de départ et le tronçon candidat :

- on vérifie que la longueur de ce raccord n'est pas aberrante
- on vérifie que ce raccord ne forme pas une trop grosse cassure de l'alignement (calcul d'angle entre le raccord avec le file de départ, et entre le raccord et la file candidate, angles devant être supérieurs à 130°)
- on indique si ce raccord est à créer ou si une arête existait déjà.

Enfin on réitère cet algorithme jusqu'à ce que le nombre de file se soit stabilisé (de nouvelles fusions apparaissant à chaque passe).

Ce procédé nous a permis d'obtenir des résultats satisfaisants, que nous allons détailler dans le chapitre III.1, bien que nous ne soyons pas encore en accord parfait avec l'identification d'un expert, que nous détaillerons dans la partie III.2.

III. Résultats

Après avoir décrit l'intégration de notre chaîne de traitement dans ImageJ, nous présenterons ici nos résultats sous un aspect qualitatif puis quantitatif. Nous prendrons surtout le temps de les discuter en comparaison avec des avis experts, ce qui nous amènera à remettre en cause certaines parties du traitement, et donc, d'ouvrir de nouvelles perspectives amélioratrices du projet Ficeler.

III.1. Intégration de la solution sous ImageJ.

La solution précédemment décrite a été implémentée et insérée sous forme de plugin dans ImageJ. L'API d'ImageJ propose facilement l'intégration de nouvelles fonctionnalités grâce à des interfaces prévues à cet effet. De même de nombreuses fonctionnalités sont utilisables quant à la manipulation d'image, d'opérateurs, de filtre… etc.

L'API d'ImageJ est construite autour des packages suivant :

- ***ij***
- ***ij.gui***
- ***ij.io***
- ***ij.macro***
- ***ij.measure***
- ***ij.plugin***
- ***ij.plugin.filter***
- ***ij.plugin.frame***
- ***ij.process***
- ***ij.text***
- ***ij.util***

La création d'un plugin se fait simplement en implémentant l'interface *ij.plugin.filter*. Cette interface implique l'implémentation d'une fonction *run* et d'une fonction *setup*. Ces fonctions sont utilisées au lancement du processus lié au plugin.

Voici le corps d'une classe « hello world » permettant la création de plugin :

```
import ij.*;
import ij.plugin.filter.PlugInFilter;
import ij.process.ImageProcessor;

/**
 * Basic class as plugin for imageJ
 */
public class Hello_ implements PlugInFilter {

	@Override
	public void run(ImageProcessor ip) {
		IJ.showMessage("Hello World en Java pour ImageJ !");
	}

	@Override
	public int setup(String arg, ImagePlus imp) {
		// TODO Auto-generated method stub
		return 0;
	}
}
```

On remarquera les imports nécessaires à ce plugin concernant les classes *ImageProcessor* et *ImagePlus* brièvement décrites ci-dessous :

La classe *ImageProcessor* est une classe abstraite du package ij.*process* permettant la manipulation d'image de type byte, short, float, et RGB. Le diagramme de classes de ce package est présenté en annexe 7. Les objets *ImageProcessor* et *ImageStack* contiennent les donnée actuelles des pixels des images, et piles d'images (dans l'ordre). Ils sont utilisés pour traiter et convertir l'image.

La classe *ImagePlus* est une classe du package *ij* portant toute les informations et données d'une image. Le diagramme de classe du package *ij* est présent en annexe 7. Elle étend les fonctionnalités de la classe java *java.awt.image* comme le montre la figure 34, permettant d'ouvrir, stocker et afficher une image (une pile d'image). Elle permet aussi son affichage à l'écran.

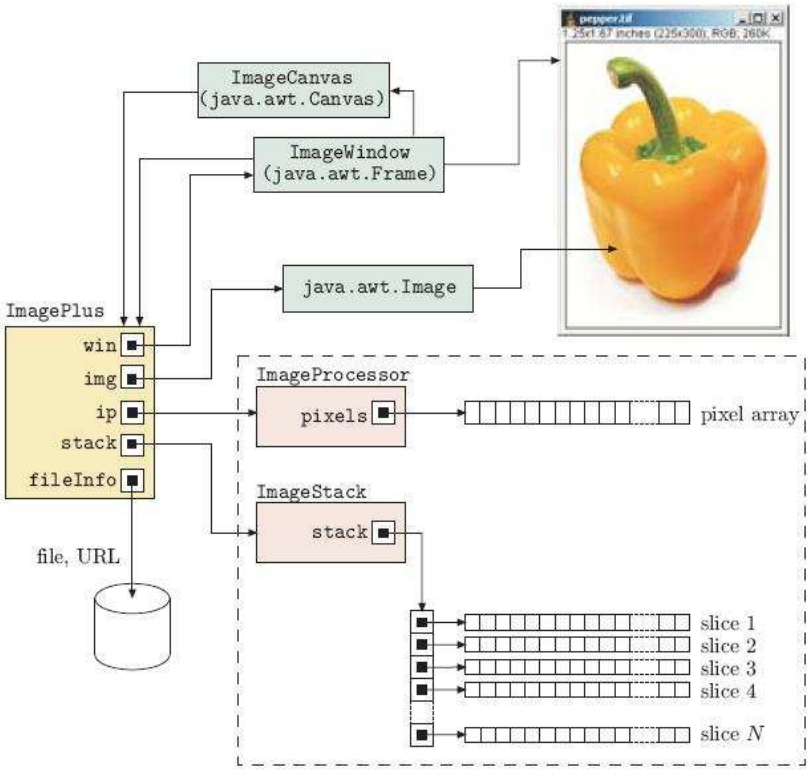


Figure 36 : *Représentation simplifiée d'une image (ou piles d'images) dans ImageJ.*

Le plugin *Ficeler* a donc est donc conçu sur cette base. La classe *FicelerRun_* implémente l'interface *ij.plugin.filter*. Nous pouvons voir les classes interagissant avec *FicelerRun_* dans le diagramme de dépendance de classes fournit en annexe 8.

III.2. Résultats obtenus

Le stage a permis de réaliser un plugin s'insérant dans l'API d'ImageJ. Il est donc d'ores et déjà utilisable par un utilisateur en allant simplement sur le menu *plugin* du logiciel et en choisissant *Ficeler* (annexe 9). Ces résultats sont sous la forme d'une sortie graphique présentant les files colorées sur l'image traitée, et d'une sortie texte regroupant les résultats numériques : une sortie tableur compatible est fournie comportant les caractéristiques des files.

III.2.1. Résultat qualitatif

La représentation des files permet d'une part de contrôler visuellement leur parcours, d'autres part de choisir les plus pertinentes à une étude statistique (cf. I.3).
Les raccords réalisés par l'algorithme de détection en chapitre II.2.3 sont signalés d'une couleur différente selon leur type (noir pour le rajout d'une arête, gris pour un raccord sans rajout). Ceci permet de qualifier la fiabilité d'une file, en effet une file ayant subit beaucoup de raccords a plus de chance d'être faussée comparée à une sans raccords. La seule autre information présente est l'identifiant de chaque file en accord avec les couleurs de celles-ci.

Les résultats obtenus sur deux images testes sont présentés ci-dessous :

Figure 37 : *Image résultat du plugin Ficeler sur Pin Noir.*

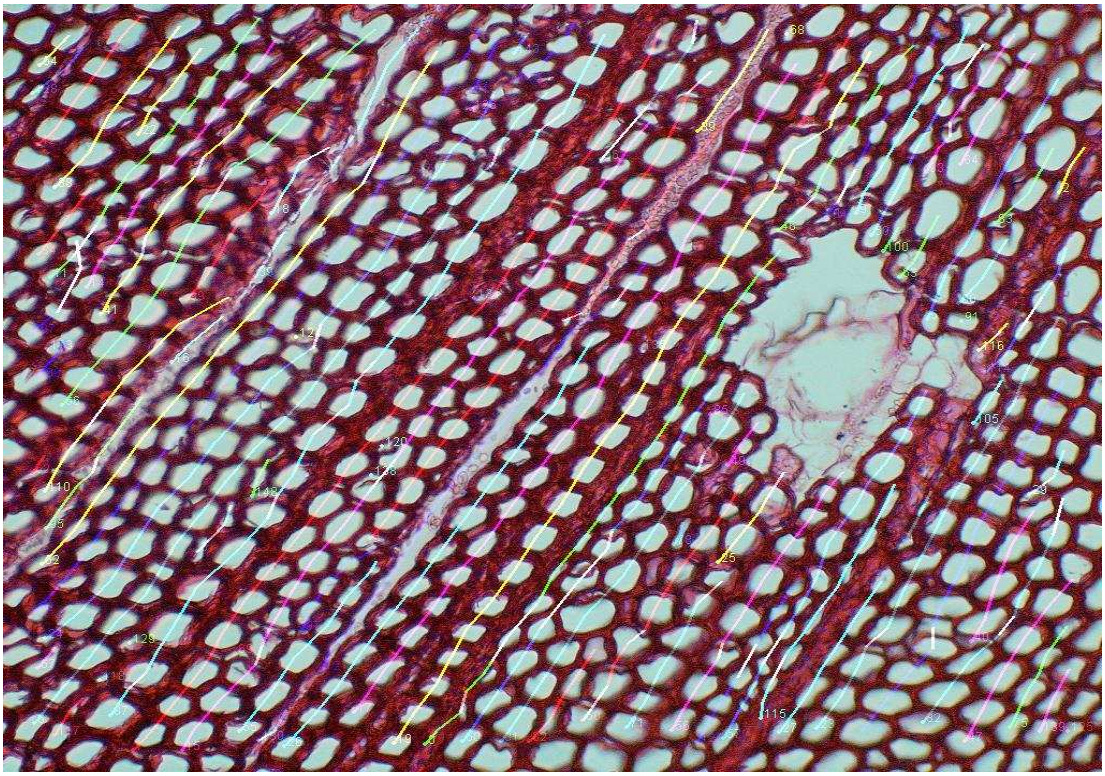


Figure 38 : *Image résultat du plugin Ficeler sur Pinus Caribensis.*

III.2.2. Résultat quantitatif

Au delà des données brutes des files, il a fallu quantifier la qualité d'une file. Nous nous somme basés sur deux critères principaux : la longueur et le type de bassins compris dans ces files.

- Indices de confiance, combinant la longueur et le typage de ses composants :

La longueur est un critère significatif car a priori les files cellulaires traversent toute l'image ou presque ; se basant sur cette information, on peut qualifier les files petites comme moins fiables : il s'agira d'ailleurs le plus souvent de tronçons parasites, de défaut de raccord… Cette longueur sera exprimée relativement à longueur de la plus grande file.

Quant au typage des bassins formant la file, il est directement tiré de la classification des régions par la méthode CART. On peut donc qualifier les composantes d'une file en pourcentage de régions « cellules » sur le nombre total de régions.

- Donnée brute :

Les données utiles permettant à l'expert de caractériser les files cellulaires ont été définies en accord avec leurs attentes et sont listées ci-dessous :

- surface du lumen
- ratio de surface lumen/paroi
- circularité du lumen

Avec les indices cités précédemment, l'expert pourra choisir les files jugées les plus fiables à inclure dans son étude statistique.

Un extrait de la sortie au format CSV est présenté en annexe 10.

- Temps d'exécution :

L'expert participant à la réalisation du projet a passé six heures à l'annotation de trois images, soit deux heures de traitement manuel par image. Le plugin Ficeler passe en moyenne deux minutes par image (variable en fonction de la résolution de l'image). Se basant sur le fait qu'une erreur commise par l'algorithme est corrigible en une minute par l'opérateur, on peut considérer que le plugin reste plus rapide qu'une annotation manuelle en dessous de 118 erreurs par image.

- Réussite de l'identification

Il est difficile d'évaluer la quantité de files détectées par notre traitement en accord avec l'identification experte. Comme nous l'avons déjà cité, l'algorithme n'est pas finalisé et beaucoup de parcelles de files sont détectées mais sont, soit mal raccordées, soit non raccordées. Une estimation grossière du pourcentage de réussite du plugin à ce stade tourne autour des 30 à 40%, ce qui reste largement améliorable.

III.3. Discussion

Nous discuterons ici des résultats obtenus en les comparant à des images traitées manuellement par un expert. Nous aurons sur chaque point difficile rencontré un avis expert en architecture des plantes et un avis technique du développeur dans le but d'identifier les causes biologiques ou techniques des problèmes.

Beaucoup d'erreurs ont été faites par le plugin au regard des images expertes. Nous ne reviendrons pas sur toutes ces erreurs mais en exposerons les plus intéressantes.

Notamment dans certains cas de figures assez récurrents que sont le choix de la bonne file en cas de bifurcation, de discontinuité, de raccord manquant…

- **Fourche :**

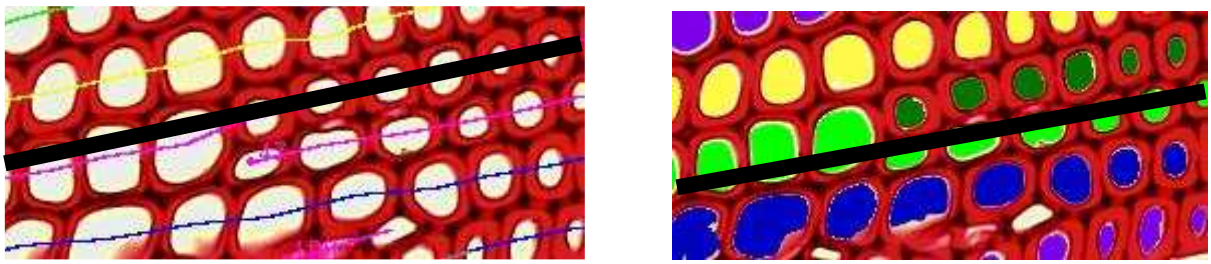


Figure 39 : *Tracé du plugin à gauche, de l'expert à droite.*

Notre algorithme choisit son chemin principalement en se basant sur des directions à respecter et des distances inter files à minimiser. L'expert considère que la branche à choisir est celle comportant les plus grosses cellules. Ce point reste donc à intégrer au suivi des files.

- **Discontinuité de files :**

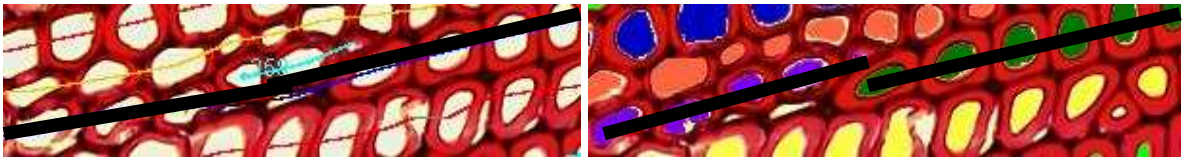


Figure 40 : *Tracé du plugin à gauche, de l'expert à droite.*

Notre algorithme ne voit qu'une continuité dans une file alors que l'expert voit deux files s'arrêtant à cet endroit car il présume qu'elles n'appartiennent pas au même plan de coupe. Ce cas semble difficile à corriger car peu d'indices semblent indiquer ce cas de figure ; une étude plus poussée du contexte peut peut-être aider à résoudre ce problème.

D'autre cas remettant uniquement le coté technique en question sont présentés. Un certain manque de temps pour finaliser les parcours des files en est la cause, mais nous pouvons déjà citer quelques problèmes « classiques » et la solution envisagée.

- **Raccord de tronçons manquants :**

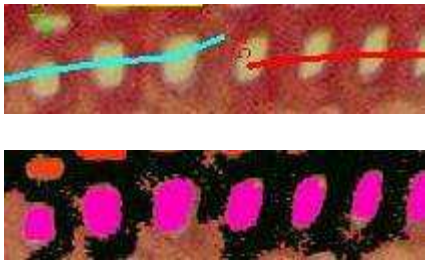


Figure 41 : *Tracé du plugin en haut, de l'expert en bas.*

Ici nous sommes aux limites des conditions d'angles imposées par l'algorithme lors de la création de raccord. Ces angles sont limités pour éviter que la file puisse emprunter des chemins qui changeraient brusquement de direction. Peut être peut-on remédier à ce problème en forçant l'arrêt de la file bleue dans un bassin identifié comme « cellule » ce qui ne semble pas le cas ici (extrémité de la file en pleine paroi), la déviation observée semblerait ainsi évitée et le raccord possible.

- **Zone non reconnue dans les files :**

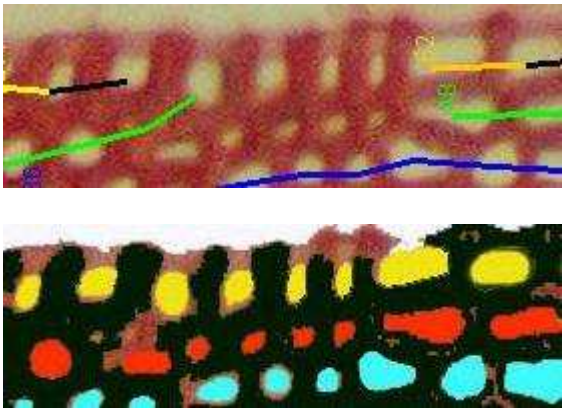


Figure 42 : *Tracé du plugin en haut, de l'expert en bas.*

Ici un sérieux problème de reconnaissance est présent dans le parcours de notre algorithme. Il semblerait que cet amas de cellules ne soit pas correctement reconnu comme « cellules » à l'étape de classification et que le parcours des files lors du premier passage n'a pas pu détecter de tronçons. On pourrait remédier à ce problème en faisait partir la détection des tronçons aussi bien sur des régions « cellules » que « non cellules ».

Alentours de canaux non gérés :

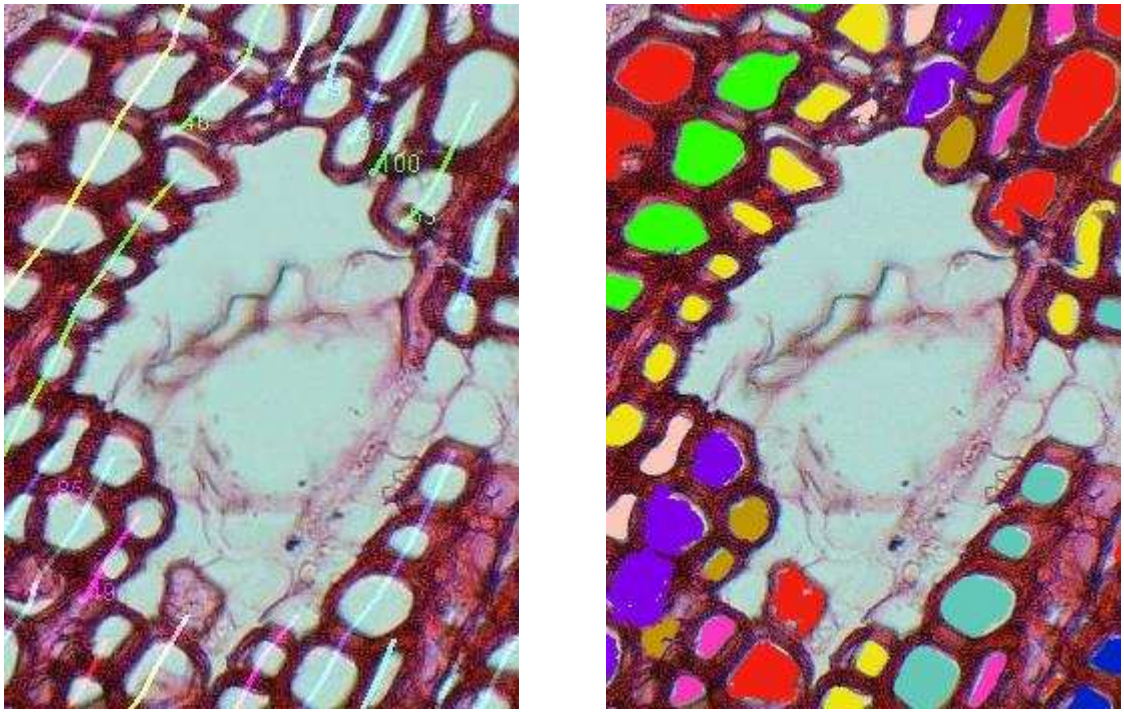


Figure 43 : *Tracé du plugin à gauche, de l'expert à droite.*

Contrairement à l'expert, notre traitement ne gère pas l'arrêt, puis la reprise de la file lorsqu'un canal résinifère est présent.

Malgré beaucoup de mauvaises détections de files, une partie des files extraites est tout de même d'ores et déjà exploitable. Et les indices de confiance établis permettent sans aucun doute de retrouver facilement ces files.

Illustration de files correctement reconnues :

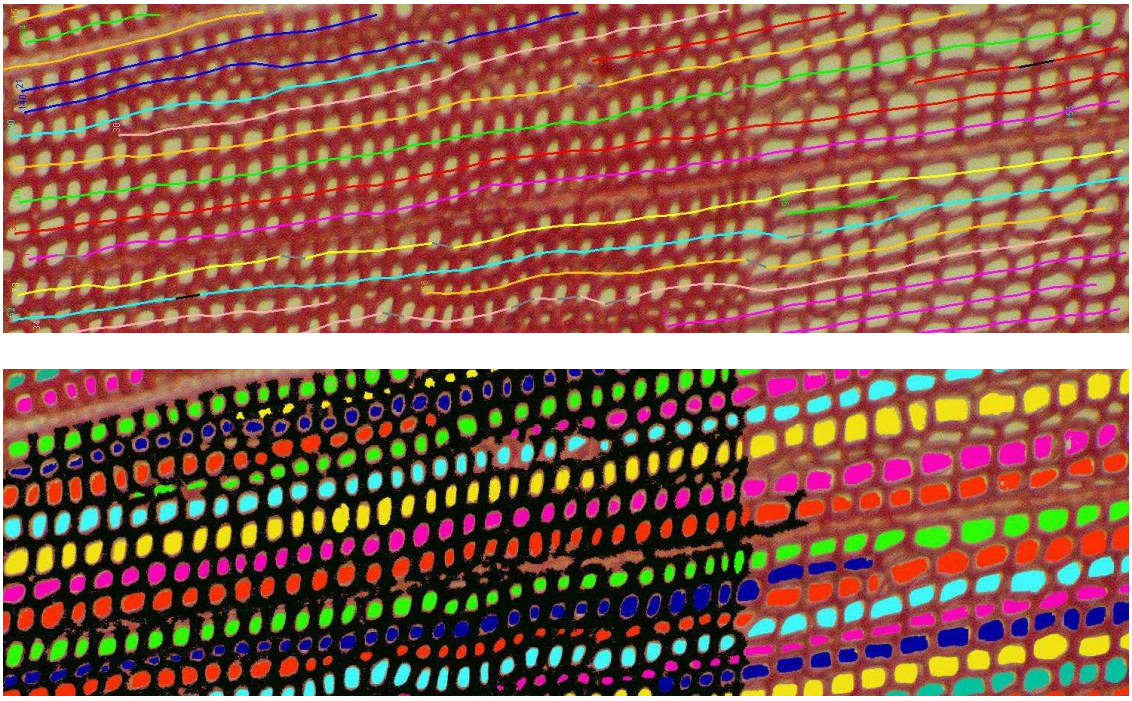


Figure 44 : *Tracé du plugin en haut, de l'expert en bas.*

III.4. Perspectives

Nous pouvons distinguer des perspectives à court terme comme à long terme. En effet bons nombres de points, plus ou moins techniques, ont été soulevés durant toute l'explication des « matériels et méthodes » qui peuvent être mis au clair dans un premier temps ; mais d'autres points sont à aborder, telle que la réalisation d'une IHM[32] ou la formalisation du protocole d'acquisition. Dans un second temps, nous verrons aussi que la chaîne de traitement est vouée à évoluer quant à sa portée applicative, à d'autres espèces et à d'autres types d'études.

III.4.1. Perspectives à court terme.

III.4.1.1. Finalisation du plugin

Le plugin mis en place est fonctionnel mais ne répond pas encore complètement aux attentes des biologistes. Dans sa version finale le plugin doit permettre :

- D'identifier et localiser les difficultés afin que l'expert ne perde pas de temps sur des zones sans problèmes.
- D'introduire facilement une connaissance ou un avis expert : requalifier un bassin, modifier une file ou une connexion…

[32] Interface Homme Machine.

- De choisir les caractéristiques à évaluer.

Toutes ses fonctionnalités font ressortir la nécessité d'étudier et d'implémenter une interface homme machine spécialisée quant à la visualisation et au traitement des files cellulaires extraites. Ceci n'a pu être fait dans les temps impartis mais est nécessaire à une réelle utilisation du plugin.

III.4.1.2. Un point sur les paramètres

La plus grande difficulté rencontrée lors de l'élaboration de cette chaîne de traitement automatique est peut être le paramétrage. En effet, bons nombres de filtres et algorithmes nécessitent bien souvent d'être paramétrés, comme le flou gaussien par exemple où l'on définit la taille du masque. La réalisation d'une méthode automatique signifie qu'il ne faudrait dans l'idéal aucune intervention de l'utilisateur lors du traitement, ce qui implique que les paramètres soient définis automatiquement, et plus exactement fixés (en pouvant justifier le choix réalisé), soit calculés dynamiquement.

Nous pouvons citer les paramètres dont nous n'avons pas trouvé de règles triviales de calibrage :

- Filtre Mean Shift : taille du masque 2D, et voisinage RGB 3D, dont nous avons déjà discuté lors de l'explication du filtre.
- Flou gaussien : taille du masque qui influe sur la « quantité » de flou appliquée. Ce flou nous servant à détourner les détails de l'image pour une meilleure segmentation, il dépend évidemment de la taille et de la résolution de l'image ; plus l'image est zoomée et/ou de grande résolution, plus on cherchera à atténuer les détails.

→ Ces paramètres sont étroitement liés à la variabilité des dimensions et échelle de l'image. Une des perspectives les plus probantes serait d'identifier pour chaque image l'échelle de prise de vue et ainsi caractériser la taille réelle d'un pixel (cette information pourrait être indiquée directement sur l'image grâce à une échelle placée par la personne prenant le cliché, ou calculée sur d'autres critères morphologiques de l'image). En combinant cette échelle avec la résolution de l'image (facilement accessible), nous pourrions sûrement trouver une façon de d'auto paramétrer ces filtres.

Nous avons, d'autre part, tenté d'extraire des histogrammes les seuils nécessaires à la réduction du graphe (chapitre II.1.2.2), ainsi que la valeur permettant d'orienter l'image (chapitre II.2.3.1). Il a fallu, dans ces deux cas, extraire des maximums, ce que nous avons simplement fait dans un cas par une détection de pics puis par une sélection des deux premiers, et dans l'autre cas en prenant la valeur la plus haute arbitrairement.

→ Après réflexion, nous aurions pu, dans un souci de rigueur mathématique et de confiance dans les valeurs extraites, approximer ces histogrammes par des gaussiennes. En effet les distributions observées semblent tout à fait de cette forme. Dans ce cas il aurait fallu appliquer une régression linéaire sur une gaussienne aux valeurs observées. Nous aurions ainsi pu extraire plus d'informations utiles dans nos utilisations de maximums car en plus d'une valeur moyenne nous aurions aussi accès au sigma de la courbe, représentant l'aplatissement de celle-ci. Dans le calcul des seuils de fusion, les sigmas nous auraient été utiles pour

déterminer avec précision les valeurs seuils à utiliser (seuils = valeur moyenne +/- sigma). Dans le cas d'extraction du pic de l'orientation, cela aurait permis de déterminer un angle moyen mais aussi de déterminer un pas (à +/- sigma) pouvant nous donner le rang d'arêtes à conserver lors de la réduction d'arêtes.

III.4.1.3. Un point sur la sur-segmentation

Nous avons beaucoup évoqué le problème de sur-segmentation lors de l'application de l'algorithme de Ligne de Partage des Eaux. Ce problème est en effet un des premiers éléments à corriger au mieux car il influe énormément sur les étapes de classification des régions et de reconnaissance de files.
Il s'agit en fait d'un problème classique dans la littérature liée à l'algorithme même. Nous pouvons aborder ce sujet de deux façons, soit améliorer le prétraitement soit améliorer le post traitement. Nous avons pour l'instant travaillé sur les deux aspects, mais ceux-ci peuvent être largement améliorés.

→ D'une part en améliorant l'application de l'algorithme LPE. La sur-segmentation est liée au surplus de minima locaux détectés sur l'image dû aux variations infimes du relief, entraînant de multiples sources d'inondation et donc de bassins versant par l'algorithme. Le problème est classiquement atténué en utilisant des marqueurs comme sources d'inondation plutôt que des minima locaux. Ces marqueurs peuvent souvent être présents sur les objets à segmenter et donc détectables par une analyse morphologique de l'image : utilisation d'une carte des distances 2D qui à chaque pixel de cette carte associe une valeur en fonction de sa distance au bord d'un zone préalablement seuillée. L'algorithme peut être appliqué à cette carte plutôt qu'à l'image originale, ou à l'image gradient ou sur toute autre fonction d'énergie étant définie de façon à avoir une valeur élevée sur les transitions dans l'image. Par exemple les variations d'intensité ou de textures localisées se traduisent par des maxima du gradient (ou de la fonction d'énergie), des crêtes en terme de relief. La valeur de la fonction d'énergie est faible dans les parties « homogènes » de l'image. Ces parties correspondent à des minima locaux.

→ D'autre part, une amélioration de la réduction du graphe d'adjacence peut être faite, d'abord en améliorant les critères de fusion actuels, mais aussi en trouvant un ensemble de critères densitométriques ou topologiques propre à chaque type de structures présent dans l'image. L'idéal serait de fusionner les bassins spécifiques aux vaisseaux, canaux ensembles, puis ceux des rayons, et enfin extraire les bassins des trachéides. Cette façon de procéder, au delà de mieux segmenter les cellules, rendrait accessibles au biologiste des informations potentiellement intéressantes sur tout autre élément structurant de l'image.

III.4.1.4. Un point sur la classification

Cette étape importante dans l'identification de la nature de chacune des régions peut être largement améliorée par une amélioration de la segmentation vue précédemment. Une segmentation plus juste, par type structurant, apporterait des critères de caractérisation plus déterminants.

D'un point de vue local, les résultats de l'algorithme Classification And Regression Tree (CART) pourraient être améliorés aussi par l'apport de nouvelles variables décrivant les régions sous un aspect plus géométrique (ce qui n'a pas été fait) en choisissant rigoureusement des facteurs de forme (circularité, diamètre de Feret, superposition d'ellipse… etc.), voir sous d'autres aspects (topologiques ?).
D'autre part nous avons évoqué le point des faux positifs et faux négatifs dans le chapitre II.3.2.1.2. Ceux-ci ne nous ont pas apporté d'informations utiles, mais il serait peut être intéressant de se pencher dessus de telle sorte qu'il y ait un minimum de faux positifs pour mieux démarrer la reconnaissance de file (car on se base sur les régions caractérisées cellules pour amorcer la reconnaissance). On pourrait forcer la méthode CART à minimiser les faux positifs en imposant une matrice de coûts favorisant les faux négatifs plutôt que les faux positifs.
Enfin une amélioration dans la continuité du cas idéal selon lequel toute structure de l'image est correctement segmentée : on pourrait classifier nos régions en autant de types d'éléments structurels qu'il en existe. Ceci sort du sujet initial mais offrirait de belles perspectives d'études aux biologistes.

Plus généralement la méthode CART en elle-même peut être remise en cause du fait de ses défauts énoncés dans le chapitre lui étant consacré II.3.2.1. Réside toujours derrière cette méthode le risque d'un sur-apprentissage et d'un sous-apprentissage. Le choix du jeu d'apprentissage est donc une étape sensible qui n'obéit a priori à aucunes règles. Quant à choisir une autre méthode de classification le débat est largement ouvert, mais justifier un choix particulier est hasardeux sans être spécialiste de ce domaine.

III.4.1.5. Un point sur le protocole d'acquisition

Les débuts du stage ont été largement perturbés par l'aspect préparatoire. Différents protocoles ont été énoncés (ponçage sans coloration, et vice-versa), donnant des images bien différentes sur plusieurs points cités au chapitre II.2. Au delà de la restriction que nous nous sommes donnée en ne traitant qu'un seul de ces protocoles (avec coloration), il serait intéressant d'optimiser, formaliser, ce protocole dans l'espoir de tirer de meilleurs traitements. Par exemple le fait de réaliser la prise de vue en alignant au mieux verticalement les files permettrait de traiter plus de files. Systématiser la pose d'échelle (à condition de la détecter correctement) lors de l'acquisition permettrait de résoudre quelques problèmes de paramétrages déjà cités, tout comme la standardisation de la résolution et niveau de zoom des prises.

III.4.2. Perspectives à long terme.

Une des perspectives les plus alléchantes de ce projet est sans doute son application sur des essences de bois différentes. Nous nous sommes restreint à ne traiter que les résineux du fait de leur relative simplicité structurale comparé aux feuillus qui, eux, comportent beaucoup plus de diversité cellulaires. Mais, comme l'illustre la figure 40 présentant les résultats obtenus sur l'acajou, la chaîne de traitement pourrait fonctionner tout aussi bien si l'on adaptait l'apprentissage à ce type d'espèces. Ainsi, la portée du plugin Ficeler sous ImageJ pourrait s'élargir à un plus large public.

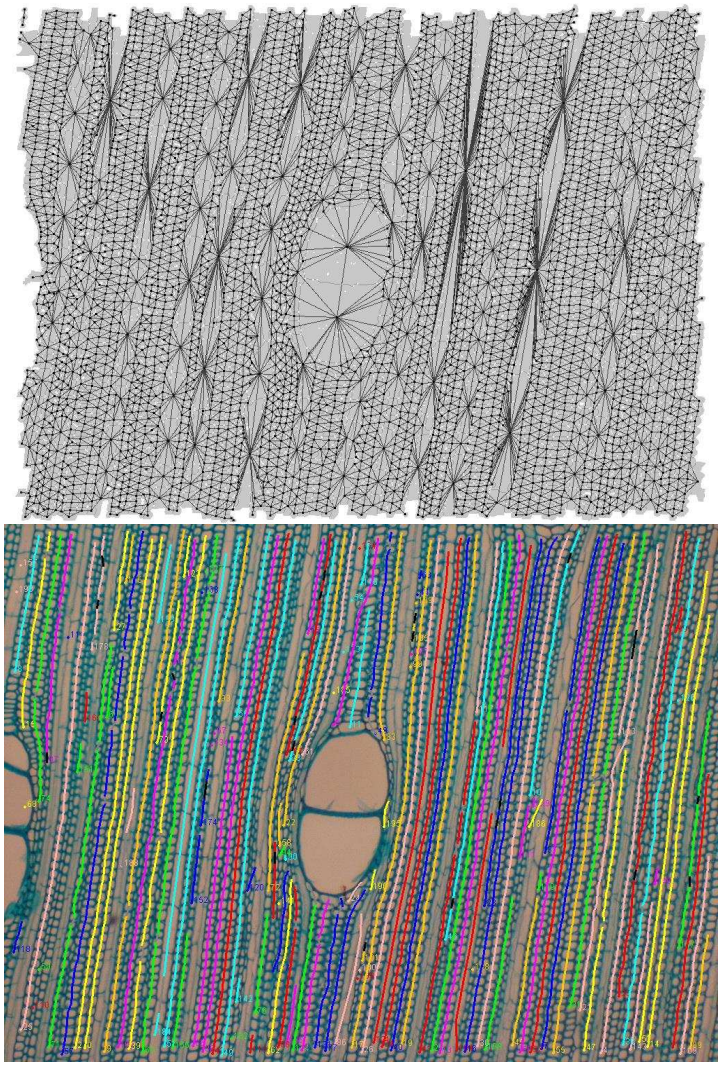
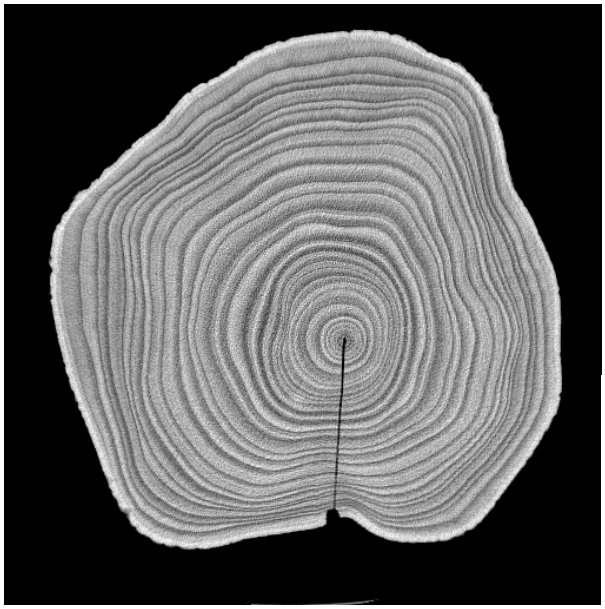


Figure 45 : *Résultat de Ficeler sur une image d'acajou ; en haut le graphe réduit, en bas le tracé des files : des erreurs apparaissent autour des vaisseaux.*

Nous pouvons aussi parler de la généricité de l'approche employée. Un autre problème ouvert se situe à une échelle macroscopique : il s'agit de l'identification automatique des cernes sur des images obtenues par balayage de rayons X. Quelques essais, illustrés ci-dessous, ont été réalisés avec la même chaîne de traitement sujette de ce rapport. Il semblerait que la méthode puisse être appliquée en spécialisant quelques étapes, telle que le prétraitement (différent pour des images de rayon X), la segmentation et la réduction de régions, la détection de structures de forme différentes (ici de cercles concentriques).



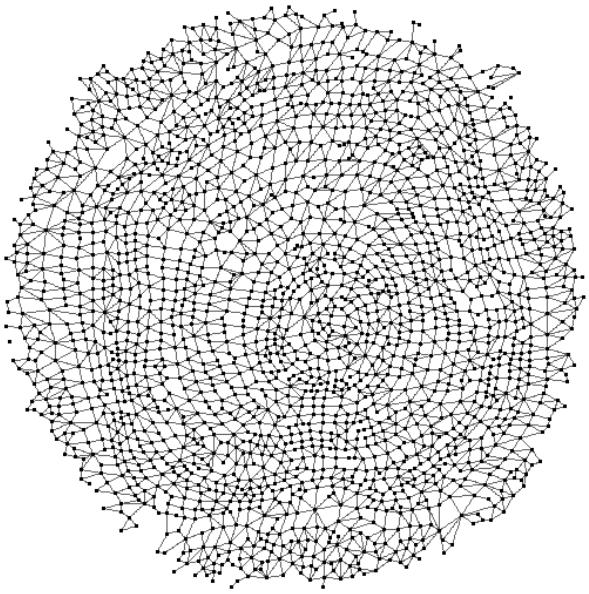


Figure 46 : *Application du plugin Ficeler sur l'identification de cernes.*

IV. Conclusion

Le projet Ficeler concernant la reconnaissance automatique de files cellulaires chez les résineux aura su capter toute mon attention durant ce stage de Master 2 bioinformatique. Il a été le produit original d'applications et de développements informatiques en traitement d'images sur un sujet aujourd'hui indispensable à la compréhension de l'homme concernant le développement et l'architecture des plantes, et ce au sein du laboratoire aujourd'hui reconnu dans ce domaine qu'est l'unité mixte de recherche « bot**A**nique et bioinfor**M**atique de l'**A**rchitecture des **P**lantes ».

Le sujet a été traité suivant une démarche de recherche encadré par un ingénieur de recherche en mathématiques appliquées à l'informatique au sein de mon laboratoire d'accueil (AMAP) ainsi que par un chercheur du CNRS spécialiste en traitement d'image appartenant à l'équipe d'imagerie du Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier (LIRMM).

Cette collaboration a permis la réalisation d'un plugin écrit en Java s'insérant dans un logiciel libre de traitement d'image intitulé *ImageJ*. La solution fournie a d'abord abordé la problématique par un travail de segmentation automatique des cellules de bois présentes sur les images microscopiques à traiter. S'en suit une approche de caractérisation des régions segmentées par une méthode d'analyse statistique intitulée « Classification And Regression Tree » (CART) largement utilisée dans ce domaine. Finalement une dernière étape a été nécessaire à la reconnaissance des files en parcourant le graphe d'adjacence servant de structure de données aux régions segmentées.

Pour ma part, ces six mois m'ont permis de progresser dans bien des domaines aussi bien techniquement qu'humainement. La diversité des méthodes employées dans la réalisation de ce plugin, au delà des progrès en développement Java, m'ont permis de d'accroître mes compétences en traitement d'images, algorithmiques, mathématiques, statistiques, mais aussi en architecture végétale. De plus j'ai eu l'occasion de côtoyer bon nombre de chercheurs aux domaines d'applications très variés et aux compétences complémentaires permettant d'insérer au mieux mon domaine de compétences encore jeune qu'est la bioinformatique.

Bibliographie

[1] Mattias K. Moël & Lloyd A.Donaldson "*Comparison of segmentation method for digital image analysis of confocal microscope images to measaure tracheid cell dimensions*", IAWA Journal, Vol. 22 (3), 2001: 267-288.

[2] M.Milgram & J.-P. Cocquerez "*Fermeture de contour par un operateur local*", Traitement du Signal, volume 3 – n°6, 1986, 303-311.

[3] Vincent L, Soille P, "*Watershed in digital spaces, an efficient algorithm based on immersion simulation*". Trans. PAMI vol 13, n° 6, jun 91.

[4] Meyer F, "*Skeletons in digital spaces. Image analysis and mathematical morphology, theoretical advances*". Serra. Academic press.1988.

[5] Ronald J. & Leanne B. "*A Graph-Based segementation of wood Micrographs*", CSIRO Division of Mathematics and Statistics.

Tables des figures

Tables des annexes

Annexe 1 : Pseudo code de l'algorithme de Ligne de Partage des Eaux [3].

Algorithm 4.1 Vincent-Soille watershed algorithm [52].

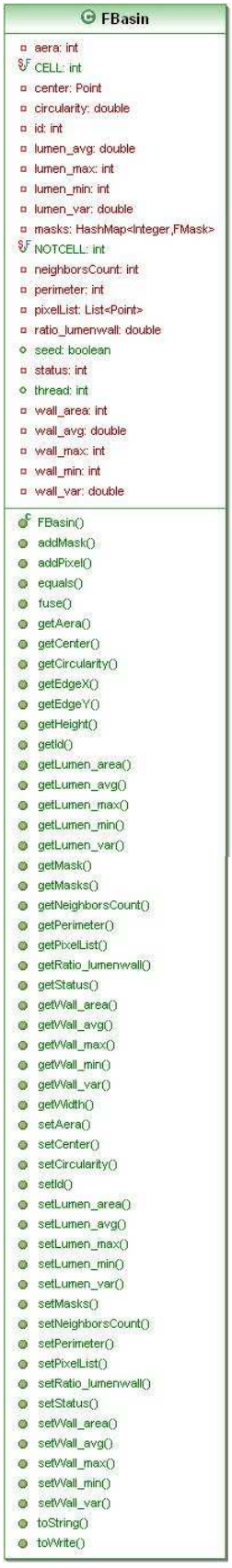
```
1: procedure Watershed-by-Immersion
2: INPUT: digital grey scale image G = (D, E, im).
3: OUTPUT: labelled watershed image lab on D.
4: #define INIT − 1                    (∗initial value of lab image∗)
5: #define MASK − 2                    (∗initial value at each level∗)
6: #define WSHED 0                     (∗label of the watershed pixels∗)
7: #define FICTITIOUS (−1, −1)         (∗fictitious pixel ∉ D∗)
8: curlab ← 0                          (∗curlab is the current label∗)
9: fifo_init(queue)
10: for all p ∈ D do
11:    lab[p] ← INIT ; dist[p] ← 0     (∗dist is a work image of distances∗)
12: end for
13: SORT pixels in increasing order of grey values (minimum h_min, maximum h_max)
14:
15: (∗Start Flooding∗)
16: for h = h_min to h_max do          (∗Geodesic SKIZ of level h − 1 inside level h∗)
17:    for all p ∈ D with im[p] = h do      (∗mask all pixels at level h∗)
18:                (∗these are directly accessible because of the sorting step∗)
19:       lab[p] ← MASK
20:       if p has a neighbour q with (lab[q] > 0 or lab[q] = WSHED) then
21:          (∗Initialize queue with neighbours at level h of current basins or watersheds∗)
22:          dist[p] ← 1 ; fifo_add(p, queue)
23:       end if
24:    end for
25:    curdist ← 1 ; fifo_add(FICTITIOUS, queue)
26:    loop          (∗extend basins∗)
27:       p ← fifo_remove(queue)
28:       if p = FICTITIOUS then
29:          if fifo_empty(queue) then
30:             BREAK
31:          else
32:             fifo_add(FICTITIOUS, queue) ; curdist ← curdist + 1 ;
33:             p ← fifo_remove(queue)
34:          end if
35:       end if
36:       for all q ∈ N_G(p) do          (∗labelling p by inspecting neighbours∗)
37:          if dist[q] < curdist and (lab[q] > 0 or lab[q] = WSHED) then
38:             (∗q belongs to an existing basin or to watersheds∗)
39:             if lab[q] > 0 then
40:                if lab[p] = MASK or lab[p] = WSHED then
41:                   lab[p] ← lab[q]
42:                else if lab[p] ≠ lab[q] then
43:                   lab[p] ← WSHED
44:                end if
45:             else if lab[p] = MASK then
46:                lab[p] ← WSHED
47:             end if
48:          else if lab[q] = MASK and dist[q] = 0 then          (∗q is plateau pixel∗)
49:             dist[q] ← curdist + 1 ; fifo_add(q, queue)
50:          end if
51:       end for
52:    end loop
53:    (∗detect and process new minima at level h∗)
54:    for all p ∈ D with im[p] = h do
55:       dist[p] ← 0          (∗reset distance to zero∗)
56:       if lab[p] = MASK then          (∗p is inside a new minimum∗)
57:          curlab ← curlab + 1 ;          (∗create new label∗)
58:          fifo_add(p, queue) ; lab[p] ← curlab
59:          while not fifo_empty(queue) do
60:             q ← fifo_remove(queue)
61:             for all r ∈ N_G(q) do          (∗inspect neighbours of q∗)
62:                if lab[r] = MASK then
63:                   fifo_add(r, queue) ; lab[r] ← curlab
64:                end if
65:             end for
66:          end while
67:       end if
68:    end for
69: end for
70: (∗End Flooding∗)
```

Annexe 2 : Algorithme de détection des bassins versants (code Java).

```
/**
 * method who full the basin structure from watershed imageprocessor
 */
private void extractBasins() {
	// initialize boolean read[W][H] array to false (W=image width, H=image
	// height) to know read pixel.
	initReadArray();
	int number = 0;
	for (int y =0; y < H; y++) {
		for (int x = 0; x < W; x++) {
			FBasin currentBasin;
			List<Point> listpix;
			// if pixel is not read and black (basins are black on the
			// image).
			if (!read[x][y] && watershedIProcessor.getPixel(x, y) == 0) {
				listpix = new ArrayList<Point>();
				// build current basin
				trackBasin(x, y, listpix);
				currentBasin = new FBasin(number, listpix);
				this.listOfBasin.add(currentBasin);
				number++;
			}
		}
	}
	System.out.println(listOfBasin.size() + " basins detected");
}
```

```
/**
* find all pixels  to include in current basin
* @param xx x position of current pixel
* @param yy y position of current pixel
* @param curentBasin FBasin currently detected
*/
private void trackBasin(int xx, int yy, List<Point> curentBasin) {
	FBasinFIFO fifo = new FBasinFIFO();
	fifo.addPoint(new Point(xx,yy));
	read[xx][yy] = true;
	Point point=null;
	while(!fifo.isEmpty()) {
		point = fifo.remove();
		curentBasin.add(point);
		int x = point.x;
		int y = point.y;
		// look on connexity*4 if regions pixel are present and add it in fifo
		// stack.
		if(x<W-1 && !read[x + 1][y] && watershedIProcessor.getPixel(x + 1, y) == 0) {
			read[x+1][y] = true;
			fifo.addPoint(new Point(x+1, y));
		}
		if(y<H-1 &&!read[x][y + 1] && watershedIProcessor.getPixel(x, y + 1) == 0) {
			read[x][y+1] = true;
			fifo.addPoint(new Point(x, y+1));
		}
		if (x>0 && !read[x - 1][y]&& watershedIProcessor.getPixel(x - 1, y) == 0) {
			read[x-1][y] = true;
			fifo.addPoint(new Point(x-1, y));
		}
		if ((y>0) && !read[x][y - 1] && watershedIProcessor.getPixel(x, y - 1) == 0) {
			read[x][y-1] = true;
			fifo.addPoint(new Point(x, y-1));
		}
	}
}
```

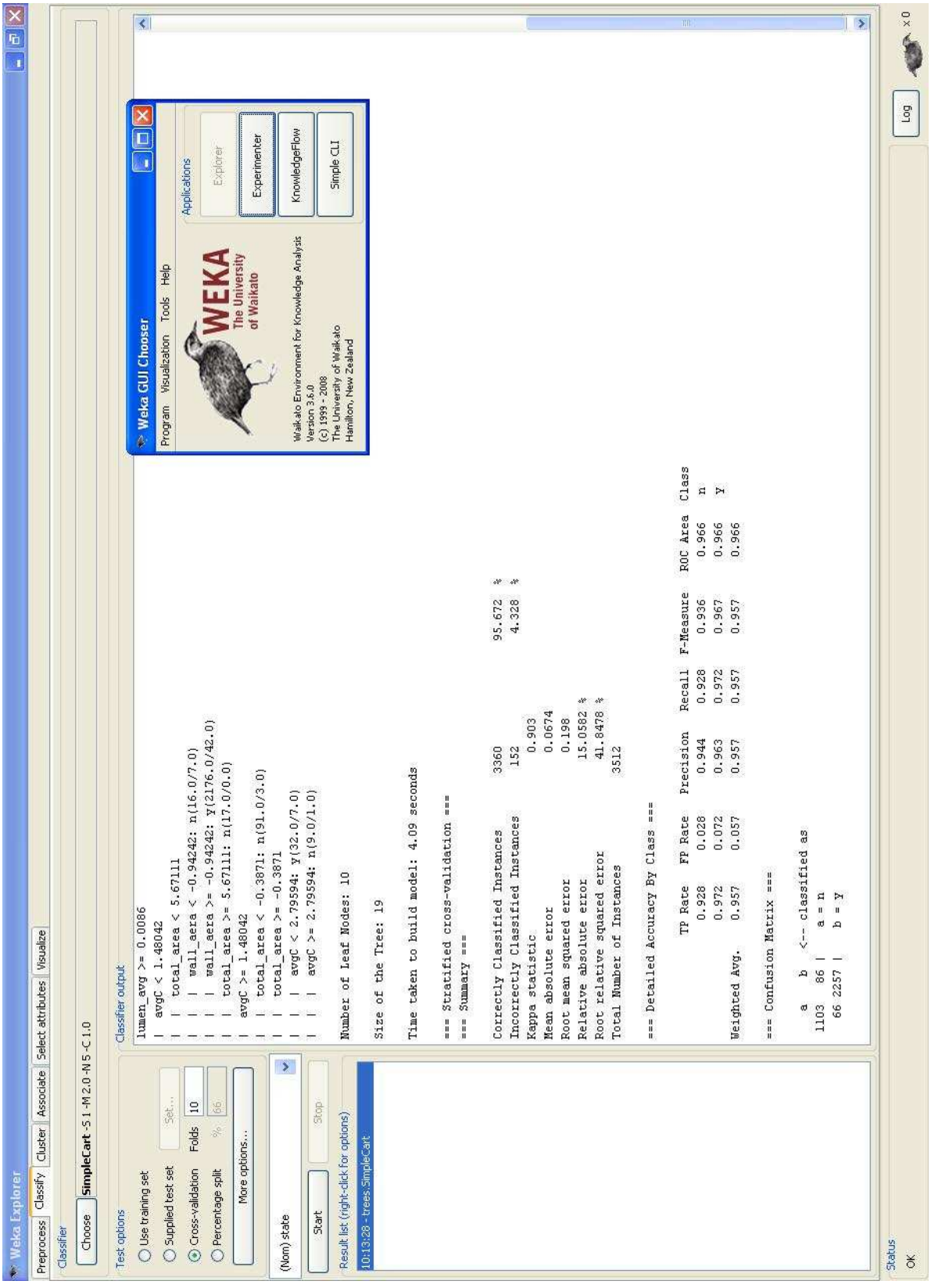
Annexe 3 : Diagramme de classe des classes *FBasin* et *FEdge.*



Annexe 4 : Variables utilisées lors la classification décrivant les bassins.

total_area : l'aire totale
neighbors : nombre de voisins
min : min d'intensité
max : max d'intensité
avg : moyenne d'intensité
var : variance d'intensité
dista : moyenne des distances des régions voisines
lumen_area : aire du lumen
lumen_min : min d'intensité du lumen
lumen_max : max d'intensité du lumen
lumen_avg : moyenne d'intensité du lumen
lumen_var : variance d'intensité du lumen
wall_area : aire de la paroi
wall_min : min d'intensité de la paroi
wall_max : max d'intensité de la paroi
wall_avg : moyenne d'intensité de la paroi
wall_var : variance d'intensité de la paroi
mins : moyenne des min d'intensité des arêtes de séparation
maxs : moyenne des max d'intensité des arêtes de séparation
avgs : moyenne des moyennes d'intensité des arêtes de séparation
vars : moyenne des variances d'intensité des arêtes de séparation
mina : moyenne des min d'intensité des arêtes d'adjacence
maxa : moyenne des max d'intensité des arêtes d'adjacence
avga : moyenne des moyennes d'intensité des arête d'adjacence
vara : moyenne des variances d'intensité des arêtes d'adjacence
minC : min d'intensité du périmètre de la région
maxC : max d'intensité du périmètre de la région
avgC : moyenne d'intensité du périmètre de la région
varC : variance d'intensité du périmètre de la région
state : flag donné lors l'étalonnage

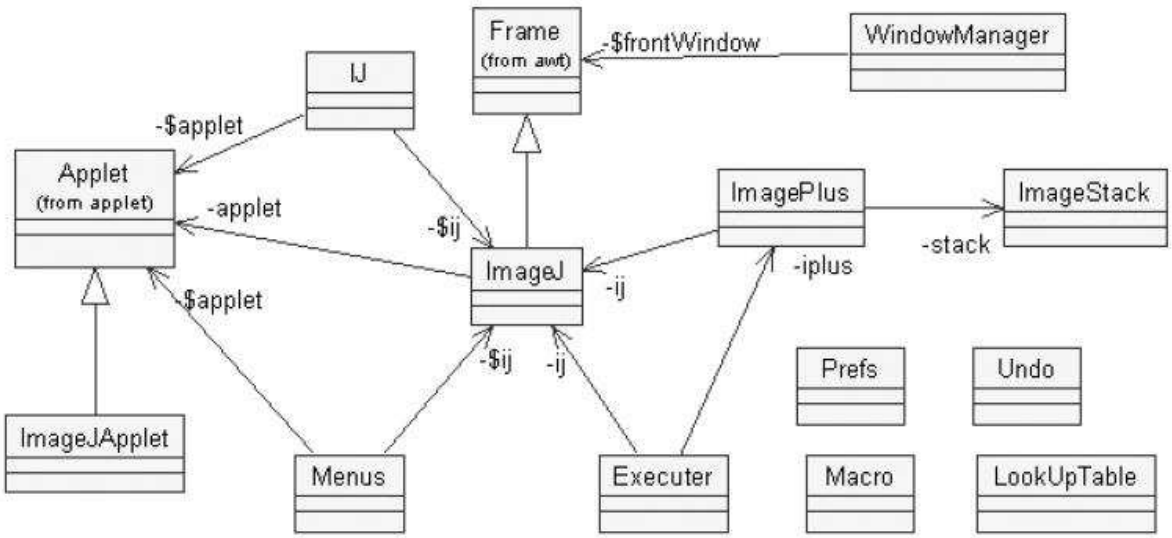
Annexe 5 : Utilisation et résultats obtenus avec le logiciel de fouille de donnée Weka.

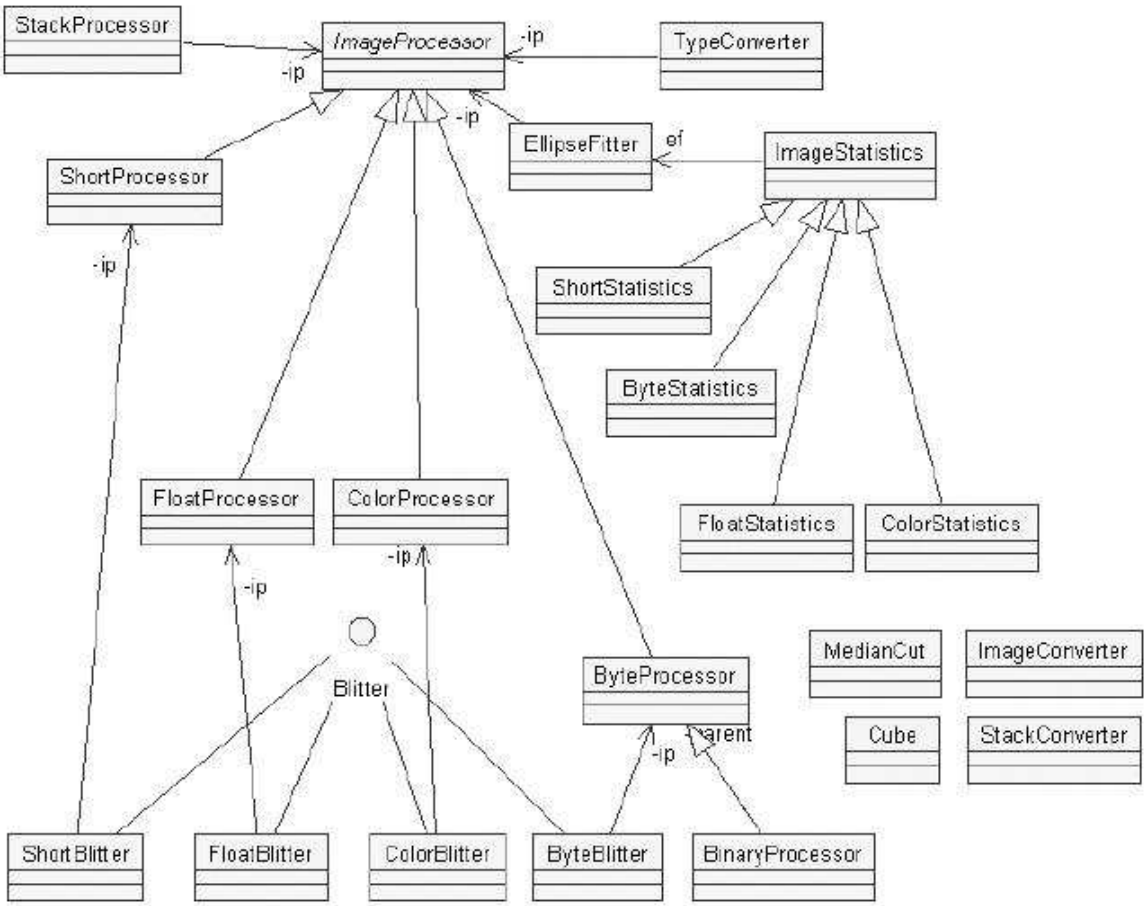


Annexe 6 : Rapport de la création d'arbre par la méthode CART sous Weka.

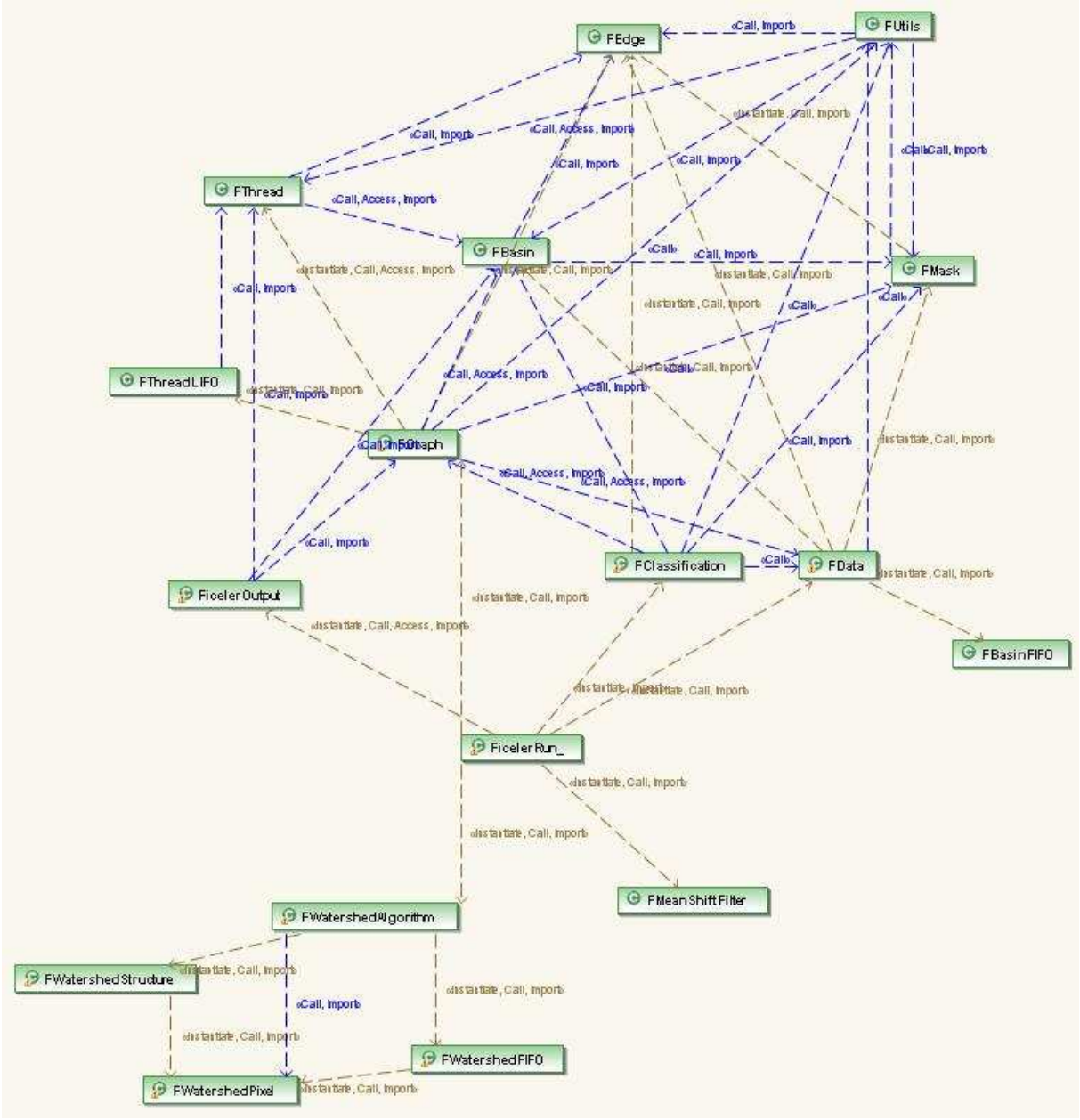
```
=== Run information ===

Scheme:       weka.classifiers.trees.SimpleCart -S 1 -M 2.0 -N 5 -C 1.0
Relation:     cart1
Instances:    3512
Attributes:   31
              Id, total_area, neighbors, min, max, avg, var, dista, lumen_area,
              lumen_min, lumen_max,
              lumen_avg, lumen_var, wall_aera,
              wall_min, wall_max, wall_avg, wall_var, mins, maxs, avgs, vars,
              mina, maxa, avga, vara, minC, maxC, avgC, varC, state

Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

CART Decision Tree

lumen_avg < 0.0086
|  lumen_var < 1.41761: n(935.0/35.0)
|  lumen_var >= 1.41761
|  |  maxC < -0.04125: y(53.0/10.0)
|  |  maxC >= -0.04125
|  |  |  lumen_avg < -1.73336: y(8.0/0.0)
|  |  |  lumen_avg >= -1.73336: n(62.0/8.0)
lumen_avg >= 0.0086
|  avgC < 1.48042
|  |  total_area < 5.67111
|  |  |  wall_aera < -0.94242: n(16.0/7.0)
|  |  |  wall_aera >= -0.94242: y(2176.0/42.0)
|  |  total_area >= 5.67111: n(17.0/0.0)
|  avgC >= 1.48042
|  |  total_area < -0.3871: n(91.0/3.0)
|  |  total_area >= -0.3871
|  |  |  avgC < 2.79594: y(32.0/7.0)
|  |  |  avgC >= 2.79594: n(9.0/1.0)

Number of Leaf Nodes: 10
Size of the Tree: 19
Time taken to build model: 4.09 seconds

=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances        3360               95.672  %
Incorrectly Classified Instances       152                4.328  %
Kappa statistic                          0.903
Mean absolute error                      0.0674
Root mean squared error                  0.198
Relative absolute error                 15.0582 %
Root relative squared error             41.8478 %
Total Number of Instances             3512

=== Detailed Accuracy By Class ===
               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.928    0.028      0.944     0.928     0.936      0.966    n
                 0.972    0.072      0.963     0.972     0.967      0.966    y
Weighted Avg.    0.957    0.057      0.957     0.957     0.957      0.966

=== Confusion Matrix ===
    a    b   <-- classified as
 1103   86 |    a = n
   66 2257 |    b = y
```

Annexe 7 : Diagramme de classe des packages *IJ* et *IJ.Processe*.

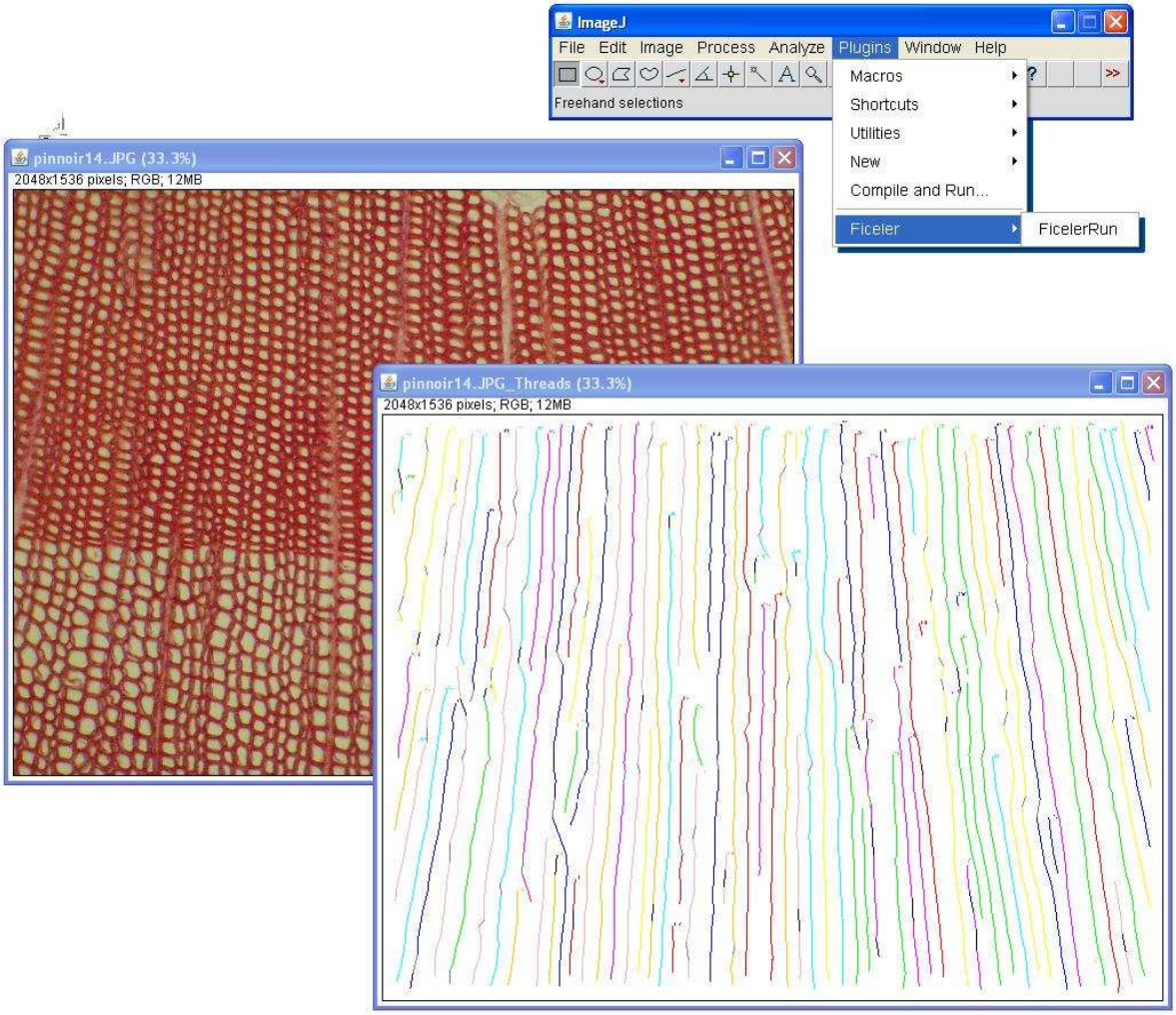




Annexe 8 : Diagramme de dépendance des classes plugin Ficeler.



Annexe 9 : Utilisation du plugin Ficeler dans ImageJ.



Annexe 10 : Sorties numériques du plugin Ficeler.

Il s'agit d'un fichier texte au format CSV : on retrouve la légende des données en première ligne, puis les données des files triée par ordre décroissant selon l'indice de longueur où une ligne correspond à une cellule de la ligne.

```
file, length_ratio, cell_ratio, cell, tag(cell=1 other=2), lumen_area,
ratio_lumen/wall, circularity
15,100,77,1513,2,48,57.0,0.6388291371290634
15,100,77,1460,1,675,50.0,0.7918704382810352
15,100,77,1394,1,694,52.0,0.7887057774219032
15,100,77,1364,1,385,51.0,0.6354649484289968
15,100,77,1357,2,76,85.0,0.7311320696387643
15,100,77,1286,1,415,50.0,0.6699937025467987
15,100,77,1277,2,79,74.0,0.7746319583191601
15,100,77,1240,1,317,47.0,0.8008651387077719
15,100,77,1189,1,387,46.0,0.7906716965636248
15,100,77,1138,1,450,47.0,0.8668162075859002
15,100,77,1063,1,786,57.0,0.8253189677408834
15,100,77,1000,1,460,52.0,0.7031526022810153
15,100,77,906,1,433,45.0,0.6499493965107728
15,100,77,848,1,374,43.0,0.8484521619267046
15,100,77,802,1,304,41.0,0.7886606656556394
15,100,77,771,1,228,35.0,0.8534358003833581
15,100,77,718,1,385,42.0,0.8335769876691648
15,100,77,644,1,407,44.0,0.7765705048034715
15,100,77,625,2,25,35.0,0.8957167302196744
15,100,77,575,1,399,44.0,0.6728225085118614
15,100,77,549,2,27,27.0,0.9095828925138423
15,100,77,475,1,587,50.0,0.7223413252224061
15,100,77,457,2,68,62.0,0.7588734637926687
15,100,77,385,1,456,47.0,0.7345114092602666
15,100,77,341,1,263,34.0,0.5835751742142197
15,100,77,262,1,309,46.0,0.7596034668182995
15,100,77,229,1,120,24.0,0.7298131021442217
7,96,76,1494,2,50,21.0,0.8130585809961919
7,96,76,1467,1,411,45.0,0.7711056045824956
7,96,76,1412,1,459,42.0,0.8611617335848307
7,96,76,1335,1,599,56.0,0.7801697427484844
7,96,76,1331,2,78,64.0,0.8580192364756423
7,96,76,1256,1,664,54.0,0.8346948525777489
7,96,76,1196,1,521,38.0,0.7981652756532083
7,96,76,1156,1,382,41.0,0.727067634120247
7,96,76,1089,1,487,50.0,0.7365194647230395
7,96,76,1028,1,575,54.0,0.8079496876053993
7,96,76,1002,2,37,29.0,0.9001028259778735
7,96,76,921,1,499,46.0,0.8190668933165339
7,96,76,855,1,508,45.0,0.7449836213797542
7,96,76,796,1,485,41.0,0.8501631051125818
7,96,76,752,1,568,44.0,0.6909270840843488
7,96,76,662,1,487,41.0,0.7541376541821551
7,96,76,599,2,142,24.0,0.560445538474158
7,96,76,594,2,77,31.0,0.6808521272395219
7,96,76,511,1,526,48.0,0.7429056755048705
7,96,76,446,1,438,43.0,0.8407394932656351
7,96,76,420,2,58,31.0,0.44689191146650853
7,96,76,336,1,510,43.0,0.7093177944678187
7,96,76,228,1,495,37.0,0.6132873721936507
```

Résumé

Le projet Ficeler concerne la reconnaissance automatique de files cellulaires chez les résineux. Il a été le produit original d'application et développement informatique en traitement d'images, statistiques, mathématiques sur un sujet aujourd'hui indispensable à la compréhension de l'homme concernant le développement et l'architecture des plantes. Ces besoins ont été émis par l'unité mixte de recherche « bot**A**nique et bioinfor**M**atique de l'**A**rchitecture des **P**lantes » spécialiste dans ce domaine. Les résultats obtenus auront permis une approche nouvelle du problème déjà réfléchi dans le passé. Une chaîne de traitement aura été mise en place, fonctionnelle sous le logiciel libre de traitement d'images *ImageJ*. Le plugin fourni pourra cependant suivre autant d'améliorations techniques que d'innovations pouvant élargir sa portée au niveau des espèces et/ou structures traitées.

Summary

The Ficeler project concerns the automatic recognition of cell lines among conifers. It was the original product of application and computer development in image processing, statistics, mathematics on a subject now necessary to the human understanding of plants' development and structure. These needs were brought to light by the mixed unit of research AMAP: an expert in the field. Acquired results had allowed a new approach on this matter already reflected in the past. A processing chain has been set up, functional under free image processing software: ImagesJ. The provided plugin will, however, be able to follow as much technical improvements as innovations that can enlarge his range on others tree species and/or treated structures.