Guilhelm Savin

Segmentation 2D et 3D par Systèmes Multi-Agents

Encadré par Marie Aimar-Beurton et Pascal Desbarats
LaBRI - Université de Bordeaux 1
12 juin 2008

Remerciements,
Je tiens à remercier en premier lieu Marie Aimar-Beurton et Pascal Desbarats pour m'avoir encadré, conseillé et supporté. Je remercie aussi Guillaume Damiand d'avoir accepté d'être mon tuteur.
Merci à Solenne de m'avoir insufflé la motivation lorsqu'elle n'était pas suffisante, ainsi que pour sa relecture. Merci à Frédéric pour son soutien et sa relecture.
Enfin, merci à ma mère pour sa correction de mes oublis orthographiques.

Résumé

Il existe un grand nombre de méthodes de segmentation d'image. Celles-ci sont généralement adaptées à un type d'image (photographie numérique, IRM, image satellite,...). À notre connaissance, il n'existe pas de méthode générique permettant de résoudre le problème de la segmentation pour n'importe quelle image.

Dans ce mémoire, nous allons étudier une méthode de segmentation reposant sur un système inspiré de colonies d'araignées sociales. Ces araignées sont capables de tisser collectivement des structures en soie lesquelles sont grandes par rapport à la taille des araignées.

Ce mémoire présente une adaptation de cette méthode au problème de la segmentation d'image.

Mot clefs segmentation, agent, vie artificielle, araignée sociale

Abstract

There are many segmentation methods in image processing. These methods are usually limited to a few image types. To our knowledge, there is no generic method which is able to solve segmentation problem for any image.

In this master thesis, a segmentation method which uses a socials spiders colonies system will be studied and improved. These spiders are able to collectively weave silk structure.

This master thesis aims to present an adaptation of this method to the segmentation problem.

Keywords artificial life, segmentation, agent, social spider

Table des matières

Table des figures

Liste des tableaux

Introduction

La segmentation d'image est un problème important dans les domaines de l'analyse d'image. Elle est par exemple utilisée en imagerie médicale afin d'analyser et de quantifier les différentes structures anatomiques présentes dans les images.

Les méthodes de segmentation existantes peuvent être classées selon le but à atteindre. Par exemple, il existe des méthodes qui déterminent des régions dans l'image alors que d'autres au contraire cherchent à déterminer les frontières des régions. Il est aussi possible de classer les méthodes selon l'approche utilisée pour résoudre le problème de la segmentation. On trouvera par exemple des méthodes basées sur des connaissances explicites de l'image, d'autres basées sur un ensemble de probabilités...

La méthode que nous intéresse et que nous développerons dans ce mémoire est d'un genre nouveau. Elle repose en effet sur un système inspiré de la nature : des colonies d'araignées sociales. Ces araignées d'environ 5mm sont capables de tisser une structure en fils de soie pouvant atteindre un volume de $100m^3$. Cette structure (figure 1) est utilisée ensuite pour piéger des proies pouvant atteindre 700 fois la taille d'une araignée.

FIG. 1 – Toile tissée par une colonie de *Anelosimus Eximius*

C. BOURJOT ET AL. ont proposé un article [3] reproduisant le mécanisme de création de la toile afin de détecter des régions dans une image. Cependant, la méthode exposée comporte des problèmes et nécessite d'être comparée à d'autres méthodes afin de pouvoir déterminer la pertinence de cette méthode et la direction à suivre pour l'améliorer.

Nous présenterons dans le chapitre 1 un état de l'art sur la segmentation d'images ainsi que la vie artificielle en informatique et les systèmes multi-agents. Il sera suivi par une description de la méthode et de son fonctionnement. Cette description sera accompagnée des différents problèmes liés à cette méthode.

Le chapitre 2 présentera la résolution de problèmes liés à cette méthode ainsi que des comparaisons avec d'autres méthodes. Ces comparaisons auront pour but de déterminer les défauts de la méthode afin de présenter en perspectives des solutions qui permettront d'améliorer cette méthode.

Chapitre 1

Segmentation d'image et araignées sociales

Nous allons dans ce chapitre présenter le contexte dans lequel se situe la problèmatique de ce mémoire. Nous présenterons dans un premier temps la segmentation d'images ainsi que les différentes approches existantes. La méthode de segmentation qui sera l'objet de ce mémoire utilise un système reproduisant le comportement de colonies d'araignées sociales, c'est pourquoi nous présenterons ensuite le domaine de la vie artificielle. Nous terminerons par la présentation des systèmes multi-agents.

1.1 Segmentation d'images

La segmentation est une des premières étapes de l'analyse d'image. Elle permet de partitionner l'image en plusieurs ensembles disjoints de pixels/voxels[1]. Ces ensembles sont appelés *régions*. Les pixels/voxels d'une même région partagent une propriété d'homogénéité. Cette propriété varie selon le domaine auquel est appliquée la segmentation, mais en règle générale elle permet de donner un sens à la région comme par exemple associer la région à un objet connu ou à déterminer.

La propriété définissant l'homogénéité des pixels varie selon l'objectif de la segmentation. Ce but est généralement associé au domaine où la segmentation est appliquée. Le domaine de l'imagerie médicale fournit un grand nombre d'applications à la segmentation d'image. Par exemple, N. Richard *et al.* utilisent la segmentation d'image afin d'étiqueter les voxels d'une image cérébrale obtenue par résonance magnétique ce qui permet d'associer les voxels à un type de tissu (matière blanche, matière grise, ...)[11]. Cependant, les applications ne se limitent pas au domaine de l'imagerie médicale. On peut par exemple citer A. Guillaud *et al.* qui utilisent la segmentation

[1] *volumetric pixel* : désigne les pixels d'images 3D

afin de compter le nombre de cercles présent dans des *otholithes*[2] afin d'en déterminer l'age [1]. Citons enfin un article de L. Sant'Anna Bins *et al.* [2] où la segmentation est appliquée aux images satellites afin d'en extraire des informations des régions détectées telles que l'aire, la forme, la texture... La segmentation permet alors de créer une abstraction d'une image.

1.1.1 Différents types d'approche

Le processus de segmentation peut être réalisé de plusieurs façons :
- par étiquetage des pixels,
- par détection des régions,
- par détection du contour des régions,
- par la combinaison de la détection de regions/contours.

La détection des régions consiste à déterminer les ensembles de pixels répondant à la propriété d'homogénéité alors que la détection du contour des régions cherche à déterminer ceux qui ne répondent pas à cette propriété c'est à dire les frontières. On trouve dans la littérature certaines méthodes combinant ces deux approches [9].

La classification précédente prend en compte le but que la méthode de segmentation cherche à atteindre. Il est cependant possible de classer les méthodes de segmentation selon l'approche utilisée comme le fait L. Germond dans sa thèse [8]. Selon cette classification, il est possible de distinguer quatre types d'approches :
- approche par classification,
- approche utilisant des modèles,
- approche à base de connaissances,
- approche coopérative.

Approche par classification

Parmi les méthodes empruntant une approche par classification, on peut citer la classification bayésienne ou par champs de Markov. La classification bayésienne repose sur la connaissance de la distribution des niveaux de gris des tissus recherchés. La théorie des champs de Markov utilise la notion de voisinage d'un pixel afin d'introduire des modèles locaux semblables à la classification bayésienne.

Approche utilisant des modèles

Parmi les approches utilisant des modèles, on distinguera principalement les **modèles énergétiques** et les **modèles paramétriques**. Ce type d'approches tend à déterminer les contours des régions. Dans le cas de modèles

[2]structure minérale, constituant de l'oreille interne des vertébrés

énergétiques, un contour représenté par une courbe paramétrique est positionné sur l'image puis soumis à deux types de forces. Le premier type représente les forces internes qui déforment le contour. Le second permet de définir l'attraction du contour vers les contours de l'image. La solution est obtenue en minimisant l'énergie du système. Cette approche reste délicate dans certains domaines comme la segmentation d'IRM, principalement à cause de la sensibilité de cette méthode aux positions initiales des courbes paramétriques.

Approche à base de connaissances

Dans les approches précédentes, les connaissances sont intrinsèques. Il est cependant possible de détacher la représentation des connaissances de la méthode de segmentation afin que celles-ci soient explicites. On obtient ainsi une base de connaissances que la méthode de segmentation peut utiliser afin de segmenter l'image. Par exemple, dans le cas de la segmentation d'images IRM, les connaissances peuvent être les quantités des différents tissus présents sur l'image.

Approche coopératives

Enfin, les approches coopératives permettent de faire coopérer plusieurs méthodes de segmentation. On peut distinguer la coopération séquentielle, qui effectue différents types de segmentation successivement, de la coopération itérative, qui effectue différents types de segmentation simultanément.

La méthode des araignées, que nous verrons par la suite, peut être classée parmi les méthodes détectant des régions dans une image. Cependant, la seconde classification est insuffisante pour classer cette méthode. En effet dans cette méthode le processus de segmentation émerge de sous-processus dont le but n'est pas la segmentation. Il nous faut donc ajouter à la classification précédente les approches émergentes qui nous permettra de regrouper les méthodes où, comme pour la méthode des araignées, la tâche de segmentation émerge d'autres tâches.

1.2 Vie artificielle

1.2.1 Définition

La vie artificielle en informatique est un domaine où la vie biologique est étudiée afin d'être modélisée. Elle prend ses sources dans la thèse de Church-Turing dont l'interprétation forte énonce qu'une machine de Turing peut reproduire le comportement de tout système physique. C'est cependant Christopher Langton qui regroupa les travaux concernant

le domaine de la vie artificielle et inventa le terme en 1987 pour la première conférence internationale *the Synthesis and Simulation of Living System*. La définition initiale de C. LANGTON concernant la vie artificielle fut :

> "La vie artificielle est l'étude des systèmes construits de mains d'homme qui exhibent des comportements caractéristiques des systèmes naturels vivants. Elle vient en complément des sciences biologiques traditionnelles, qui *analysent* des organismes vivants, en tentant de *synthétiser* des comportements semblables au vivant au sein d'ordinateurs et d'autres substrats artificiels. En étendant les fondements empiriques sur lesquels la biologie est basée au-delà de la vie à base de carbone qui a évolué sur Terre, la vie artificielle peut contribuer à la biologie théorique en positionnant *la vie telle que nous la connaissons* au sein d'un espace plus large : *la vie telle qu'elle pourrait être.*"[3]

Cependant, cette définition fut trop générale pour définir formellement le domaine de la vie artificielle. Aujourd'hui encore ce domaine est mal défini et se contente de regrouper les travaux s'inspirant de la biologie. Il s'agit d'un domaine connexe à l'intelligence artificielle. On distingue deux interprétations de la vie artificielle : l'**interprétation forte** et l'**interprétation faible**[4].

L'interprétation forte vient directement des buts initialement fixés par LANGTON concernant la vie artificielle :

> "Le but ultime de l'étude de la vie artificielle serait de créer la vie dans un autre substrat, idéalement un substrat *virtuel* où l'essence de la vie aurait été abstraite des détails de sa mise en œuvre dans *quelque substrat que ce soit.* Nous aimerions construire des modèles qui sont si semblables au vivant qu'ils cesseraient d'être des simulations de la vie pour en devenir des *exemples.*"[5]

Cette interprétation prétend donc qu'il est possible de reproduire l'organisation logique d'un organisme vivant sur un autre substrat (un ordinateur par exemple), ce qui rendrait ce substrat vivant. Cette interprétation est fortement critiquée. Dans le cadre du sujet qui nous intéresse, la segmentation d'image, cette interprétation ne nous intéresse guère car elle ne constitue pas une façon de résoudre un problème mais plutôt une certaine forme de philosophie.

L'interprétation faible de la vie artificielle cherche simplement à reproduire les mécanismes du vivant afin de résoudre des problèmes de façon autonome. Par exemple, l'évolution des espèces telle que la décrite DARWIN a inspiré le domaine des algorithmes évolutionnaires[6]. Dans ce type d'algorithme, on manipule une population de solutions. Une fonction dite de *fitness* permet de définir la qualité des individus. Les meilleurs individus de

[3] LANGTON C.G., 1989-2,-[II-21],p.1,italiques originales
[4] strong alife, weak alife
[5] LANGTON C.G.,1986.-[II-20],p.147,italiques originales
[6] evolutionary computation

la population sont selectionnés puis croisés afin de créer une nouvelle population. Les algorithmes génétiques représentent la branche la plus connue des algorithmes évolutionnaires.

Des mécanismes du vivant particulièrement intéressants sont ceux que l'on peut observer dans les sociétés. Il s'agit en effet de mécanismes propres au vivant permettant d'organiser une certaine population d'individus dans le but de faire survivre et prospérer cette population. L'inspiration des sociétés a ouvert la voie vers un nouveau domaine, *l'intelligence artificielle distribuée*, qui permet de répartir l'intelligence globale sur un ensemble d'individus. En effet, l'étude d'espèces d'insectes sociaux a permis de dégager un comportement collectif intelligent de ces espèces à partir d'un comportement simple des individus et d'une forme de communication entre ces mêmes individus. Les systèmes multi-agents, que nous verrons par la suite, sont un bon outil pour modéliser ces individus ainsi que les intéractions entre eux. L'exemple le plus présent dans la littérature est sûrement celui des fourmis utilisées pour la résolution de problème d'optimisation combinatoire [5].

La méthode de segmentation des araignées que nous allons présenter repose sur un système qui reproduit les mécanismes de colonies sociales d'araignées. Cette méthode entre donc dans l'interprétation faible de la vie artificielle.

Dans un contexte général, l'utilisation de la vie artificielle, en particulier de l'intelligence artificielle distribuée, permet de bénéficier du caractère autonome et adaptatif du vivant. Jean-Philippe Rennard consacre un livre [14] à la vie artificielle dans lequel on trouve une étude plus approfondie sur ce domaine. D'après lui, l'auto-organisation des colonies d'insectes repose sur quatre points fondamentaux :

- existence d'interactions multiples
- rétroaction positive
- rétroaction négative
- amplification des fluctuations

La rétroaction[7] permet d'attribuer un effet retour à un évenement. Une **rétroaction positive** amplifiera le phénomène lié à l'évenement déclencheur. Par exemple dans le cas des fourmis, celles-ci sont attirées par les chemins ayant une forte concentration de phéromones, mais en suivant ce chemin elles renforcent la concentration de celui-ci. La **rétroaction négative** atténue les effets d'un phénomène. Par exemple, les phéromones déposées par les fourmis s'évaporent dans le temps.

[7] anglais : *feedback*

1.2.2 La vie artificielle dans la segmentation

Dans le domaine de la segmentation, la vie artificielle offre certains avantages [7]. Le premier est la possibilité d'utiliser une forme de communication entre différents organismes. Cette communication permet à partir d'échanges d'informations locales d'obtenir des informations globales qui à leur tour permettront aux organismes d'être plus efficaces dans leur tâche. Un second avantage de la vie artificielle pour la segmentation d'image est l'autonomie dont font preuve les organismes. Par exemple, dans le cas d'un système utilisant des organismes se déplaçant dans un environnement, l'état final du système ne sera pas influencé par les positions initiales des organismes.

Un exemple de méthode utilisant la vie artificielle est la méthode des organismes déformables [17] basés sur les modèles déformables de Terzopoulos [10] très présents dans la littérature. Cette méthode combine la géométrie, la physique et la théorie de l'approximation afin de créer des organismes élastiques soumis à des forces. Ces organismes vont se déformer afin de prendre la forme du contour d'une région.

1.3 Systèmes multi-agents

Les agents sont des entités partiellement autonomes disposant d'un nombre restreint de fonctionnalités, capables de communiquer et d'évoluer dans un environnement. Les agents peuvent percevoir et modifier leur environnement. On distingue deux catégories d'agents : les **agents cognitifs** et les **agents réactifs**. Les agents réactifs se contentent de réagir à des stimuli de l'environnement. Les agents cognitifs héritent de ce comportement mais possèdent en plus une couche d'apprentissage ainsi qu'une couche décisionnelle leur permettant d'évoluer.

Les systèmes multi-agents désignent les sytèmes où un ensemble d'agents exécutent des tâches locales dans un environnement afin de résoudre une tâche globale.

1.3.1 Interactions entre agents

La puissance des systèmes multi-agents vient en partie de la capacité des agents à interagir entre eux. Ces interactions peuvent être de deux formes : directes ou indirectes.

Les **interactions directes** sont en général caractéristiques d'agents cognitifs qui, œuvrant dans un but précis, sont capables de communiquer intentionellement. Ce type d'interactions se rapproche d'un acte de langage. Les **interactions indirectes** sont au contraire caractéristiques d'agents réactifs qui en réponse à des stimuli de l'environnement déposent des informations dans celui-ci. En 1959, P.P. Grassé définit ce type d'interactions indirectes dans les sociétés (en particulier les insectes) sous le nom de *stigmergie* :

« La coordination des tâches, la régulation des constructions ne dépendent pas directement des ouvriers, mais des constructions elles-mêmes. "L'ouvrier ne dirige pas son travail, il est guidé par lui." C'est à cette stimulation d'un type particulier que nous donnons le nom de stigmergie. »[8]

Vincent Chevrier consacre une partie de son mémoire de HDR [4] à la description de son travail sur les interactions entre agents. Ce travail met en avant l'importance des interactions entre agents et le besoin de modéliser ces interactions.

Les buts des interactions entre agents peuvent être classés en deux catégories : **coopération** et **compétition**. Un agent peut en effet collaborer en vue de résoudre un but commun ou au contraire chercher à être le plus performant pour résoudre une tâche.

1.3.2 Avantages des systèmes multi-agents

Les systèmes multi-agents offrent divers avantages, le principal étant de permettre une distribution des tâches entre plusieurs entités. Par exemple, lors de la réalisation d'un programme complexe, il peut être intéressant de répartir les différentes fonctionnalités du programme entre différentes entités. Les systèmes multi-agents deviennent alors un bon outil pour réaliser cette repartition des fonctionnalités et optimiser les interactions entre ces entités.

Les systèmes multi-agents possèdent un autre avantage important. Il s'agit de leur capacité à modéliser les sociétés qu'elles soient humaines ou animales. En effet, les colonies d'insectes tels que les fourmis sont souvent constituées de classes d'individus ou **castes** proposant des services. On parlera par exemple de fourmi ouvrière, guerrière, etc... Chaque fourmi est donc un agent qui propose les services qu'il est capable de réaliser et cela en vue d'un but commun à savoir la survie de la colonie. De ce fait, les systèmes multi-agents sont un bon outil pour modéliser des systèmes de vie artificielle ou d'intelligence artificielle distribuée généralement composés de diverses entités qui interagissent entre elles. Un exemple d'utilisation de la vie artificielle et de modélisation agents dans le cadre de la segmentation d'image est donné par J. Fripp [7].

1.3.3 Systèmes multi-agents et segmentation

Les sytèmes multi-agents sont utilisés dans un grand nombre de méthodes de segmentation [11, 1, 7, 15]. La méthode proposée par Nathalie Richard *et al* [11] utilise un système multi-agents afin de mettre en place diverses castes d'agents dans le but de segmenter des images cérébrales à trois dimensions. Dans cet article, les auteurs proposent une méthode de segmen-

[8] Grassé P.P.,1959.-[VII-33]

tation basée régions[9] et qui utilise un système multi-agents avec trois castes d'agents afin d'identifier les différents tissus d'un cerveau humain à partir d'images IRM 3D. La première caste n'est représentée que par un individu et permet un contrôle global. Un individu de la seconde est affecté à chaque voxel et calcule un modèle local. Enfin, chaque agent attribué aux voxels crée pour ces derniers un agent de la troisième caste pour chaque type de tissu connu.

Dans la section suivante, nous nous intéresserons particulièrement à la méthode présentée par C. Bourjot *et al* [3]. Cette méthode s'inspire du comportement d'araignées sociales, *Anelosimus Eximius*, afin de détecter des régions dans une image.

1.4 Méthodes des araignées

Nous allons présenter maintenant la méthode de segmentation que nous utiliserons et améliorerons par la suite. Nous verrons les différentes composantes de cette méthode, son fonctionnement puis les problèmes qui se posent.

1.4.1 Vue d'ensemble

L'article de C. Bourjot [3] présente une nouvelle méthode de segmentation d'image. Cette méthode est basée régions, c'est à dire qu'elle va détecter les régions de l'image. Elle s'inspire pour cela du comportement d'araignées sociales, Anelosimus Eximius, afin de créer un système multi-agents qui aura pour but la segmentation de l'image.

Les agents seront donc les araignées. Ces araignées évolueront dans un environnement créé à partir de l'image à segmenter. Elles possèdent un cycle de vie qui décrit les actions qu'elles doivent effectuer. Le système ainsi défini possède aussi un cycle de vie qui consiste à exécuter le cycle de vie de chaque araignée (cf. figure 1.1).

1.4.2 Description des entités

Environnement

L'environnement est une matrice de pixels en niveau de gris. Chaque pixel est une position potentielle pour une araignée et permettra à cette dernière d'atteindre d'autres pixels grâce au voisinage qui est défini. Ce voisinage est composé de deux ensembles comme le montre la figure 1.2 :

[9] on entendra par *basée régions* que le but de la méthode est de détecter les régions de l'image

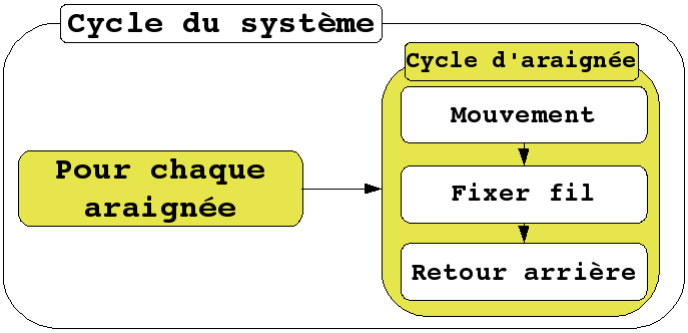


FIG. 1.1 – Vue d'ensemble du système

– un ensemble statique de pixels définissant le voisinage direct. Ce voisinage est 8-connexe. Cet ensemble sera nommé `Neigh` par la suite.
– un ensemble dynamique de pixels qui pourra s'agrandir au fur et à mesure que les araignées tisseront des fils entres les pixels. Cet ensemble sera nommé `SCuts`.

L'union de ces deux ensembles sera notée `Access`. La figure 1.2 présente le voisinage d'un pixel. Les pixels gris foncé représente l'ensemble `Neigh`, cet ensemble est défini à la création du l'environnement et restera identique tout au long des itérations du système. Les pixels gris clair représente l'ensemble `SCuts`. Cet ensemble est constitué des extrémités des fils reliés au pixel considéré. Cet ensemble est donc vide à l'itération zéro, ce sont les araignées qui augmenteront cet ensemble en tissant des fils à chaque itération du système. Les pixels noirs représentent les pixels présente à la fois dans `Neigh` et dans `SCuts`.

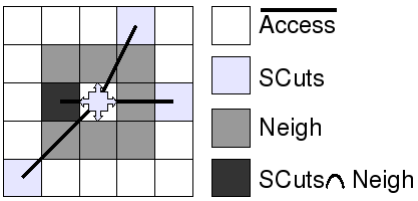


FIG. 1.2 – Voisinage d'un pixel

Araignée

Une araignée est constituée d'un état interne et d'un ensemble de fonctionnalités. L'état interne comprend une position courante et la dernière position où l'araignée a tissé. Les fonctionnalités dont dispose l'araignée sont les suivantes :

1. mouvement,
2. tissage d'un fil entre deux pixels,
3. retour à la dernière position où il y a eu tissage.

L'ensemble des araignées est partitionné en sous-ensembles que l'on appellera colonies. Chaque colonie est associé aux caractéristiques d'une région qui sera détectée par les araignées de cette colonie. Il y a donc autant de colonies que de régions à détecter.

Le cycle de vie d'une araignée consiste, comme le montre la figure 1.1, à exécuter successivement chacune des fonctionnalités de l'araignée, dans le même ordre que celui donné précédemment.

Nous allons maintenant détailler ces trois fonctionnalités.

Mouvement Le mouvement permet à l'araignée de se déplacer dans l'environnement. Cette fonctionnalité consiste à mettre à jour la position de l'araignée avec un pixel du voisinage de la position courante.

Le choix de la nouvelle position est d'abord dépendant du nombre de colonies présentes dans l'environnement : une colonie ou plusieurs.

Dans le cas où une seule colonie est présente, l'araignée doit choisir entre se déplacer vers un pixel de `Neigh` ou vers un pixel de `SCuts`. La probabilité de choisir `SCuts` est un paramètre de la colonie, nommé `pdraglines`.
Ensuite, chaque pixel de l'ensemble choisi aura la probabilité suivante d'être choisi :

- si l'ensemble est `Neigh`, tous les pixels ont la même chance d'être selectionnés,
- si l'ensemble est `SCuts`, chaque pixel p a la probabilité $\frac{draglines_p}{draglines_{all}}$ d'être selectionné, où $draglines_p$ correspond au nombre de fils allant de la position courante à p et $draglines_{all}$ est le nombre de fils total partant de la position courante.

Dans le second cas, on utilisera une fonction $w()$ permettant d'associer un poids à chaque pixel. Plus le poids d'un pixel est élévé, plus la probabilité que ce pixel soit choisi sera grande. Pour chaque pixel p, la probabilité $\mathscr{P}$ de se déplacer vers ce pixel est

$$\mathscr{P} = \frac{w(p)}{\sum_{a \in \texttt{Access}} w(a)}$$

Cette fonction de poids va permettre de favoriser les pixels en fonction de leur appartenance à `Neigh` ou `Scuts`, et dans le cas de `Scuts`, des fils reliant la position courante à ce pixel. Si $p \in$ `Neigh`, alors $w(p)$ est constant, sa valeur étant définie par le paramètre `constantwp` de la colonie. Autrement, si $p \in SCuts$, on distinguera le nombre de fils $draglines_{self}$ tissés par la colonie de l'araignée considérée entre p et la position courante, et le nombre

de fils $draglines_{other}$ tissés par les autres colonies.

$$\begin{aligned} self &= \texttt{attractselft} \times F(draglines_{self}) \\ other &= \texttt{attractother} \times F(draglines_{other}) \\ w(p) &= self + other \end{aligned}$$

`attractself` et `attractother` sont des paramètres de la colonie qui définissent l'attraction que produisent les fils sur l'araignée. Nous avons vu que nous distinguons deux types de fils, chaque type pouvant avoir son propre facteur d'attraction sur l'araignée. Ainsi une araignée pourra être plus attirée par les fils de sa colonie, ou au contraire préférer ceux des autres colonies. Ce mécanisme permet de définir dans quelle proportion l'araignée peut utiliser les fils des autres colonies afin d'explorer l'environnement.

Tissage Le tissage permet de créer un lien entre la position courante et le dernier pixel où un fil a été tissé. Il dépend de deux paramètres de la colonie de l'araignée : `reflevel` et `selectivity`. La probabilité de tisser suit une loi normale dont la moyenne est `reflevel` et la variance `selectivity`, deux paramètres de la colonie. `reflevel` définit le niveau de gris moyen de la région détectée par la colonie.

Retour arrière Le retour arrière consiste à retourner à la dernière position où un fil a été tissé. La probabilité d'un retour arrière est définie par le paramètre `backprobability` de la colonie.

La méthode des araignées prend donc plusieurs arguments en entrée. Il faut d'abord connaitre le nombre de colonies que l'on souhaite utiliser puis pour chaque colonie il est nécessaire de connaître les paramètres suivant :

- `reflevel` défini l'intensité de référence des pixels de la région que la colonie doit détecter,
- `selectivity` permet de définir la variance de la loi normale utilisée lors du tissage,
- `attractself` défini l'attraction exercée par les fils de la colonie sur les araignées de cette colonie,
- `attractother` défini l'attraction exercée par les fils des autres colonie sur les araignées de cette colonie,
- `backprobability` défini la probabilité d'un retour arrière,
- `constantwp` permet d'attribuer un poids au pixel du voisinage direct (`Neigh`) lors du mouvement,
- `saturationvalue` défini le poids maximum d'un pixel du voisinage indirect (`SCuts`).

Il est possible de séparer ces paramètres en deux catégories : ceux définissant les caractéristiques de la régions à détecter (`reflevel`, `selectivity`) et ceux définissant la façon dont la région sera détectée (tous les autres).

1.4.3 Problématique

La méthode des araignées possède l'avantage d'être une méthode reposant sur des mécanismes simples facilitant sa compréhension et son implantation. Cependant, la méthode telle que décrite dans [3] soulève plusieurs questions importantes :

- comment déterminer l'ensemble des paramètres d'entrée,
- comment déterminer le nombre d'itérations nécessaires ou fixer une condition d'arrêt.

Nous avons vu que chaque colonie possède 7 paramètres. Pour pouvoir procéder à la segmentation d'une image il faut donc d'abord déterminer le nombre de colonies à utiliser, puis les paramètres de chaque colonie. Le calcul des paramètres peut poser problème si celui-ci est fait de façon manuelle. En effet, nous effectuerons par la suite des mesures sur les résultats de segmentations par la méthode des araignées. Afin de rendre ces mesures pertinentes, il est important que le calcul des paramètres repose sur une méthode explicite.

Pour les mêmes raisons, il est important que la condition d'arrêt puisse être explicité. Dans le cas contraire, les résultats pourraient être remis en cause : un nombre d'itérations insuffisant pourrait entrainer une analyse qualitative mauvaise, de même qu'un nombre d'itérations excessif pourrait augmenter le temps d'exécution de la méthode sans améliorer la qualité du résultat.

Chapitre 2

Segmentation des araignées et premiers résultats

Nous commencerons par résoudre les problèmes liés à la méthode des araignées. Nous proposerons une méthode permettant de calculer certains paramètres essentiels de la méthode qui permettra de segmenter une image sans prérequis. Puis nous proposerons une condition d'arrêt qui permettra de se dégager d'un nombre d'itérations spécifié par l'utilisateur.

Nous comparerons ensuite la méthode des araignées à des méthodes *classiques*. Ces comparaisons ne seront pas exhaustives dans le sens où elles ne prendront pas en compte tous les aspects de la segmentation. Ces comparaisons nous permettront de déterminer l'efficacité de la méthode des araignées.

Pour terminer, nous étendrons la méthode des araignées aux images en trois dimensions. La méthode sera utilisée telle quelle hormis une modification concernant le voisinage des voxels nécessaire au passage en trois dimensions. Il n'y aura donc pas d'optimisation spécifique à la 3d dans ces résultats.

2.1 Résolution des problèmes

2.1.1 Détection automatique des paramètres

La méthode des araignées possède plusieurs paramètres qui vont définir les régions qui vont être détectées. Ces paramètres sont présentés dans le tableau 2.1. Il faut d'abord connaître le nombre de colonies (le nombre de types de régions) et les propriétés de chacune en particulier le niveau de gris de référence et le paramètre `selectivity`. Nous avons vu que le calcul de ces paramètres de façon empirique pouvait être un obstacle à la comparaison de résultats de la méthode des araignées : si les paramètres ne sont pas calculés de la même façon dans chaque cas, les comparaisons deviennent

Paramètre	Description
`reflevel`	niveau de gris de référence de la colonie
`selectivity`	permet de définir la variance de la loi normale utilisée lors du tissage
`attractself`	attraction vers les fils de sa colonie
`attractother`	attraction vers les fils des autres colonies
`backprobability`	probabilité d'un retour arrière
`constantwp`	poids d'un pixel du voisinage direct
`saturationvalue`	poids maximum d'un pixel du voisinage indirect

TAB. 2.1 – Paramètres d'une colonie

peu fiables car dépendantes de la méthode de calcul. Nous allons donc voir dans cette partie une méthode permettant de déterminer les paramètres de la simulation. Cette méthode sera utilisée par la suite dans toutes les segmentations.

Chaque colonie possède un niveau de gris (ou intensité) de référence dont vont se servir les araignées de la colonie afin de déterminer si elles doivent fixer un fil. Nous avons alors deux possibilités :

1. déterminer "manuellement" les intensités, ce qui se révèle peu intéressant hormis dans le cas où les intensités sont connues de l'utilisateur,
2. utiliser une méthode permettant de déterminer automatiquement ces intensités.

La détection automatique va permettre d'obtenir des paramètres optimaux pour l'image sans imposer à l'utilisateur de connaissances sur l'image.

Par la suite, nous nommerons degré d'une intensité le nombre de pixels/voxels de cette intensité.

Nous allons utiliser l'histogramme de l'image afin de déterminer les intensités présentant un intérêt à être détectées. En effet, une région dans une image se traduit en général par un pic plus ou moins important dans l'histogramme de l'image. Nous allons donc détecter successivement chaque pic, en éliminant les intensités composant le pic des futurs choix possibles. La figure 2.1 montre la détection des maxima (sommets des pics) de l'histogramme :

`A :` on détecte l'intensité n de degré maximal de l'histogramme,

`B :` on détecte les intensités voisines v telles que
- si $v < n$ alors $v + 1$ est détectée et $degre(v) \leq degre(v + 1)$
- si $v > n$ alors $v - 1$ est détectée et $degre(v) \leq degre(v - 1)$

`C :` on marque les intensités détectées afin de ne pas les utiliser par la suite,

`D :` on recommence l'étape A jusqu'à ce que toutes les intensités aient été marquées.

Cette méthode nous permet d'obtenir la liste des maxima de l'histogramme. Cependant cette méthode est particulièrement sensible au bruit lors de

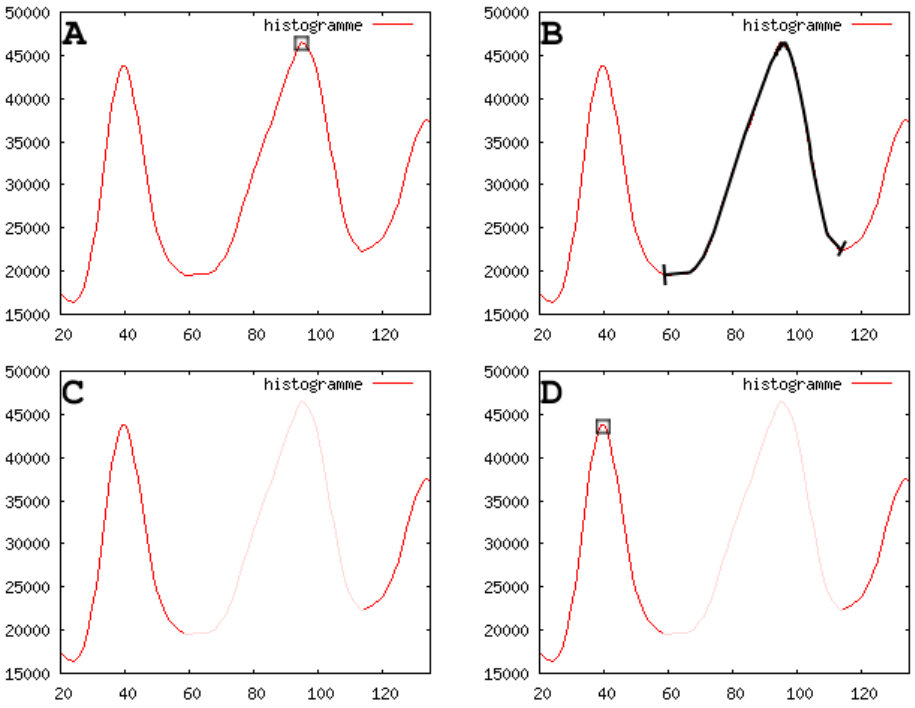


FIG. 2.1 – Détection des maxima de l'histogramme

l'étape `B`. Nous allons donc introduire une méthode permettant de réduire les effets du bruit sur l'histogramme en lissant ce dernier et permettant ainsi d'obtenir un nombre de maxima représentatif du nombre de classes d'intensités de l'image.

Lissage de l'histogramme

Le lissage va permettre de réduire les pics provoqués par le bruit.

La méthode proposée ici s'effectue par itération. A chaque itération, le degré d'une intensité n devient la moyenne des degrés des intensités $n-1, n, n+1$ (cf. annexe C).

Nous proposons ici une méthode permettant de détecter automatiquement les paramètres optimaux pour la segmentation de l'image. La méthode permet de déterminer le nombre de colonies ainsi que les paramètres `reflevel` qui détermine l'intensité de référence associée à la colonie, `selectivity` qui permet d'agir sur la probabilité de tisser un fil, et le nombre d'araignées de chaque colonie.

Nous allons d'abord déterminer les maxima de l'histogramme de la façon décrite ci-dessus. L'histogramme sera lissé jusqu'à ce que la diminution du

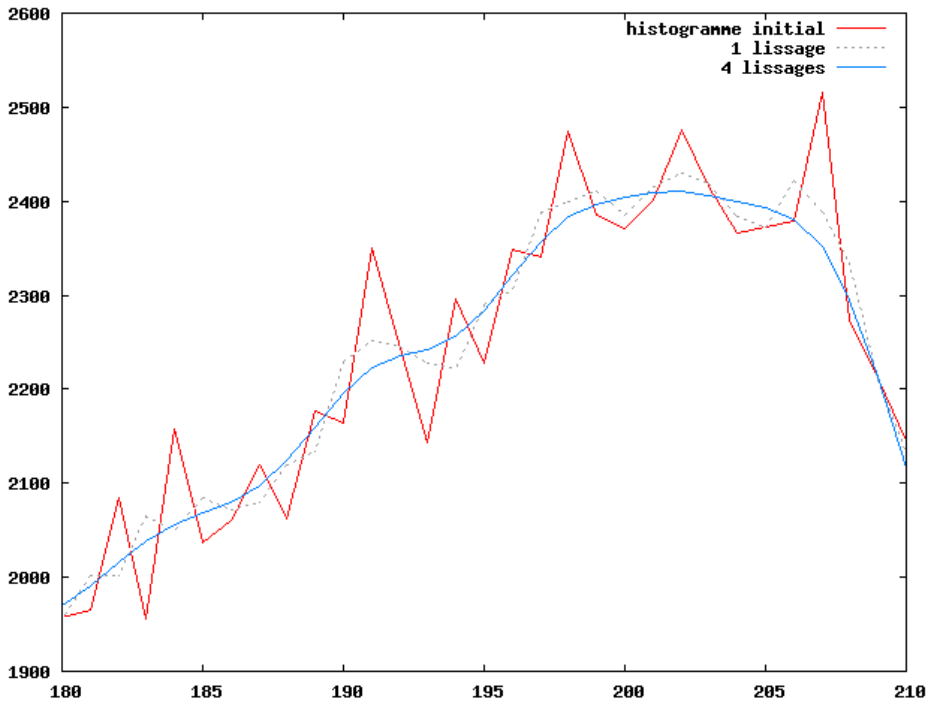


FIG. 2.2 – Lissage progressif de l'histogramme

nombre de maxima soit significative. En effet, les maxima provoqués par le bruit se situent sur des pics de l'histogramme dont la pente est peu importante et qui seront effacés par le lissage en peu d'itérations. Les maxima représentant une région se situent sur des pics de pente plus importante qui nécessitent un nombre de lissages plus important afin d'être effacés. Cependant, le risque d'effacer un maximum *intéressant* n'est pas nul.

Afin de ne pas tomber dans des aberrations, nous ajouterons la condition que le nombre de maxima, et donc le nombre de régions à détecter, doit être supérieur à 2. En effet, l'image contient au minimum un fond et un premier plan.

Nous obtenons une suite de maxima $M_1, M_2, \ldots, M_k$. Le nombre de maxima, k, détermine le nombre de colonies d'araignées qui vont être utilisées. Chaque maximum correspond au paramètre `reflevel` de la colonie. Nous allons ensuite partitionner les intensités en autant de classes que de maxima détectés. Pour cela, il suffit de chercher l'intensité de degré minimal entre deux maxima. Nous obtenons ainsi une suite de minimums $m_1, m_2, \ldots, m_{k-1}$ de telle sorte que $0 < M_1 < m_1 < \ldots < m_{k-1} < M_k < l$ où l est l'intensité maximale. Les k classes d'intensités obtenues sont alors $[0; m_1]$, $]m_1; m_2]$, $\ldots$, $]m_{k-1}; l]$.

La variance de chaque classe nous fournit le paramètre `selectivity` de chaque colonie correspondante. Le nombre d'araignées par colonie peut être adapté en fonction de la somme des degrés de la classe.

Nous pouvons désormais déterminer les paramètres "vitaux" de la méthode des araignées. Il reste à déterminer les nombres d'araignées par région, ainsi que l'attraction des fils de soie sur les araignées.

2.1.2 Condition d'arrêt

Nous avons vu que la méthode des araignées possède un cycle de vie qui est itéré un certain nombre de fois afin de segmenter l'image au fur et à mesure. Le nombre d'itérations va influencer deux points importants du résultat de la segmentation :

1. la qualité de la segmentation,
2. le temps d'exécution nécessaire pour aboutir à ce résultat.

En effet, à chaque itération les araignées vont tisser des fils entre les pixels qui seront utilisés pour déterminer à quelle région appartient le pixel. Si le nombre d'itérations n'est pas suffisamment important, le nombre de pixels n'appartenant à aucune région sera important et le résultat sera de mauvaise qualité. Au contaire, si le nombre d'itérations est trop important, les araignées ne feront que renforcer au bout d'un certain temps les fils déjà existants sans apporter de nouvelles informations. Ce dernier point aura pour effet un temps d'exécution plus long pour une qualité de résultat identique.

De même que pour les valeurs des paramètres, le calcul du nombre d'itérations est un point important pour obtenir des mesures sur les résultats qui soient pertinentes. Plutôt que de fixer un nombre d'itérations, il est possible de déterminer une condition d'arrêt qui sera vérifiée avant d'itérer une nouvelle fois le système. Nous allons voir la condition d'arrêt que nous utiliserons par la suite.

Pour simplifier la lecture, nous appelerons β le nombre de fils tissés entre des pixels dont le degré est nul lors d'une itération.

Le résultat de la segmentation des araignées dépend des fils qui auront été tissés entre les pixels. Lors d'une itération, lorsque β tend vers zéro, on peut considérer que le système se stabilise les araignées ne font que renforcer les fils déjà existant.

La figure 2.3 montre l'évolution de β lors d'une segmentation.

Il est possible de détecter le moment où β reste à zéro. Ce moment détermine l'arrêt de la simulation. On peut améliorer cette condition en ajoutant deux paramètres :

– un seuil pour β qui détermine quand β peut être considéré comme nul.
– le nombre de β nuls autorisés avant l'arrêt de la simulation.

Le seuil peut être conditionné par le nombre d'araignées. En effet, lors d'une itération, chaque araignée à la possibilité de tisser un fil. Le nombre de fils tissés lors d'une itération est donc borné par le nombre d'araignées.

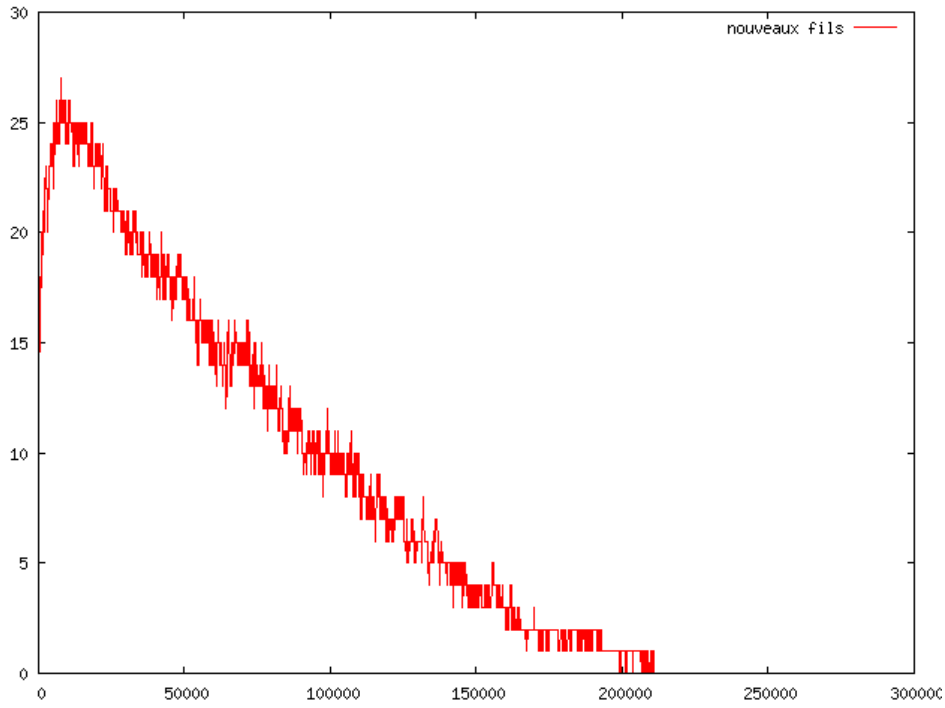


FIG. 2.3 – Nombre de fils tissés lors d'une simulation

2.1.3 Construction de l'image segmentée

Les araignées permettent de définir une mesure par colonie c : $d_{p,c}$ (degré) sur chaque pixel p. On peut alors définir le degré global d'un pixel comme la somme des degrés pour chaque colonie.

$$d_p = \sum_{c \in colonies} d_{p,c} \tag{2.1}$$

On peut donc déterminer pour chaque pixel la colonie dominante, c'est à dire la colonie c tel que $d_{p,c} = \max_{v \in colonies}(d_{p,v})$.

Une première image résultat peut être créée de la façon suivante :

1. attribution d'une couleur à chaque colonie,
2. on définit une couleur pour les pixels de degré nul,
3. on colorie chaque pixel avec la couleur de sa colonie dominante.

Cette méthode permet d'obtenir un résultat ayant un défaut : les araignées peuvent avoir détecté des régions non connexes ayant les mêmes propriétés, on obtient alors une région composée de plusieurs composantes non connexes, comme nous le montre la figure 2.4.

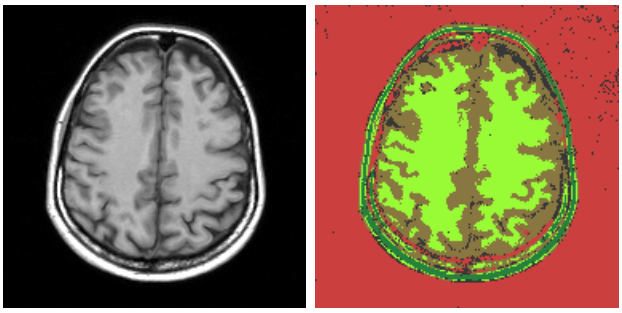


FIG. 2.4 – Image originale et résultat basique

Pour résoudre ce problème, nous allons utiliser un algorithme d'étiquetage en composantes connexes.

La figure 2.5 montre le résultat de cette étiquetage sur la segmentation des araignées.

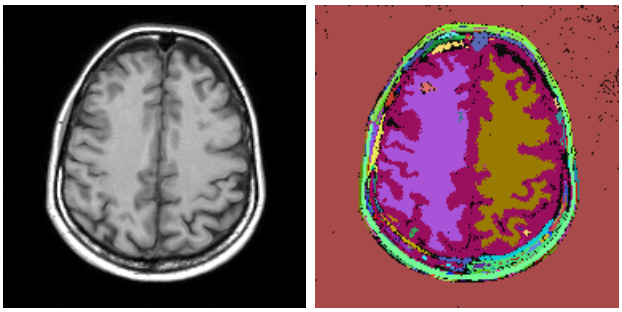


FIG. 2.5 – Image originale et résultat avec régions connexes

Nous disposons désormais d'une représentation de l'image segmentée que nous pourrons utiliser pour comparer la méthode des araignées avec d'autres méthodes de segmentation.

2.2 Comparaison avec les méthodes existantes

Dans cette partie, nous allons comparer la méthode des araignées avec d'autres méthodes de segmentation. Ces comparaisons vont permettre de déterminer l'efficacité de la méthode des araignées par rapport aux méthodes de segmentation classiques.

Nous allons utiliser pour ces comparaisons deux autres méthodes :
– une méthode de classification par seuillage : la méthode de Otsu[12]
– une méthode basée régions : la croissance de région (Region Growing)
Ces deux méthodes seront décrites en 2.2.3.

La comparaison des méthodes nécessite de mettre en place des critères de comparaison qui seront utilisés sur l'ensemble des images de tests. Nous allons comparer les résultats sur plusieurs points :

1. nombre de régions
2. correspondance entre les régions du modèle utilisé (dans le cas d'images synthétiques) et du résultat de la segmentation
3. temps d'exécution

Pour plus de clarté, nous désignerons par $\mathscr{M}$ l'image modèle et par $\mathscr{R}$ l'image résultant de la segmentation. $\mathscr{M}_i$ correspond à la région i du modèle et $\mathscr{R}_j$ correpond à la région j du résultat.

Le nombre de régions va nous permettre de déterminer si la méthode considérée détecte un nombre de régions proche de la réalité. Une région sera considérée non-significative si sa taille est inférieure à 10 pixels. Dans le cas d'images bruitées, il est possible que certaines méthodes détectent des régions non-significatives. C'est pourquoi nous indiquerons en supplément du nombre total de régions, le nombre de régions de taille non-significative. Le calcul du nombre de régions s'effectuera sur le résultat de la méthode de segmentation à laquelle s'ajoutera un traitement d'étiquetage en composantes connexes du résultat afin de considérer les régions connexes.

La correspondance entre modèle et résultat va nous permettre de déterminer si les régions détectées par la méthode correspondent aux régions définies dans le modèle. Ceci n'est possible que dans le cas d'images de synthèse pour lesquelles le modèle est connu. Cette correspondance sera nommée *précision.*

Pour calculer cette précision, il faut déterminer quelle région $\mathscr{R}_j$ correspond le mieux à la région $\mathscr{M}_i$. Cette région est déterminée à partir de deux mesures :
– n^i représente le nombre total de pixels/voxels de $\mathscr{M}_i$,
– n_j représente le nombre total de pixels/voxels de $\mathscr{R}_j$,
– n_j^i représente le nombre de pixels/voxels communs $\mathscr{M}_i$ et $\mathscr{R}_j$.
Il est alors possible de calculer les proportions $\delta_{i,j} = n_j^i/n^i$ et $\gamma_{i,j} = n_j^i/n_j$ qui représentent respectivement la proportion de pixels/voxels de $\mathscr{M}_i$ appartenant à $\mathscr{R}_j$ et la proportion de pixels/voxels de $\mathscr{R}_j$ qui appartenant à $\mathscr{M}_i$. Nous avons alors deux façons de choisir la région $\mathscr{R}_j$ qui correspond le mieux à $\mathscr{M}_i$:

1. $\mathscr{R}_k$ telle que la valeur de $\delta_{i,k}$ soit maximale : on privilégie dans ce cas uniquement le fait que $\mathscr{M}_i$ et $\mathscr{R}_j$ ont un maximum de pixels/voxels en commun,

2. $\mathscr{R}_k$ telle que la somme $\delta_{i,k}+\gamma_{i,k}$ est maximale : de même que précédemment, mais on ajoute la condition que $\mathscr{R}_k$ doit avoir un minimum de ses pixels/voxels dans d'autres régions que $\mathscr{M}_i$.

Dans nos résultats, nous indiquerons deux précisions, $precision_\delta$ et $precision_{\delta+\gamma}$, qui correspondront respectivement aux deux choix de $\mathscr{R}_k$ décrits ci-dessus.

Dans les deux cas, la précision sera la moyenne des valeurs de δ pour l'ensemble des régions du modèle.

Les temps d'exécution qui seront donnés proviennent de l'exécution des méthodes sur une machine équipée de deux Intel(R) Xeon(R) E5345 (8 cores au total à 2.33GHz) ainsi que de 8Go de mémoire vive. Le système d'exploitation de cette machine est Linux avec un noyau 2.6.21 x86_64.

2.2.1 Présentation des images de tests

Nous allons maintenant présenter les images qui seront utilisées pour les comparaisons. Nous utiliserons une image synthétique basique (figure 2.6) représentant plusieurs formes géométriques ainsi que du texte. Cette image sera utilisée dans le but de vérifier le bon fonctionnement des méthodes. C'est pourquoi nous ajouterons ensuite à cette image un bruit normallement distribué[1] dont le niveau peut atteindre 20% (figure 2.7) afin d'étudier la résistance au bruit de la méthode des araignées. Ces deux images ont une dimension de 256×256 pixels. Les résultats concernant cette image sont présentés dans la sous-partie 2.2.4.

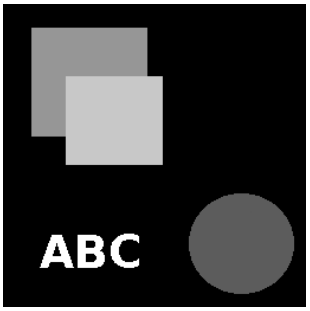


FIG. 2.6 – Image synthétique de test

Nous utiliserons ensuite le modèle BRAINWEB.

[1] normally distributed noise

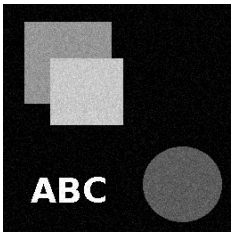


FIG. 2.7 – Image synthétique bruitée de test

2.2.2 BrainWeb

BRAINWEB[2] est un simulateur d'acquisition d'images cérébrales par IRM. Il permet, entre autres, d'aider à la validation d'algorithmes de segmentation. En effet, l'image est obtenue à partir d'un modèle dont on connait les différentes composantes. Une coupe de ce modèle est présentée figure 2.8.

FIG. 2.8 – Modèle BRAINWEB (coupe)

On peut distinguer dix éléments dans ce modèle qui sont répertoriés dans la table 2.2.

Les paramètres du modèle qui sera utilisé par la suite sont les suivants :
- modality : T1,
- slice thickness : 1mm,
- noise : 3%,
- intensity non-uniformity : 20%,
- dimensions : $181 \times 217 \times 181$ voxels.

Nous allons d'abord segmenter une coupe 2d de cette image (figure 2.9) puis nous segmenterons cette image en trois dimensions. Cette coupe correspond à la 90^{eme} slice de l'image. Cette image est utilisée afin de déterminer si la méthode est capable de segmenter des images présentant des formes plus complexes ce qui nous permettra de déterminer s'il est envisageable de

[2] `http://www.bic.mni.mcgill.ca/brainweb/`

élément	pourcentage		élément	pourcentage	
background	42.2%		muscle skin	8.7%	
csf	5.2%		skin	10.2%	
grey matter	12.7%		skull	5.1%	
white matter	9.5%		glial matter	0.1%	
fat	2.1%		connective	4.2%	

TAB. 2.2 – Modèle Brain Web : les différentes composantes

segmenter des images en trois dimensions.

FIG. 2.9 – Coupe 2d du modèle BRAINWEB (181 × 217px)

2.2.3 Présentation des autres méthodes

Croissance de région (Region Growing)

La méthode *croissance de région* consiste à construire une région à partir de pixels de base, puis en ajoutant récursivement les voisins dont la différence de niveau de gris avec les pixels initiaux est inférieure à un certain seuil. L'algorithme utilisé pour la croissance de région est disponible en annexe (A.2).

Cette méthode consiste à faire croître une région initiale en ajoutant à cette région les pixels qui n'appartiennent à aucune région mais appartenant au voisinage des pixels déjà présents dans la région et dont le niveau de gris

est suffisamment proche de celui de la région. Lorsqu'il n'est plus possible d'ajouter des pixels, on crée une nouvelle région avec un pixel qui n'a pas encore été sélectionné puis on fait croître cette région.

La méthode se termine quand tous les pixels sont dans une région.

Méthode de Otsu

La méthode de Otsu est une méthode de seuillage multi-niveaux. Elle consiste à déterminer, pour un nombre de régions donné, les valeurs optimales des différents seuils en se basant sur la variance des subdivisions ainsi créées.

La méthode de base consiste à séparer le premier plan du fond. On cherche dans ce cas le seuil optimal pour partager les pixels en deux régions. Pour un seuil t, il est possible de calculer la *variance inter-classes*[3] $\sigma^2(t)$. Cette mesure est obtenue à partir des intensitsé moyennes μ_1, μ_2 et μ des classes $[0;t]$, $[t+1;L]$ et $[0;L]$ où L représente l'intensité maximale. L'équation 2.2 montre le calcul de σ^2, où ω_1 et ω_2 représente respectivement la proportion de pixels dans la classe $[0;t]$ et $[t+1;L]$ par rapport au nombre total de pixels.

$$\sigma^2(t) = \omega_1(t)(\mu_1(t) - \mu)^2 + \omega_2(t)(\mu_2(t) - \mu)^2 \tag{2.2}$$

Otsu montre que le seuil optimal t^* est obtenu pour une variance inter-classe maximale. La méthode consiste donc à calculer cette variance pour tous les seuils possibles ($t \in \{1, \ldots, L-1\}$) et d'en déterminer la valeur maximale.

Cette méthode s'étend facilement au calcul de M classes avec $M-1$ seuils $\{t_1, t_2, \ldots, t_{M-1}\}$ ($t_1 < t_2 < \ldots < t_{M-1}$). La variance inter-classes est alors définie de la façon suivante :

$$\sigma^2(t_1, \ldots, t_{M-1}) = \sum_{k=1}^{M} \omega_k(\mu_k - \mu)^2 \tag{2.3}$$

où ω_k représente la proportion de pixels dans la classe $[t_{k-1}; t_k]$[4], μ_k l'intensité moyenne de cette même classe et μ l'intensité moyenne de la classe $[0;L]$.

On calcule pour chaque $M-1$-uplet de seuils la variance inter-classe correspondante. Les seuils optimaux, $(t_1^*, \ldots, t_{M-1}^*)$, correspondent à la valeur maximale de la variance inter-classe :

L'article de Ping-Sung Liao, et al. propose un algorithme qui minimise le nombre de calculs nécessaires, permettant d'obtenir un algorithme plus rapide [13]. L'implémentation donnée en annexe A.3 repose sur cet algorithme.

[3] between-class variance

[4] $t_0 = 0$ et $t_M = L$

	Régions	Régions $> 10px$	Précision$_\delta$	Precision$_{\delta+\gamma}$	Exécution
Araignées	11	11	99.75%	99.75%	312 s
Region Growing	10	10	100%	100%	0.5 s
Otsu	10	10	100%	100%	0.5 s

Tab. 2.3 – Comparaison des résultats pour l'image synthétique

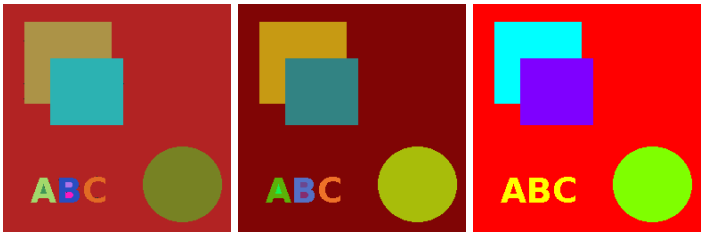
2.2.4 Segmentation d'image synthétique

Dans cette sous-partie, nous allons présenter les résultats des différentes segmentations de l'image synthétique présentée dans la figure 2.6. Dans un premier temps, nous verrons les résultats de la segmentation sur l'image non bruitée afin de nous assurer du bon fonctionnement des méthodes, puis nous verrons les résultats sur l'image bruitée afin de déterminer la résistance au bruit de la méthode des araignées.

L'image est composée de 10 régions connexes.

Image non bruitée

La figure 2.10 nous montre les résultats des différentes segmentations. Le tableau 2.3 donne les valeurs des mesures effectuées sur ces résultats.



a b c

Fig. 2.10 – Segmentations d'image synthétique : a) araignées, b) croissance de région, c) Otsu

Les résultats des méthodes croissance de région et Otsu ont une précision maximale avec un nombre de régions qui correspond à celui du modèle. Les temps d'exécution de ces méthodes sont faibles.

L'image résultante de la segmentation des araignées possède une région supplémentaire qui correspond aux pixels qui n'ont été détectés par aucune colonie. Cette région n'est pas connexe, les pixels qui la composent sont éparpillés dans l'image. Il est donc envisageable d'effectuer un post-traitement qui permettrait de rattacher ces pixels à la colonie la plus présente dans leurs voisinages.

Bien que la différence soit faible, la précision du résultat de la méthode des araignées est moins bonne que celles des deux autres méthodes. Cepen-

	Régions	Régions $> 10px$	Précision$_\delta$	Precision$_{\delta+\gamma}$	Exécution
Araignées	3741	274	86.6%	86.6%	512 s
Region Growing	185	172	82.94%	82.94%	0.7 s
Otsu	1615	266	80.8%	76.8%	0.5 s

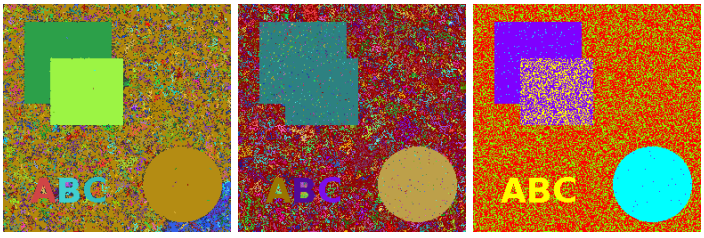
Tab. 2.4 – Comparaison des résultats pour l'image synthétique bruitée

dant, comme la méthode des araignées est une méthode stochastique, nous ne nous attendons pas à obtenir une précision maximale. En revanche, cette précision doit rester stable lors de l'ajout de bruit à l'image (cf. figure 2.7).

En ce qui concerne le temps d'exécution de la méthode des araignées, celui-ci est 600 fois plus important que les autres méthodes pour une précision qui est inférieure ce qui peut constituer un problème pour l'utilisation de cette méthode.

Image bruitée

Nous allons utiliser l'image présentée figure 2.7 qui ajoute un bruit à l'image précédente. Les résultats des segmentations sont présentés dans la figure 2.11. Les mesures utilisées pour les comparaisons sont présentées dans le tableau 2.4.



a b c

Fig. 2.11 – Segmentations d'image synthétique bruitée : a) araignées, b) croissance de région, c) Otsu

L'ajout d'un bruit à l'image a provoqué une diminution de la précision des résultats des trois méthodes. Le résultat de la méthode des araignées possède la meilleure précision. De plus, la différence entre la précision pour l'image non-bruitée (tableau 2.3) et celle de l'image bruitée est minimale pour la méthode des araignées. Ces deux points nous permettent de dire que la méthode des araignées a été la moins sensible à l'ajout du bruit dans l'image.

Le nombre de régions a augmenté pour les trois méthodes par rapport à la segmentation de l'image non-bruitée. On remarque que c'est la méthode des araignées puis la méthode Otsu qui ont produit le plus de régions. Ces deux

	Régions	Régions $> 10px$	Précision$_\delta$	Precision$_{\delta+\gamma}$	Exécution
Araignées	4361	434	60.8%	54.9%	50 s
Region Growing	908	664	46.2%	21.1%	0.5 s
Otsu	1875	567	90.7%	89.3%	0.3 s

Tab. 2.5 – Comparaison des résultats pour la coupe BrainWeb

méthodes, contrairement à la croissance de région, ont produit un nombre important de régions non-significatives ce qui implique une forte sursegmentation de l'image. Ces régions non-significatives peuvent être éliminées à l'aide d'un post-traitement du résultat.

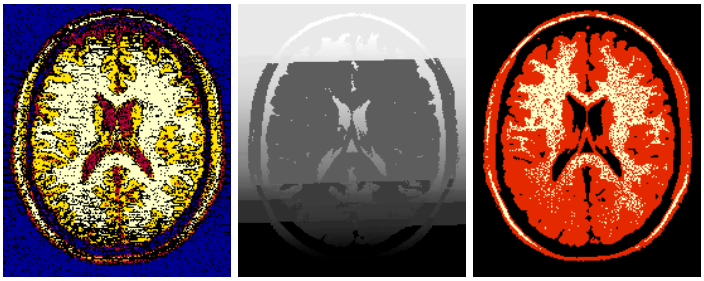
Les temps d'exécution des méthodes croissance de région et Otsu sont faibles (inférieur à 1s) et sont restés stables par rapport à la segmentation de l'image non-bruitée. La méthode des araignées, dont le temps d'exécution pour l'image non-bruitée était déjà important, nécessite cette fois-ci un temps d'exécution encore plus important. Cette augmentation du temps d'exécution est dû à un tissage plus progressif des fils dans l'image. En effet, dans l'image non-bruitée les intensités des pixels sont égales à une des intensités de référence (`reflevel`) des colonies. La probabilité qu'une araignée tisse un fil lorsqu'elle se trouve sur un pixel dont l'intensité est égale au paramètre `reflevel` de sa colonie est donc maximale. Dans le cas de l'image bruitée, les intensités des pixels sont plus ou moins éloignées des paramètres `reflevel` des colonies, par conséquent la probabilité pour une araignée de tisser un fil est réduite. Ce fait retarde le moment de la méthode où le nombre de fils tissés en une itération est suffisamment faible pour provoquer la condition d'arrêt prolongeant ainsi l'exécution de la méthode.

On peut donc retenir de cette segmentation que la méthode des araignées a mieux resisté au bruit que les autres méthodes. Cette résistance a cependant entrainé une sursegmentation de l'image ainsi qu'un temps de calcul plus long.

2.2.5 Coupe 2d de BrainWeb

Nous allons maintenant appliquer les différentes méthodes à une coupe 2d du modèle BrainWeb. Ce modèle est composé de 10 régions qui sont exposées dans la partie 2.2.2. Les paramètres liés à l'image produite à partir du modèle sont aussi exposés dans la partie 2.2.2. Les résultats de cette segmentation vont nous permettre de déterminer si la méthode des araignées supporte des images plus complexes, en particulier des images IRM. Les résultats sont présentés dans la figure 2.12.

La précision du résultat de la méthode des araignées a diminué par rapport aux segmentations précédentes où il s'agissait de formes simples. Cette diminution de la précision peut être due à la présence de régions de faible



a b c

FIG. 2.12 – Segmentations d'une coupe de BRAINWEB : a) araignées, b) croissance de région, c) OTSU

taille qui ne sont peu ou pas détectées. De plus, il est possible de certaines régions aient été détectées par une même colonie et que lors de la construction du résultat elles aient été fusionnées.

La méthode des araignées a produit le meilleur nombre de régions significatives, c'est à dire celui le plus proche du nombre réel de régions. Cependant, c'est aussi cette méthode qui a produit le plus grand nombre de regions non-significatives c'est à dire que les araignées ont sursegmenté l'image. Nous avons cependant vu qu'il est possible d'éliminer ces régions non-significatives à l'aide d'un post-traitement. Cette sursegmentation de l'image n'a pas été amplifiée par les formes plus complexes du modèle BRAINWEB.

Le passage à un type d'image plus aux formes plus complexes n'a pas affecté le temps d'exécution des méthodes croissance de région et OTSU. Le temps d'exécution de la méthode des araignées a diminué comparé aux tests précédents tout en restant supérieur aux temps d'exécution des deux autres méthodes. Cette diminution peut s'expliquer par un nombre de fils tissés moins important ce qui en réduisant le voisinage indirect des pixels réduit la complexité de la méthode. Le nombre de fils tissés est moins important car l'image est composée d'un plus grand nombre de régions, chaque région représentant donc une plus petite proportion de l'image. Les araignées mettent donc plus de temps (nombre d'itérations) à trouver des pixels correspondant à leur colonie.

2.3 Segmentation d'image 3D

Les araignées considèrent une relation de voisinage entre les pixels pour leurs déplacements. Dans l'article de C. BOURJOT ET AL. [3], les auteurs ne considèrent que des images à deux dimensions en utilisant un voisinage 8-connexe.

Il est possible d'étendre cette notion de voisinage pour les araignées en trois dimensions en utilisant un voisinage 26-connexe.

	Régions	Régions $> 10px$	Précision$_\delta$	Precision$_{\delta+\gamma}$	Exécution
Araignées	54575	13391	87.3%	87.3%	8 h
Region Growing	7866	1634	75.4%	46.7%	102 s
Otsu	8497	865	83.9%	83.9%	10 s

TAB. 2.6 – Résultat de la segmentation 3D

Nous allons donc étendre ce voisinage afin de segmenter des images en trois dimensions puis nous comparerons les résultats de la même façon que celle utilisée avec les images en deux dimensions dans la partie 2.2.

Segmentation de l'image 3D obtenue à partir de BrainWeb

Nous allons voir dans cette partie les résultats des différentes segmentations de l'image 3D obtenue à partir de BRAINWEB.

La précision dont fait preuve la méthode des araignées est comparable aux autres méthodes. On peut constater cependant un nombre de régions plutôt élevé qui peut s'expliquer par un nombre de voxels non-détectés plus important, entrainant une déconnexion des régions.

Cette précision a un coût important sur le temps d'exécution. Par conséquent, de même que pour les résultats sur les images à deux dimensions, nous obtenons un bon résultat avec un mauvais temps d'exécution. La méthode de OTSU, bien que souffrant d'une précision de quelques pourcents moins bonne, produit un bon résultat pour un temps de calcul environ 4000 fois moins long. De plus, le nombre de régions produites par les araignées est plus grand d'un facteur 10 comparé aux autres méthodes.

Conclusion et perspectives

Nous avons présenté dans ce mémoire une méthode qui présentait divers problèmes de mise en place. En effet, cette méthode nécessite un grand nombre de paramètres dépendant de l'image à traiter. De plus la méthode ne disposait pas de réelle condition d'arrêt ce qui imposait à l'utilisateur de fixer à l'avance le nombre d'itérations nécessaires.

Nous avons dans un premier temps résolu le problème du calcul des paramètres en proposant une méthode permettant de calculer automatiquement les paramètres. Cette méthode permet de déterminer les paramètres les plus importants de la méthode à savoir :

- le nombre de colonies,
- le paramètre `reflevel` de chaque colonie qui défini l'intensité de référence,
- le paramètre `selectivity` de chaque colonie qui régit la probabilité de tisser un fil.

Les autres paramètres ont été fixés car ils possèdent une importance moindre dans le résultat de la segmentation.

Nous avons ensuite vu une méthode basée sur le nombre de fils tissés lors d'une itération afin de déterminer l'arrêt de la méthode. Cette condition d'arrêt nous a permis d'éliminer le problème du nombre d'itérations nécessaires pour produire une bonne segmentation.

Nous avons effectué des comparaisons entre les résultats de la méthode des araignées, de la croissance de région et de la méthode Otsu. Ces comparaisons étaient axés sur la précision des méthodes, le nombre de régions produites ainsi que le temps d'exécution des méthodes. Ce ne sont pas des comparaisons exhaustives dans le sens où tous les aspects de la segmentation ne sont pas pris en compte.

Ces comparaisons nous ont permis de mettre en avant certains points de la méthode des araignées à améliorer. En particulier, nous avons remarqué que cette méthode produisait un nombre important de régions et que le temps d'exécution était plus long que celui des autres méthodes.

Cependant, la méthode des araignées repose sur une architecture qui est idéale pour être parallélisée. Cette méthode est en effet composée d'un ensemble d'agents qu'il est possible de répartir sur plusieurs processeurs.

Le fait que ces agents exécutent tous le même algorithme laisse envisager d'utiliser une architecture SIMD[5] et donc d'utiliser cette méthode sur des processeurs graphique (GPU).

Des prototypes de version multi-threads de la méthode des araignées ont été réalisés lors du développement de Jism. Ces prototypes ne sont pour le moment pas optimisés mais permettent de repartir l'exécution des araignées sur plusieurs processeurs.

Nous avons vu que le second défaut de la méthode des araignées est de produire une sursegmentation de l'image. Il est possible de résoudre ce problème en procédant à une fusion des régions insignifiantes afin de les rattacher à des régions de tailles significatives ou de les fusionner entre elles selon le cas. L'opération de fusion n'est pas un traitement coûteux en temps processeur, par conséquent l'ajout d'un tel post-traitement à la méthode des araignées ajouterait un temps de calcul négligeable comparé à celui de la méthode elle même.

Certaines solutions ont été envisagées afin d'améliorer la méthode. Il est envisageable de guider le déplacement des araignées à l'aide d'un gradient ou d'un laplacien. En effet, ces mesures permettraient d'obtenir des informations sur l'eventuelle présence de contours. Il serait alors possible d'utiliser les araignées de deux façons :

1. le gradient aurait un effet répulsif ce qui permettrait de *cloisonner* les araignées d'une colonie dans une région,
2. à l'inverse, le gradient pourrait avoir un effet attractif. Les araignées seraient dans ce cas utilisées dans le but de détecter les contours des régions.

Enfin, nous avons vu une première approche de la segmentation d'images en trois dimensions par la méthode des araignées. La méthode a été utilisée sans autre modification que l'extension du voisinage. Il n'y a donc aucune optmisation liée à l'ajout d'une dimension supplémentaire. Cependant, la segmentation du modèle BrainWeb a montré que la méthode était capable de produire un résultat avec une bonne précision.

Des optimisations sont nécessaires afin de réduire le nombre de régions produites et surtout afin de réduire le temps d'exécution de la méthode. Nous avons proposé des solutions qui pourraient permettre de résoudre ou au moins réduire ces défauts.

Ce mémoire a donc permis de résoudre certains problèmes de la méthode des araignées (calcul des paramètres et condition d'arrêt). Il a aussi permis de révéler certains défauts de cette méthode (sursegmentation, temps

[5]Single Instruction, Multiple Data

d'exécution important). Cependant, nous avons vu qu'il était envisageable de résoudre ou du moins réduire ces défauts. Nous avons enfin vu que la méthode des araignées pouvait être utilisée afin de segmenter des images en trois dimensions, bien que cette utilisation nécessite la résolution des défauts cités plus haut.

Annexe A

Implémentation

A.1 Algorithme des Araignées

A.1.1 Représentation de l'environnement

L'environnement est un ensemble de pixels. Pour une meilleure manipulation des pixels, cet ensemble se présente sous la forme d'une matrice. On peut par conséquent accèder directement au pixel via ses coordonnées. Le nombre d'objets `Pixel` créés sera relativement élévé (particulièrement en envisageant un passage à trois dimensions), il est donc important qu'un pixel ait le moins possible d'attributs sans que cela entraine des pertes de performances trop importantes. De plus (cf. figure A.1) l'environnement contient la liste des colonies et des araignées.

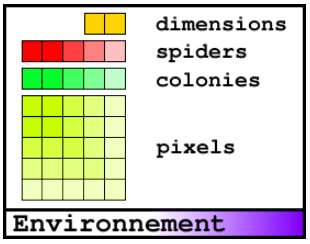


Fig. A.1 – Représentation de l'environnement

Comme le montre la figure A.2, un pixel est composé d'un niveau de gris, de ses coordonnées et de ses voisins. Les voisins sont repartis en deux catégories : voisinage direct (neigh) qui sont les pixels connexes à celui considéré et le voisinage indirect correspondant aux extrémités des fils de soie (draglines) connectés au pixel considéré.

L'objet `Dragline` associé à chaque voisin indirect contient les informations concernant le degré du fil et les colonies qui l'ont tissé. En effet, si un fil est déjà présent entre deux pixels et que l'on souhaite en ajouter un

nouveau, on ne rajoutera pas un nouveau fil mais on augmentera le degré de celui qui est présent. De même si plusieurs colonies tissent un même fil (i.e. tissent entre une même paire de pixels), on augmentera le degré propre à la colonie.

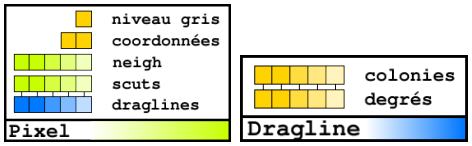


FIG. A.2 – Représentation des pixels et des fils

A.1.2 Araignée

Une araignée est composée de sa position courante, la dernière position où elle a tissé, sa colonie et l'environnement dans lequel elle évolue (cf. figure A.3).

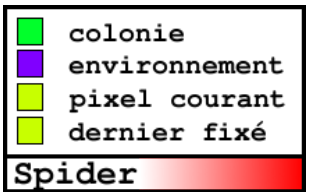


FIG. A.3 – Représentation des araignées

A.1.3 Pseudo-code

INPUT :
`pixels` : matrice de pixels
`width`,`height` : dimension de la matrice
`config` : paramètres de la méthode
`ite` : nombre d'itérations
BEGIN
`scuts [width][height]` : matrices des draglines
création des colonies et des araignées à partir de `config`
Tant que `ite` $-- > 0$, faire
Pour chaque araignée `s`, faire
mouvement(`s`,`pixels`,`scuts`)
silkfixing(`s`,`scuts`)
comeback(`s`)

Fin pour
Fin tant que
END
Complexité : $0(\texttt{ite} \times n_{araignees} \times (C_{mouvement} + C_{silkfixing} + C_{comeback})$

Procédure **mouvement()**
INPUT :
`s` : araignée cible
`pixels` : matrice de pixels
`scuts` : matrice des draglines
BEGIN
`weights [neightSize(s.position)+scutsSize(s.position)]` :
↪ poids des pixels du voisinage
Pour i $\in \{0, \ldots, neighSize(s.position) - 1\}$, faire
`weights [i]` = `s.colonie.constantwp`
Fin pour
Pour `i` $\in \{neighSize(s.position), \ldots, \texttt{weights}.size - 1\}$, faire
`weights [i]` = **fontionW**(`s`)
Fin pour
choisir un pixel dans le voisinage en fonction de `weights` et
↪ déplacer l'araignée sur ce pixel
END
Complexité : 0(nombre max de voisins)

Procédure **silkfixing()**
INPUT :
`s` : araignée cible
`pixels` : matrice de pixels
`scuts` : matrice des draglines
BEGIN
`p` = tirer un nombre au hasard ($[0, 1]$)
Si $p < gaussienne(s.position.level)$, faire
ajouter une dragline entre `s.position` et `s.lastFixed`
Fin si
END
Complexité : 0(1)

Procédure **comeback()**
INPUT :
`s` : araignée cible
BEGIN
`p` = tirer un nombre au hasard ($[0, 1]$)
Si $p < s.colonie.backprobability$, faire
`s.position` = `s.lastFixed`

Fin si
END
Complexité : 0(1)

A.2 Croissance de région

Algorithme de croissance de région implanté :

INPUT :
`pixels` : matrice de pixels
`width`,`height` : dimension de la matrice
`regions` : ensemble des régions (vide à l'entrée)
BEGIN
Pour `x` de 0 à `width`, faire
Pour `y` de 0 à `height`, faire
Si `pixels`[x][y] n'est dans aucune région, faire
GrowRegion(`pixels`[x][y],`regions`)
Fin si
Fin pour
Fin pour
END

Procédure **GrowRegion**
INPUT :
`start` : pixel initial
`regions` : ensemble des régions (vide à l'entrée)
BEGIN
`candidates` : QUEUE<Pixel>
`region` : nouvelle région
`candidates`.ajouter(`start`)
`regions`.ajouter(`region`)
Tant que candidates n'est pas vide, faire
retirer un pixel de `candidates`, le mettre dans `p`
`region`.ajouter(`p`)
Pour chaque voisin `v` de `p`, faire
Si `v`$\notin$`regions` ET |niveau(v)-niveau(start)| <`THRESHOLD`, alors
`candidates`.ajouter(`v`)
Fin si
Fin pour
Fin tantque
END
niveau() est le niveau de gris du pixel, `THRESHOLD` est un paramètre.

Complexité : $0(n^2 \times v)$ où v est la taille du voisinage.

A.3 Méthode de Otsu

Algorithme rapide multi-niveaux :

INPUT :
`k` : nombre de classes
`L` : intensité maximale
`histogram` : distribution des pixels
OUTPUT :
`(k-1)`-uplet donnant une variance inter-classe optimale
BEGIN
`p` : matrice triangulée de dimension `L`
`s` : matrice triangulée de dimension `L`
`h` : matrice triangulée de dimension `L`
// Précalculs
`p [0][0]` = `histogram [0]`
`s [0][0]` = `histogram [0]`
`h [0][0]` = `histogram [0]`
Pour i $\in \{1, \ldots, L\}$, faire
`p [0][i]` = `p [0][i-1]` + `histogram [i]`
`s [0][i]` = `s [0][i-1]` + i$\times$ `histogram [i]`
`h [0][i]` = (`s [0][i]`)^2 / `p [0][i]`
Fin pour
Pour i $\in \{1, \ldots, L\}$, faire
Pour j $\in \{i, \ldots, L\}$, faire
`p [i][j]` = `p [0][j]` - `p [0][i-1]`
`s [i][j]` = `s [0][j]` - `s [0][i-1]`
`h [i][j]` = (`s [i][j]`)^2 / `p [i][j]`
Fin pour
Fin pour
// Recherche de l'optimal
`max` : valeur max
`opt` : distribution optimale
Pour chaque d $= (t_1, \ldots, t_{k-1})$ tel que $0 < t_1 < \ldots < t_{k-1} < L$, faire
`somme` = `h [0]`$[t_1]$
Pour i $\in \{1, \ldots, k-1\}$, faire
somme = somme + `h` $[t_i + 1]$`[`t_{i+1}`]`
Fin pour
somme = somme + `h` $[t_k]$`[L]`
Si *somme* $>$ *max*, faire
max = somme
opt = d
Fin si
Fin pour

Retourner `opt` **END**

Complexité : $0(k \times (L - k + 1)^{k-1})$

Annexe B

Java Image Segmentation Methods

Afin de pouvoir au mieux utiliser la méthode des araignées ainsi que les différentes méthodes de segmentation utilisées lors de comparaisons, j'ai implémenté un programme permettant de manipuler des méthodes de segmentation. Ce programme est composé d'un noyau permettant de charger une image puis d'invoquer des méthodes de segmentation sur cette image. Les méthodes de segmentation se présentent sous la forme de plugins qui sont chargés automatiquement lorsque l'on fait appel à eux.

Un plugin, décrit par l'interface `org.miv.jism.core.SegmentationMethod`, est composé de trois méthodes principales. La première consiste à l'initialisation du plugin avec le contexte courant. La seconde exécute le plugin, puis la dernière permet d'écrire (sous forme d'un fichier image) et éventuellement afficher la segmentation résultante.

JisM est disponible en tant que projet open-source sous licence GNU GPL 3.0[1], et est hébergé par SourceForge à l'adresse

`http ://jism.sourceforge.net`.

B.1 Noyau de JisM

Le noyau de JisM permet de charger un contexte, puis de lui appliquer différentes méthodes.

B.1.1 Architecture multi-threads

Le noyau repose sur une architecture multi-threads qui permet d'exécuter des plugins en parallèle ainsi que d'exécuter des plugins multi-threadés.

[1] GNU General Public Licence

Cette architecture est composée d'un `ThreadPoolExecutor`[2] qui permet de gérer l'ensemble des tâches à effectuer ainsi que le nombre de threads à utiliser.

B.1.2 Contexte

Afin d'optmiser le stockage des informations liées au contexte (valeurs des pixels/voxels), ces données sont stockées dans un objet `ByteBuffer`[3]. Un pixel est alors représenté par son indice dans le buffer.

Cette technique nous permet de gérer la mémoire nécessaire à la création du contexte sans avoir de fuites mémoires liées à l'instanciation d'un grand nombre d'objets java. Si les valeurs des pixels/voxels sont stockées sur 16bits, alors la mémoire nécessaire pour une image de n pixels/voxels sera $2 \times n$ octets.

B.2 JisM Plugins

Par dessus le noyau de JISM vient se greffer une couche de plugins. Ces plugins sont différentes méthodes de segmentation. L'interêt de ce système de plugins est de permettre à l'utilisateur d'implémenter ses propres plugins. Un objet est un plugin s'il implémente l'interface `SegmentationMethod`[4]. Les méthodes qui devront être implémentées correspondent principalement à l'initialisation, l'exécution et l'écriture du résultat de la méthode.

B.2.1 Plugin Python

Afin de s'ouvrir à un large public, JISM est capable d'exécuter des plugins écrits en Python. Le plugin JISM à utiliser sera `PythonPlugin` qui prendra en paramètre le nom du fichier python correspondant au script. Ce script doit contenir les fonctions définies dans l'interface. Le contexte d'exécution est défini dans la variable `ctx` qui est accessible depuis le script. La figure B.1 montre un exemple basic d'un plugin Python.

[2] `java.util.concurrent.ThreadPoolExecutor`
[3] `java.nio.ByteBuffer`
[4] `org.miv.jism.core.SegmentationMethod`

```
# début du fichier monplugin.py
from org.miv.jism import Args
def init() :
       print "initialisation de monplugin.py"
       print "dimension de l'image : ",ctx.getWidth(),"x",ctx.getHeight()
def run() :
       print "lancement de la méthode run()"
def write() :
       print "exécution de la méthode d'écriture"
def getoutputname() :
       return Args.GET("prefix").value() + "-monpluginpython.png"
```

FIG. B.1 – Exemple de plugin python

Annexe C

Lissage de l'histogramme détaillé

FIG. C.1 – Segment de l'histogramme initial

L'histogramme d'une image peut être bruité. Dans ce cas certains pics seront éventuellement détectés comme niveau gris de référence alors qu'ils ne présentent aucun intêret. La figure C.1 montre un segment de l'histogramme de l'image 3d BrainWeb (c.f. 2.2.2).

Un premier lissage (figure C.2) permet de réduire les pics de l'histogramme provoqués par un bruit. Une fois le lissage effectué, on détecte les maxima

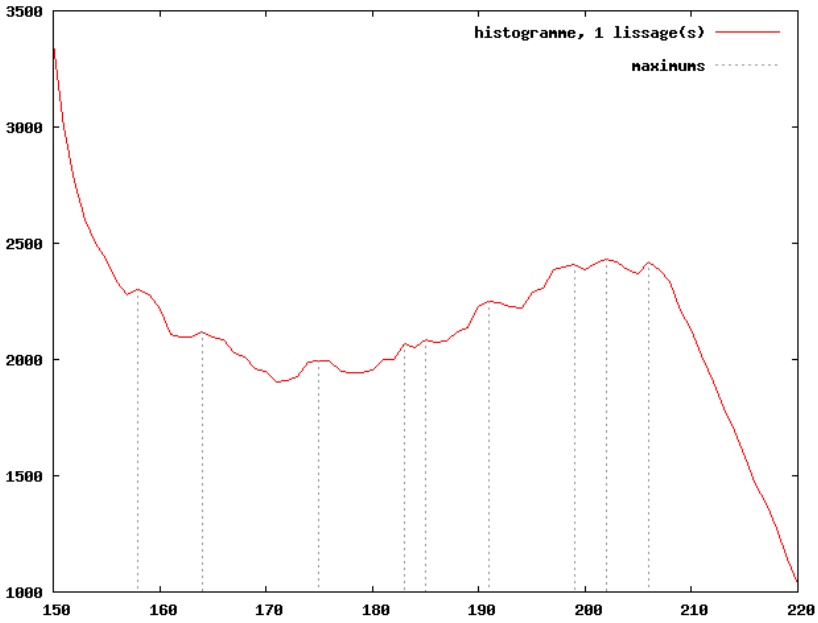


FIG. C.2 – Segment de l'histogramme après 1 lissage

locaux de l'histogramme. On effectue le lissage (figures C.3 C.4 C.5) jusqu'à obtenir le nombre de maxima voulu ou que la diminution de ce nombre ne soit plus significative.

La figure C.6 permet de comparer le segment de l'histogramme avant et après le lissage.

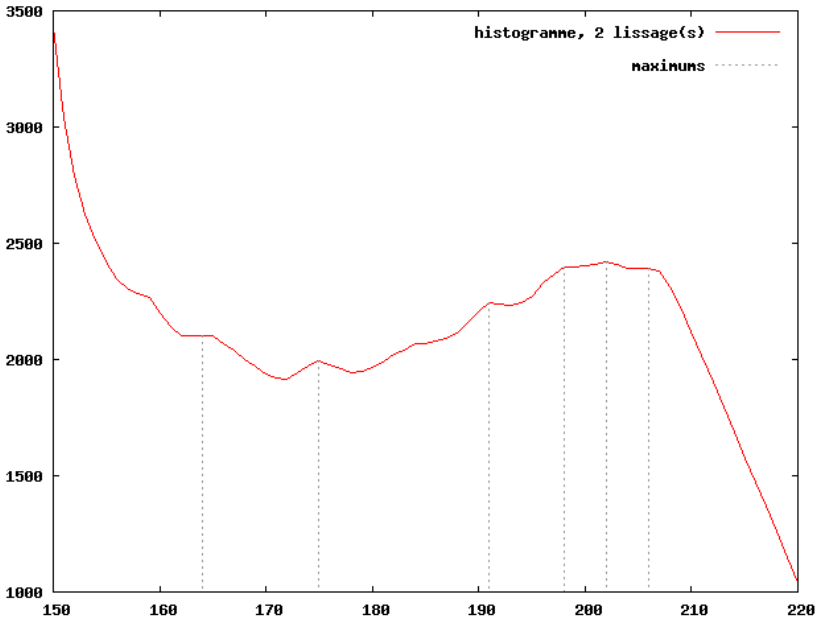


FIG. C.3 – Segment de l'histogramme après 2 lissages

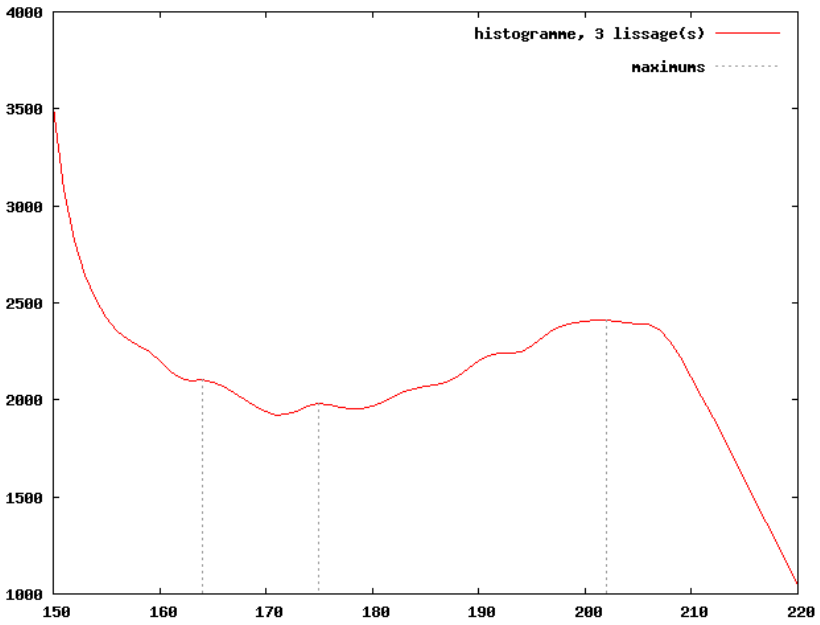


FIG. C.4 – Segment de l'histogramme après 3 lissages

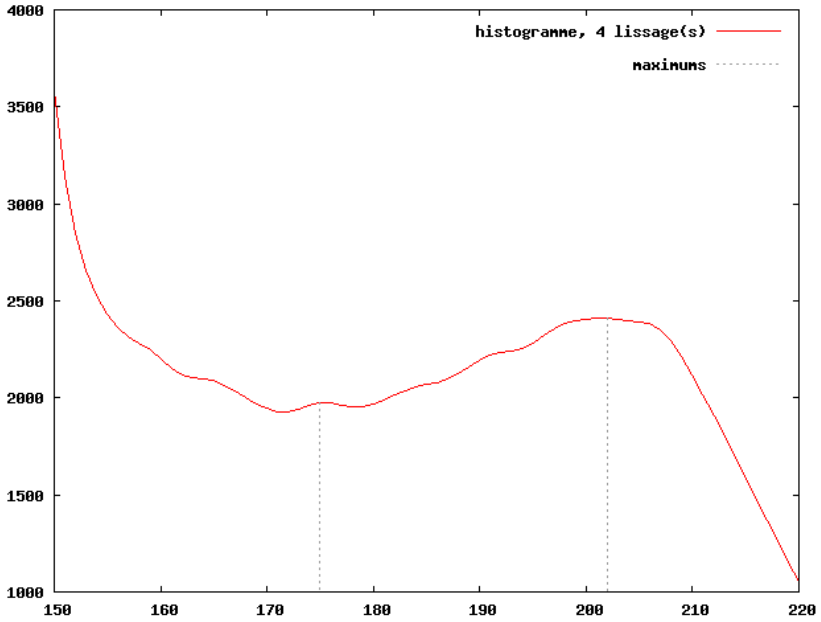


FIG. C.5 – Segment de l'histogramme après 4 lissages

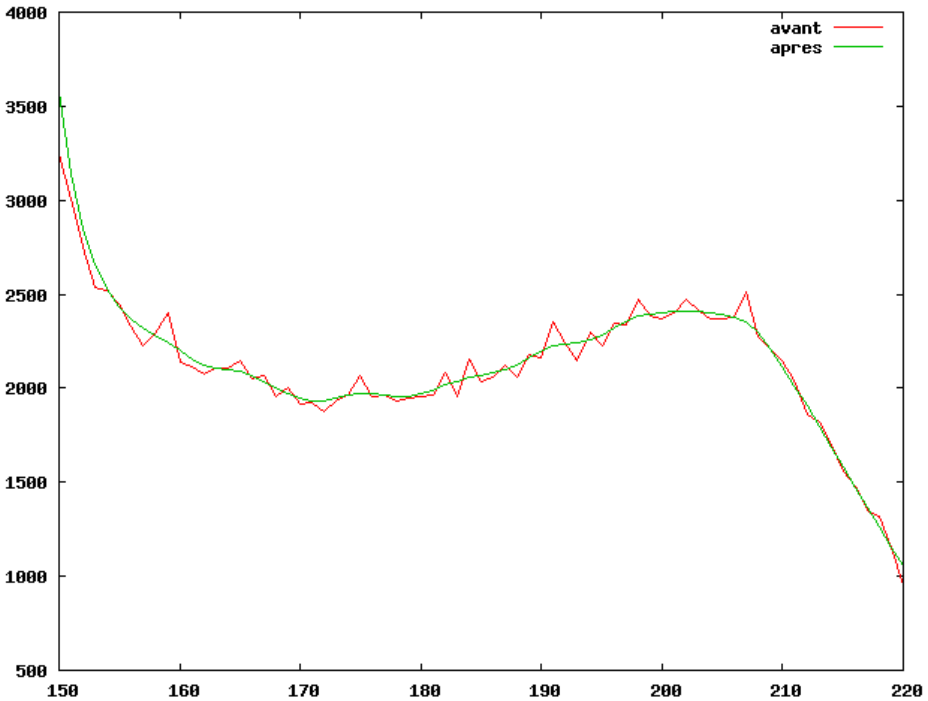


FIG. C.6 – Comparaison de l'histogramme avant et après le lissage

Bibliographie

[1] Anne Guillaud ; Abdessalam Benzinou ; Herve Troadeb ; Vincent Rodin ; Jean Le Bihan. Autonomous agents for edge detection and continuity perception on otolith images. *Image and Vision Computing*, 20(13-14) :955–968, December 2002.

[2] Leonardo Sant'Anna Bins, Leila M. Garcia Fonseca, Guaraci José Erthal, and Fernando Mitsuo II. Satellite imagery segmentation : a region growing approach. *Anais VIII Simpósio Brasileiro de Sensoriamento Remoto*, pages 677–680, April 1996.

[3] Christine Bourjot, Vincent Chevrier, and Vincent Thomas. A new swarm mechanism based on social spiders colonies : from web weaving to region detection. *Web Intelligence and Agent Systems : An International Journal - WIAS*, 1(1) :47–64, Mar 2003.

[4] Vincent Chevrier. *Contributions au domaine des systemes multi-agents.* Hdr, Université Henry-Poincarré - Nancy 1, Janvier 2002.

[5] Colorni A. Dorigo M., Maniezzo M. The ant system : optimization by a colony of cooperating agent. *Man and Cybernetics part B : xybernetics 26(1)*, pages 29–41, 1996. IEEE Transactions on Systems.

[6] Gerhard Weiss et al. *Multi-agent Systems : A Modern Approach to Distributed Artificial Intelligence.* MIT Press, July 2000. ISBN-10 : 0-262-73131-2 ; ISBN-13 : 978-0-262-73131-7.

[7] Jurgen Fripp. *Artificial Life and Agent based Modelling for Image Segmentation.* PhD thesis, The University of Queensland, 2003.

[8] Germond Laurence. *Trois principes de cooperation pour la segmentation en imagerie de resonance magnetique cerebrale.* PhD thesis, Universite Joseph Fourier - Grenoble 1, 1999. Octobre.

[9] J. Le Moigne and J.C Tilton. Refining image segmentation by integration of edge and region data. *Geoscience and Remote Sensing, IEEE Transactions on*, 33(3) :605–615, May 1995. Digital Object Identifier 10.1109/36.387576.

[10] Tim McInerney and Demetri Terzopoulos. Deformable models in medical image analysis : a survey. *Medical Image Analysis*, 1(2) :91–108, 1996.

[11] Catherine Garbay Nathalie Richard, Michel Dojat. Automated segmentation of human brain mr images using a multi-agent approach. *Artificial Intelligence In Medicine*, 30(2) :153–176, february 2004.

[12] N. Otsu. A threshold selection method from gray-level histogram. *IEEE Transactions on System Man Cybernetics*, SMC-9(1) :62–66, 1979.

[13] Tse-Sheng Chen Ping-Sung Liao and Pau-Choo Chung. A fast algorithm for multilevel thresholding. *Journal of Information Science and Engineering*, 17 :713–727, 2001.

[14] Jena-Philippe Rennard. *Vie artificielle : où la biologie rencontre l'informatique*. Vuibert Informatique, 2002. ISBN :2-7117-8694-3.

[15] Yi Shang and Hongchi Shi. A web-based multi-agent system for interpreting medical images. *World Wide Web*, 2(4) :209–218, december 1999. 10.1023/A :1019261025204.

[16] Fabien Michel Smaine Mazouzi, Zahia Guessoum and Mohamed Batouche. *Advanced Concepts for Intelligent Vision Systems : A Multi-agent Approach for Range Image Segmentation with Bayesian Edge Regularization*, volume 4678/2007, chapter 41, pages 449–460. 2007. isbn :978-3-540-74606-5.

[17] Ghassan Hamarneh Tim. Intelligent deformable organisms : An artificial life approach to medical image analysis, 2001. Technical report CSRG-432.

[18] Martha Shenton Tim McInerney, Ghassan Hamarneh and Demetri Terzopoulos. Deformable organisms for automatic medical image analysis. *Medical Image Analysis*, 6(3) :251–266, Sept. 2002.

[19] Filipe Almeida Vitorino Ramos. Artificial ant colonies in digital image habitats – a mass behaviour effect study on pattern recognition. In *Proceedings of ANTS2000 - 2nd InternationalWorkshop on Ant Algorithms (From Ant Colonies to Artificial Ants)*, pages 113–116, sept. 2000.

Index