

ECOLE POLYTECHNIQUE DE L'UNIVERSITÉ FRANÇOIS RABELAIS DE TOURS

Département Informatique

64 avenue Jean Portalis

37200 Tours, France

Tél. +33 (0)2 47 36 14 14

polytech.univ-tours.fr

Projet Recherche & Développement

2018-2019

**Implémentation de nouvelles
fonctionnalités dans la plateforme
NeuroBrainSeg de segmentation
d'images médicales 3D**



POLYTECH[®]
TOURS

Tuteurs académiques

Gaëtan GALISOT

Jean-Yves RAMEL

Étudiant

Yuanyuan LI (DI5)

29 mars 2019



Liste des intervenants

Nom	Email	Qualité
Yuanyuan LI	yuanyuan.li@etu.univ-tours.fr	Étudiant DI5
Gaëtan GALISOT	gaetan.galisot@univ-tours.fr	Tuteur académique, Département Informatique
Jean-Yves RAMEL	jean-yves.ramel@univ-tours.fr	Tuteur académique, Département Informatique



Avertissement

Ce document a été rédigé par Yuanyuan LI susnommé l'auteur.

L'Ecole Polytechnique de l'Université François Rabelais de Tours est représentée par Gaëtan GALISOT et Jean-Yves RAMEL susnommés les tuteurs académiques.

Par l'utilisation de ce modèle de document, l'ensemble des intervenants du projet acceptent les conditions définies ci-après.

L'auteur reconnaît assumer l'entière responsabilité du contenu du document ainsi que toutes suites judiciaires qui pourraient en découler du fait du non respect des lois ou des droits d'auteur.

L'auteur atteste que les propos du document sont sincères et assument l'entière responsabilité de la véracité des propos.

L'auteur atteste ne pas s'approprier le travail d'autrui et que le document ne contient aucun plagiat.

L'auteur atteste que le document ne contient aucun propos diffamatoire ou condamnable devant la loi.

L'auteur reconnaît qu'il ne peut diffuser ce document en partie ou en intégralité sous quelque forme que ce soit sans l'accord préalable des tuteurs académiques et de l'entreprise.

L'auteur autorise l'école polytechnique de l'université François Rabelais de Tours à diffuser tout ou partie de ce document, sous quelque forme que ce soit, y compris après transformation en citant la source. Cette diffusion devra se faire gracieusement et être accompagnée du présent avertissement.



Pour citer ce document

Yuanyuan LI, *Implémentation de nouvelles fonctionnalités dans la plateforme NeuroBrainSeg de segmentation d'images médicales 3D*, Projet Recherche & Développement, Ecole Polytechnique de l'Université François Rabelais de Tours, Tours, France, 2018-2019.

```
@mastersthesis{
  author={LI, Yuanyuan},
  title={Implémentation de nouvelles fonctionnalités dans la plateforme NeuroBrainSeg
    de segmentation d'images médicales 3D},
  type={Projet Recherche \& Développement},
  school={Ecole Polytechnique de l'Université François Rabelais de Tours},
  address={Tours, France},
  year={2018-2019}
}
```


Table des matières

Liste des intervenants	a
Avertissement	b
Pour citer ce document	c
Table des matières	i
Table des figures	vi
1 Introduction	1
1 Contexte	1
2 Objectifs	2
3 Hypothèse	2
4 Bases méthodologiques	3
2 Description générale	4
1 Environnement du projet	4
2 Caractéristiques des utilisateurs	4
3 Fonctionnalités du système	4
3.1 Learning	5
3.2 Segmentation incrémentale et interactive.....	5
4 Structure générale du système	6
3 État de l’art	8
1 Introduction.....	8
2 Segmentation d’image médicale 3D à Atlas.....	8
2.1 Introduction	8

2.2	Recalage	8
3	Méthode de l'Atlas	9
3.1	Atlas Topologique	9
3.2	Multi-atlas	10
3.3	Atlas probabiliste	11
3.4	Atlas probabiliste locale	12
3.5	Comparaison de différentes méthodes d'Atlas	13
4	Modélisation	14
4.1	Graphe de connaissance a priori	14
4.2	Relation spatiale	15
5	Conclusion	16
4	Analyse et conception	17
1	Analyse et conception d'existant	17
1.1	Introduction d'existant	17
1.2	Modélisation existante	18
1.2.1	Graphe de connaissance a priori	18
1.3	Fonctionnalités existantes	18
1.3.1	Fonctionnalités existantes de Learning	18
1.3.2	Fonctionnalités existantes de segmentation incrémentale et interactive	22
1.4	Résumé	27
2	Analyse et conception d'ajoute	28
2.1	Graphe de connaissance a priori modifié	28
2.2	Sauvegarde de résultat	29
3	Comparaison l'existant et l'ajoute (ou la modification)	32
5	Mise en oeuvre	33
1	Outils et libraires	33
2	Implémentation	33
2.1	Implémentation version 1.0 d'apprentissage	34
2.2	Implémentation version 2.0 d'apprentissage	36
2.2.1	Création du tableau	37
2.2.2	Détermination de la relation	38
2.2.3	Implémentation de classes	40
2.3	Implémentation de segmentation	42
2.3.1	Sauvegarder les segmentations	42
2.3.2	Labelliser le voxel	42
2.3.3	Implémentation de classe	43
3	Analyse de résultats	43

3.1	Analyse le résultat d'apprentissage	43
3.1.1	Résultat de relation.....	44
3.2	Analyse le résultat de segmentation.....	45
3.2.1	Segmentation incrémentale.....	45
3.2.2	La qualité de segmentation	47
4	Limite.....	48
6	Bilan et conclusion	49
1	Bilan du semestre 9	49
1.1	Faits	49
1.2	Planning du semestre 10	49
2	Bilan du semestre 10	50
2.1	Faits	50
3	Bilan sur la qualité	50
4	Bilan auto-critique su la gestion du projet.....	50
	Annexes	51
A	Spécification fonctionnelle	52
1	Description des fonctionnalités du programme d'apprentissage.....	52
1.1	Définition de la fonction 1 : readLabelImages.....	52
1.2	Définition de la fonction 2 : findRelations.....	52
1.3	Définition de la fonction 3 : createGraph	53
1.4	Définition de la fonction 4 : readGraph	53
1.5	Définition de la fonction 5 : updateGraph.....	53
2	Description des fonctionnalités du programme de segmentation	53
2.1	Définition de la fonction 1 : classVoxels	53
2.2	Définition de la fonction 2 : saveSegmentation	54
2.3	Définition de la fonction 3 : readSegmentation	54
B	B. Spécification non fonctionnelle	55
1	Performances	55
2	Capacités.....	55
3	Sécurité	55
C	Planification	56
1	Découpage en tâche.....	56
1.1	Tâche 1 : Gestion de projet	56
1.2	Tâche 2 : Étudier d'état de l'art de segmentation médicale	56
1.3	Tâche 3 : Étudier le document de logiciel existant	57

1.4	Tâche 4 : Étudier modélisation méthode d'atlas	57
1.5	Tâche 5 : Étudier segmentation méthode de segmentation	57
1.6	Tâche 6 : Installer et compiler le logiciel	57
1.7	Tâche 7 : Étudier et utiliser le logiciel.....	58
1.8	Tâche 8 : Modifier le modèle de graphe.....	58
1.9	Tâche 9 : Modifier le modèle de segmentation	58
1.10	Tâche 10 : Validation des modifications.....	58
1.11	Tâche 11 : Rédaction du rapport final mi-parcours.....	59
1.12	Tâche 12 : Préparation de la soutenance mi-parcours.....	59
1.13	Tâche 13 : Programme de modification d'apprentissage et tests.....	59
1.14	Tâche 14 : Programme de modification de segmentation et tests	59
1.15	Tâche 15 : Validation des programmes	60
1.16	Tâche 16 : Rédaction du rapport final.....	60
1.17	Tâche 17 : Préparation de la soutenance finale.....	60
2	Diagramme de Gantt	61
D	Document d'installation et de déploiement	62
1	Outils et libraires	62
1.1	IDE	62
1.2	ITK	62
1.3	ITK SNAP.....	63
1.4	Lemon graph library.....	63
1.5	CMake	63
2	Installation.....	64
3	Document Doxygen	66
4	Explications de Classe	67
E	Document d'utilisation	69
1	Utilisation d'apprentissage.....	69
1.1	Préparations.....	69
1.2	Exécution	70
1.3	Visualisation de résultat.....	71
2	Utilisation de segmentation	72
2.1	Préparations.....	72
2.2	Exécution	73
2.3	Visualisation de résultat.....	75
F	Document de tests	78
1	Tests fonctionnels	78
1.1	Test de créer et sauvegarder du graphe.....	79

1.2	Test de relation spatiale	79
1.3	Test 1 d'analyser relation	80
1.4	Test 2 d'analyser relation	81
1.5	Test de sauvegarder la segmentation.....	82
1.6	Test le résultat de segmentation	83
Webographie		84
Bibliographie		85

Table des figures

1 Introduction

1	Le processus de segmentation existant	2
2	Le processus de segmentation incrémentale existant.....	2
3	La segmentation de la Méthode A.....	3
4	La segmentation de la Méthode B	3

2 Description générale

1	Le diagramme de cas d'utilisateur - Learning	5
2	Le diagramme de cas d'utilisateur - Segmentation	6
3	Le diagramme de composants.....	6

3 État de l'art

1	Le processus de la méthode Atlas Topologique	10
2	Le processus de la méthode Multi-Atlas	10
3	Le processus de la méthode Multi-Atlas(l'image est tirée de [8])	11
4	Le processus de la méthode Atlas probabiliste.....	11
5	La carte de probabilité (l'image est tirée de [7])	12
6	Le processus de la méthode Atlas probabiliste locale	13
7	L'Atlas probabiliste locale.....	13
8	La comparaison de différentes méthodes de l'Atlas	14
9	Graphe de connaissance a priori	15
10	Relation spatiale	15
11	Le processus général de segmentation à l'Atlas	16

4 Analyse et conception

1	Le graphe de connaissance a priori	18
2	Le diagramme de classe - Learning.....	19
3	La procédure de Learning	20
4	Ligne commande de démarrer - Learning	20
5	Le processus de 'Learning'	21
6	La structure du résultat	21
7	La structure de fichier '.lgf'	22
8	Le diagramme de classe - Segmentation	23
9	La procédure de Segmentation.....	24
10	Ligne commande de démarrer - Segmentation.....	24
11	Les configurations de paramètres	25
12	Le processus d'exécution	25
13	Le résultat	26
14	Le fichier : CurrentGraph.lgf (visualiser par Notepad).....	26
15	Le fichier : LabelImage.nii.gz (visualiser par ITK SNAP)	27
16	Les fonctionnalités existantes de Learning	27
17	Les fonctionnalités existantes de Segmentation	28
18	Graphe de connaissance a priori modifié	28
19	La classe 'SegmentationResultBase' modifiée.....	29
20	Le tableau pour déterminer le changement de voxel	29
21	Le processus de segmentation avec la 'Relation' ajoutée.....	30
22	La comparaison de modélisation du graphe	32
23	La comparaison de fonctionnalités	32

5 Mise en oeuvre

1	Git Branche	33
2	Exemple d'image 1	34
3	La version 1.0 de la modification de classes	36
4	Exemple d'image 2	37
5	Le tableau initial.....	37
6	Mettre à jour le tableau	38
7	Le tableau de voxels complets.....	38
8	Exemple 1 : La région 2 et la région 9.....	39
9	Exemple 2 : La région 6 et la région 7.....	39
10	Exemple 3 : La région 2 et la région 4.....	39
11	Exemple 4 : La région 1 et la région 9.....	40
12	L'ordre d'appel de méthodes	40
13	La version 2.0 de la modification de classes	41

14	L'exemple	42
15	Le tableau de l'exemple	42
16	Le critère	43
17	Le résultat d'apprentissage.....	43
18	Le dossier 'Nodes'	43
19	Les nœuds	44
20	Les arcs.....	44
21	Le tableau croisé dynamique.....	45
22	Le résultat sur une même image	45
23	Le dossier de résultat.....	46
24	Le résultat sur deux images.....	46
25	Le résultat sur deux images.....	46
26	Le résultat sur deux images.....	46
27	Le résultat de segmentation	47
28	Le fichier 'CurrentGraph.lgf'	47
29	Le résultat de segmentation	47
30	Test de 'TestSegmentationResult2()'	48
 C Planification		
1	Le diagramme de Gantt.....	61
 D Document d'installation et de déploiement		
1	IDE	62
2	ITK	62
3	ITK SNAP	63
4	Visualiser l'image avec ITK SNAP	63
5	Lemon	63
6	CMake.....	64
7	Modifier le chemin de ITK	64
8	Modifier le chemin de Lemon 1	64
9	Modifier le chemin de Lemon 2	64
10	Copier 'CMakeLists.txt'	64
11	Configurer CMake.....	65
12	Exécuter 'NeuroGeo.sln'.....	65
13	Modifier le chemin de segmentations vérités	65
14	Modifier le chemin de sauvegarde	65
15	Commenter la ligne 129	66
16	Doxygen	66
17	Les Classes et les méthodes	66
18	Le diagramme de classe.....	68

E Document d'utilisation

1	Images et images labellisées.....	69
2	Images à étudier	69
3	Dossiers d'images labellisees	70
4	Images labellisées d'image '1000_3_template.nii'.....	70
5	L'image '1000_3_template.nii'	70
6	Images labellisées d'image '1000_3_template.nii'.....	71
7	La commande complète.....	71
8	Le processus d'exécution	71
9	Le résultat d'apprentissage.....	71
10	Les nœuds	72
11	Les arcs.....	72
12	Préparer d'images.....	72
13	Images à segmenter	73
14	Segmentations vérités	73
15	L'image '1000_3.nii'	74
16	Le résultat d'apprentissage de l'image '1000_3_template.nii'	74
17	Lire boundingBox vers segmentation vérité	74
18	Sauvegarde du résultat	74
19	Segmentation incrémentale.....	75
20	Charger la segmentation	75
21	Quitter	75
22	Le résultat	75
23	Charger l'image principale.....	76
24	Charger l'image segmenté.....	76
25	La région 13 (le label avec la couleur bleu), 6 (le label avec la couleur rose) et 155 (le label avec la couleur vert).....	76
26	La région 59 (le label avec la couleur vert) et 57 (le label avec la couleur beige)	77

F Document de tests

1	La structure de tests	78
2	Test.....	78
3	Test de nœuds	79
4	Test de relation spatiale.....	80
5	Test relation séparée.....	81
6	Test le graphe de statut de segmentation.....	83
7	Test la qualité de segmentation.....	83

1

Introduction

Ce document présente les spécifications du projet « Implémentation de nouvelles fonctionnalités dans la plateforme NeuroBrainSeg de segmentation d'images médicales 3D ».

Ce projet est proposé par Monsieur Gaëtan GALISOT, membre de Laboratoire d'Informatique Fondamental et Appliqué de Tours (EA 6300) pour aider les biologistes de l'INRA (Institut National de la Recherche Agronomique) à faire la segmentation d'image médicale 3D. Ce projet est encadré par Monsieur Gaëtan GALISOT et Monsieur Jean-Yves RAMEL, professeur de l'Université de Tours.

1 Contexte

Avec le développement rapide et la vulgarisation du matériel d'image médicale, les technologies d'images telles que l'imagerie par résonance magnétique (IMR), la tomomodensitométrie (TDM) et la tomographie par émission de positons (TEP) sont devenues des méthodes médicales principales pour le diagnostic médical et la planification chirurgicale. Une grande quantité d'informations d'image médicale est produite chaque jour dans le monde, et il a été rapporté que la quantité d'informations d'image médicale représente plus d'un cinquième de la totalité des informations dans le monde.

Le traitement des images médicales est la première étape de l'analyse des images médicales, ce qui permet de rendre les images plus intuitives et plus claires et d'améliorer l'efficacité du diagnostic médical. La segmentation des images est une partie importante du traitement des images, mais aussi un point difficile, un problème de goulot d'étranglement affectant la reconstruction médicale 3D.

La segmentation de l'image médicale basée sur la connaissance a priori de la forme de l'objet, de la distance entre les objets, etc. peut améliorer la qualité des résultats de la segmentation. La quantité et la qualité des connaissances a priori sont un facteur important dans la segmentation de l'image. Par conséquent, la façon d'extraire les connaissances a priori utiles et nécessaires de l'image d'apprentissage constitue une étape importante de la segmentation des images.

Pour faire la segmentation médicale basée sur connaissance a priori, le projet 'NeuroGeo' financé par 'Centre-Val-de-Loire' qui permet de segmenter plusieurs types de cerveaux, de structures est en trains d'être développé par des biologistes de l'INRA (Institut National de la Recherche Agronomique), des chercheurs en neuro-anatomie de l'INSERM (Institut National de la Santé et de la Recherche Médicale) et des chercheurs en informatique du Laboratoire

d'Informatique Fondamentale et Appliqué de Tours (LIFAT)[4]. Le logiciel 'NeuroBrainsSeg' (au sein de projet 'NeuroGeo') développé par Monsieur Gaëtan GALISOT permet de faire la segmentation 3D (segmentation incrémentale et interactive) des régions anatomiques vers l'image IRM de cerveaux efficacement. Les méthodes de segmentation déjà implémentées sont assez efficaces pour la segmentation des régions anatomiques.

2 Objectifs

Le logiciel 'NeuroBrainsSeg' permet de faire la segmentation d'image médicale 3D (incrémentale et interactive) des régions anatomiques vers l'image IRM de cerveaux. Maintenant, les fonctionnalités existantes du logiciel 'NeuroBrainsSeg' peut segmenter incrémentalement et efficacement, c'est-à-dire, il peut segmenter région par région comme Figure 1.

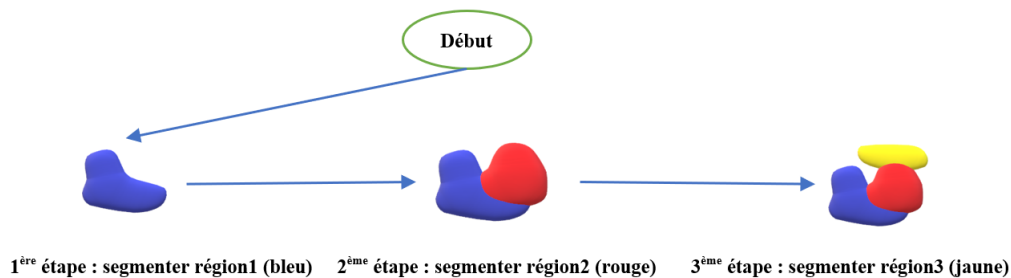


Figure 1 – Le processus de segmentation existant

Il d'abord segmente la région1, après, il segmente la région2, finalement, il segmente la région3. Tous les résultats sont sauvegardés dans une seule image, l'intensité de chaque voxels égale le numéro de la région.

Mais, le problème est : si la région1 (par exemple, cordon antérieur) est incluse ou imbriquée dans la région2 (par exemple, Substance blanche). D'abord, on segmente la région1, après, on segmente la région2.

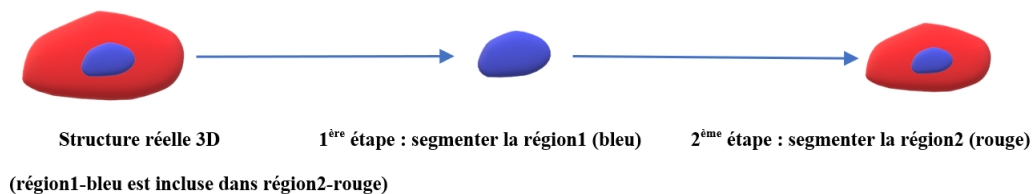


Figure 2 – Le processus de segmentation incrémentale existant

Enfin, on obtient une seule image (le processus est comme Figure2), l'intensité représente le numéro de la région. Donc, vers l'intensité de cette image, il n'y a pas d'espace commun entre la région1 et la région2. Mais, en effet, il y a des espaces communs entre la région1 et la région2 évidemment.

Pour résoudre ce problème, l'objectif du projet est d'implémenter des nouvelles fonctionnalités au sein du logiciel 'NeuroBrainsSeg' qui permettent de segmenter si une région est incluse ou imbriquée dans l'autre.

3 Hypothèse

Pour résoudre ce problème, on essayer de chercher deux méthodes :

- Méthode A (comme Figure 3) : sauvegarder les résultats dans plusieurs images, chaque image correspond à une région, l'intensité égale le numéro de la région.

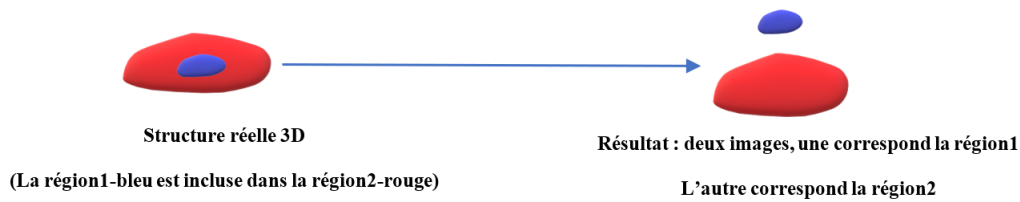


Figure 3 – La segmentation de la Méthode A

- Méthode B (comme Figure 4) : sauvegarder les résultats dans une seule image, mais il a besoins un fichier pour expliquer s'il y a des espaces communs entre des régions.

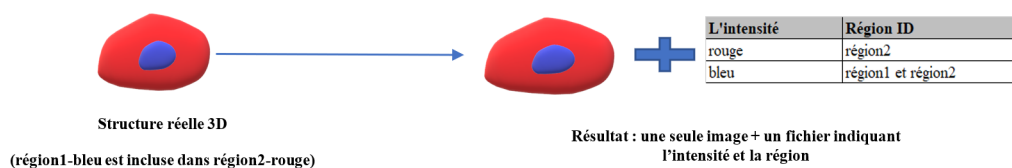


Figure 4 – La segmentation de la Méthode B

On veut d'abord essayer de développer la méthode A, parce que le résultat de la méthode A est plus simple à comprendre pour l'utilisateur. Si la méthode A ne peut pas être réalisée, on change à développer la méthode B.

4 Bases méthodologiques

- Gérer : pour bien gérer le projet, j'utilise la méthode Agile qui permet de consulter et de discuter régulièrement avec le client et l'encadrant. Tous les détails de gestion de projet sont écrits dans l'annexe – C **Planification**.

- Programmer : pour programmer et améliorer la lisibilité du code, je respecterai la convention de 'Camel Case'. Je vais utiliser Git pour gérer les versions de code.

2

Description générale

Je vais présenter quatre parties dans ce chapitre : l'environnement du projet, les caractéristiques des utilisateurs, les fonctionnalités du système, la structure générale du système.

1 Environnement du projet

- Ce projet sera développé en C++ sous Windows.
- L'IDE est Visual Studio (Version 2017).
- Pour gérer les traitements du graphe et de l'image, j'utiliserai deux libraires :
 - Insight Segmentation and Registration Toolkit (ITK).
 - Lemon Graph.

2 Caractéristiques des utilisateurs

Les utilisateurs sont des biologistes de l'INRA. Ils ne sont pas experts en traitement d'images et en informatique.

Donc, quand on développe le logiciel, on doit faire attention à l'ergonomie du logiciel. Par exemple, l'interface ou les commandes du logiciel doit être facile et pratique à comprendre et utiliser pour les biologistes non experts en informatique et en traitement d'images.

3 Fonctionnalités du système

Il y a deux fonctionnalités principales.

1. Learning :

- Entrer : une image de cerveau IRM + plusieurs images labellisées L_i .
- Retour : l'Atlas Probabiliste Local (Plusieurs couples de l'image Template locale T et la carte de probabilité locale P) + graphe de connaissance a priori modifié.

- Description : cela permet d'étudier et d'analyser l'image d'apprentissages pour obtenir toutes les connaissances a priori nécessaires pour la segmentation.

2. Segmentation incrémentale et interactive :

- Entrer : l'image à segmenter I + régions à segmenter R_i + connaissance a priori.
- Retour : les images avec des régions segmentées labellisées, chaque image correspond une région.
- Description : il segmente région par région, c'est-à-dire, il d'abord segmente région R_1 , après, R_2 , R_3 , etc... l'utilisateur définit l'ordre de la segmentation de région.

3.1 Learning

D'abord, l'utilisateur faudrait entrer le chemin de l'image IRM de cerveau et le chemin de dossier qui contient des images labellisées, aussi il faudrait indiquer le chemin où il veut sauvegarder le résultat.

Après, toutes les connaissances a priori qui seront utilisées pour la segmentation suivante sont enregistrées sous le chemin que l'utilisateur indique.

Le diagramme de cas d'utilisateur est comme l'image ci-dessous (Figure 1).

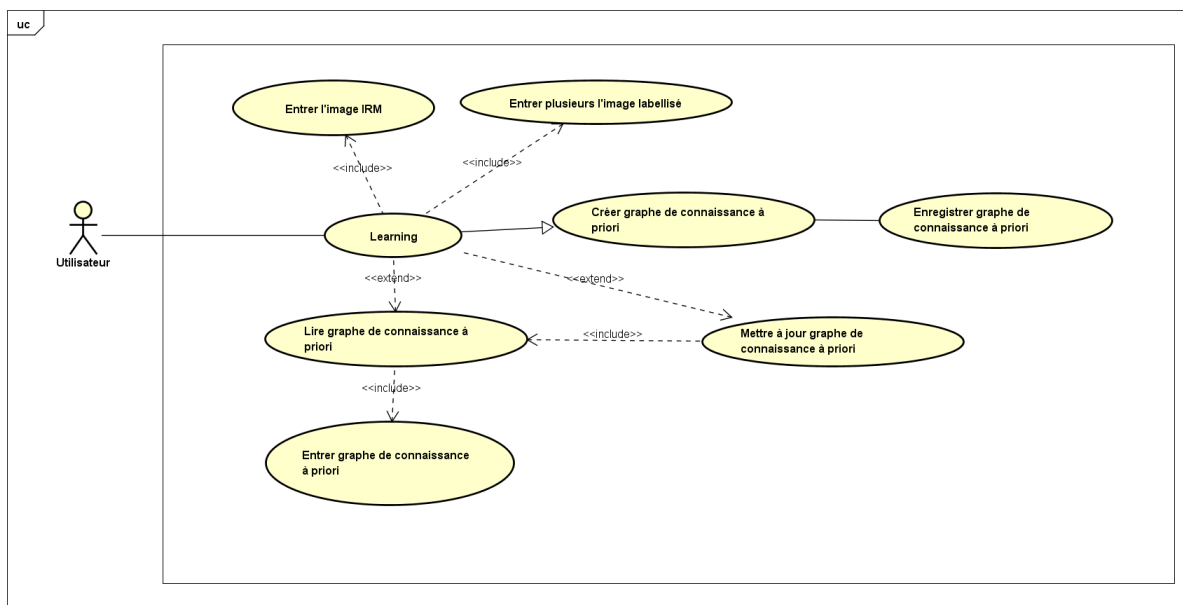


Figure 1 – Le diagramme de cas d'utilisateur - Learning

3.2 Segmentation incrémentale et interactive

Le diagramme de cas d'utilisateur est comme l'image ci-dessous(Figure 2).

D'abord, l'utilisateur faudrait entrer le chemin de connaissance a priori et l'image IRM à segmenter.

Après, l'utilisateur faudrait configurer les paramètres pour segmenter :

1. La région à laquelle l'utilisateur veut segmenter.
2. Le bord de la boîte de la région (c'est obligatoire pour la première région à segmenter).
3. La méthode utilisée pour segmentation.
4. La méthode utilisée pour classifier des voxels.

A la fin, l'utilisateur peut utiliser la commande 'saveSegmentation + le chemin (où l'utilisateur veut sauvegarder le résultat)' pour sauvegarder le résultat de segmentation.

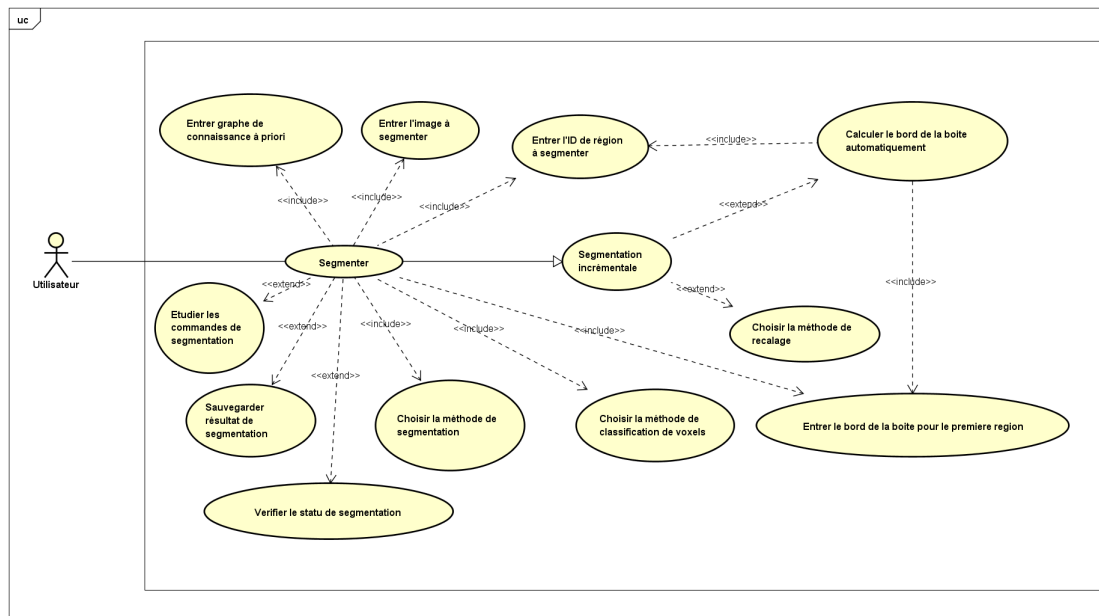


Figure 2 – Le diagramme de cas d'utilisateur - Segmentation

4 Structure générale du système

Il y a deux composants principaux (comme Figure 3) dans le système.

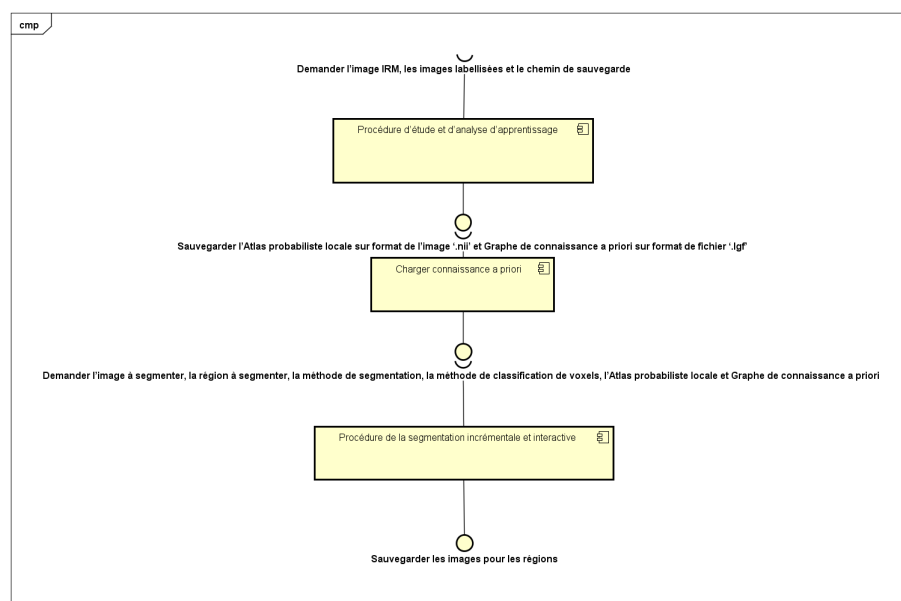


Figure 3 – Le diagramme de composants

Un composant de faire l'étude de l'image d'apprentissage pour obtenir les connaissances a priori, l'autre fait la segmentation.

La procédure de la segmentation incrémentale et interactive demande le résultat de la procédure d'étude et d'analyse d'apprentissage.

3

État de l'art

Dans ce chapitre je vais présenter les différentes méthodes de segmentation médicale à Atlas et les différentes modélisations pour segmentation.

1 Introduction

Segmentation médicale à Atlas est une pratique méthode basée sur connaissance a priori pour réaliser la segmentation d'image médicale 3D.

2 Segmentation d'image médicale 3D à Atlas

2.1 Introduction

Le processus de segmentation à Atlas contient des étapes principales comme ci-dessous :

- Créer (chercher, choisir) Atlas : Atlas est un type de donnée (des images) qui permet d'utiliser comme la connaissance a priori pour faire la segmentation. Il y a plusieurs types de l'Atlas, le type de l'Atlas plus simple est l'Atlas Topologique.
- Faire recalage : trouver une transformation qui permet de transférer l'image Atlas à l'image cible.
- Appliquer la même transformation sur image Atlas pour obtenir le résultat de segmentation.

Je vais expliquerai les différentes méthodes de créer l'Atlas dans secteur 3 - **Méthode de l'Atlas**.

2.2 Recalage

Recalage est un processus de convertir une image à une autre. On suppose qu'on veut convertir Image1 à Image2. Le processus de faire recalage se compose de 3 parties :

- Faire une transformation $T()$ qui permet de convertir Image1 à Image2.

- Quantifier la similitude entre deux images ($T(Image1)$ et $Image2$).
 - Optimiser la transformation $T()$ pour améliorer recalage.
- (1). Faire une transformation : le document [5] décrit 3 types de transformation :
- Rigid [WWW5] : juste contient la rotation, la translation, la réflexion et la combinaison de ces trois transformations.
 - Affine [WWW1] : faire la transformation linéaire.
 - Curved : faire la transformation non-linéaire.
- (2). Quantifier la similitude : pour estimer la qualité du recalage, on peut utiliser plusieurs types de méthodes. Par exemple, le document [5] introduit d'utiliser la distance Euclidienne pour estimer la distance entre $T(Image1)$ et $Image2$. Le document [1] indique qu'on peut aussi utiliser 'Indice de Sørensen-Dice'[WWW3] (en anglais, the Dice similarity coefficient (DSC)) pour quantifier.
- (3). Optimiser la transformation : il existe plusieurs algorithmes qui permettent d'optimiser la transformation, par exemple, Algorithme déterministe, Algorithme Heuristique (par exemple, Algorithme de colonies de fourmis).

3 Méthode de l'Atlas

La segmentation de l'image basée sur des connaissances a priori peut profiter la précision et la crédibilité de la segmentation d'images. Méthode de l'Atlas est une méthode qui utilise connaissance a priori pour segmenter l'image médicale.

L'Atlas est créé par l'étude et l'analyse d'images apprentissages. Chaque région (une structure de cerveau) possède un Atlas qui stocke connaissance a priori de cette région. Il existe plusieurs types de l'Atlas pour stocker connaissance a priori. Pour expliquer les méthodes de différents types de l'Atlas plus spécifiquement et clairement, on suppose qu'on veut segmenter la régions numéro 1 (notée comme la région1).

3.1 Atlas Topologique

Atlas Topologique est un couple de l'image qui contient une image d'intensité Template T et une image labellisée L (une image avec les régions labellisées)[1].

Le processus de segmentation basée sur l'Atlas Topologique (comme l'image Figure1) :

- 1) Choisir Atlas Topologique : d'abord, on parcourt tous les images d'apprentissages, si la région1 est labellisée dans une image, on choisit cette image comme l'image labellisée L , et on choisit l'image d'intensité qui est associée avec l'image L comme l'image Template T .
- 2) Faire recalage : trouver une transformation $F()$ permettant de transférer l'image T à l'image I .
- 3) Appliquer la même transformation $F()$ sur l'image L .
- 4) Obtenir le résultat de segmentation R .

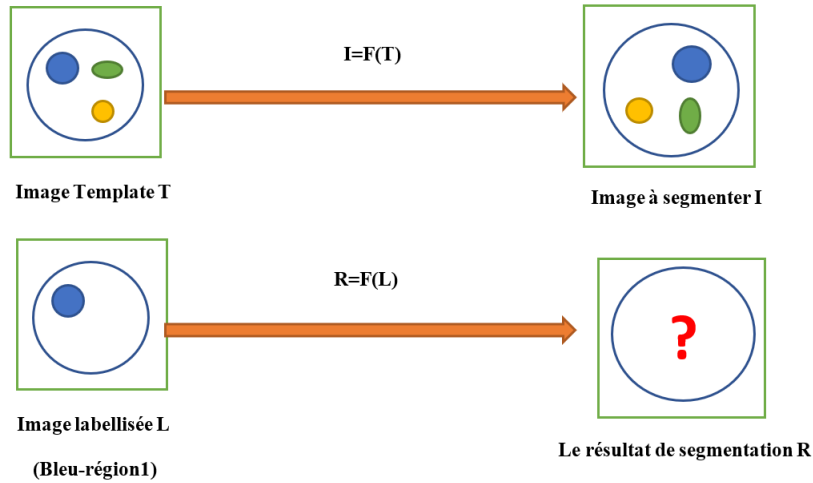


Figure 1 – Le processus de la méthode Atlas Topologique

3.2 Multi-atlas

Multi-Atlas contient plusieurs couples de l'Atlas topologique. Le processus de segmentation basée sur Multi-Atlas (comme l'image ci-dessous Figure 2) :

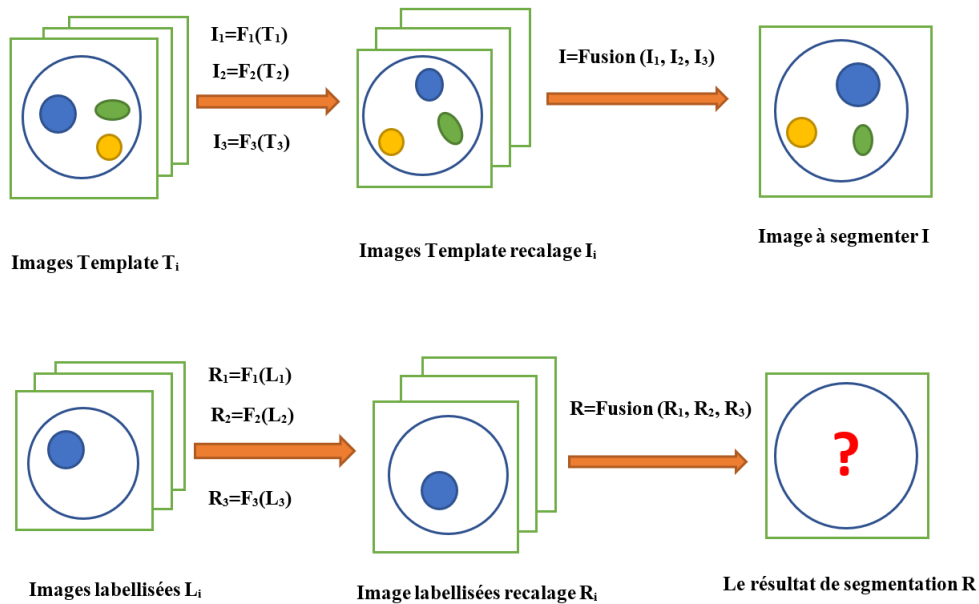


Figure 2 – Le processus de la méthode Multi-Atlas

- 1) Choisir l'image Template T_i et l'image labellisées L_i : la même étape que l'Atlas Topologique, mais, dans ce cas-là, on choisit plusieurs l'Atlas Topologique comme Multi-Atlas.
- 2) Faire recalage : pour chaque l'image T_i , trouver une transformation $F_i()$ qui permet de convertir l'image T_i à l'image I qu'on veut segmenter. Après cette étape, tous les résultats sont notés comme I_i .
- 3) Fusionner tous les image I_i à une seule image. La fusion est noté comme $Fusion()$.
- 4) Appliquer la transformation $F_i()$ sur l'image labellisée L_i . Après cette étape, tous les résultats sont notés comme R_i .

5) Fusionner (utiliser la même *Fusion()* que la 3ème étape) tous les images R_i pour obtenir le résultat final R .

Le document [4], le document [8] et le document [2] utilise Multi-Atlas pour faire la segmentation. Pendant la segmentation, le document [4] utilise la méthode de vote [WWW6] pour classifier les voxels (pour déterminer le voxels est dans la région intérieur ou extérieur). La segmentation du document [8] est comme l'image ci-dessous.

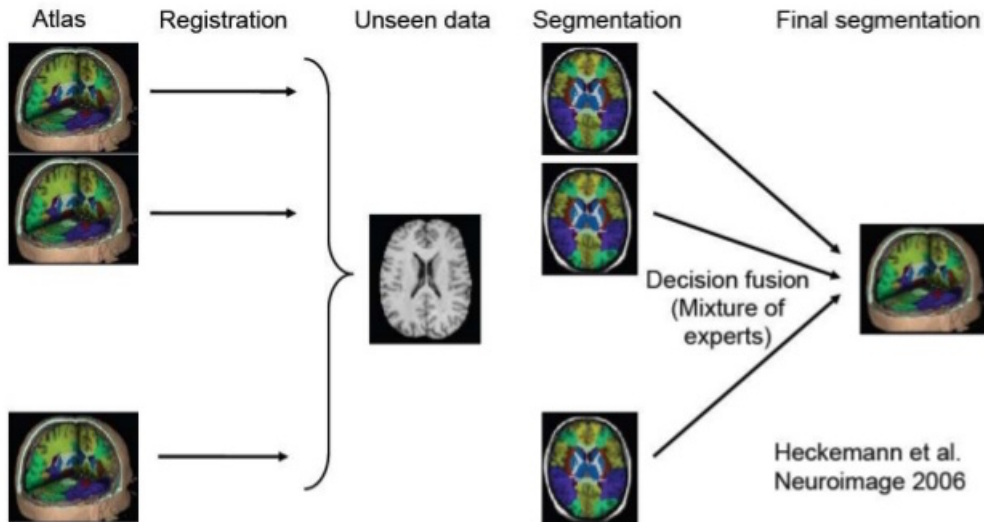


Figure 3 – Le processus de la méthode Multi-Atlas(l'image est tirée de [8])

3.3 Atlas probabiliste

Atlas probabiliste contient couple de l'image Template T et une carte de probabilité P .

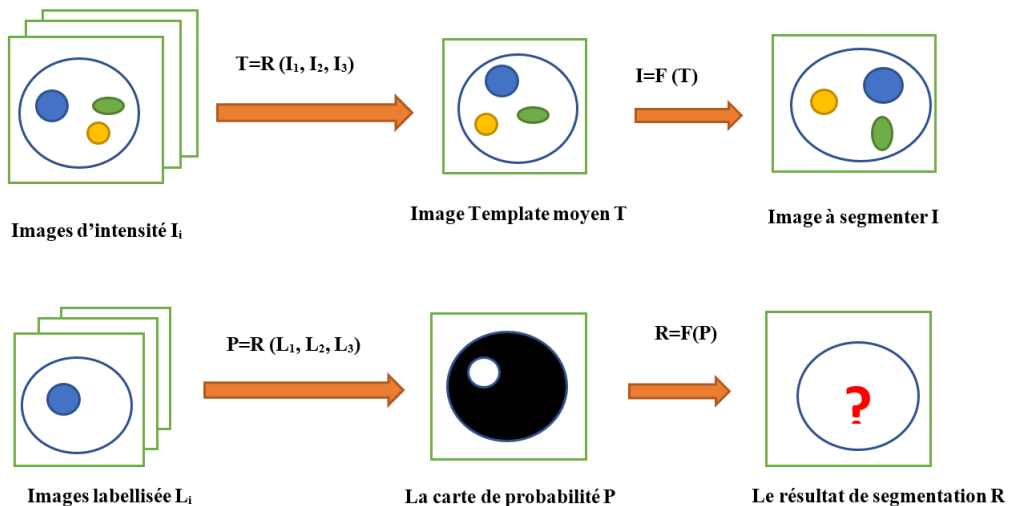


Figure 4 – Le processus de la méthode Atlas probabiliste

Le processus de segmentation basée sur Atlas Probabiliste (comme l'image ci-dessus Figure 4) :

1) Créer l'image Template T : faire recalage entre tous les image I_i , c'est à dire fusionner tous les images I_i . Le résultat est comme l'image Template T .

2) Trouver la transformation $F()$ qui permet de transférer l'image Template T à l'image qu'on veut segmenter I .

3) Créer la carte de probabilité P : pour fusionner L_i , appliquer les mêmes recalages sur les images L_i . Le résultat est noté comme la carte de probabilité P . Dans cette carte, l'intensité de chaque voxels représente la probabilité que cette voxel est dans la région1. Plus l'intensité est claire, plus il est probable que ce voxel est dans la région1.

4) Appliquer la même transformation $F()$ que la 2ème étape sur la carte P , après, on peut obtenir le résultat final R .

On peut trouver que ce type de l'Atlas fonctionne bien dans le document [7] et le document [6]. La carte de probabilité est comme l'image ci-dessous Figure5, l'intensité de l'image est entre 0 et 1.

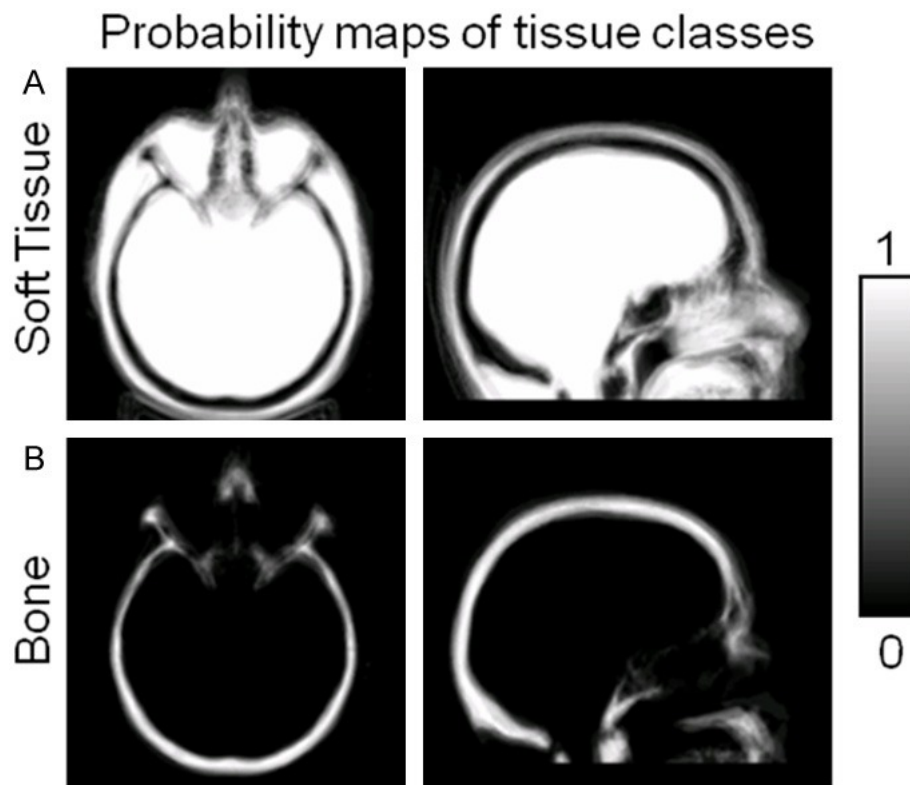


Figure 5 – La carte de probabilité (l'image est tirée de [7])

3.4 Atlas probabiliste locale

Atlas probabiliste locale [4] contient une image Template locale T et la carte de probabilité locale P .

Le processus de segmentation basée sur l'Atlas probabiliste locale (comme l'image ci-dessous Figure6) :

- 1) Créer l'image Template locale T :
 - D'abord, si la région1 est dans des images d'apprentissages A_i , extraire la région1 vers ces l'images A_i . Les résultats sont notés comme l'image a_i .
 - Fusionner l'image a_i pour créer l'image Template locale T .
- 2) Créer la carte de probabilité locale P :
 - D'abord, si la région1 est labellisée dans l'image, on choisit ces images notées comme L_i .

- Extraire la région l_i vers l'image L_i notées comme l'image l_i .
- Fusionner ces image l_i pour obtenir la carte de probabilité locale P.

3) Trouver la transformation $F()$ permettant de convertir l'image T à l'image I et appliquer $F()$ sur la carte de probabilité locale P pour obtenir le résultat R .

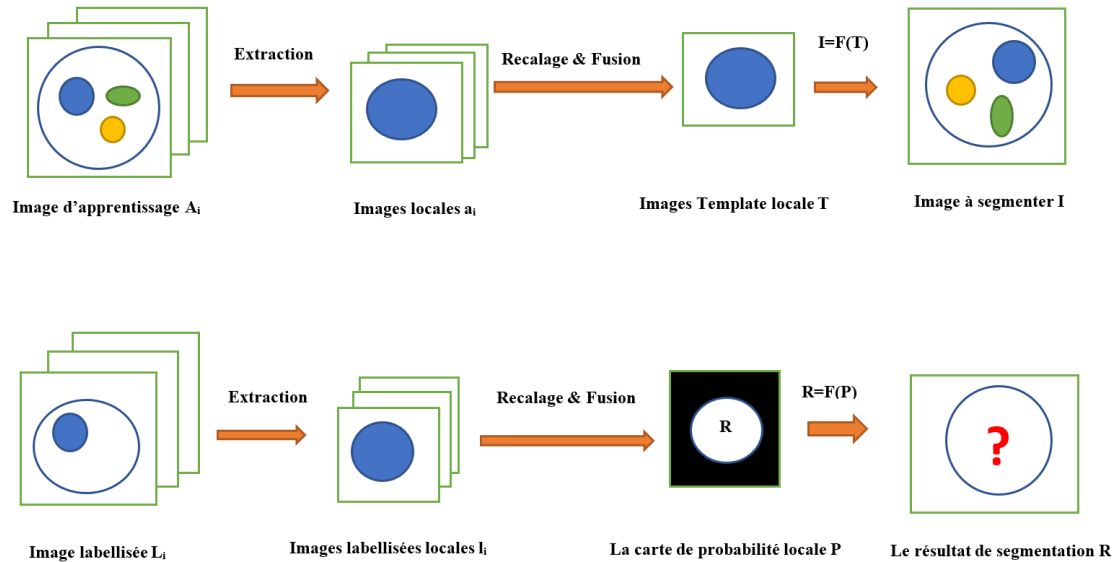


Figure 6 – Le processus de la méthode Atlas probabiliste locale

On peut aussi vérifier que cette Atlas fonctionne bien dans le document [4]. Pendant la segmentation, le document [4] utilise connaissance de 'Champs de Markov cachés' [WWW4] pour classifier les voxels. En vue d'utiliser connaissance de 'Champs de Markov cachés', il a besoin d'initialiser 3-4 classes par appliquer Algorithme k-means [WWW2].

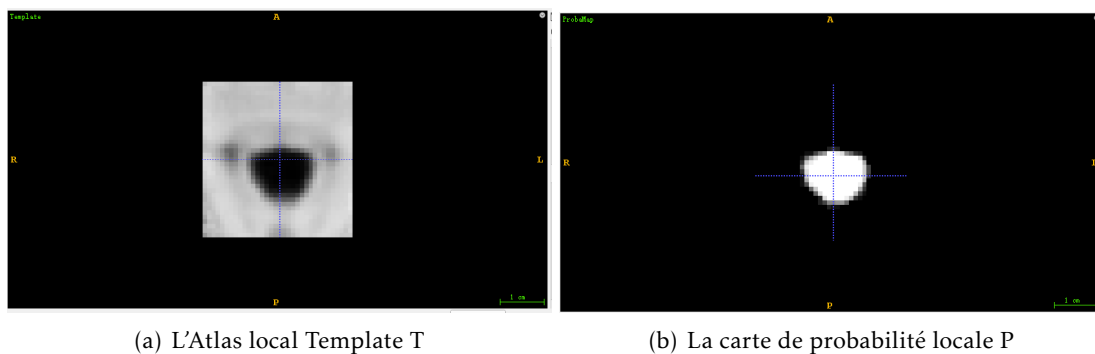


Figure 7 – L'Atlas probabiliste locale

3.5 Comparaison de différentes méthodes d'Atlas

La comparaison de différentes méthodes de l'Atlas est comme le tableau ci-dessous Figure8 :

Méthode	Composant (Pour chaque région)	Avantage	Inconvénient
Atlas Topologique	Image Template + Image labellisée	Le processus est plus simple et plus facile à comprendre.	Le résultat n'est pas très précis. La connaissance a priori n'est pas très suffisante.
Multi-Atlas	Plusieurs images Template + Plusieurs images labellisées	Il contient beaucoup de données a priori. Donc, il est précis que l'Atlas Topologique.	Le résultat est influencé par la quantité et la qualité de l'image Template et l'image labellisée.
Atlas probabiliste	Image Template + La carte de probabilité	La vitesse est plus rapide que Multi-Atlas. Parce qu'il fait recalage juste sur une seule région.	Le résultat est un peu moins précis.
Atlas probabiliste locale	Image Template locale + La carte de probabilité locale	Il extrait sous-image, donc, cela permet d'éviter de beaucoup de changement de la carte de probabilité.	Le temps d'exécution est influencé par le nombre de régions.

Figure 8 – La comparaison de différentes méthodes de l'Atlas

4 Modélisation

La modélisation de comment on stocke tous les informations de l'image, de cerveau est plus importante et nécessaire. La théorie des graphes est une matière très pratique, qui est appliquée dans divers domaines tels que les sciences naturelles et les sciences sociales. De plus, elle est souvent utilisée dans la modélisation de segmentation d'image médicale.

Le document [3] indique que le 'Graphe d'adjacence de régions' peut représenter une image. Dans ce graphe, chaque nœud représente une région ou une structure, les arêtes représentent la α – *adjacents* relation entre des régions. Pour distinguer les différents niveaux de relations, il y a plusieurs types de graphe : Graphe d'adjacence de régions simple, Multigraphe d'adjacence des régions et Graphe duaux. Aussi, il indique qu'on peut conduire un 'quadtree' qui permet de représenter les relation hiérarchies entre des régions ou des structures.

Si on déjà sait qu'il y a certaine relation entre deux régions, et si on déjà segmente une de ces deux régions, on peut facilement segmente l'autre. Donc, dans le document [4], il propose segmentation incrémentale, c'est-à-dire, on segmente l'image région par région. Donc, il a besoins d'étudier la relation (les distances) entre des régions comme des connaissance a priori. De ce fait, le document [4] propose 'Graphe de connaissance a priori' qui permet de stocker L'Atlas et les structures anatomiques dans un seul graphe et il utilise ce graphe pour la segmentation. Je vais introduire cette modélisation spécifiquement.

4.1 Graphe de connaissance a priori

Graphe de connaissance a priori (Figure9) est un graphe orienté. Il contient toutes les connaissances a priori de structure anatomique. Le nœud de ce graphe représente une structure anatomique de cerveau, l'arc de ce graphe représente la relation de nœuds.

Les connaissances a priori se compose de deux parties principales : Atlas probabiliste locale et relation spatiale. Pendant la segmentation, la relation spatiale est pour positionner un sous-espace dans l'image globale, après, on peut utiliser Atlas probabiliste locale pour déterminer si le voxel du sous-espace est dans la région qu'on veut segmenter.

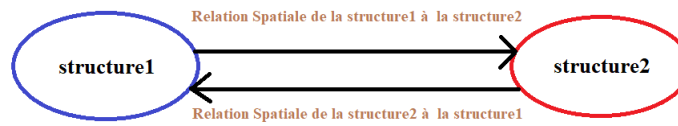


Figure 9 – Graphe de connaissance a priori

4.2 Relation spatiale

On suppose qu'on veut étudier la relation spatiale de la région1 à la région2. On étudie la relation spatiale à partir l'image d'apprentissage.

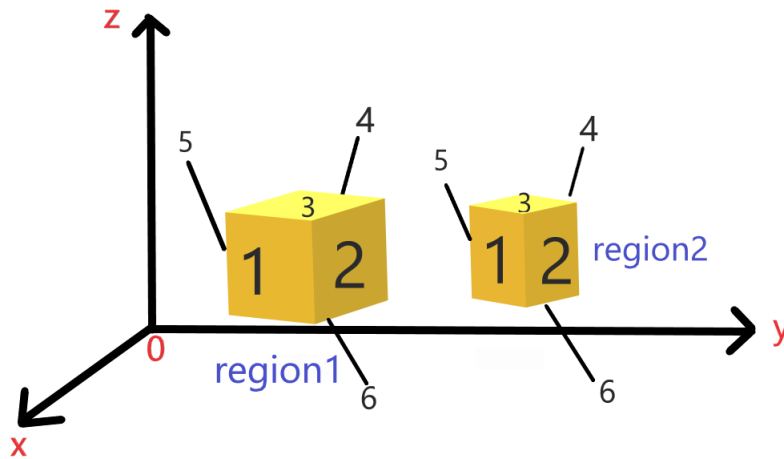


Figure 10 – Relation spatiale

1) Si la région1 et la région2 apparaissent dans une même image, on choisit ces images notées I_i .

2) Après, pour chaque l'image I_i , pour chaque région, parcourir tous les voxels, trouver la boîte qui peut couvrir la région cible. (Comme l'image ci-dessous Figure10).

3) La relation spatiale de la région1 à la région2 est décrite par les 24 distances (chaque boîte possède 6 surfaces, chaque surface possède 4 distances), par exemple, calculant la distance de la surface1 de la région1 à la région2 est comme ci-dessous :

- Pour chaque l'image I_i , la distance entre la surface 1 de la région1 à celle de la région 2 est notée comme d_i^{11} , la distance entre la surface 1 de la région1 à la surface 4 de la région 2 est notée comme d_i^{14} ;

- Parmi toutes les distances calculées vers l'image d'apprentissage, il juste enregistre $\text{Max}(d_i^{11})$, $\text{Max}(d_i^{14})$, $\text{Min}(d_i^{11})$, $\text{Min}(d_i^{14})$.

4) Donc, la relation spatiale contient 24 distances.

5) Mais, pour plus précisément, tous les distances sont divisées par la dimension de la direction de la région1. Par exemple, $\text{Max}(d_i^{11})$, $\text{Max}(d_i^{14})$, $\text{Min}(d_i^{11})$, $\text{Min}(d_i^{14})$ sont divisées par la dimension de la région1 dans la direction x.

5 Conclusion

Le processus de la segmentation médicale à Atlas est composé par deux étapes principale (Figure 11) :

- 1) Étude d'apprentissage pour obtenir de connaissances a priori.
- 2) Faire la segmentation basée sur de connaissances a priori.

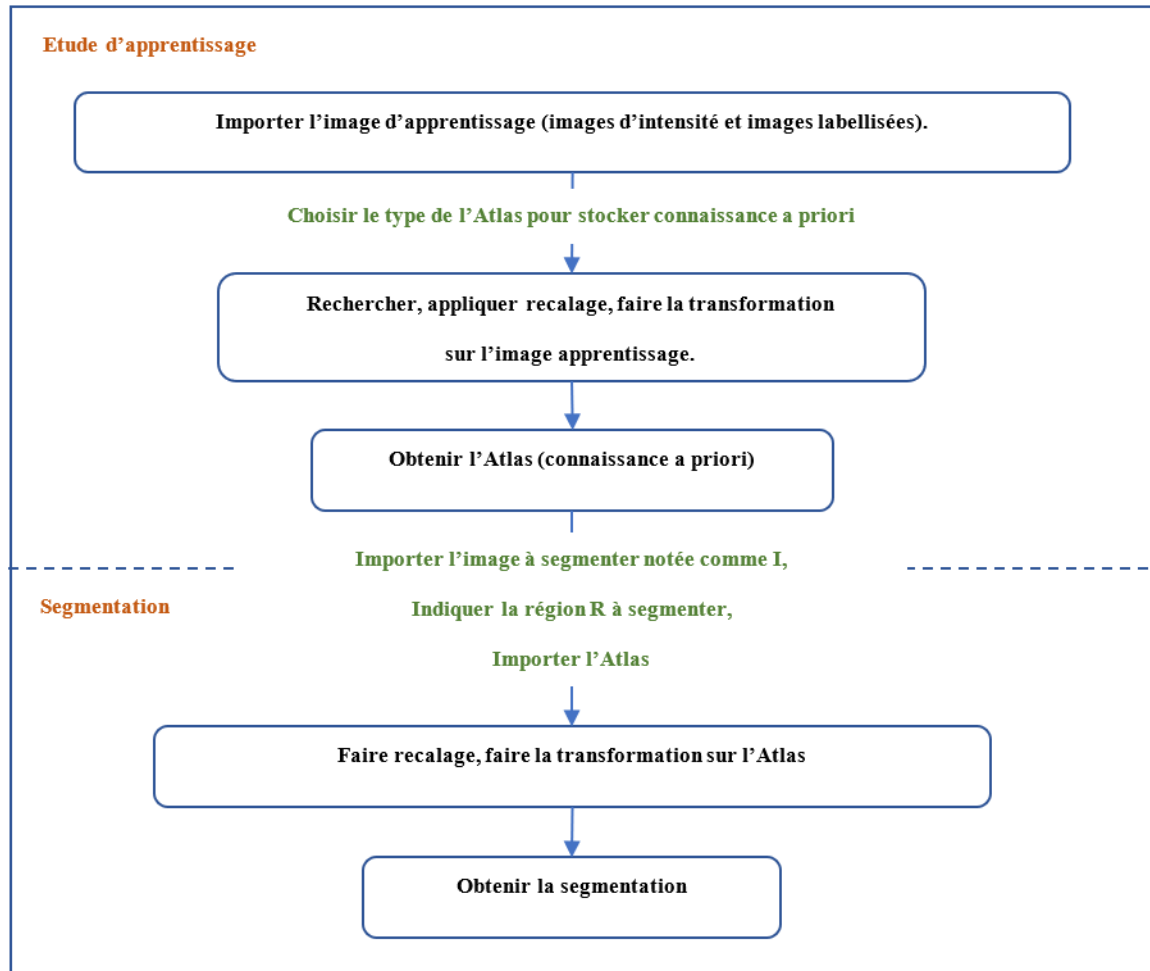


Figure 11 – Le processus général de segmentation à l'Atlas

4

Analyse et conception

Le logiciel 'NeuroBrainsSeg' est développé par Monsieur Gaëtan GALISOT. Donc, pour faire ce projet, j'ai besoin d'analyser les fonctionnalités existantes du logiciel 'NeuroBrainsSeg'.

Donc, dans ce chapitre, je vais d'abord analyser le logiciel existant. Après, je vais expliquer comment je vais modifier ou ajouter les fonctionnalités au sein du logiciel 'NeuroBrainsSeg'.

1 Analyse et conception d'existant

Dans cette partie je vais analyser les fonctionnalités existantes du logiciel 'NeuroBrainsSeg' spécifiquement.

1.1 Introduction d'existant

Le logiciel 'NeuroBrainsSeg' utilise les connaissances a priori pour faire la segmentation de l'image. Les connaissances a priori se compose de deux parties principales :

- L'Atlas probabiliste locale.
- Relation Spatiale.

La procédure d'étude de ces deux connaissances a priori est introduite dans secteur **Atlas probabiliste locale** et **Relation spatiale**.

Ce logiciel permet de faire deux fonctionnalités principales :

- Learning : Étudier et analyser l'image d'apprentissage. Le résultat se compose d'Atlas probabiliste locale et le graphe de connaissance a priori.
- Segmentation : segmenter l'image IRM incrémentalement (région par région) et interactivement. La segmentation est basée sur l'Atlas probabiliste locale et graphe de connaissance a priori qu'on obtient dans la première fonctionnalité.

1.2 Modélisation existante

1.2.1 Graphe de connaissance a priori

Vers le document [4] et le secteur **Graphe de connaissance a priori**, on peut savoir qu'il construit un graphe de connaissance a priori pour stocker toutes les connaissances a priori.

Le graphe de connaissance a priori :

1. Nœud : chaque nœud représente une structure anatomique.
2. L'attribut de nœud : Atlas probabiliste locale de ce nœud (une structure anatomique).
3. Arc : chaque arc représente le lien entre les deux nœuds.
4. L'attribut d'arc : la relation spatiale entre les deux nœuds.

Par exemple, comme l'image ci-dessous Figure1 :

Un nœud représente la partie de 'Lobe temporal' de cerveau, l'autre représente la partie de 'Lobe pariétal'. Chaque nœud comporte son Atlas probabiliste locale. Chaque arc comporte sa relation spatiale.

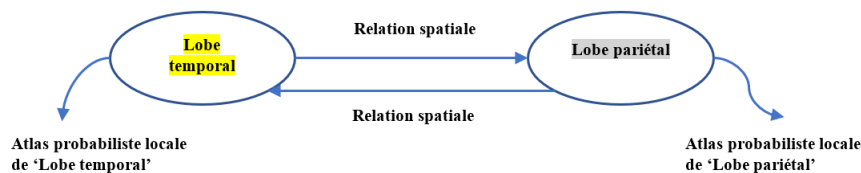


Figure 1 – Le graphe de connaissance a priori

1.3 Fonctionnalités existantes

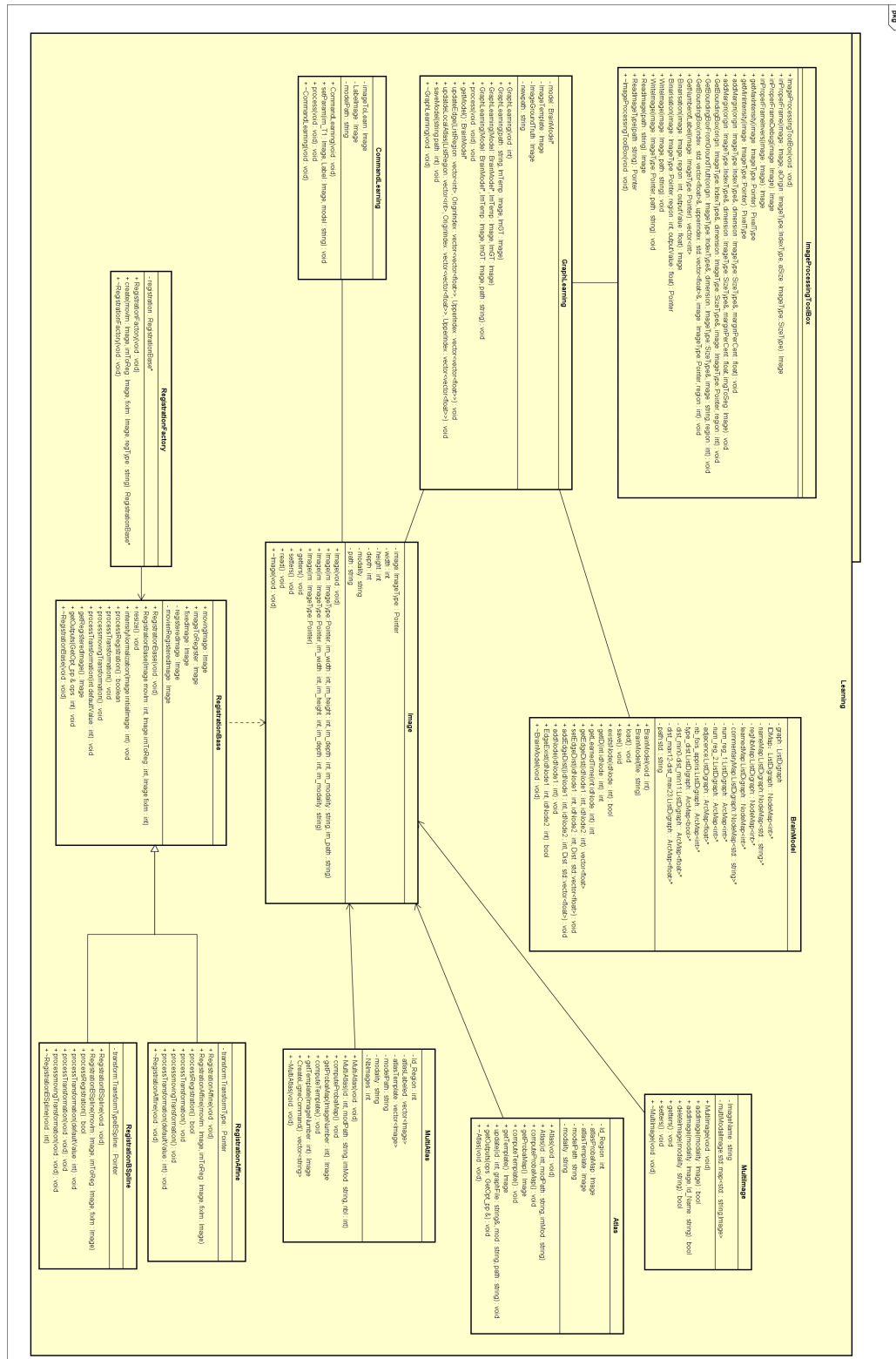
Maintenant, l'interface graphique n'est encore pas opérationnelle. Donc j'ai utilisé l'invite de commande pour démarrer et exécuter le logiciel.

1.3.1 Fonctionnalités existantes de Learning

1. Diagramme de classe

- Image : une classe pour enregistrer les attributs d'image (chemin, longueur, largeur, hauteur). L'attribut principale de cette classe est le pointeur sur une image. Cela permet de charger une image en mémoire.
- MultiImage : l'attribut principale de cette classe est un Map - contient plusieurs objets de la classe 'Image'. Donc cela permet de charger plusieurs images en mémoire.
- GraphLearning : une classe pour exécuter le processus 'Learning', par exemple, créer ou mettre à jour le graphe à priori.
- BrainModel : une classe pour enregistrer le graphe de connaissance a priori. Il enregistre les nœuds (attributs : id, nom, région...), les arcs (attributs : numéro de régions, relation spatiale – 24 distances).

Le diagramme de classe est comme ci-dessous Figure2 :



2. Fonctionnalité

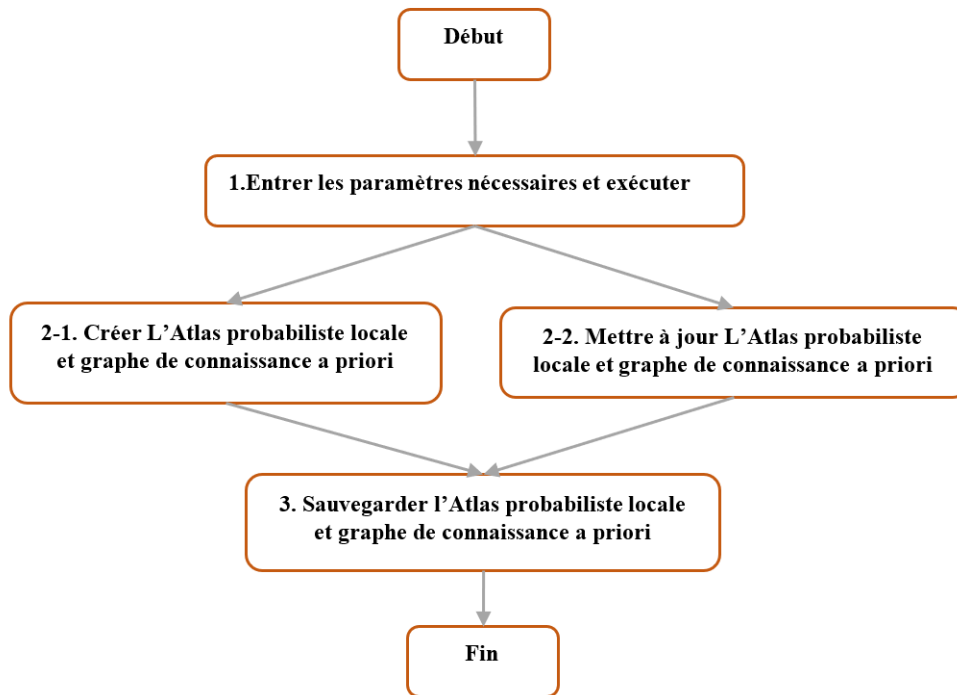


Figure 3 – La procédure de Learning

(1). Entrer les paramètres et démarrer :

Pour démarrer, il a besoins de 3 paramètres :

Ligne commande :

```

--MODE Learning
--INPUT_T1 D:\ImagePRD\test\Image\1003_3.nii
--LABEL_IM D:\ImagePRD\train\label_training\1000_3_glm.nii.gz
  
```

Figure 4 – Ligne commande de démarrer - Learning

–MODE Learning : c'est-à-dire, l'utilisateur veut exécuter le processus d'étude et d'analyse d'apprentissage.

–INPUT_T1 + le chemin de l'image IRM : ce paramètre pour charger l'image IRM (une seule image).

–LABEL_IM + le chemin de l'image labellisée : ce paramètre pour charger l'image labellisée (une seule image).

Après, ces commandes (Figure 4), le logiciel démarre dans le mode 'Learning'.

(2). Créer ou mettre à jour L'Atlas probabiliste locale et graphe de connaissance a priori :

```

D:\VS2017_workspace\NeuroGeo\build\Debug\NeuroGeo.exe
... Creating Model ...
... Local Atlas Generation ...
region : 157
region : 156
region : 45
region : 44
region : 197
region : 145
region : 115
region : 129
region : 109
region : 161
region : 135
region : 43
region : 196
region : 114
region : 108
region : 134
region : 128
region : 42
region : 199
region : 144
region : 39
region : 38
region : 169
region : 168
region : 40
region : 198
region : 41
region : 107

```

Figure 5 – Le processus de 'Learning'

(3). Sauvegarder l'Atlas probabiliste locale et graphe de connaissance a priori :

La structure du résultat est comme Figure 6.

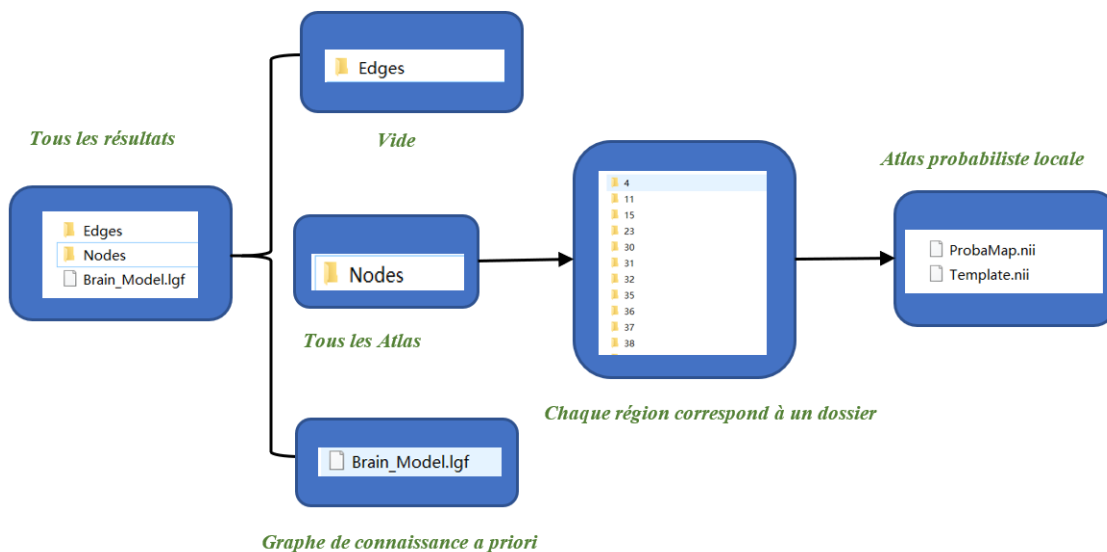


Figure 6 – La structure du résultat

Explication :

- Dossier 'Edges' : vide.
 - Dossier 'Nodes' : il contient plusieurs sous-dossiers, chaque sous-dossier est nommé par l'ID de la région. Chaque sous-dossier contient l'Atlas probabiliste locale (une image Template locale une carte de probabilité locale).
 - Brain_Model.lgf : un fichier d'enregistrer le graphe de connaissance a priori, La structure de ce fichier est comme Figure7 :
- Les nœuds : les attributs de nœud – ID de région, nom de région, numéro de région.
 - Les arcs : les attributs d'arc – numéro de la région1 (noeud1), numéro de la région2 (noeud2), la relation spatiale (24 distances).



Figure 7 – La structure de fichier '.lgf'

1.3.2 Fonctionnalités existantes de segmentation incrémentale et interactive

1. Diagramme de classe

Les classes principales pour le processus 'Segmentation' sont des classes : SegmentationResultBase et LocalSegmentationMethode.

- SegmentationResultBase : la classe pour sauvegarder le résultat.
 - segmentedImage : un objet de la classe 'Image', pour sauvegarder l'image avec des régions segmentées.
 - CurrentGraph : pour sauvegarder l'état de segmentation. Par exemple, le numéro de région segmentée, le bord de la boîte de la région segmentée.
- LocalSegmentationMethode : le main processus d'exécuter la segmentation locale. Par exemple, segmenter l'image ou mettre à jour la segmentation.

Le diagramme de classe est comme ci-dessous Figure8 :

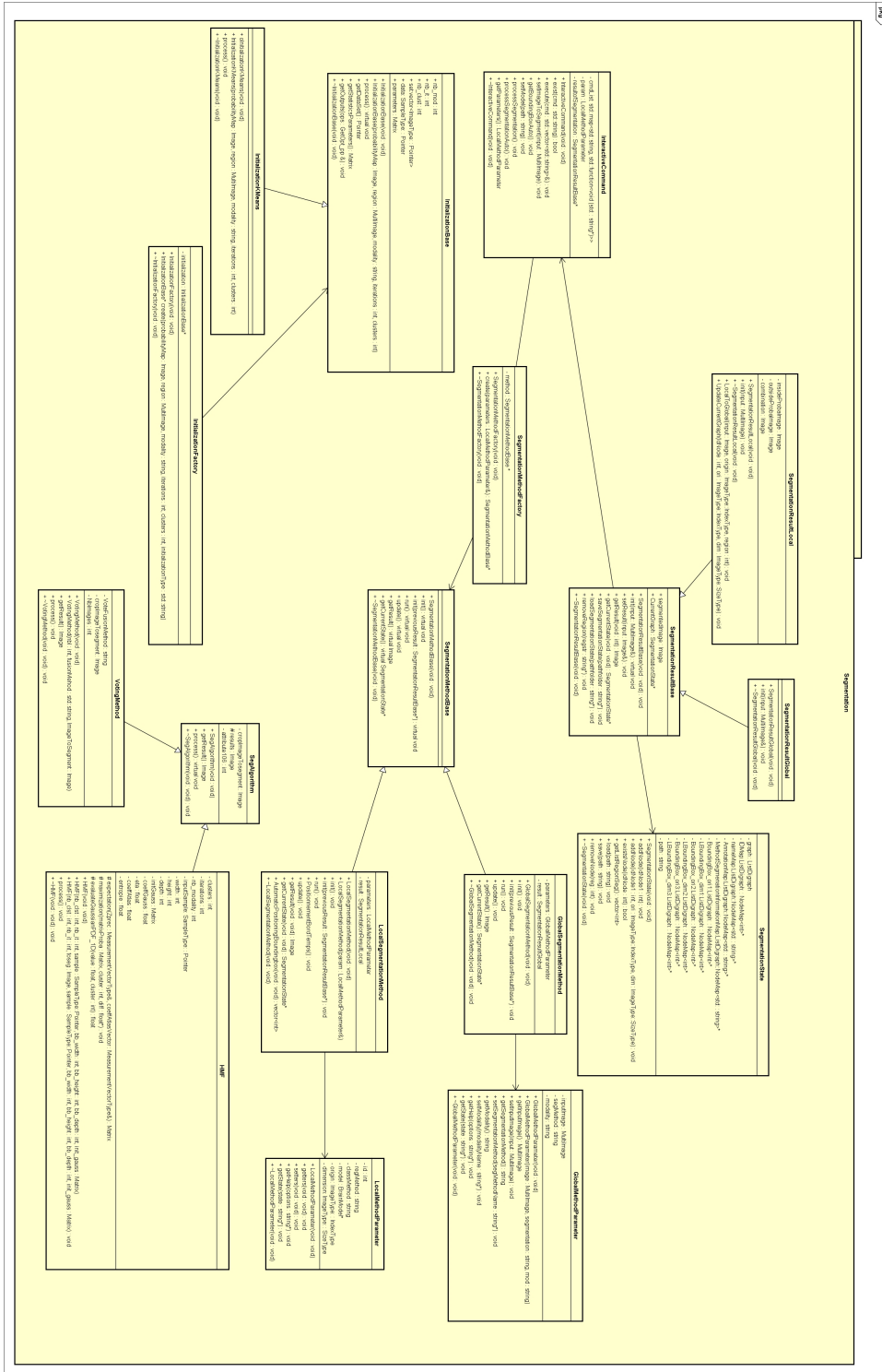


Figure 8 – Le diagramme de classe - Segmentation

2. Fonctionnalité

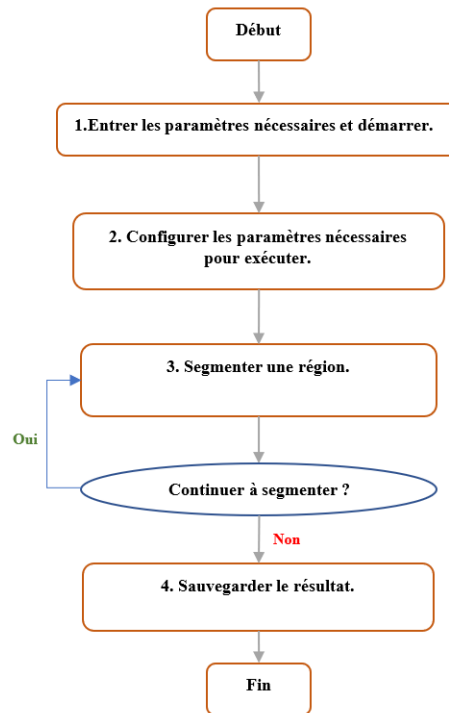


Figure 9 – La procédure de Segmentation

(1). Entrer les paramètres nécessaires et démarrer :

- MODE Seg : c’est-à-dire, l’utilisateur veut segmenter l’image.
- INPUT_T1 + le chemin de l’image à segmenter : pour charger l’image à segmenter.
- MODEL_PATH + le chemin du graphe a priori : pour charger connaissance a priori.

Ligne commande :

```

--MODE Seg
--INPUT_T1 D:\ImagePRD\test\Image\1003_3.nii
--MODEL_PATH D:\ImagePRD\ModelTest
  
```

Figure 10 – Ligne commande de démarrer - Segmentation

Après ces commandes (Figure 10), le logiciel démarre dans le mode de segmentation et charge les images nécessaires.

(2). Configurer les paramètres nécessaires pour exécuter (comme Figure 11) :

Après démarrer le logiciel et charger les images, l’utilisateur doit configurer les paramètres nécessaires pour la segmentation, il peut configurer les paramètres selon les différentes options :

- Configurer la méthode de segmentation : setSegmentationMethod + Local ou Global.
- Configurer la région à segmenter : setCurrentRegion + le numéro de région.
- Configurer le bord de la boîte de la région : setBoundingBox + les coordonnées de boîte.
- Configurer la méthode de classification de voxels : setClssMethod + HMF ou MultiAtlas.

```

D:\VS2017_workspace\NeuroGeo\build\Debug\NeuroGeo.exe
NeuroGeo > setSegentationMethod Local
/!\ ERROR : Non-existent command
D:\VS2017_workspace\NeuroGeo\source\src\InteractiveCommand.cpp(42)
NeuroGeo > setSegmentationMethod Local
... Segmentation method = Local ...
... Loading Model ...
... Model loaded ...
NeuroGeo > setCurrentRegion 51
... Region ID = 51 ...
NeuroGeo > setBoundingBox (1,2,3) (100,100,100)
... Bounding-box origin = (1,2,3) ...
... Bounding-box dimension = (100,100,100) ...
NeuroGeo > setClassMethod HMF

```

Figure 11 – Les configurations de paramètres

(3). Segmenter une région : le processus d'exécution est comme Figure 12.

```

D:\VS2017_workspace\NeuroGeo\build\Debug\NeuroGeo.exe
NeuroGeo > setClassMethod HMF
NeuroGeo > execute
... Get Bounding Box From Ground Truth ...
... -----> Generation OK ...
Positionnement Bounding Box : [83, 86, 85] [28, 27, 100]
Positionnement extended Bounding Box : [80, 83, 75] [34, 33, 120]
... Probability Map Generation ...
D:\ImagePRD\ModelTest\Nodes\51\T1\ProbaMap.nii
... -----> Generation OK ...
... Template Generation ...
... -----> Generation OK ...
atlasTemplate : Image (0000027F952DCC10)
RTTI typeid: class itk::Image<float,3>
Reference Count: 3
Modified Time: 1631
Debug: Off
Object Name:
Observers:
  none
Source: (none)
Source output name: (none)
Release Data: Off
Data Released: False
Global Release Data: Off
PipelineMTime: 1475
UpdateMTime: 1630
RealTimeStamp: 0 seconds
LargestPossibleRegion:
  Dimension: 3
  Index: [0, 0, 0]

```

Figure 12 – Le processus d'exécution

Le processus de segmentation contient deux sous-processus : un est de classer les voxels, l'autre est d'enregistrer la segmentation sur mémoire.

Sous-processus 1 – classer les voxels :

- Entrée : sous-image de la région à segmenter I_s , toutes les connaissances a priori.
- Retourne : une image R_1 en mémoire, l'intensité de chaque voxel égale 0 ou 1. 0 signifie que ce voxel n'est pas dans la région, 1 signifie que ce voxel est dans la région.
- Processus : il utilise les connaissances de 'K-means' et 'Champs de Markov caché' pour déterminer si le voxel est dans la région.

Sous-processus 2 – sauvegarder en mémoire :

- Entrée : le résultat de sous-processus 1 noté R_1 , l'image globale I_g , le numero de la region r .
- Retourne : l'image avec des régions segmentée noté 'segmentedImage'.
- Processus : chaque fois, quand on segmente une région, il mettre à jour l'attribut 'segmentedImage' (un objet de la classe 'Image') de 'SegmentationResultBase'.

Algorithme de sauvegarde le label de voxel est comme ci-dessous Algorithme **Sauvegarder le résultat** :

Algorithme 1 Sauvegarder le résultat**Entrée:**

R_1 : le résultat de sous-processus 1 ;
 I_g : l'image globale à segmenter ;
 r : le numéro de la région ;

Retourne:

segmentedImage : mettre à jour ;

```

0: function LOCALTOGLOBAL( $R_1, I_g, r$ )
1: for  $i = 0$  to  $R_1$ .Depth do
2:   for  $j = 0$  to  $R_1$ .Height do
3:     for  $k = 0$  to  $R_1$ .Width do
4:        $segmentedImage.currentVoxel = getPositionOfTheGlobalImage(i, j, k)$ 
5:       if  $segmentedImage.currentVoxel.Intensite == 0$  then
6:          $segmentedImage.currentVoxel \leftarrow setIntensite(r * R_1(i, j, k))$ 
7:       end if
8:     end for
9:   end for
10: end for
10: return  $segmentedImage$ 
10: end function=0

```

Donc, vers l'algorithme existant (le code de la ligne 5), on peut savoir que chaque fois, quand on segmente une région, il juste mettre à jour l'intensité de voxels encore non labellisée.

(4). Sauvegarder le résultat :

Quand on tape la commande 'saveSegmentation', il sauvegarde deux fichiers (Figure13) sur disque. Un est l'image de segmentation, l'autre est le fichier qui sauvegarde : l'ID de région à segmenter, le borde de la boîte de segmentation (les coordonnées de la boîte).

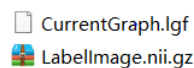


Figure 13 – Le résultat

Le fichier 'CurrentGraph.lgf' enregistre le numéro de la région, le bord de la boîte de la région (Figure14).

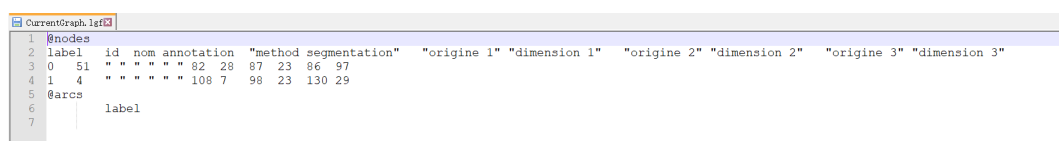


Figure 14 – Le fichier : CurrentGraph.lgf (visualiser par Notepad)

On peut vérifier vers la fenêtre gauche (Figure 15) que l'intensité est 51, égale au numéro de la région.

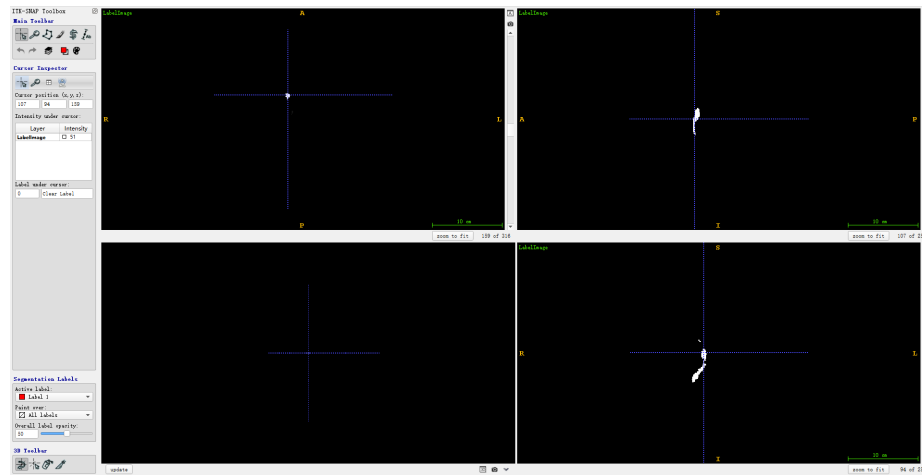


Figure 15 – Le fichier : LabelImage.nii.gz (visualiser par ITK SNAP)

1.4 Résumé

Les deux images dessous ((Figure 16) et (Figure 17)) contiennent les fonctionnalités principales existantes.

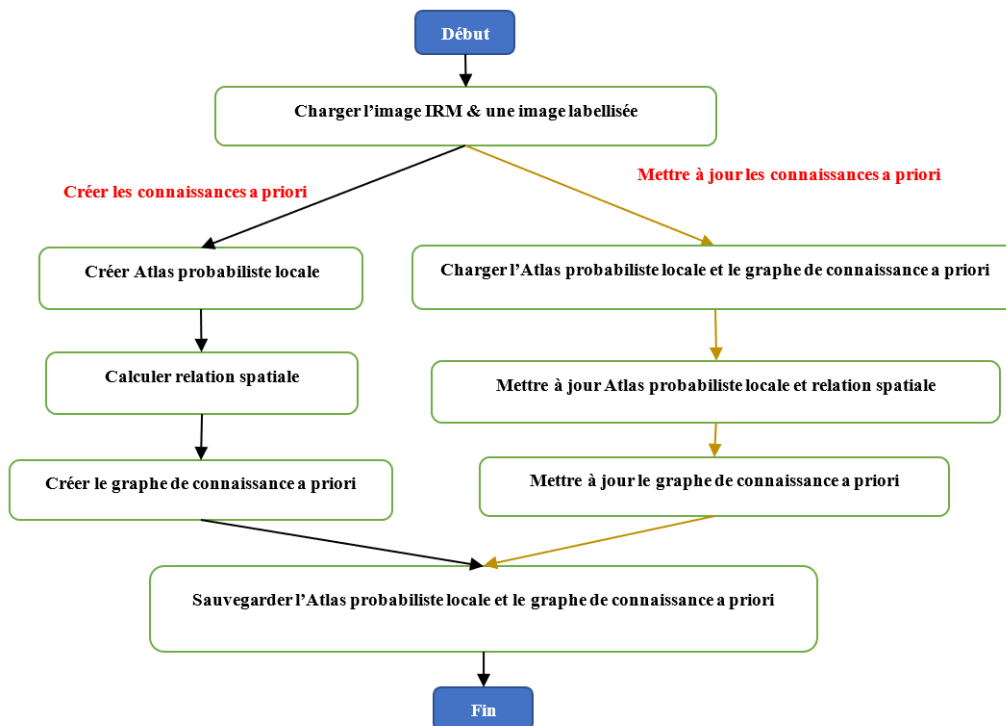


Figure 16 – Les fonctionnalités existantes de Learning

Vers l'image 'Les fonctionnalités existantes de Segmentation' (Figure 17), on peut trouver qu'il segmente région par région et toutes les fonctionnalités marchent bien. Mais, toutes les régions segmentées sont représentées sur une seule image, l'intensité correspond le numéro de région. Donc, il ne peut pas actuellement faire des segmentations lorsque des régions sont incluses ou imbriquées les unes dans les autres.

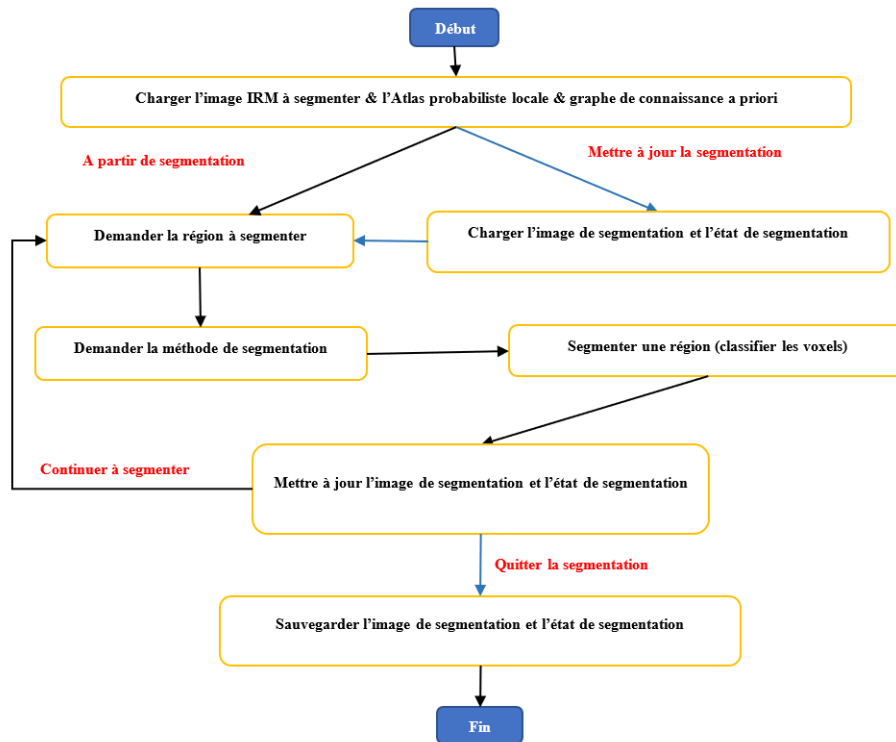


Figure 17 – Les fonctionnalités existantes de Segmentation

Par conséquent, pour résoudre ce problème, je vais faire des modifications sur la base de fonctionnalités existantes.

2 Analyse et conception d'ajoute

2.1 Graphe de connaissance a priori modifié

Pour gérer les relations entre des structures anatomiques, j'ajouterai un attribut dans l'arc qui s'appelle 'relation' (en rouge dans Figure 18) dans le graphe de connaissance a priori. Il peut évaluer 0, 1, 2, 3.

- 0 : il signifie que la Région1 est séparée avec la Région 2.
- 1 : il signifie que la Région1 est imbriquée dans Région 2.
- 2 : il signifie que la Région1 est incluse dans Région 2.
- 3 : il signifie que la Région1 contient Région 2.

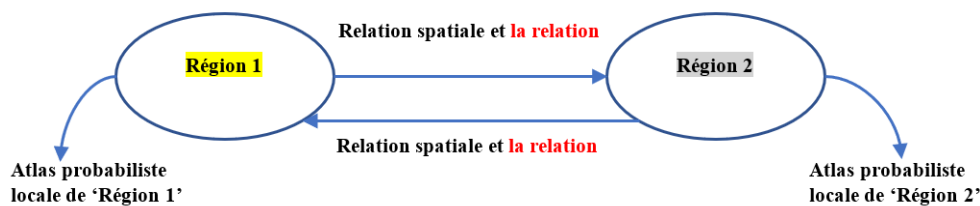


Figure 18 – Graphe de connaissance a priori modifié

Donc, pour programmer, je vais ajouter un attribut dans la classe 'BrainModel'. De plus, toutes les fonctionnalités associées de gérer la classe 'BrainModel' seront modifiées. Il est expliqué précisément dans l'annexe – A **Spécification fonctionnelle**.

2.2 Sauvegarde de résultat

Vers le secteur 1.3.1. **Fonctionnalités existantes de segmentation incrémentale et interactive**, on peut savoir que le logiciel utilise l'objet de la classe 'Image' pour sauvegarder les régions segmentées dans une seule image. Mais, on veut stocker plusieurs niveaux de région, chaque région est présentée par une image. De plus, vers le secteur 1.3.1. **Fonctionnalités existantes de Learning**, on peut savoir que la classe 'MultiImage' contient un attribut de type 'Map' qui peut initialiser plusieurs l'objet de la classe 'Image'. Donc, pour enregistrer plusieurs l'image de résultat, je changerai le type de l'attribut 'segmentedImage' de la classe 'SegmentationResultBase' à 'MultiImage'. Aussi, pour programmer, toutes les fonctionnalités associées à cet attribut seront modifiées.

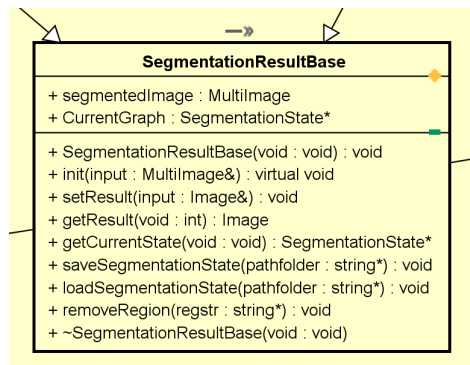


Figure 19 – La classe 'SegmentationResultBase' modifiée

Après, on modifie cette classe comme image dessus (Figure 19), chaque fois quand on segmente une région, on ajoute un nouvel objet de la classe 'Image' dans l'attribut 'segmentedImage'. On met l'intensité de l'image vers l'attribut – 'relation'(ajoutée dans le graphe de connaissance a priori). Par exemple, on suppose que on veut segmenter la région1 et on a déjà segmenté la région2 (l'image de la région2 segmentée est notée comme I_2).

Relation entre région1 et région2	L'intensité de l'image I_2	Changer Ou Pas
0 (R_1 est séparée avec R_2)	R_2	Non
	0	Oui
1 (R_1 est imbriquée avec R_2)	R_2	Oui
	0	Oui
2 (R_1 est incluse dans R_2)	R_2	Oui
	0	Non
3 (R_1 contient R_2)	R_2	Oui
	0	Oui

Figure 20 – Le tableau pour déterminer le changement de voxel

Donc, on parcourt chaque relation entre la région cible et la région déjà segmentée, si toutes les relations disent, on peut changer l'intensité de ce voxel, on change. Sinon, on ne peut pas modifier.

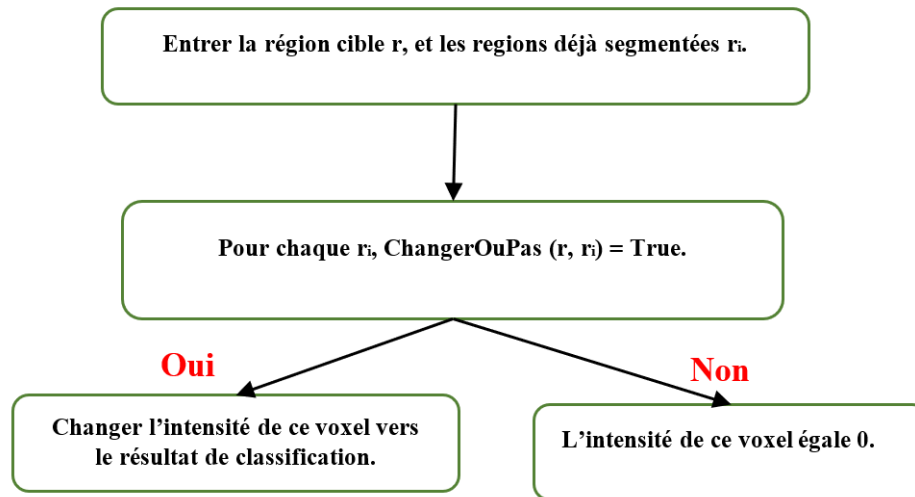


Figure 21 – Le processus de segmentation avec la 'Relation' ajoutée

L'algorithme ajouté de déterminer le changement de voxel est comme ci-dessous **Algorithme de déterminer si change l'intensité** :

Algorithme 2 Algorithme de déterminer si change l'intensité

Entrée:

r_1 : le numéro de région cible ;
 r_2 : le numéro de région déjà segmentée ;
 $intensite$: l'intensité de voxel de la région déjà segmentée ;

Retourne:

$setOrNot$: true or false ;

```

0: function DETERMINCHANGE( $r_1, r_2, intensite$ )
1:  $setOrNot \leftarrow False$ 
2: if  $graphe.getRelation(r_1, r_2) == 0$  then
3:   if  $intensite == 0$  then
4:      $setOrNot \leftarrow True$ 
5:   end if
6: else if  $graphe.getRelation(r_1, r_2) == 1$  then
7:   if  $intensite == r_2$  OR  $intensite == 0$  then
8:      $setOrNot \leftarrow True$ 
9:   end if
10: else if  $graphe.getRelation(r_1, r_2) == 2$  then
11:   if  $intensite == 0$  then
12:      $setOrNot \leftarrow True$ 
13:   end if
14: else if  $graphe.getRelation(r_1, r_2) == 3$  then
15:   if  $intensite == r_2$  OR  $intensite == 0$  then
16:      $setOrNot \leftarrow False$ 
17:   end if
18: end if
18: return  $setOrNot$ 
18: end function=0
  
```

L'algorithme de sauvegarder le label de voxel est comme ci-dessous **Algorithme modifiée de sauvegarder le résultat** :

Algorithme 3 Algorithme modifiée de sauvegarder le résultat

Entrée:

R_1 : le résultat de sous-processus 1 ;
 I_g : l'image globale à segmenter ;
 r : le numéro de la région ;

Retourne:

segmentedImage : ajouter un nouveau objet de la classe 'Image' dans 'segmentedImage' ;

```

0: function LOCALToGLOBAL( $R_1, I_g, r$ )
1:  $currentRegSegImage \leftarrow newImage(R_1.Depth, R_1.Heigh, R_1.Width, intensite = 0)$ 
2: for  $i = 0$  to  $R_1.Depth$  do
3:   for  $j = 0$  to  $R_1.Height$  do
4:     for  $k = 0$  to  $R_1.Width$  do
5:        $setOrNot \leftarrow False$ 
6:       for  $l = 0$  to  $segmentedImage.size$  do
7:          $currentVoxel = getPositionOfTheGlobalImage(i, j, k)$ 
8:          $region2 \leftarrow segmentedImage.get(l).getNumero$ 
9:          $intensite \leftarrow segmentedImage.get(l).getVoxel(currentVoxel).getIntensite$ 
10:         $setOrNot \leftarrow DeterminChange(r, region2, intensite)$ 
11:        if  $setOrNot == False$  then
12:          Break
13:        end if
14:      end for
15:      if  $setOrNot == True$  then
16:         $currentRegSegImage.currentVoxel \leftarrow setIntensite(r * R_1(i, j, k))$ 
17:      end if
18:    end for
19:  end for
20: end for
21:  $segmentedImage.add(currentRegSegImage)$ 
21: return  $segmentedImage$ 
21: end function=0
  
```

Tous les détails de fonctionnalités modifiées ou ajoutées sont expliqués dans l'annexe A **Spécification fonctionnelle**.

3 Comparaison l'existant et l'ajoute (ou la modification)

Vers le tableau dessous (Figure 22), on peut regarder que j'ajoute une relation dans l'attribut de l'arc pour indiquer la relation (séparée, incluse, imbriquée ou contient) entre les régions.

Composant	Graphe de connaissance a priori	Graphe de connaissance a priori modifié
Nœud	Représente une région.	
Les attributs du nœud	ID, le numéro, le nom, l'Atlas probabiliste locale de la région.	
Arc	Représente les relations du nœud.	
Les attributs de l'arc	Relation spatiale (24 distances)	Relation spatiale (24 distances) + <i>Relation (séparée, inclus, imbriquée ou contient)</i>
<i>Tous les différentes entre Graphe de connaissance a priori et Graphe de connaissance a priori modifié sont indiquées en brun.</i>		

Figure 22 – La comparaison de modélisation du graphe

Vers le tableau dessous (Figure 23), on peut trouver les différentes principales en brun.

Composant	Nom de fonctionnalité	Entrer (E)	Retour (E)	Entrer (M)	Retour (M)
Learning	Créer Atlas probabiliste a priori	Une image IRM + Une image labellisée	Une image Template locale T + Une carte de probabilité locale P	Une image IRM + <i>Plusieurs images labellisées</i>	Une image Template locale T + Une carte de probabilité locale P
	Créer le graphe de connaissance a priori	Une image IRM + Une image labellisée	Un fichier '.lgf' enregistre le graphe de connaissance a priori.	Une image IRM + <i>Plusieurs images labellisées</i>	<i>Un fichier '.lgf' enregistre le graphe de connaissance a priori modifié.</i>
Segmentation	Segmentation incrémentale et interactive	Une image IRM + Le numéro de la région à segmenter + Le graphe de connaissance a priori	Une seule image avec les régions segmentées labellisées + Un fichier '.lgf' enregistre l'état de segmentation	Une image IRM + Le numéro de la région à segmenter + <i>Le graphe de connaissance a priori modifié</i>	<i>Plusieurs images avec les régions segmentées labellisées</i> + Un fichier '.lgf' enregistre l'état de segmentation
<i>Dans ce tableau, (E) indique 'l'existant', (M) indique la modification. Tous les différentes entre l'existant et l'ajoute (ou la modification) sont indiquées en brun.</i>					

Figure 23 – La comparaison de fonctionnalités

On peut aussi trouver les différentes entre l'algorithme **Sauvegarder le résultat** et l'algorithme **Algorithme modifiée de sauvegarder le résultat**.

5

Mise en oeuvre

1 Outils et libraires

Ce projet est programmé sur Visual Studio 2017 en C++, aussi, il utilise les libraires : 'ITK' et 'Lemon graph library'. Pour visualiser les images médicales 3D, j'utilise l'outil 'ITK SNAP'. (Les détails de ces libraires sont dans l'annexe [Document d'installation et de déploiement](#))

2 Implémentation

Dans cette partie, je vais présenter l'implémentation de mon projet. La version de programme est contrôlée par GitHub. Pour bien réaliser les fonctionnalités de différentes parties et maintenir le code, j'ai créé différentes branches sur Git.

- Branche master : code source version 0.0 (la version avant mes implémentations).
- Branche lyyBranche : pour implémenter la fonctionnalité d'apprentissage.
- Branche segBranche : pour implémenter la fonctionnalité de segmentation, cette branche est basée sur la branche 'lyyBranche'.

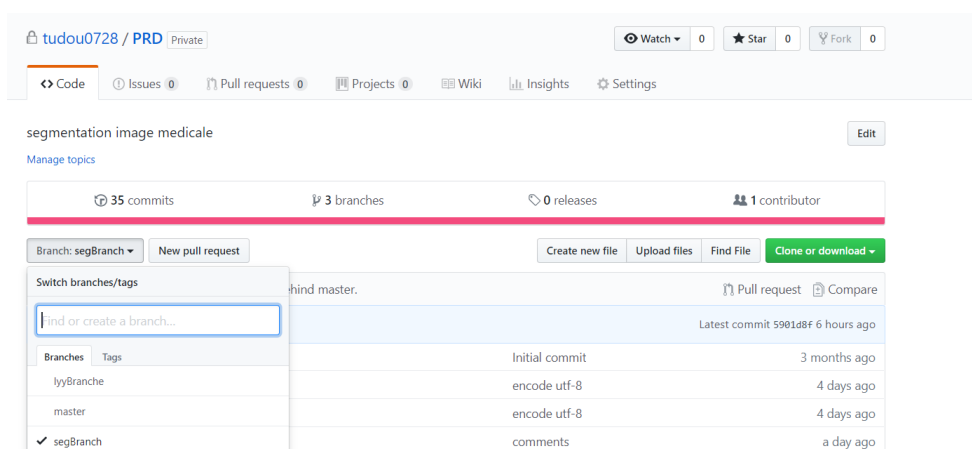


Figure 1 – Git Branche

Je vais présenter les détails de l'implémentation dans les sections suivantes.

Vers la section **Graphe de connaissance a priori modifié**, on peut savoir que pour bien segmenter, on peut ajouter un attribut 'relation' (qui présente la relation entre les deux régions) dans le graphe de connaissance a priori.

- *- 0 : il signifie que la région R_1 est séparée avec la région R_2 .
- *- 1 : il signifie que la région R_1 est imbriquée dans la région R_2 .
- *- 2 : il signifie que la région R_1 est incluse dans la région R_2 .
- *- 3 : il signifie que la région R_1 contient la région R_2 .

2.1 Implémentation version 1.0 d'apprentissage

Par exemple, si on a trois images labellisées L_1 , L_2 et L_3 pour faire d'apprentissage, comme Figure 2.

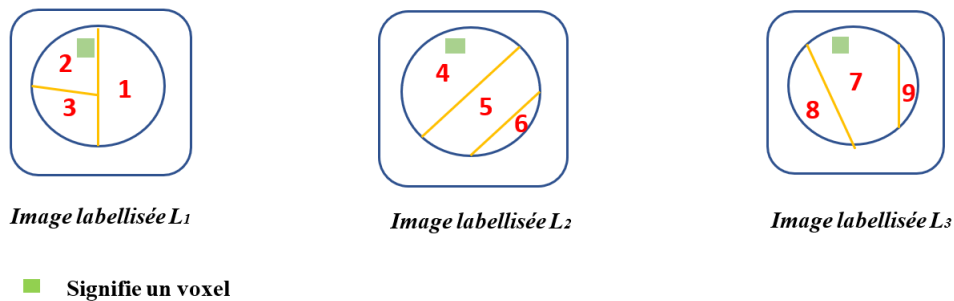


Figure 2 – Exemple d'image 1

Par exemple, si on va trouver la relation entre 'la région 2' et 'la région 4'.

- (1). On peut d'abord parcourir l'image L_1 et L_2 .
- (2). Après, on récupère toutes les coordonnées de la région 2 et de la région 4, noté comme 'list2' et 'list4'.
- (3). On compare 'list2' et 'list4' :
 - S'il n'y a pas d'élément commun entre 'list2' et 'list4', la relation égale 0.
 - S'il y a d'élément(s) commun(s) entre 'list2' et 'list4', la relation égale 1.
 - Si 'list2' est inclus dans 'list4', la relation égale 2.
 - Si 'list2' contient 'list4', la relation égale 3.

Pour implémenter cette fonctionnalité, j'ai modifié le code de la partie d'apprentissage, et je vais présenter les modifications principales que j'ai faites.

- J'ai modifié le type de l'attribut 'ImageGroundTruth' de la classe 'GraphLearning' : type 'Image' \rightarrow type 'MultiImage'.

- J'ai ajouté l'attribut qui s'appelle 'relation' dans la classe 'BrainModel'. En plus, j'ai ajouté la méthode 'setEdgeRelation ()' et la méthode 'getEdgeRelation ()' dans cette classe.

- J'ai ajouté la méthode 'findRealtionsTest ()' dans la classe 'GraphLearning' qui correspond à l'algorithme **Algorithme de chercher la relation entre les deux régions** ci-dessus.

L'algorithme de la méthode 'findRealtionsTest ()' est comme ci-dessous **Algorithme de chercher la relation entre les deux régions** :

Algorithme 4 Algorithme de chercher la relation entre les deux régions

Entrée:

$region_1$: le numéro de la région R_1 ;
 $region_2$: le numéro de la région R_2 ;
 L_1 : l'image de la région R_1 ;
 L_2 : l'image de la région R_2 ;

Retourne:

$relation$: la relation entre la région R_1 et la région R_2 ;

```

0: function FINDRELATIONS( $region_1, region_2, L_1, L_2$ )
1:  $listCoordonnees1 \leftarrow listedecoordonnesdelargion1$ 
2:  $listCoordonnees2 \leftarrow listedecoordonnesdelargion2$ 
3:  $coordonneeCommune \leftarrow 0$ 
4: for  $Coordonneecoordonnee1 : listCoordonnees1$  do
5:   if  $find(listCoordonnees2, begin(), listCoordonnees2.end(), coordonnee1)!$   $=$ 
      $listCoordonnees2.end()$  then
6:      $coordonneeCommune \leftarrow coordonneeCommune + 1$ 
7:   end if
8: end for
9: if  $elementCommune == 0$  then
10:   return 0
11: else if  $elementCommune == listCoordonnees1.size()$  then
12:   return 2
13: else if  $elementCommune == listCoordonnees2.size()$  then
14:   return 3
15: else
16:   return 1
17: end if
18: return  $coordonneeCommune$ 

```

Mais, après cette implémentation, quand j'ai exécuté cet algorithme, je trouve que le temps d'exécution était très long. Pendant 30 minutes, il ne trouve qu'environ 100 relations de régions. Donc, si on a vingt mille relations à chercher, il va prendre presque 10 heures.

Donc, pour obtenir une bonne performance, je vais implémenter d'autre version pour étudier la relation entre les deux régions. Mais, le résultat de cet algorithme est plus précis. Donc, je vais utiliser cet algorithme pour tester (c'est pourquoi le nom de cette méthode est 'findRelationTest'.) et vérifier le résultat. Je vais présenter les tests dans l'annexe **Document de tests**.

En conséquence, après cette implémentation, les classes principales de la partie d'apprentissage sont comme Figure 3 :

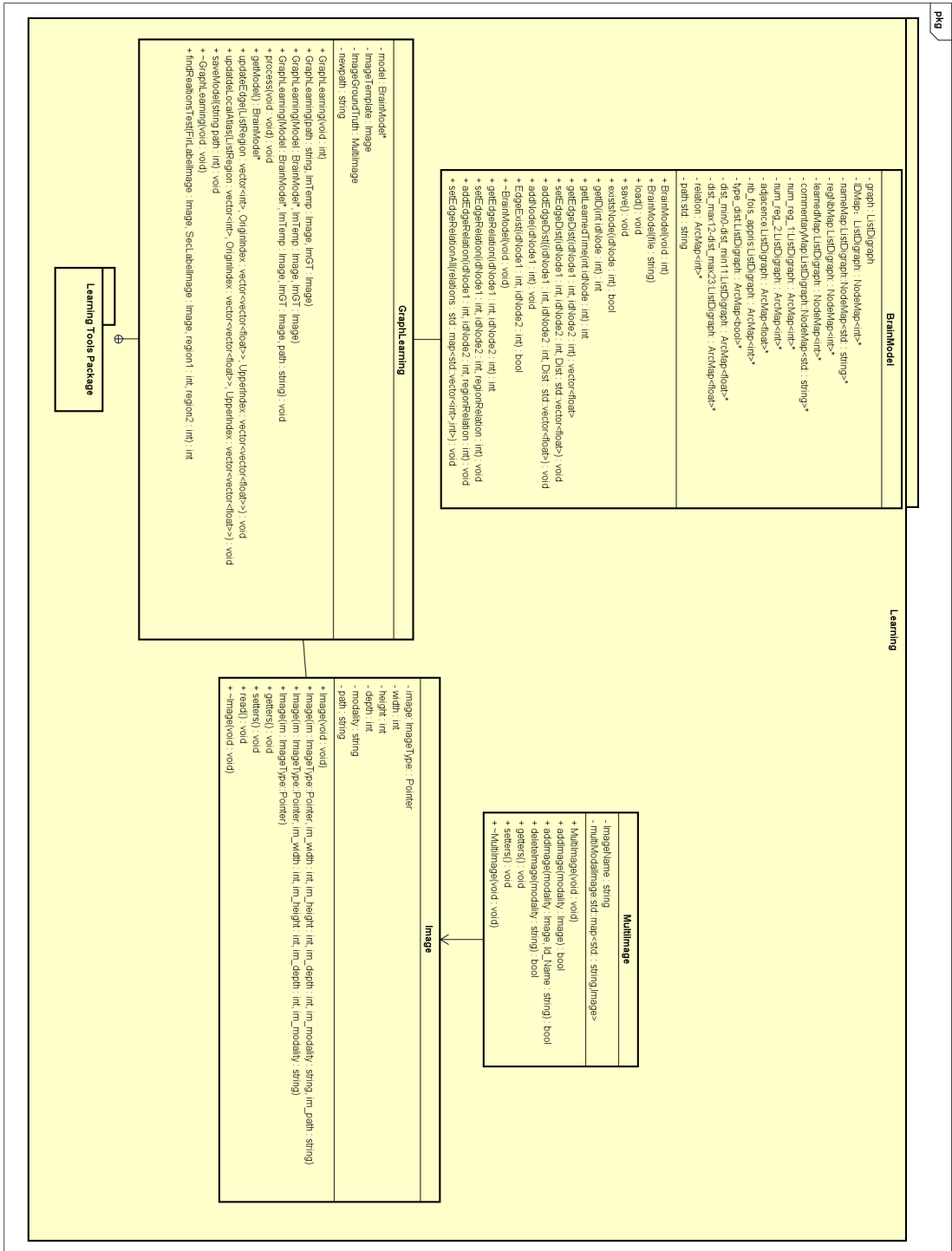


Figure 3 – La version 1.0 de la modification de classes

2.2 Implémentation version 2.0 d'apprentissage

Pour bien améliorer la performance de fonctionnalité d'apprentissage, j'utilise d'autre façon d'étudier la relation et je vais présenter dans cette section.

2.2.1 Création du tableau

Pour trouver la relation entre les deux régions d'images différentes (par exemple, l'image labellisée L_1, L_2, L_3), j'ai créé un tableau de régions (comme Figure 5) :

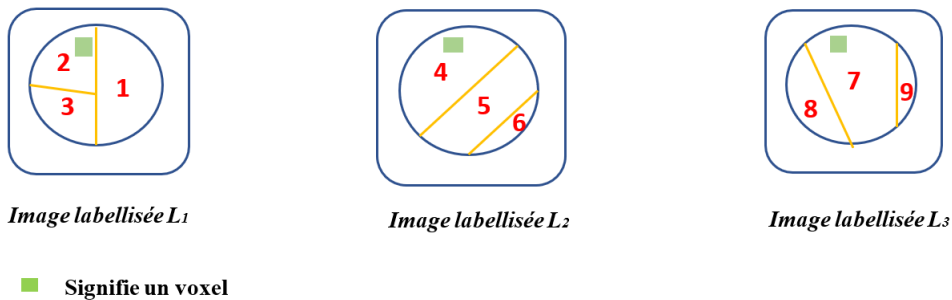


Figure 4 – Exemple d'image 2

(1). Initialiser le tableau de régions : d'abord, on initialise le tableau avec tous les numéros de régions (comme Figure 5).

<i>Noté comme région 1</i>	Numéro de région qui touche région 1 (<i>noté comme list1</i>)
1	[Liste vide]
2	[Liste vide]
3	[Liste vide]
4	[Liste vide]
5	[Liste vide]
6	[Liste vide]
7	[Liste vide]
8	[Liste vide]
9	[Liste vide]

Figure 5 – Le tableau initial

(2). Mettre à jour le tableau de régions : suivante, on parcourt tous les voxels d'images. Par exemple, quand on parcourt les voxels, on lit le numéro de région de voxel vert comme Figure 4. On peut savoir que 'la région 2' touche 'la région 4', aussi, 'la région 2' touche 'la région 7', donc, on doit mettre à jour le tableau, c'est-à-dire :

- Ajouter 4 et 7 dans la liste de la région 2.
- Ajouter 2 et 7 dans la liste de la région 4.
- Ajouter 2 et 4 dans la liste de la région 7.

Donc, le tableau est mis à jour comme Figure 6 :

<i>Noté comme région 1</i>	Numéro de région qui touche région 1 (<i>noté comme list1</i>)
1	[Liste vide]
2	[4,7]
3	[Liste vide]
4	[2,7]
5	[Liste vide]
6	[Liste vide]
7	[2,4]
8	[Liste vide]
9	[Liste vide]

Figure 6 – Mettre à jour le tableau

Pour bien ménager la mémoire, quand on met à jour le tableau, on doit respecter le critère : chaque numéro apparaît juste une fois.

(3). Après parcourir tous les voxels d'images labellisées, le tableau de voxels complets est comme Figure 7 :

<i>Noté comme région 1</i>	Numéro de région qui touche région 1 (<i>noté comme list1</i>)
1	[4,5,6,7,9]
2	[4,7,8]
3	[4,5,7,8]
4	[1,2,3,7,8]
5	[1,3,7,8,9]
6	[1,7,9]
7	[1,2,3,4,5,6]
8	[2,3,4,5]
9	[1,5,6]

Figure 7 – Le tableau de voxels complets

2.2.2 Détermination de la relation

Pour déterminer la relation entre les deux régions, on peut juste analyser le tableau de voxels complets. On sait qu'il y a 4 types de relations, donc, pour chaque type de relation, je vais expliquer spécifiquement.

(1). Type 1 - relation séparée

Description : la région r_1 ne touche pas la région r_2 .

Le critère :

- Le numéro de r_1 n'est pas dans la liste 'list1' de r_2 .
- Et le numéro de r_2 n'est pas dans la liste 'list1' de r_1 .
- Et il n'y a pas d'élément commun entre ces deux listes.

Noté comme région 1	Numéro de région qui touche région 1 (noté comme list1)
2	[4,7,8]
9	[1,5,6]

Figure 8 – Exemple 1 : La région 2 et la région 9

Ex : vers l'exemple 1 (comme Figure 8)

Description : dans ce cas, r_1 ='la région 2', r_2 ='la région 9'.

Analyse : on peut savoir que le numéro 2 n'est pas dans la liste de la région 9, aussi, le numéro 9 n'est pas dans la liste de la région 2. Et il n'y a pas de numéros communs dans ces deux listes.

Conclusion : donc, on peut dire que la relation entre la région 2 et la région 9 est la relation de type 1, et vice-versa.

(2). Type 2 - relation imbriquée

Description : la région r_1 est imbriquée dans la région r_2 .

Le critère :

- Le numéro de r_1 est dans la liste 'list1' de r_2 .
- Et le numéro de r_2 est dans la liste 'list1' de r_1 .
- Et il y a des éléments communs entre ces deux listes.

Noté comme région 1	Numéro de région qui touche région 1 (noté comme list1)
6	[1,7,9]
7	[1,2,3,4,5,6]

Figure 9 – Exemple 2 : La région 6 et la région 7

Ex : vers l'exemple 2 (comme Figure 9)

Description : dans ce cas, r_1 ='la région 6', r_2 ='la région 7'.

Analyse : on peut savoir que le numéro 6 est dans la liste de la région 7, aussi, le numéro 7 est dans la liste de la région 6. Et ils ont un élément commun - 'le numéro 1'.

Conclusion : donc, on peut dire que la relation entre la région 6 et la région 7 est la relation de type 2, et vice-versa.

(3). Type 3 - relation incluse

Noté comme région 1	Numéro de région qui touche région 1 (noté comme list1)
2	[4,7,8]
4	[1,2,3,7,8]

Figure 10 – Exemple 3 : La région 2 et la région 4

Description : la région r_1 est incluse dans la région r_2 .

Le critère :

- Le numéro de r_1 est dans la liste 'list1' de r_2 .
- Et le numéro de r_2 est dans la la liste 'list1' de r_1 .

- Et 'list1' de r_1 est inclus dans 'list1' de r_2 .

Ex : vers l'exemple 3 (comme Figure 10)

Description : dans ce cas, r_1 ='la région 2', r_2 ='la région 4'.

Analyse : on peut savoir que le numéro 2 est dans la liste de la région 4, aussi, le numéro 4 est dans la liste de la région 2. Et la liste1 de la région 2 est incluse dans la liste1 de la région 4.

Conclusion : donc, on peut dire que la relation entre la région 2 et la région 4 est la relation de type 3.

(4). Type 4 - relation contient

Description : la région r_1 contient la région r_2 .

Le critère :

- Le numéro de r_1 est dans la liste 'list1' de la r_2 .
- Et le numéro de r_2 est dans la liste 'list1' de r_1 .
- Et 'list1' de r_1 contient 'list1' de r_2 .

Noté comme région 1	Numéro de région qui touche région 1 (noté comme list1)
1	[4,5,6,7,9]
9	[1,5,6]

Figure 11 – Exemple 4 : La région 1 et la région 9

Ex : vers l'exemple 4 (comme Figure 11)

Description : dans ce cas, r_1 ='la région 1', r_2 ='la région 9'.

Analyse : on peut savoir que le numéro 1 est dans la liste de la région 9, aussi, le numéro 9 est dans la liste de la région 1. Et 'liste1' de la région 1 contient 'liste1' de la région 9.

Conclusion : donc, on peut dire que la relation entre la région 1 et la région 9 est la relation de type 4.

2.2.3 Implémentation de classes

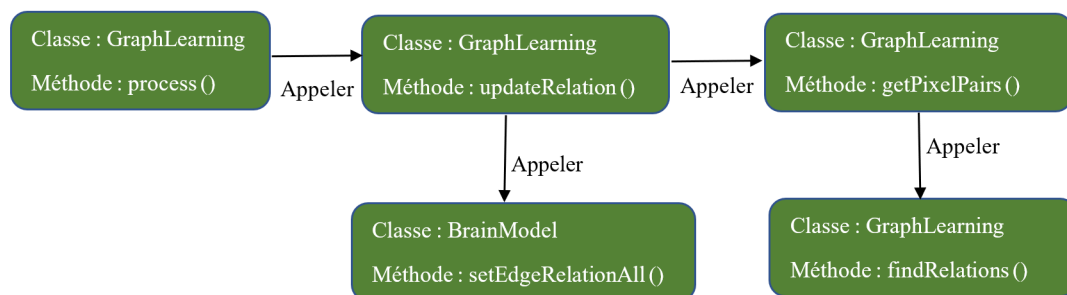


Figure 12 – L'ordre d'appel de méthodes

Selon cette version d'apprentissage, l'implémentation principale est comme ci-dessous :

- J'ai ajouté la méthode 'getPixelPairs ()' dans la classe 'GraphLearning' : cela permet de créer le tableau de voxels complets.
- J'ai ajouté la méthode 'findRelations ()' dans la classe 'GraphLearning' : cela permet de déterminer la relation vers le tableau de voxels complets.

- J'ai ajouté la méthode 'updateRelation ()' dans la classe 'GraphLearning' : cela permet de mettre à jour la relation pour certain arc du graphe.

- J'ai ajouté la méthode 'setEdgeRelationAll ()' dans la classe 'BrainModel' : cela permet de mettre à jour toutes les relations informations d'arcs du graphe.

En conséquence, après cette implémentation, les classes principales de la partie d'apprentissage sont comme Figure 13 :

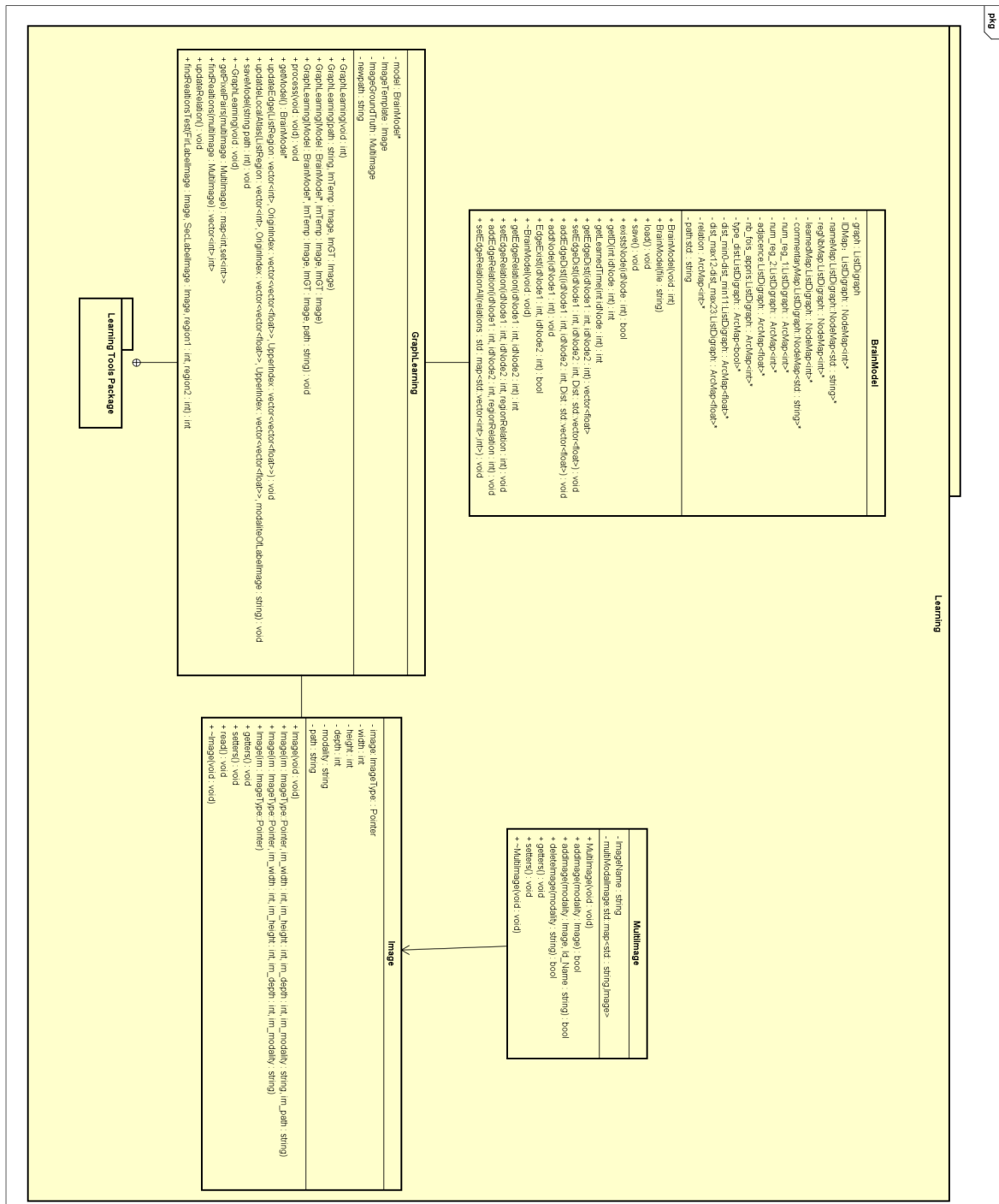


Figure 13 – *La version 2.0 de la modification de classes*

2.3 Implémentation de segmentation

2.3.1 Sauvegarder les segmentations

Pour segmenter le cerveau en plusieurs niveaux, je vais sauvegarder les régions segmentées dans plusieurs images. Aussi, pour bien ménager la mémoire, le critère est comme ci-dessous :

- Si la région (qu'on veut segmenter) touche au moins une région qui est déjà segmentée, je sauvegarde cette région dans une nouvelle image.

Par exemple, Si on déjà segmente la région 2,8,7,9 comme Figure 14, et après, on veut segmenter la région 3.

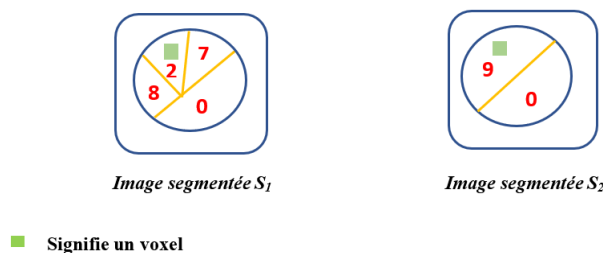


Figure 14 – L'exemple

Donc, il y aura trois probabilités :

- La région 3 ne touche pas la région 2, 7 et 8, on peut labelliser la région 3 sur l'image segmentée 1.
- La région 3 touche la région 2 ou 7 ou 8, mais la région 3 ne touche pas la région 9, on peut labelliser la région 3 sur l'image segmentée 2.
- La région 3 touche la région 2 ou 7 ou 8, aussi, la région 3 touche la région 9. Donc, on doit créer une nouvelle image pour labelliser la région 3.

L'image qui est utilisées pour sauvegarde	Les régions déjà segmentées	Si toucher ou pas	Labelliser ou non
Image segmentée 1	[2,8,7]	Non	Oui
Image segmentée 2	[9]	Oui	Non

Figure 15 – Le tableau de l'exemple

2.3.2 Labelliser le voxel

Quand on étiquette le voxel, si toutes les régions (associées avec ce voxel) disent que nous touchons la région à segmenter, on peut labelliser ce voxel, si non, on ne peut pas le labelliser. Donc, le critère est comme Figure 16.

Nombre de l'image sauvegarde	La somme de la relation	Etiqueter ou non
N	L'autre	Non
N	La somme du nombre de la relation imbriquée, incluse et contienne = n	Oui

Figure 16 – Le critère

2.3.3 Implémentation de classe

Donc, pour réaliser la fonctionnalité de segmentation :

- J'ai modifié la méthode 'run ()' et la méthode 'AutomaticPositioningBoundingBox ()' de la classe 'LocalSegmentationMethod'.
- Aussi, pour labelliser les voxels, j'ai modifié la méthode 'init ()' et la méthode 'LocalTo-Global ()' de la classe 'SegmentationResultLocal'.
- Pour sauvegarder, charger et modifier le résultat de segmentation, j'ai modifié la méthode 'saveSegmentationState ()', la méthode 'loadSegmentationState ()' et la méthode 'removeRegion ()' de la classe 'SegmentationResultBase'.

Après mes implémentations, le diagramme de classes complètes est représenté dans la section **Explications de Classe**.

3 Analyse de résultats

On a différentes images à étudier, et pour chaque image, il y a deux images labellisées.

Donc, pour l'exécution, on entre l'image '1000_3_template.nii' comme l'image à étudier. Et on entre le dossier '1000' qui comporte l'images labellisées de l'image '1000_3_template.nii'.

3.1 Analyse le résultat d'apprentissage

Après l'exécution d'apprentissage, on peut obtenir deux dossiers et un fichier.

> étude (D:) > ImagePRD > imagePreparer > apprentissage > result > 1000				
名称	修改日期	类型	大小	
Edges	2019/3/20 5:51	文件夹		
Nodes	2019/3/20 5:58	文件夹		
Brain_Model.lgf	2019/3/20 6:01	LGF 文件	4,478 KB	

Figure 17 – Le résultat d'apprentissage

- Le dossier 'Edges' et le dossier 'Nodes' comportent toutes les images de l'Atlas Local et la carte probabilité locale.

> étude (D:) > ImagePRD > imagePreparer > apprentissage > result > 1000 > Nodes				
名称	修改日期	类型	大小	
4	2019/3/20 5:58	文件夹		
6	2019/3/20 5:58	文件夹		
10	2019/3/20 5:58	文件夹		
11	2019/3/20 5:57	文件夹		
13	2019/3/20 5:58	文件夹		
15	2019/3/20 5:58	文件夹		
16	2019/3/20 5:58	文件夹		

Figure 18 – Le dossier 'Nodes'

On peut vérifier que le dossier 'Edges' comporte 143 dossiers, chaque dossier représente une région.

- Le fichier 'Brai_Model.lgf' : ce fichier enregistre toutes les informations de nœuds et d'arcs de graphe de connaissance a priori. Pour bien visualiser ce fichier, on peut copier ce fichier à l'Excel (comme Figure 19 et Figure 20).

A	B	C	D	E	F	G
label	id	nom	numeroRegion	fois_appris	commentaires	
0	157	Structure 157	157	1		
1	156	Structure 156	156	1		
2	45	Structure 45	45	1		
3	44	Structure 44	44	1		
4	197	Structure 197	197	1		
5	145	Structure 145	145	1		
6	115	Structure 115	115	1		
7	129	Structure 129	129	1		
8	109	Structure 109	109	1		
9	161	Structure 161	161	1		
10	135	Structure 135	135	1		
11	43	Structure 43	43	1		
12	196	Structure 196	196	1		
13	114	Structure 114	114	1		
14	108	Structure 108	108	1		
15	134	Structure 134	134	1		
16	128	Structure 128	128	1		
17	42	Structure 42	42	1		
18	199	Structure 199	199	1		
19	144	Structure 144	144	1		
20	39	Structure 39	39	1		
21	38	Structure 38	38	1		
22	169	Structure 169	169	1		
23	168	Structure 168	168	1		
24	40	Structure 40	40	1		
25	198	Structure 198	198	1		
26	41	Structure 41	41	1		
27	107	Structure 107	107	1		

Figure 19 – Les nœuds

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
num1	numero d'arc	label	numero	numero	relation	Adjacer	nbfoisa	type_di	dist_Mi	dist_Mi	dist_Mi	dist_Mi	dist_Mi	dist_Mi	dist_Mi	dist_Mi	dist_Mi	dist_Mi	dist_Mi	dist_Mi	dist_Mi	dist_Mi
0	1	0	157	156	0	0	1	0	-1.27586	-2.24138	-0.10345	-1.06897	-0.14286	-1.11429	1.02857	0.057143	0.5	-0.33333	2.16667	1.33333	-1.27586	-2.2
0	2	1	157	45	0	0	1	0	-0.06897	-1.03448	2.31034	1.34483	-1.51429	-2.48571	1.42857	0.457143	0.666667	-0.16667	29.8333	29	-0.06897	-1.0
0	3	2	157	44	0	0	1	0	-2.37931	-3.34483	0	-0.96552	-1.48571	-2.45714	1.45714	0.485714	1	0.166667	29.6667	28.8333	-2.37931	-3.3
0	4	3	157	197	0	0	1	0	0.068966	-0.89655	1.03448	0.068966	-0.62857	-1.6	0.6	-0.37143	1.16667	0.333333	5.5	4.66667	0.068966	-0.8
0	5	4	157	145	0	0	1	0	0.241379	-0.72414	2.03448	1.06897	-0.54286	-1.51429	0.657143	-0.31429	1.16667	0.333333	7.33333	6.5	0.241379	-0.7
0	6	5	157	115	0	0	1	0	-0.06897	-1.03448	0.793103	-0.17241	-0.51429	-1.48571	0.685714	-0.28571	1.16667	0.333333	7.33333	6.5	-0.06897	-1.0
0	7	6	157	129	0	0	1	0	0.724138	-0.24138	1.93103	0.965517	0.342857	-0.62857	1.17143	0.2	1.16667	0.333333	7.5	6.66667	0.724138	-0.2
0	8	7	157	109	0	0	1	0	-0.06897	-1.03448	0.896552	-0.06897	0.142857	-0.82857	0.971429	0	1.16667	0.333333	7.33333	6.5	-0.06897	-1.0
0	9	8	157	161	0	0	1	0	0.206897	-0.75862	1.72414	0.758621	0.628571	-0.34286	1.2	0.228571	1.16667	0.333333	7.33333	6.5	0.206897	-0.7
0	10	9	157	135	0	0	1	0	-0.03448	-1	1.55172	0.586207	0.228571	-0.74286	1.11429	0.142857	1.16667	0.333333	11.8333	11	-0.03448	-1.0
0	11	10	157	43	0	0	1	0	-0.06897	-1.03448	2.31034	1.34483	-1.31429	-2.28571	0.971429	0	1.33333	0.5	29.3333	28.5	-0.06897	-1.0
0	12	11	157	196	0	0	1	0	-1.37931	-2.34483	-0.34483	-1.31034	-0.8	-1.77143	0.6	-0.37143	2.33333	1.5	6.5	5.66667	-1.37931	-2.3
0	13	12	157	114	0	0	1	0	-0.96552	-1.93103	0	-0.96552	-0.57143	-1.54286	0.685714	-0.28571	2.33333	1.5	8.66667	7.83333	-0.96552	-1.5
0	14	13	157	108	0	0	1	0	-1.03448	-2	0	-0.96552	0.028571	-0.94286	0.742857	-0.22857	2.33333	1.5	7.83333	7	-1.03448	-1.0
0	15	14	157	134	0	0	1	0	-1.24138	-2.2069	0.068966	-0.89655	0.171429	-0.8	1.05714	0.085714	2.33333	1.5	12.5	11.6667	-1.24138	-2.0
0	16	15	157	128	0	0	1	0	-2	-2.96552	-0.68966	-1.65517	0.057143	-0.91429	1.08571	0.114286	2.33333	1.5	7.33333	6.5	-2	-2.5
0	17	16	157	42	0	0	1	0	-2.27586	-3.24138	0	-0.96552	-1.22857	-2.2	1.08571	0.114286	2.33333	1.5	29	28.1667	-2.27586	-3.2
0	18	17	157	199	0	0	1	0	0.034483	-0.93104	1.58621	0.62069	-1.42857	-2.4	-0.02857	-1	3	2.16667	12.3333	11.5	0.034483	-0.5
0	19	18	157	144	0	0	1	0	-1.89655	-2.86207	-0.68966	-1.65517	-0.62857	-1.6	0.542857	-0.42857	3	2.16667	7.33333	6.5	-1.89655	-2.8
0	20	19	157	39	0	0	1	0	-0.06897	-1.03448	1.82759	0.862069	0.514286	-0.45714	2.28571	1.31429	3	2.16667	13.6667	12.8333	-0.06897	-1.0
0	21	20	157	38	0	0	1	0	-1.93103	-2.89655	0	-0.96552	0.485714	-0.48571	2.22857	1.25714	3	2.16667	14	13.1667	-1.93103	-2.8
0	22	21	157	169	0	0	1	0	-0.06897	-1.03448	0.862069	-0.10345	-1.42857	-2.4	0.457143	-0.51429	3.33333	2.5	10.5	9.66667	-0.06897	-1.0
0	23	22	157	168	0	0	1	0	-0.96552	-1.93103	0.034483	-0.93104	-1.48571	-2.45714	0.428571	-0.54286	3.33333	2.5	11.1667	10.3333	-0.96552	-1.5
0	24	23	157	40	0	0	1	0	-1.58621	-2.55172	-0.03448	-1	0.657143	-0.31429	2.02857	1.05714	3.66667	2.83333	14.5	13.6667	-1.58621	-2.5
0	25	24	157	198	0	0	1	0	-1.55172	-2.51724	-0.13793	-1.10345	-1.54286	-2.51429	0.257143	-0.71429	3.83333	3	12.3333	11.5	-1.55172	-2.5
0	26	25	157	41	0	0	1	0	-0.10345	-1.06897	1.41379	0.448276	0.685714	-0.28571	2.08571	1.11429	3.83333	3	14.5	13.6667	-0.10345	-1.0
0	27	26	157	107	0	0	1	0	0.862069	-0.10345	2	1.03448	-1	-1.97143	0.114286	-0.85714	4	3.16667	9.33333	8.5	0.862069	-0.1
0	28	27	157	106	0	0	1	0	-2.34483	-3.31034	-0.75862	-1.72414	-1	-1.97143	0.2	-0.77143	4.33333	3.5	11.1667	10.3333	-2.34483	-3.3

Figure 20 – Les arcs

On peut vérifier qu'il enregistre 143 nœuds et 20306 arcs.

Et le temps d'exécution d'étudier la relation entre les deux régions environ égale 3 minutes.

3.1.1 Résultat de relation

Pour bien visualiser les données de relation, j'utilise le tableau croisé dynamique, et le résultat est comme Figure 21 :

Donc, on peut observer que :

- Le nombre de 'relation 2' (100) = le nombre de 'relation 3' (100).

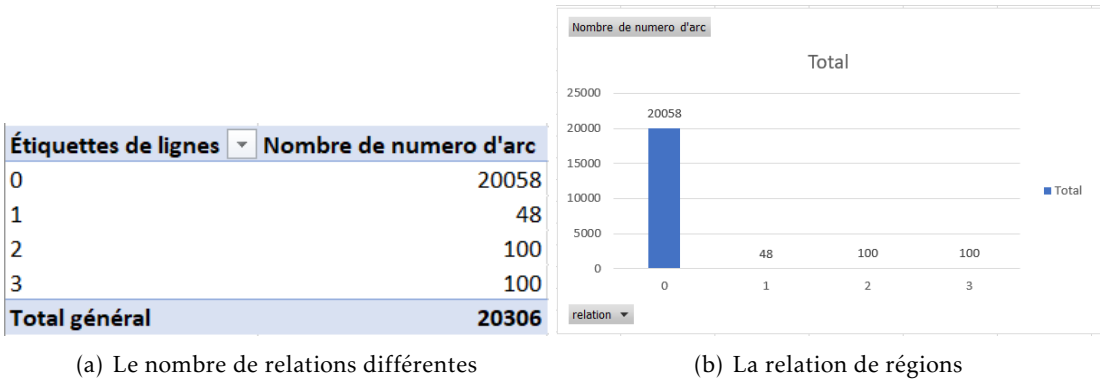


Figure 21 – Le tableau croisé dynamique

- Le nombre de 'relation 1' est un nombre pair (48).

Aussi, pour vérifier si ces relations sont correctes, j'ai fait le test et j'utilise la première version d'apprentissage comme le modèle de référence. Et ce test est valide (plus détails sont dans annexe [Document de tests](#)).

3.2 Analyse le résultat de segmentation

3.2.1 Segmentation incrémentale

Notre logiciel segmente l'image IRM région par région, donc, il y aura plusieurs cas quand on choisir les régions à segmenter.

(1). Cas 1 : la région r_1 ne touche pas la région r_2 .

Par exemple, d'abord, on segmente la région 58 (le label avec la couleur violet). Ensuite, on segmente la région 155 (le label avec la couleur vert). Il sauvegarde ces deux régions sur une même image comme Figure 22.

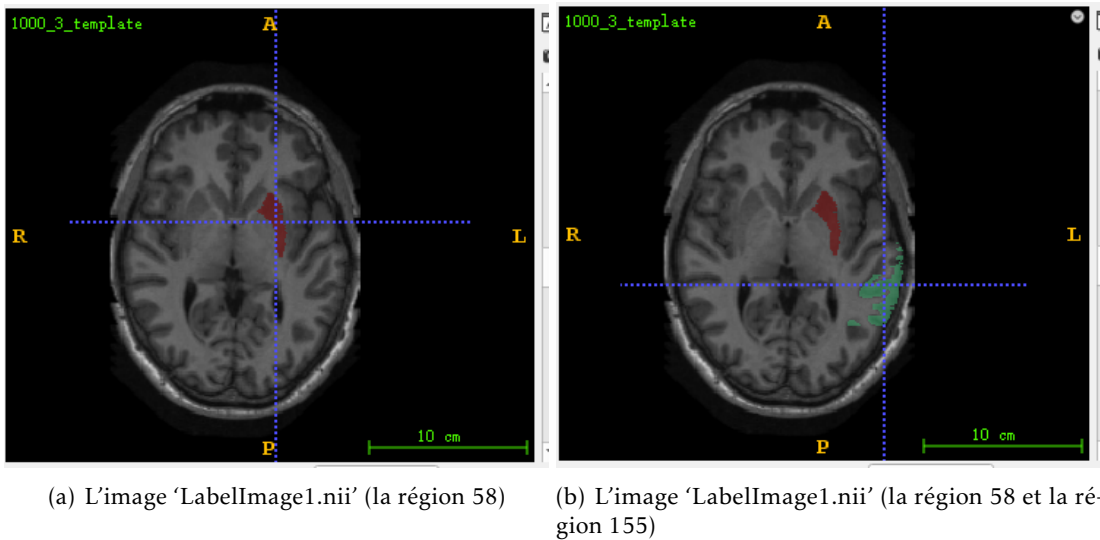


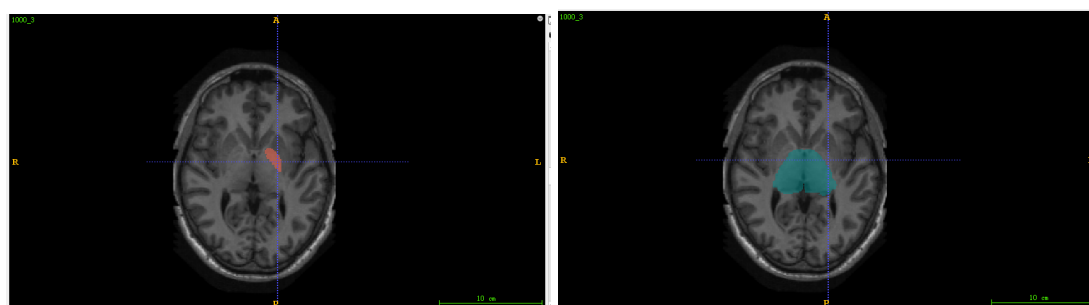
Figure 22 – Le résultat sur une même image

(2). Cas 2 : la région r_1 est imbriquée dans la région r_2 .

Par exemple, d'abord, on segmente la région 56 (le label avec la couleur rose), ensuite, on segmente la région 13 (le label avec la couleur vert) : il y a deux images de résultat (comme Figure 23).

CurrentGraph.lgf	2019/3/21 17:36
LabelImage1.nii.gz	2019/3/21 17:35
LabelImage2.nii.gz	2019/3/21 17:36

Figure 23 – Le dossier de résultat



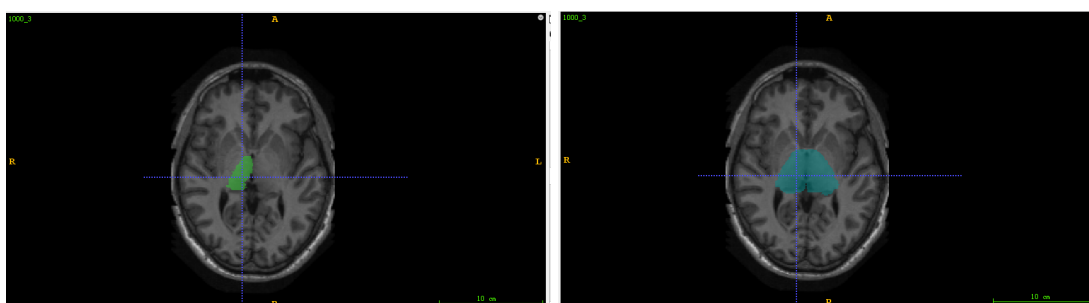
(a) L'image 'LabelImage1.nii' (la région 56)

(b) L'image 'LabelImage2.nii' (la région 13)

Figure 24 – Le résultat sur deux images

(3). Cas 3 : la région r_1 est incluse dans la région r_2 .

Par exemple, d'abord, on segmente la région 59 (le label avec la couleur vert), ensuite, on segmente la région 13 (le label avec la couleur bleu), il y a aussi deux images de résultat (comme Figure 25) :



(a) L'image 'LabelImage1.nii' (la région 59)

(b) L'image 'LabelImage2.nii' (la région 13)

Figure 25 – Le résultat sur deux images



(a) L'image 'LabelImage1.nii' (la région 10)

(b) L'image 'LabelImage2.nii' (la région 154)

Figure 26 – Le résultat sur deux images

(4). Cas 4 : la région r_1 contient la région r_2 .

Par exemple, d'abord, on segmente la région 10 (le label avec la couleur rose), ensuite, on segmente la région 154 (le label avec la couleur vert), il y a aussi deux images de résultat (comme Figure 26).

(5). Un exemple complet de segmentation (le résultat est comme Figure 27 et Figure 28) :

- *- 0 : il signifie que la région R_1 est séparée avec la région R_2 .
- *- 1 : il signifie que la région R_1 est imbriquée dans la région R_2 .
- *- 2 : il signifie que la région R_1 est incluse dans la région R_2 .
- *- 3 : il signifie que la région R_1 contient la région R_2 .

étude (D:) > ImagePRD > imagePreparer > seg > result > 1000Cas1

名称	修改日期	类型	大小
CurrentGraph.lgf	2019/3/20 14:58	LGF 文件	1 KB
LabelImage1.nii.gz	2019/3/20 14:58	好压 GZ 压缩文件	90 KB
LabelImage2.nii.gz	2019/3/20 14:58	好压 GZ 压缩文件	76 KB

Figure 27 – Le résultat de segmentation

```

#nodes
label id nom annotation "method segmentation" "origine 1" "dimension 1" "origine 2" "dimension 2" "origine 3" "dimension 3"
0 13 " " " " 74 62 104 36 108 46
1 57 " " " " 71 22 115 24 130 41
2 155 " " " " 139 39 109 56 83 74
3 6 " " " " 69 60 104 31 69 111
4 59 " " " " 79 26 110 24 116 33
#arcs
label

```

Figure 28 – Le fichier 'CurrentGraph.lgf'

L'ordre de segmentation est : la région $13 \rightarrow 57 \rightarrow 155 \rightarrow 6 \rightarrow 59$.

Les images de résultat est Figure 29.

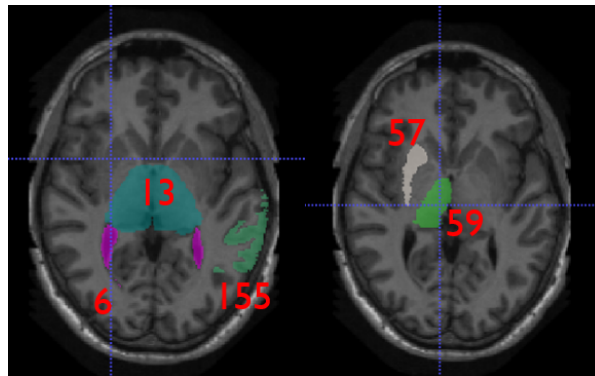


Figure 29 – Le résultat de segmentation

3.2.2 La qualité de segmentation

J'utilise le coefficient Sørensen-Dice pour calculer la distance entre la segmentation qui est obtenue par notre logiciel et la segmentation correcte. J'utilise la méthode 'TestSegmentationResult2 ()' pour tester, et tous les détails sont dans l'annexe Document de tests.

Vers Figure 30, on peut savoir que le coefficient Sørensen-Dice égale 0.979361.



Figure 30 – Test de 'TestSegmentationResult2()'

4 Limite

La limite principale est la performance de segmentation.

(1). La qualité

La qualité de segmentation est affectée par deux points principaux. Un est l'ordre de régions à segmenter, l'autre est de 'Bounding Box de la région'. Donc, pour obtenir le mieux résultat, on doit segmenter à partir la région qui est plus facile à segmenter.

(2). Le temps d'exécution

Le temps d'exécution est affecté principalement par recalage, si le résultat de recalage n'est pas très bon, le temps d'exécution sera plus long. Aussi, si la région est plus grande, le temps d'exécution sera plus long. Et le résultat de recalage est affecté par 'Bounding Box de région'.

En conséquence, quand on segmente la première région, on doit choisir la région qui est plus facile à segmenter. Et aussi, quand on configure 'Bounding Box', on doit le mettre plus précisément et correctement. Si non, la performance sera mauvaise.

6

Bilan et conclusion

1 Bilan du semestre 9

1.1 Faits

Pendant le semestre 9, j'ai fait des tâches comme ci-dessous :

- Gérer le projet, faire la planification du projet.
- Étudier les documents de segmentation médicale 3D à l'Atlas.
- Étudier et analyser le document du logiciel 'NeuroBrainsSeg'.
- Installer et étudier du logiciel 'NeuroBrainsSeg'.
- Lire le code du logiciel, comprendre la structure du logiciel.
- Chercher et trouver la modélisation et la modification du code pour résoudre : si une région est imbriquée ou incluse dans une autre, réaliser la segmentation du logiciel.
- Écrire les documents : l'état de l'art, cahier de spécification et la planification.
- Préparer la soutenance S9.

Tous les détails de ces tâches sont expliqués dans l'annexe – C **Planification**.

Parmi ces tâches, la tâche d'étude du logiciel existant et d'étude de segmentation à l'Atlas m'ont pris beaucoup de temps. Parce que ce sont la base du projet.

1.2 Planning du semestre 10

Les tâches pour le semestre 10 sont listées comme ci-dessous :

- Programmer les fonctionnalités d'étude d'apprentissage.
- Tester et améliorer les fonctionnalités.
- Programmer les fonctionnalités de segmentation incrémentale et interactive.
- Tester et améliorer les fonctionnalités.
- Écrire le rapport final.

- Préparer tous les documents nécessaires pour la soutenance finale.

Tous les détails de la planification sont expliqués dans l'annexe – C **Planification**.

2 Bilan du semestre 10

2.1 Faits

Pendant le semestre 10, j'ai fait des tâches comme ci-dessous :

- Dans la partie d'apprentissage, implémenter la fonctionnalité de charger plusieurs images labellisées.
- Dans la partie d'apprentissage, implémenter la fonctionnalité d'étudier la relation (entre chaque deux régions) qui sera utilisée dans la partie de segmentation.
- Dans la partie de segmentation, implémenter la fonctionnalité de étiqueter le voxel tenant compte de la relation entre les régions.
- Dans la partie de segmentation, implémenter la fonctionnalité de sauvegarder les régions segmentées dans plusieurs images par utiliser les relations entre ces régions.
- Écrire le document d'installation et déploiement.
- Écrire le document d'utilisation.
- Écrire le document de tests.
- Écrire le rapport final.
- Préparer tous les documents nécessaires pour la soutenance finale.

3 Bilan sur la qualité

Pour la partie d'apprentissage, la fonctionnalité d'étudier la relation entre les deux régions execute environ de 3 minutes (plus de 20000 arcs).

Pour la partie de segmentation, la fonctionnalité de sauvegarder les régions segmentées sur la disque execute bien. En plus, le coefficient de Dice de segmentation est supérieure à 95% (Les détails sont dans l'annexe **Document de tests**).

Mais, pour la partie de segmentation, la fonctionnalité d'obtenir 'Bounding Box' automatiquement ne fonctionne pas bien, le résultat de cette fonctionnalité n'est pas très précis. En plus, si la région est plus grande, le temps d'exécution sera plus long.

4 Bilan auto-critique su la gestion du projet

Au cours de l'année, j'ai fait mon PRD par suivre la planification de projet pour bien mener mon projet. Bien que j'ai rencontré de nombreux problèmes, tels que la compréhension du sujet, la théorie Atlas de la segmentation d'images médicales ou la rédaction du rapport, etc. Cependant, grâce à l'aide de l'encadrant M.Galisot et M.Ramel, ils m'ont expliqué très patiemment, ces problèmes ont été résolus. Je voudrais remercier M.Galisot et M.Ramel pour leurs aides sur la compréhension et la modélisation de projet.

Annexes

A

Spécification fonctionnelle

1 Description des fonctionnalités du programme d'apprentissage

1.1 Définition de la fonction 1 : readLabelImages

Identification de la fonction 1

Cette fonction permet de charger plusieurs images labellisées passés en entrée du programme.

Description de la fonction 1.

Entrée : le chemin de dossier qui contient plusieurs images labellisées.

Retourne : un objet de la classe 'MultiImage'.

Explication de la fonction 1

Maintenant, chaque fois, le logiciel ne peut que charger une image labellisée, donc, je vais modifier des attributs de classe 'GraphLearning' et les fonctions de classe 'GraphLearning' pour charger plusieurs images à la fois.

1.2 Définition de la fonction 2 : findRelations

Identification de la fonction 2

Cette fonction permet d'étudier les relations entre chaque deux régions.

Description de la fonction 2

Entrée : on objet de la classe 'MultiImage' qui enregistre toutes les images labellisées.

Retourne : une liste de 'int' qui enregistre toutes les relations entre chaque deux régions.

Explication de la fonction 2

pour trouver les relations entre chaque deux régions, je vais ajouter une fonction dans la classe 'GrapheLearning' qui permet de trouver les relations.

1.3 Définition de la fonction 3 : createGraph

Identification de la fonction 3

Cette fonction permet d'enregistrer les relations dans l'objet de classe 'BrainModel'.

Description de la fonction 3

Entrée : une liste de 'int' qui enregistre toutes les relations entre chaque deux régions.

Retourne : un objet de classe 'BrainModel' qui enregistre toutes les informations de graphe de connaissance a priori modifié.

Explication de la fonction 3

Maintenant, dans le logiciel, il y a déjà une fonction qui permet de créer le graphe. Mais, j'ajouterai un attribut dans ce graphe. Donc, je vais modifier des fonctions de la classe 'BrainModel' et de la classe 'GraphLearning' pour enregistrer les relations dans ce graphe.

1.4 Définition de la fonction 4 : readGraph

Identification de la fonction 4

Cette fonction permet de charger graphe de connaissance a priori modifié vers le fichier 'Model.lgf'.

Description de la fonction 4

Entrée : le fichier 'Brain_Model.lgf' qui enregistre toutes les informations de Graph.

Retourne : un objet de classe 'BrainModel' qui enregistre le graphe.

Explication de la fonction 4

Maintenant, dans le logiciel, il y a déjà une fonction qui permet de lire le graphe. Mais, j'ajouterai un attribut dans ce graphe. Donc, je vais modifier la fonction 'load ()' de la classe 'BrainModel' pour lire les relations vers le fichier et les enregistrer dans l'objet de 'BrainModel'.

1.5 Définition de la fonction 5 : updateGraph

Identification de la fonction 5

Cette fonction permet de mettre à jour le graphe.

Description de la fonction 5

Entrée : un objet de classe 'BrainModel'.

Retourne : un objet de classe 'BrainModel'.

Explication de la fonction 5

Maintenant, dans le logiciel, il y a déjà une fonction qui permet de mettre à jour le graphe. Mais, j'ajouterai un attribut dans ce graphe. Donc, je vais modifier la fonction 'process ()' de classe 'GraphLearning'.

2 Description des fonctionnalités du programme de segmentation

2.1 Définition de la fonction 1 : classVoxels

Identification de la fonction 1

Cette fonction permet de classifier et labelliser les voxels.

Description de la fonction 1

Entrée : l'image à segmenter.

Retourne : un objet de classe 'MltiImage' qui enregistre images de résultat de segmentation. Chaque image correspond une région.

Explication de la fonction 1

Maintenant, dans le logiciel, il y a déjà une fonction qui permet de classifier les voxels. Mais, si le voxel est déjà labellisé et il appartient de deux régions, nous ne pouvons pas relabelliser. Donc, pour réaliser l'objectif du projet et déterminer si ce voxel est dans deux régions, ou dans le fond, ou juste dans une région, je modifierai la fonction 'LocalToGlobal()' de classe 'SegmentationResultBase'.

2.2 Définition de la fonction 2 : saveSegmentation

Identification de la fonction 1

Cette fonction permet de sauvegarder le résultat de segmentation sur image de type '. nii.gz'.

Description de la fonction 2

Entrée : un objet de 'MultiImage' qui enregistre tous les résultats de segmentation.

Retourne : plusieurs images de segmentation, chaque image correspond une région.

Explication de la fonction 2

Maintenant, le logiciel juste sauvegarde une seule image, et toutes les régions segmentées sont présentées sur cette image. Donc, pour réaliser de sauvegarder les régions, si certaine région est imbriquée ou incluse dans l'autre région, je vais sauvegarder une région par une image. Donc, il a besoins de modifier la fonction 'saveSegmentationState()' de classe 'SegmentationResultBase'.

2.3 Définition de la fonction 3 : readSegmentation

Identification de la fonction 1

Cette fonction permet de charger les images (format '. nii.gz') de segmentation.

Description de la fonction 3

Entrée : le fichier 'currentGraph.lgf' qui enregistre toutes les informations de l'état de segmentation, les images (format '. nii.gz') de régions segmentées.

Retourne : un objet de classe 'SegmentationResultBase' qui enregistre l'état de segmentation et les images segmentés.

Explication de la fonction 3

Pour charger plusieurs images de segmentation à la fois, il a besoins de modifier la fonction 'loadSegmentationState()' de la classe 'SegmentationResultBase'.

B

B. Spécification non fonctionnelle

1 Performances

- Pour l'utilisateur : le résultat de segmentation est influencé par l'ordre de segmentation. L'ordre de segmentation est défini par l'utilisateur. Pour obtenir la mieux qualité de segmentation, l'utilisateur faudrait segmenter à partir des régions plus simples (la probabilité de présence de la plupart de voxels est plus grande.) à segmenter.

- Pour logiciel : maintenant, le logiciel est développé en C++ sur Windows, pour l'exécuter sur Linux, il faudrait changer des codes, par exemple, le séparateur de chemin.

2 Capacités

En ce moment, le logiciel sauvegarde toutes les images avec les régions segmentées et l'état de segmentation sur mémoires. Jusqu'à que l'utilisateur tape la commande 'saveSegmentation', le logiciel sauvegarde toutes les images sur disque. Donc, cela occupe beaucoup de mémoire pendant l'exécution. Après segmenter 5 régions, cela occupe 1Go de mémoires. Mais si on a besoins de segmenter plusieurs régions, on pourrait sauvegarder tous les images et l'état de segmentation sur disque.

3 Sécurité

Maintenant, il n'y a pas de problème de niveau de confidentialité du système.

C

Planification

Ce chapitre présente la planification de projet.

1 Découpage en tâche

Pour gérer le projet, je découpe ce projet en plusieurs tâches.

1.1 Tâche 1 : Gestion de projet

Description de la tâche

Durant cette tâche, il faut d'abord connaître le projet, gérer et planifier le projet.

Livrables

Un compte rendu sera à rendre.

Estimation de charge

Cette tâche est estimée à une semaine/homme.

1.2 Tâche 2 : Étudier d'état de l'art de segmentation médicale

Description de la tâche

Durant cette tâche, il faut recherche et lire les documents de segmentation médicale.

Livrables

Un compte rendu sera à rendre.

Estimation de charge

Cette tâche est estimée à une semaine/homme.

1.3 Tâche 3 : Étudier le document de logiciel existant

Description de la tâche

Cette tâche est découpée à deux petites tâches (tâche 4 et tâche 5).

Livrables

A la fin de cette tâche, une présentation de segmentation médicale à Atlas sera à faire à l'encadrant.

Estimation de charge

Cette tâche est estimée à deux semaines/homme.

1.4 Tâche 4 : Étudier modélisation méthode d'atlas

Description de la tâche

Durant cette tâche, il faut étudier le document de logiciel. Il faut étudier et comprendre l'Atlas. Aussi, il faut comprendre comment modéliser pour faire segmentation dans ce logiciel. C'est-à-dire, il faut bien comprendre les étapes d'obtenir les connaissances a priori.

Livrables

Un compte rendu sera à rendre.

Estimation de charge

Cette tâche est estimée à une semaine/homme.

1.5 Tâche 5 : Étudier segmentation méthode de segmentation

Description de la tâche

Il faut étudier quelles méthodes sont utilisées pour faire segmentation. Aussi, il faut étudier le processus (les étapes) de faire segmentation dans le logiciel.

Livrables

Faire une présentation de segmentation à Atlas.

Estimation de charge

Cette tâche est estimée à une semaine/homme.

1.6 Tâche 6 : Installer et compiler le logiciel

Description de la tâche

Durant cette tâche, il faut installer et compiler le logiciel sur l'ordinateur personnel. Aussi, il faut savoir quelles libraires sont utilisées pour programmer ce logiciel.

Livrables

Présenter le résultat à l'encadrant.

Estimation de charge

Cette tâche est estimée à une semaine/homme.

1.7 Tâche 7 : Étudier et utiliser le logiciel

Description de la tâche

Durant cette tâche, il faut comprendre la structure de logiciel et savoir comment utiliser ce logiciel.

Livrables

Un compte rendu sera à rendre par semaine. Dans le compte rendu, il faut décrire les résultats d'utiliser ce logiciel.

Estimation de charge

Cette tâche est estimée à deux semaines/homme.

1.8 Tâche 8 : Modifier le modèle de graphe

Description de la tâche

Durant cette tâche, il faut trouver la méthode de modifier le modèle de graphe pour réaliser d'enregistrer les relations entre chaque deux régions (une région = un composant de cerveaux).

Livrables

Un compte rendu sera à rendre. Dans ce compte rendu, il faut décrire comment modifier le modèle de graphe.

Estimation de charge

Cette tâche est estimée à une semaine/homme.

1.9 Tâche 9 : Modifier le modèle de segmentation

Description de la tâche

Durant cette tâche, il faut réfléchir et trouver la méthode de segmentation, de sauvegarder le résultat de segmentation.

Livrables

Un compte rendu sera à rendre par semaine. Dans ce compte rendu, il faut décrire comment modifier les codes existants pour réaliser de segmentation à plusieurs niveaux.

Estimation de charge

Cette tâche est estimée à deux semaines/homme.

1.10 Tâche 10 : Validation des modifications

Description de la tâche

Présenter et expliquer les modifications à l'encadrant.

Livrables

Valider les modifications par l'encadrant.

Estimation de charge

Cette tâche est estimée à une semaine/homme.

1.11 Tâche 11 : Rédaction du rapport final mi-parcours**Description de la tâche**

Écrire le document : rapport final mi-parcours.

Livrables

Rapport final mi-parcours sera à rendre.

Estimation de charge

Cette tâche est estimée à une semaine/homme.

1.12 Tâche 12 : Préparation de la soutenance mi-parcours**Description de la tâche**

Préparer les documents de la soutenance mi-parcours.

Livrables

Soutenance de mi-parcours.

Estimation de charge

Cette tâche est estimée à une semaine/homme.

1.13 Tâche 13 : Programme de modification d'apprentissage et tests**Description de la tâche**

Programmer sur logiciel pour réaliser les fonctions : charger plusieurs images labellisées à la fois ; étudier les relations entre chaque deux régions ; charger, mettre à jour et enregistrer graphe.

Livrables

Un compte rendu sera à rendre par semaine. À la fin, il faut être validé par l'encadrant.

Estimation de charge

Cette tâche est estimée à quatre semaines/homme.

1.14 Tâche 14 : Programme de modification de segmentation et tests**Description de la tâche**

Programmer sur logiciel pour réaliser les fonctions : classifier les voxels par utiliser les relations entre les régions ; enregistrer et charger les résultats de segmentation.

Livrables

Un compte rendu sera à rendre par semaine. À la fin, il faut être validé par l'encadrant.

Estimation de charge

Cette tâche est estimée à quatre semaines/homme.

1.15 Tâche 15 : Validation des programmes**Description de la tâche**

Durant cette tâche, il faut assurer qu'il n'a pas des erreurs et warnings dans code. Aussi, le résultat doit être correcte et validée par l'encadrant.

Livrables

Les codes ressources seront à rendre.

Estimation de charge

Cette tâche est estimée à une semaine/homme.

1.16 Tâche 16 : Rédaction du rapport final**Description de la tâche**

Écrire le rapport final du projet.

Livrables

Le rapport final sera à rendre.

Estimation de charge

Cette tâche est estimée à une semaine/homme.

1.17 Tâche 17 : Préparation de la soutenance finale**Description de la tâche**

Préparer tous les documents de la soutenance finale : rapport, PPT.

Livrables

Soutenance finale.

Estimation de charge

Cette tâche est estimée à une semaine/homme.

2 Diagramme de Gantt

Le diagramme de Gantt est comme ci-dessous - Figure 1.

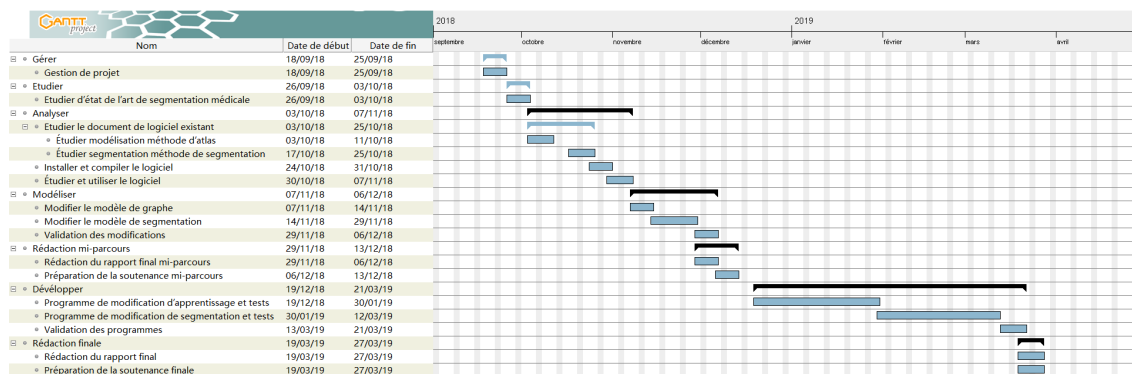


Figure 1 – Le diagramme de Gantt

D

Document d'installation et de déploiement

1 Outils et libraires

Je vais présenter les informations (par exemple, la fonctionnalité, la version, etc.) d'outils et de librairies que j'utilise pour mon projet.

1.1 IDE

Ce projet est programmé en C++, sur Visual studio (version 2017).



Figure 1 – IDE

1.2 ITK

ITK est un système multiplateforme à source ouverte offrant aux développeurs une suite complète d'outils logiciels pour l'analyse d'images. ITK fournit un certain nombre de classes et de méthodes de traitement d'images que les développeurs peuvent installer, compiler et appeler directement. Les développeurs peuvent télécharger des fichiers binaires ou des archives de code source dans le web site de ITK ou via Git. Dans ce cas-là, j'utilise ITK source (version 4.13.1).



Figure 2 – ITK

1.3 ITK SNAP

ITK-SNAP est une application logicielle utilisée pour segmenter des structures dans des images médicales 3D. Avec ce logiciel, nous pouvons observer des images, étiqueter des parties de l'image et segmenter l'image. J'utilise ITK SNAP (version 3.8) pour visualiser les images médicales.



Figure 3 – ITK SNAP

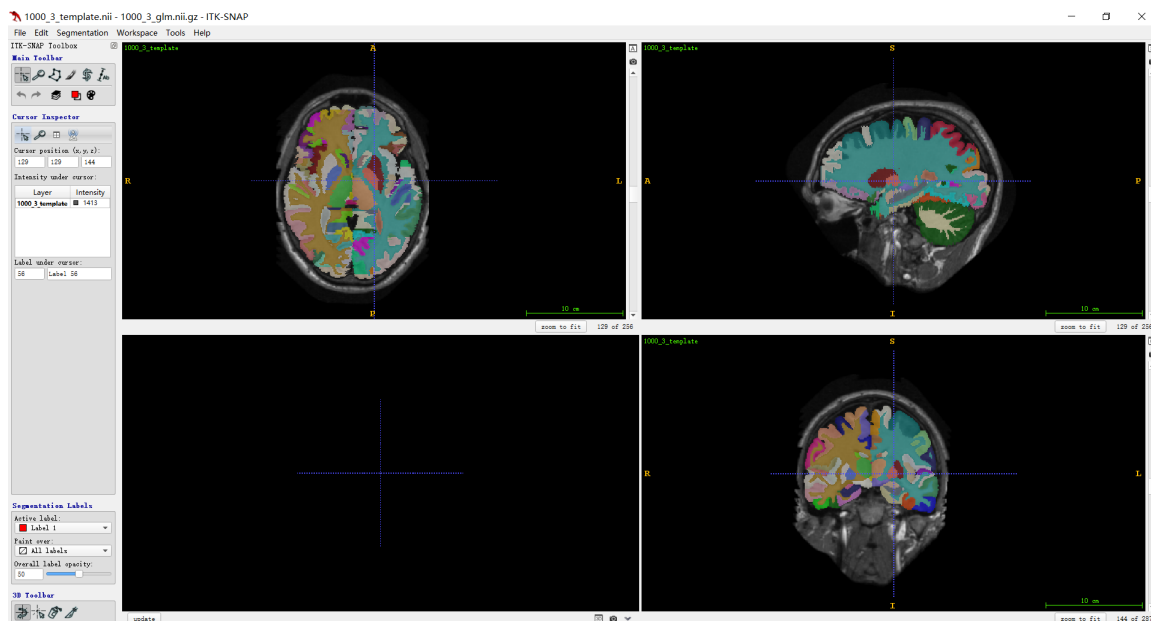


Figure 4 – Visualiser l'image avec ITK SNAP

1.4 Lemon graph library

LEMON est une librairie pratique pour la modélisation efficace et l'optimisation dans les réseaux. Il s'agit d'une librairie de modèles C++ offrant de nombreuses classes et méthodes de traitement des graphes que les développeurs peuvent installer, compiler et appeler directement. Dans ce cas-là, j'utilise Lemon graph library (version 1.3.1)



Figure 5 – Lemon

1.5 CMake

CMake est une famille d'outils multi-plateformes, à source ouverte, conçues pour créer, tester et conditionner des logiciels. On utilise CMake pour compiler le code source.



Figure 6 – CMake

2 Installation

Je vais présenter les étapes d'installation.

(1). D'abord, il faut modifier le fichier 'CMakeLists.txt'. Il faut configurer des paramètres comme Figure 7, Figure 8 et Figure 9 :

- Le chemin de librairie ITK :

```
#LXX ADD BEGIN: ITK
SET(ITK_DIR F:/informatique3/ITK/InsightToolkit-4.13.1/bin/lib/cmake/ITK-4.13/)
#LXX ADD END: ITK
```

Figure 7 – Modifier le chemin de ITK

- Le chemin de librairie Lemon :

```
SET(LEMON_INCLUDE_DIR F:/informatique3/LemonGraph/lemon-1.3.1/lemonbuild/ CACHE PATH "LEMON include directory")
SET(LEMON_INCLUDE_DIRS "${LEMON_INCLUDE_DIR}")
```

Figure 8 – Modifier le chemin de Lemon 1

```
SET(LEMON_LIBRARY F:/informatique3/LemonGraph/lemon-1.3.1/lemonbuild/lemon/Debug/${LEMON_LIB_NAME} CACHE FILEPATH "LEMON library")
SET(LEMON_LIBRARIES "${LEMON_LIBRARY}")
FILE(GLOB SRC ${CMAKE_CURRENT_SOURCE_DIR}/src/*.cpp)
```

Figure 9 – Modifier le chemin de Lemon 2

Après il faut modifier le fichier 'CMakeLists.txt', on doit le copier dans le dossier de code source. Par exemple, dans notre projet, il faut le copier dans le dossier 'source' (comme Figure 10).

étude (D:) > VS2017_workspace > NeuroGeo > source				
名称	修改日期	类型	大小	
src	2019/3/17 22:37	文件夹		
CMakeLists.txt	2018/10/25 12:03	文本文档	2 KB	

Figure 10 – Copier 'CMakeLists.txt'

(2). Utiliser CMake (comme Figure 11) :

- 'Where is the code source' : il faut entrer le chemin de dossier qui contient le fichier 'CMakeLists.txt'. Par exemple, 'D : \VS2017_workspace\NeuroGeo\source'.
- 'Where to build the binaire' : il faut entrer le chemin de dossier où il construit le projet. Par exemple, 'D : \VS2017_workspace\NeuroGeo\source\build'.
- Après, on peut lancer Cmake pour compiler le code source.

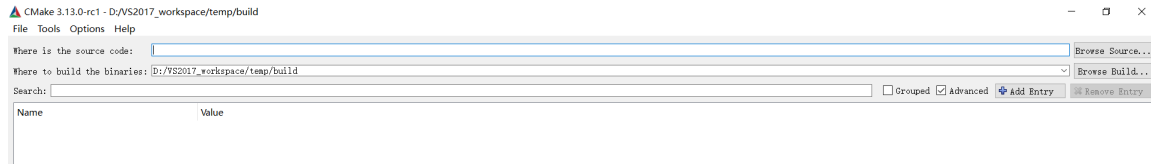


Figure 11 – Configurer CMake

(3). Après, on peut trouver qu'il crée le fichier 'NeuroGeo.sln'. On clique sur ce fichier 'NeuroGeo.sln' directement pour programmer.

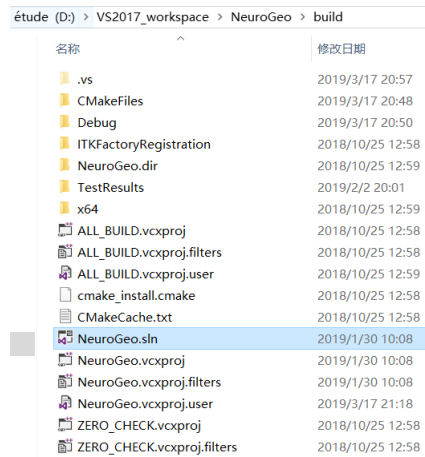


Figure 12 – Exécuter 'NeuroGeo.sln'

(4). En plus, dans le code de la méthode 'getBoundingFromTruth()' de la classe 'LocalSegmentationMethod' (ligne 481), il faut modifier le chemin de image de segmentation vérité.

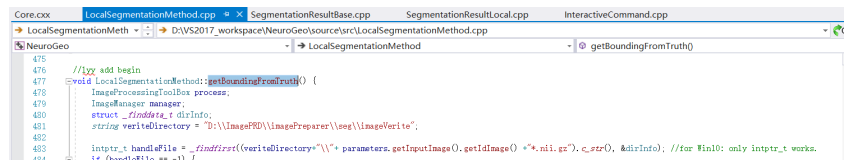


Figure 13 – Modifier le chemin de segmentations vérités

(5). Aussi, il faut changer le chemin de sauvegarder l'image 'global_HMFtest.nii.gz' (ligne 150, la méthode 'LocalToGlobal', la classe 'SegmentationResultLocal').



Figure 14 – Modifier le chemin de sauvegarde

(6). Parce que maintenant, la fonctionnalité de 'AutomaticPositioningBoundingBox ()' ne fonctionne pas bien, donc, on utilise la segmentation vérité pour obtenir 'Bounding Box' (la méthode 'getBoundingFromTruth ()').

Pour utiliser et tester la fonctionnalité de 'AutomaticPositioningBoundingBox ()', il faut commenter la ligne 129, sinon, il toujours obtenir 'Bounding Box' vers la segmentation vérité.

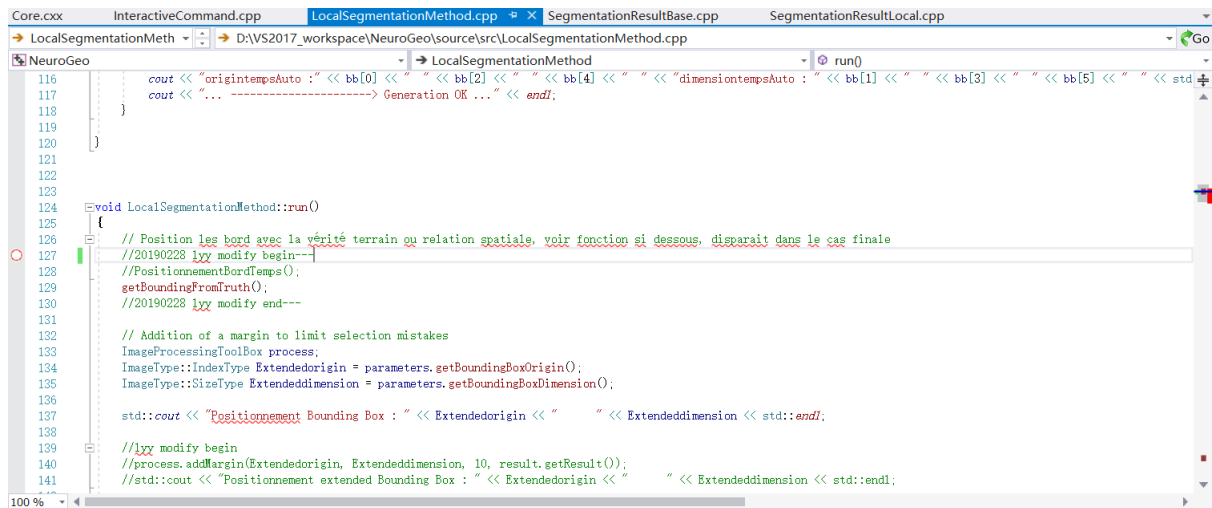
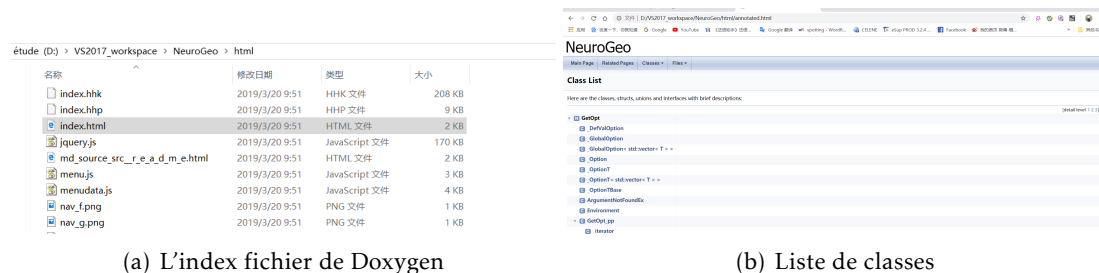


Figure 15 – *Commenter la ligne 129*

3 Document Doxygen

Pour bien connaître la structure de projet et lire le code, vous pouvez cliquer sur le fichier 'index.xml' dans le dossier 'html'. On peut obtenir toutes les informations de classes et de méthodes.

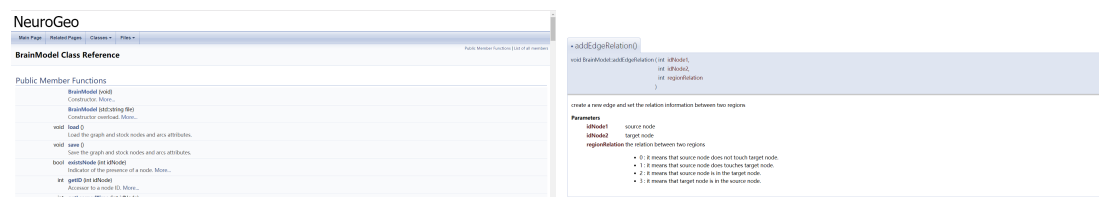


(a) L'index fichier de Doxygen

(b) Liste de classes

Figure 16 – Doxygen

Par exemple, la page (comme Figure 17) écrit la classe ‘BrainModel’.



(a) La classe 'BrainModel'

(b) La méthode 'addEdgeRelation()' de la classe 'BrainModel'

Figure 17 – Les Classes et les méthodes

4 Explications de Classe

Il y a pleines de classes dans notre projet, je vais présenter des classes principales.

(1). Les classes principales d'apprentissage :

- La classe 'Image' : cela permet de créer un objet d'image.
- La classe 'MultiImage' : cela permet de créer un 'Map' objet qui contient plusieurs objets d'image.
- La classe 'GraphLearning' : la classe principale qui initialise et exécute d'apprentissage. C'est comme le contrôleur d'apprentissage.
- La classe 'BrainModel' : il crée un graphe pour sauvegarder le numéro de régions, la distance entre de régions, la relation entre de régions, etc.
- La classe 'Core' : c'est la classe principale de recevoir et traiter les paramètres d'entrés.

(2). Les classes principales de segmentation :

- La classe 'InteractiveCommand' : il définit et configure les paramètres de fonctionnalité de segmentation.
- La classe 'LocalSegmentationMethod' : c'est la classe principale qui initialise et exécute la segmentation. C'est comme le contrôleur de segmentation.
- La classe 'SegmentationResultLocal' : dans cette classe, la méthode 'LocalToGlobal()' est pour étiqueter les voxels.
- La classe 'SegmentationResultBase' : il contient des fonctionnalités : sauvegarder le résultat de segmentation, lire la segmentation, supprimer quelque région qui est déjà segmentée.

Après mes implémentations, le diagramme est comme Figure 18.

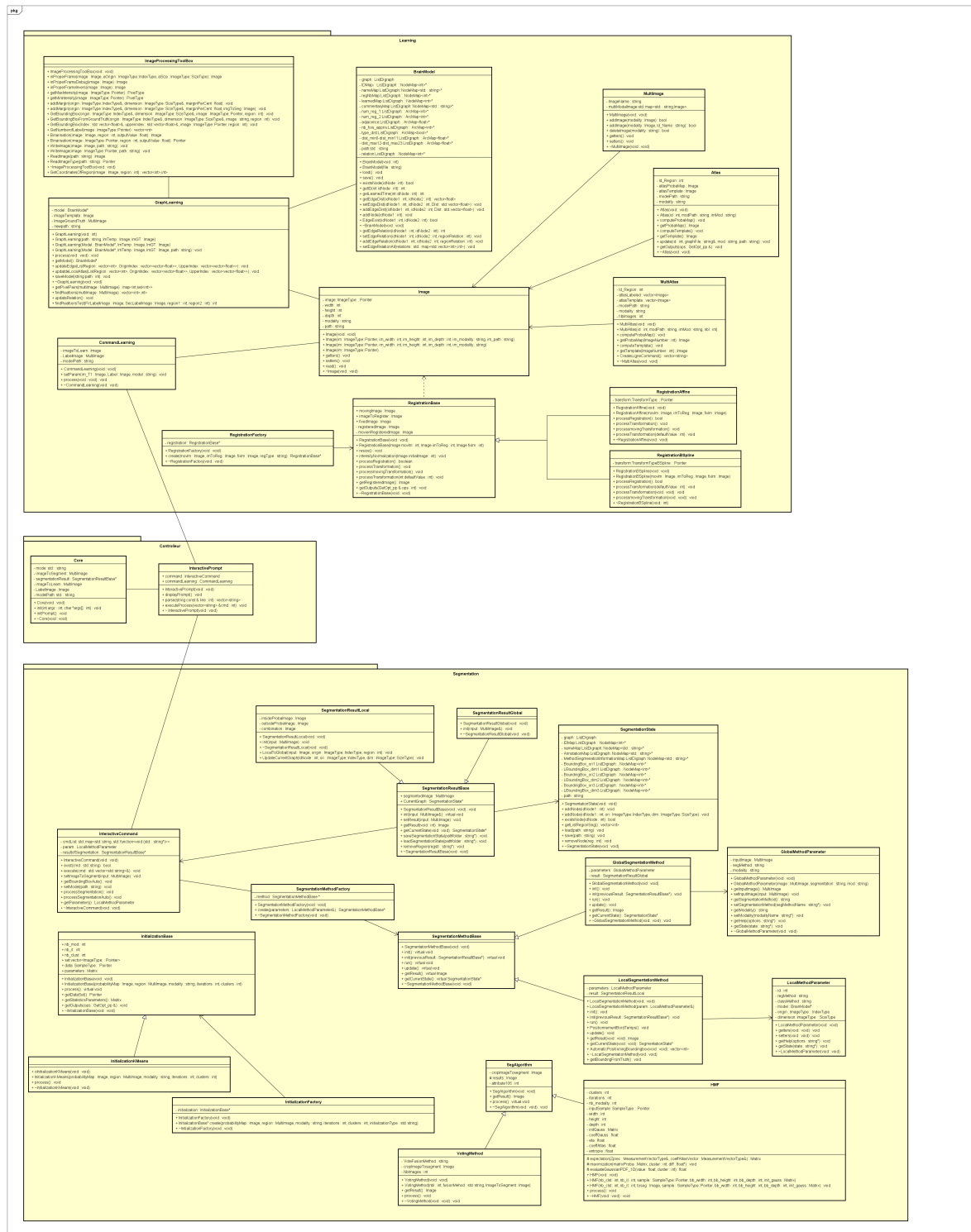


Figure 18 – Le diagramme de classe

E

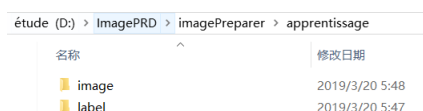
Document d'utilisation

1 Utilisation d'apprentissage

Dans ce chapitre, je vais présenter comment on utilise notre logiciel pour faire d'apprentissage d'image médicale 3D.

1.1 Préparations

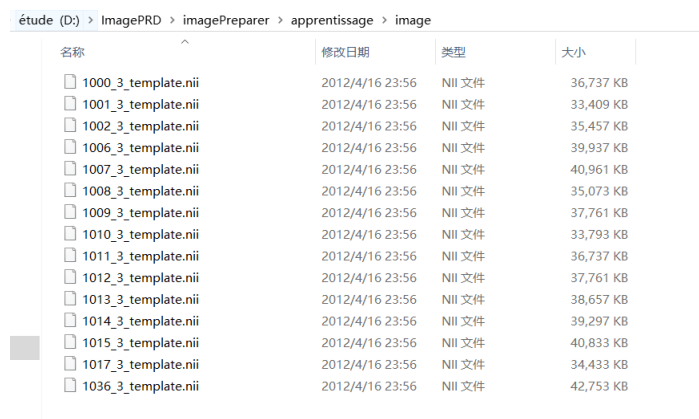
Pour faire d'apprentissage, on a deux dossiers, un est pour enregistrer les images à étudier, l'autre est pour enregistrer les images labellisées.



nom	modification
image	2019/3/20 5:48
label	2019/3/20 5:47

Figure 1 – Images et images labellisées

- Le dossier 'image' : il contient plusieurs images à étudier.



nom	modification	type	taille
1000_3_template.nii	2012/4/16 23:56	NII 文件	36,737 KB
1001_3_template.nii	2012/4/16 23:56	NII 文件	33,409 KB
1002_3_template.nii	2012/4/16 23:56	NII 文件	35,457 KB
1006_3_template.nii	2012/4/16 23:56	NII 文件	39,937 KB
1007_3_template.nii	2012/4/16 23:56	NII 文件	40,961 KB
1008_3_template.nii	2012/4/16 23:56	NII 文件	35,073 KB
1009_3_template.nii	2012/4/16 23:56	NII 文件	37,761 KB
1010_3_template.nii	2012/4/16 23:56	NII 文件	33,793 KB
1011_3_template.nii	2012/4/16 23:56	NII 文件	36,737 KB
1012_3_template.nii	2012/4/16 23:56	NII 文件	37,761 KB
1013_3_template.nii	2012/4/16 23:56	NII 文件	38,657 KB
1014_3_template.nii	2012/4/16 23:56	NII 文件	39,297 KB
1015_3_template.nii	2012/4/16 23:56	NII 文件	40,833 KB
1017_3_template.nii	2012/4/16 23:56	NII 文件	34,433 KB
1036_3_template.nii	2012/4/16 23:56	NII 文件	42,753 KB

Figure 2 – Images à étudier

- Le dossier 'label' : il contient plusieurs dossiers, comme Figure 3.

étude (D:) > ImagePRD > imagePreparer > apprentissage > label			
名称	修改日期	类型	大小
1000	2019/3/20 5:47	文件夹	
1001	2019/3/20 5:47	文件夹	
1002	2019/3/20 5:47	文件夹	
1006	2019/3/20 5:47	文件夹	
1007	2019/3/20 5:47	文件夹	
1008	2019/3/20 5:47	文件夹	
1009	2019/3/20 5:47	文件夹	
1010	2019/3/20 5:47	文件夹	
1011	2019/3/20 5:47	文件夹	
1012	2019/3/20 5:47	文件夹	
1013	2019/3/20 5:47	文件夹	
1014	2019/3/20 5:47	文件夹	
1015	2019/3/20 5:47	文件夹	
1017	2019/3/20 5:47	文件夹	
1036	2019/3/20 5:47	文件夹	

Figure 3 – Dossiers d'images labellisées

Chaque dossier contient des images labellisées (comme Figure 4). Il faut faire attention que les images labellisées dans le même dossier doivent provenir de la même image à étudier.

étude (D:) > ImagePRD > imagePreparer > apprentissage > label > 1000			
名称	修改日期	类型	大小
1000_3_glm.nii.gz	2012/9/5 16:28	好压 GZ 压缩文件	312 KB
1000_4.nii.gz	2019/1/29 15:05	好压 GZ 压缩文件	209 KB

Figure 4 – Images labellisées d'image '1000_3_template.nii'

1.2 Exécution

Les étapes d'utilisation d'apprentissage sont comme ci-dessous :

(1). Exécuter le fichier exécutable 'NeuroGeo.exe' avec les paramètres ci-dessous :

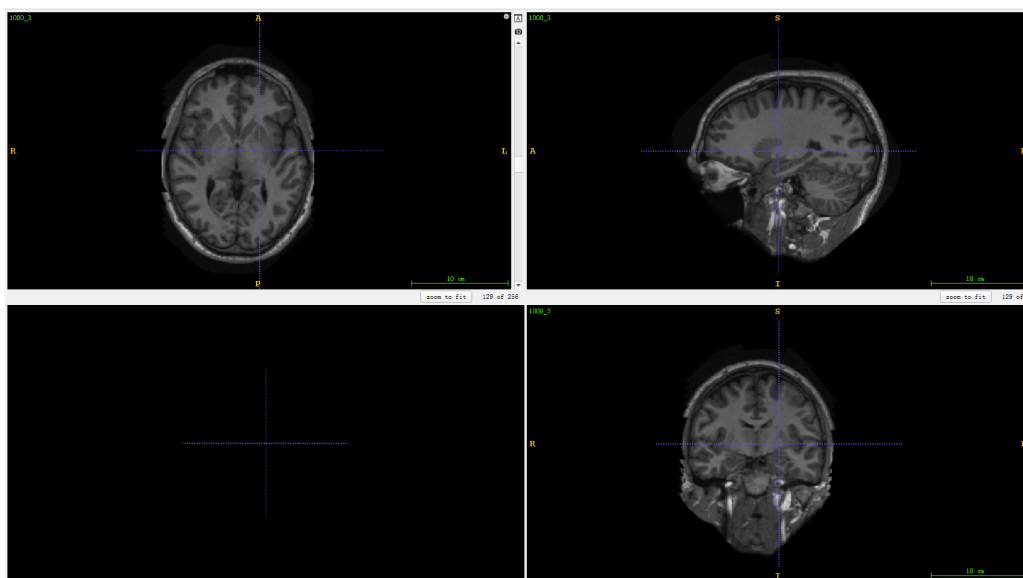


Figure 5 – L'image '1000_3_template.nii'

- -MODE Learning

C'est-à-dire qu'on exécute la fonctionnalité d'apprentissage.

- -INPUT_T1 D : \ImagePRD\imagePreparer\apprentissage\image\1000_3_template.nii

Il faut entrer le chemin d'image IRM noté comme L.

- -LABEL_IM D : \ImagePRD\imagePreparer\apprentissage\label\1000

Il faut entrer le chemin de dossier qui contient les images labellisées. Toutes ces images labellisées doivent être associées avec l'image L.



(a) L'image labellisée '1000_3_glm.nii.gz'

(b) L'image labellisée '1000_4.nii.gz'

Figure 6 – Images labellisées d'image '1000_3_template.nii'

- -MODEL_PATH D : \ImagePRD\imagePreparer\apprentissage\result\1000

Il faut entrer le chemin où on veut sauvegarder le résultat d'apprentissage.

```
D:\VS2017_workspace\NeuroGeo\build\Debug>NeuroGeo.exe --MODE Learning --INPUT_T1 D:\ImagePRD\imagePreparer\apprentissage\image\1000_3_template.nii --LABEL_IM D:\ImagePRD\imagePreparer\apprentissage\label\1000 --MODEL_PATH D:\ImagePRD\imagePreparer\apprentissage\result\1000
... Creating Model ...
... Local Atlas Generation ...
region : 157
region : 156
region : 45
region : 44
```

Figure 7 – La commande complète

(2). Le processus d'exécution :

```
region : 6
region : 13
region : 19
...> Generation OK ...
... Spatial Relationships Generation ...
...> Generation OK ...
... Region Relationships Generation ...
end getPixelPairs
...> Generation OK ...
D:\VS2017_workspace\NeuroGeo\build\Debug>
```

Figure 8 – Le processus d'exécution

1.3 Visualisation de résultat

Après l'exécution, on peut obtenir le résultat comme Figure 9 :

- 'Edges' et 'Nodes' : ils contiennent tous les Atlas local et toutes les cartes de probabilités.
- 'Brain_Model.lgf' : c'est un fichier qui sauvegarde toutes les informations de régions. Tous ces informations sont modélisée comme un graphe.

étude (D:) > ImagePRD > imagePreparer > apprentissage > result > 1000			
名称	修改日期	类型	大小
Edges	2019/3/20 5:51	文件夹	
Nodes	2019/3/20 5:58	文件夹	
Brain_Model.lgf	2019/3/20 6:01	LGF 文件	4,478 KB

Figure 9 – Le résultat d'apprentissage

- Le contenu du fichier 'Brain_Model.lgf' : il y a deux parties principales.

(1). '@nodes' : ce sont des numéros de régions.

@nodes						
label	id	nom	numeroRegion	fois_appris	commentaires	
0	157	"Structure 157"	157	1	""	
1	156	"Structure 156"	156	1	""	
2	45	"Structure 45"	45	1	""	
3	44	"Structure 44"	44	1	""	
4	197	"Structure 197"	197	1	""	
5	145	"Structure 145"	145	1	""	
6	115	"Structure 115"	115	1	""	
7	129	"Structure 129"	129	1	""	

Figure 10 – Les nœuds

(2). '@arcs' : chaque ligne représente un arc, le contenu de la colonne 'relation' représente la relation entre chaque deux régions.

Explications du contenu de la colonne 'relation' :

- *- 0 : il signifie que la région R_1 est séparée avec la région R_2 .
- *- 1 : il signifie que la région R_1 est imbriquée dans la région R_2 .
- *- 2 : il signifie que la région R_1 est incluse dans la région R_2 .
- *- 3 : il signifie que la région R_1 contient la région R_2 .

@arcs													
	label	numeroRegion1	numeroRegion2	relation	Adjacence	nbfoisappris	type_dist	dist_Min0	dist_Min1	dist_Min2	dist_Min3	dist_Min4	dist_Min5
0	1	157	156	0	1	0	-1.27586	-2.24138	-0.103448	-1.06897	-0.142857	-1.11429	1.02857 0.0571429 0.5 -0.333333 2.16667
0	2	1	157	45	0	1	0	-0.0689655	-1.03448	2.31034	1.34483	-1.51429	-2.48571 1.42857 0.457143 0.666667 -0.166667 29.8333
0	3	2	157	44	0	1	0	-2.37931	-3.34483	0	-0.965517	-1.48571	-2.45714 1.45714 0.485714 1 0.166667 29.6667 28.8333
0	4	3	157	197	0	1	0	0.0689655	-0.896552	1.03448	0.0689655	-0.628571	-1.6 0.6 -0.371429 1.16667 0.333333 5.5 4.66667 0.06667
0	5	4	157	145	0	1	0	0.241379	-0.724138	2.03448	1.06897	-0.542857	-1.51429 0.657143 -0.314286 1.16667 0.333333 7.33333
0	6	5	157	115	0	1	0	-0.0689655	-1.03448	0.793103	-0.172414	-0.514286	-1.48571 0.685714 -0.285714 1.16667 0.333333
0	7	6	157	129	0	1	0	0.724138	-0.241379	1.93103	0.965517	0.342857	-0.628571 1.17143 0.2 1.16667 0.333333 7.5 6.66667 0.724138
0	8	7	157	109	0	1	0	-0.0689655	-1.03448	0.896552	-0.0689655	0.142857	-0.828571 0.971429 0 1.16667 0.333333 7.33333
0	9	8	157	161	0	1	0	0.206897	-0.758621	1.72414	0.758621	0.628571	-0.342857 1.2 0.228571 1.16667 0.333333 7.33333 6.5

Figure 11 – Les arcs

2 Utilisation de segmentation

Dans ce chapitre, je vais présenter comment on utilise notre logiciel pour faire la segmentation (étiqueter le voxel) incrémentale d'image médicale 3D (par utiliser le résultat d'apprentissage).

2.1 Préparations

Pour cette utilisation, on a deux dossiers aussi. Un est pour enregistrer les images à segmenter, l'autre est pour enregistrer les images de segmentation vérités.

étude (D:) > ImagePRD > imagePreparer > seg			
名称	修改日期	类型	大小
image	2019/3/20 5:35	文件夹	
imageVerite	2019/3/20 5:38	文件夹	

Figure 12 – Préparer d'images

- Le dossier 'image' : il contient des images a segmenter, comme Figure 13.
- Le dossier 'imageVerite' : il contient plusieurs segmentations vérités comme Figure 14 :

Il faut faire attention au nom de segmentation vérité, le nom doit commencer par le numéro d'image à segmenter, par exemple, image '1000_3_glm.nii.gz' et image '1000_4.nii.gz' sont des segmentations vérités d'image '1000_3.nii'.

> étude (D:) > ImagePRD > imagePreparer > seg > image

名称	修改日期	类型	大小
1000_3.nii	2012/4/16 23:56	NII 文件	36,737 KB
1001_3.nii	2012/4/16 23:56	NII 文件	33,409 KB
1002_3.nii	2012/4/16 23:56	NII 文件	35,457 KB
1006_3.nii	2012/4/16 23:56	NII 文件	39,937 KB
1007_3.nii	2012/4/16 23:56	NII 文件	40,961 KB
1008_3.nii	2012/4/16 23:56	NII 文件	35,073 KB
1009_3.nii	2012/4/16 23:56	NII 文件	37,761 KB
1010_3.nii	2012/4/16 23:56	NII 文件	33,793 KB
1011_3.nii	2012/4/16 23:56	NII 文件	36,737 KB
1012_3.nii	2012/4/16 23:56	NII 文件	37,761 KB
1013_3.nii	2012/4/16 23:56	NII 文件	38,657 KB
1014_3.nii	2012/4/16 23:56	NII 文件	39,297 KB
1015_3.nii	2012/4/16 23:56	NII 文件	40,833 KB
1017_3.nii	2012/4/16 23:56	NII 文件	34,433 KB
1036_3.nii	2012/4/16 23:56	NII 文件	42,753 KB

Figure 13 – Images à segmenter

> étude (D:) > ImagePRD > imagePreparer > seg > imageVerite

名称	修改日期	类型	大小
1000_3_glm.nii.gz	2012/9/5 16:28	好压 GZ 压缩文件	312 KB
1000_4.nii.gz	2019/1/29 15:05	好压 GZ 压缩文件	209 KB
1001_3_glm.nii.gz	2012/9/5 16:28	好压 GZ 压缩文件	314 KB
1001_4.nii.gz	2019/1/29 15:06	好压 GZ 压缩文件	220 KB
1002_3_glm.nii.gz	2012/9/5 16:28	好压 GZ 压缩文件	288 KB
1002_4.nii.gz	2019/1/29 15:07	好压 GZ 压缩文件	196 KB
1006_3_glm.nii.gz	2012/9/5 16:28	好压 GZ 压缩文件	298 KB
1006_4.nii.gz	2019/1/29 15:08	好压 GZ 压缩文件	208 KB
1007_3_glm.nii.gz	2012/9/5 16:28	好压 GZ 压缩文件	285 KB
1007_4.nii.gz	2019/1/29 15:09	好压 GZ 压缩文件	194 KB
1008_3_glm.nii.gz	2012/9/5 16:28	好压 GZ 压缩文件	289 KB
1008_4.nii.gz	2019/1/29 15:09	好压 GZ 压缩文件	196 KB
1009_3_glm.nii.gz	2012/9/5 16:28	好压 GZ 压缩文件	268 KB
1009_4.nii.gz	2019/1/29 15:10	好压 GZ 压缩文件	186 KB
1010_3_glm.nii.gz	2012/9/5 16:28	好压 GZ 压缩文件	287 KB
1010_4.nii.gz	2019/1/29 15:11	好压 GZ 压缩文件	197 KB
1011_3_glm.nii.gz	2012/9/5 16:28	好压 GZ 压缩文件	293 KB
1011_4.nii.gz	2019/1/29 15:12	好压 GZ 压缩文件	199 KB
1012_3_glm.nii.gz	2012/9/5 16:28	好压 GZ 压缩文件	257 KB
1012_4.nii.gz	2019/1/29 15:13	好压 GZ 压缩文件	171 KB
1013_3_glm.nii.gz	2012/9/5 16:28	好压 GZ 压缩文件	289 KB
1013_4.nii.gz	2019/1/29 15:14	好压 GZ 压缩文件	199 KB
1014_3_glm.nii.gz	2012/9/5 16:28	好压 GZ 压缩文件	263 KB
1014_4.nii.gz	2019/1/29 15:15	好压 GZ 压缩文件	183 KB
1015_3_glm.nii.gz	2012/9/5 16:28	好压 GZ 压缩文件	296 KB
1015_4.nii.gz	2019/1/29 15:16	好压 GZ 压缩文件	205 KB
1017_3_glm.nii.gz	2012/9/5 16:28	好压 GZ 压缩文件	288 KB
1017_4.nii.gz	2019/1/29 15:17	好压 GZ 压缩文件	193 KB
1036_3_glm.nii.gz	2012/9/5 16:28	好压 GZ 压缩文件	311 KB
1036_4.nii.gz	2019/1/29 15:18	好压 GZ 压缩文件	218 KB

Figure 14 – Segmentations vérifiées

2.2 Exécution

Les étapes d'exécution de segmentation sont comme ci-dessous :

(1). Exécuter le fichier exécutable 'NeuroGeo.exe' avec les paramètres ci-dessous :

- –MODE Seg

C'est-à-dire qu'on exécute la fonctionnalité de segmentation.

- –INPUT_T1 D : \ImagePRD\imagePreparer\seg\image\1000_3.nii

Il faut entrer le chemin d'image IRM à segmenter.

- –MODEL_PATH D : \ImagePRD\imagePreparer\apprentissage\result\1000

Il faut entrer le chemin de résultat d'apprentissage.

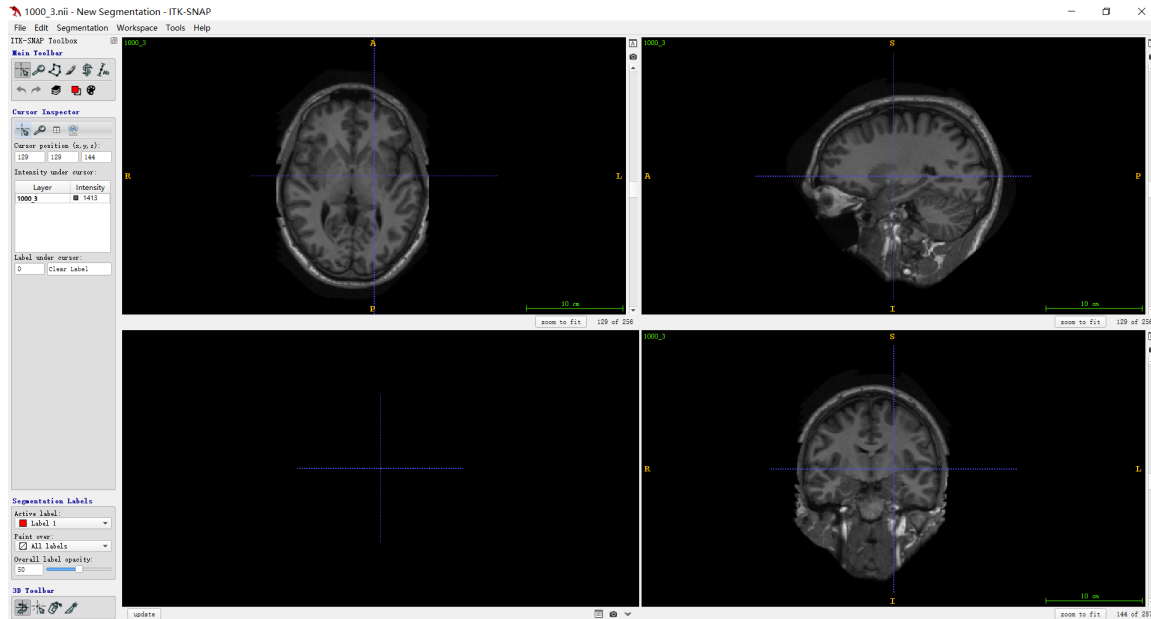


Figure 15 – L'image '1000_3.nii'

étude (D:) > ImagePRD > imagePreparer > apprentissage > result > 1000

名称	修改日期	类型	大小
Edges	2019/3/20 5:51	文件夹	
Nodes	2019/3/20 5:58	文件夹	
Brain_Model.lgf	2019/3/20 6:01	LGF 文件	4,478 KB

Figure 16 – Le résultat d'apprentissage de l'image '1000_3_template.nii'

```

D:\VS2017_workspace\NeuroGeo\build\Debug\NeuroGeo.exe --MODE Seg --INPUT_T1 D:\ImagePRD\imagePreparer\seg\image\1000_3.n
ii --MODEL_PATH D:\ImagePRD\imagePreparer\apprentissage\result\1000
NeuroGeo > setSegmentationMethod Local
... Segmentation method = Local ...
... Loading Model ...
... Model loaded ...
NeuroGeo > setCurrentRegion 13
... Region ID = 13 ...
NeuroGeo > setBoundingBox (78,110,114) (40,30,50)
... Bounding-box origin = (78,110,114) ...
... Bounding-box dimension = (40,30,50) ...
NeuroGeo > setClassMethod HMF
NeuroGeo > execute
... Get Bounding Box From Ground Truth ...
...> Generation OK ...
Positionnement Bounding Box : [74, 104, 108] [62, 36, 46]
Positionnement extended Bounding Box : [67, 100, 103] [76, 44, 56]
... Probability Map Generation ...
D:\ImagePRD\imagePreparer\apprentissage\result\1000\Nodes\13\T1\ProbaMap.nii
...> Generation OK ...
... Template Generation ...

```

Figure 17 – Lire boundingBox vers segmentation vérité

(2). Après l'exécution, l'utilisateur peut utiliser la commande 'saveSegmentation + chemin' pour sauvegarder le résultat de segmentation. Par exemple, on peut entrer 'saveSegmentation D:\ImagePRD\imagePreparer\seg\result\1000Cas1', donc, le résultat sera sauvegardé dans le dossier 'state1'.

```

24 / 25
25 / 25
Positionnement original Bounding Box : [79, 110, 116] [26, 24, 33]
NeuroGeo > saveSegmentation D:\ImagePRD\lyyTest\seg\state1
NeuroGeo >

```

Figure 18 – Sauvegarde du résultat

(3). Parce que chaque fois, il juste segmente une région, donc, après une exécution de segmentation, on peut continuer à segmenter par la commande 'setCurrentRegion XX' et 'execute'.

Par exemple, on d'abord segmente la région 57 et on sauvegarde le résultat, on peut continuer à segmenter la région 6.

```
cmd.exe
17 / 25
18 / 25
19 / 25
20 / 25
21 / 25
22 / 25
23 / 25
24 / 25
25 / 25
Positionnement original Bounding Box : [139, 109, 83] [39, 56, 74]
NeuroGeo > saveSegmentation D:\ImagePRD\imagePreparer\seg\result\1000Cas1
NeuroGeo > setCurrentRegion 6
... Region ID = 6 ...
NeuroGeo > execute
... Get Bounding Box From Ground Truth ...
... Generation OK ...
Positionnement Bounding Box : [70, 104, 69] [61, 31, 111]
Positionnement extended Bounding Box : [63, 100, 57] [75, 39, 135]
... Probability Map Generation ...
D:\ImagePRD\imagePreparer\apprentissage\result\1000\Nodes\6\T1\ProbaMap.nii
... Generation OK ...
```

Figure 19 – Segmentation incrémentale

(4). En plus, il y a d'autre commande :

- loadSegmentation D:\ImagePRD\imagePreparer\seg\result\1000Cas1

C'est-à-dire, il va lire le résultat de segmentation vers le chemin entré. Et on peut continuer à segmenter à partir de ce statu de segmentation.

```
D:\VS2017_workspace\NeuroGeo\build\Debug\NeuroGeo.exe --MODE Seg --INPUT_T1 D:\ImagePRD\imagePreparer\seg\image\1000_3.n
ii --MODEL_PATH D:\ImagePRD\imagePreparer\apprentissage\result\1000
NeuroGeo > loadSegmentation D:\ImagePRD\imagePreparer\seg\result\1000Cas1
NeuroGeo > setSegmentationMethod Local
... Segmentation method = Local ...
... Loading Model ...
... Model loaded ...
NeuroGeo > setCurrentRegion 155
... Region ID = 155 ...
NeuroGeo > setClassMethod HMF
NeuroGeo > execute
... Get Bounding Box From Ground Truth ...
... Generation OK ...
Positionnement Bounding Box : [139, 109, 83] [39, 57, 74]
Positionnement extended Bounding Box : [135, 103, 75] [47, 69, 90]
... Probability Map Generation ...
```

Figure 20 – Charger la segmentation

- removeRegion 59

C'est-à-dire, il va supprimer la région 59 qui est déjà segmentée.

(5). Pour quitter l'exécution, on a besoins d'entrer 'exit'.

```
24 / 25
25 / 25
Positionnement original Bounding Box : [79, 110, 116] [26, 24, 33]
NeuroGeo > saveSegmentation D:\ImagePRD\llyTest\seg\state1
NeuroGeo > exit
D:\VS2017_workspace\NeuroGeo\build\Debug>
```

Figure 21 – Quitter

2.3 Visualisation de résultat

nom	modification	type	taille
CurrentGraph.lgf	2019/3/20 8:03	LGF fichier	1 KB
LabelImage1.nii.gz	2019/3/20 8:03	bonne GZ compression	79 KB
LabelImage2.nii.gz	2019/3/20 8:03	bonne GZ compression	74 KB

Figure 22 – Le résultat

On peut regarder le dossier de résultat 'D:\ImagePRD\imagePreparer\seg\result\1000Cas1' Il contient l'image de segmentation ('LabelImage1.nii' et 'LabelImage2.nii') et le fichier qui enregistre le numéro et la dimension de la région qui est déjà segmentée.

Pour visualiser le résultat, on utilise l'outil 'ITK Snap'.

(1). D'abord, ouvrir l'image à segmenter comme 'main image'.

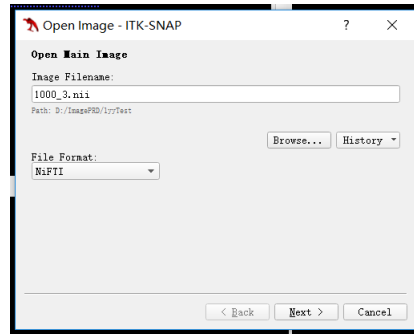


Figure 23 – Charger l'image principale

(2). Ouvrir l'image de segmentation ('LabelImage1.nii.gz').

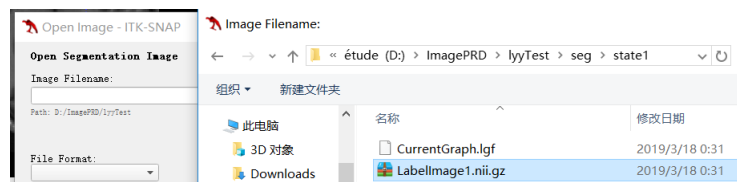


Figure 24 – Charger l'image segmenté

(3). La visualisation de 'ITK Snap' :

Sur le côté gauche de la fenêtre, nous pouvons voir que les voxels verts sont étiquetés 13 (le numéro de la région segmentée).

L'image ci-dessous est l'image 'LabelImage1.nii.gz' :

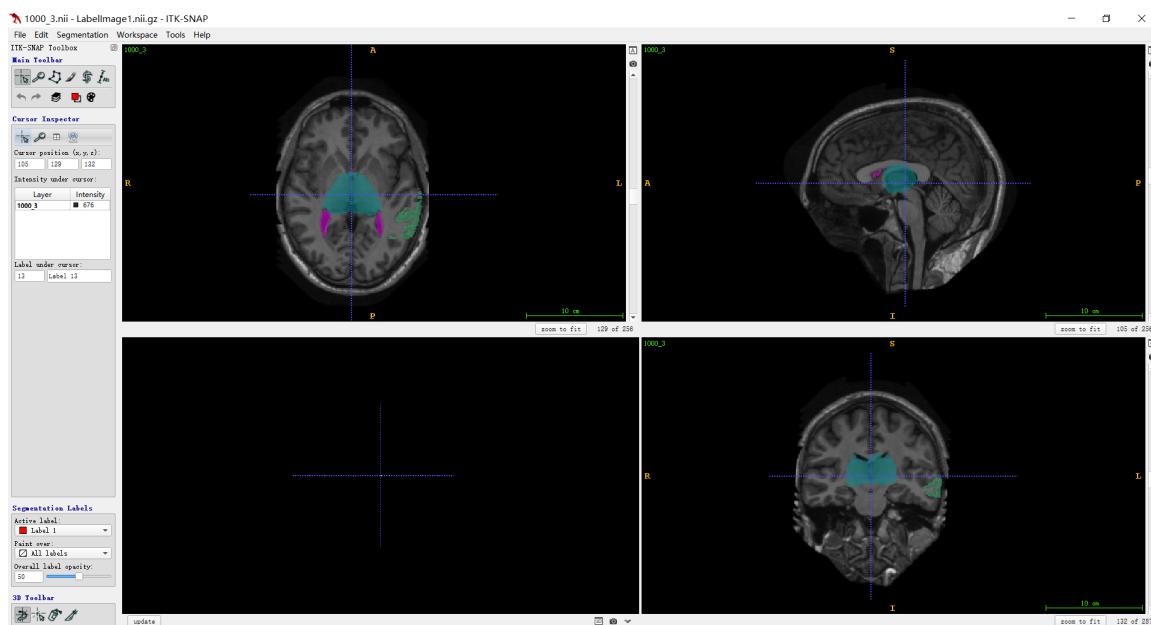


Figure 25 – La région 13 (le label avec la couleur bleu), 6 (le label avec la couleur rose) et 155 (le label avec la couleur vert)

L'image ci-dessous est l'image 'LabelImage2.nii.gz' :

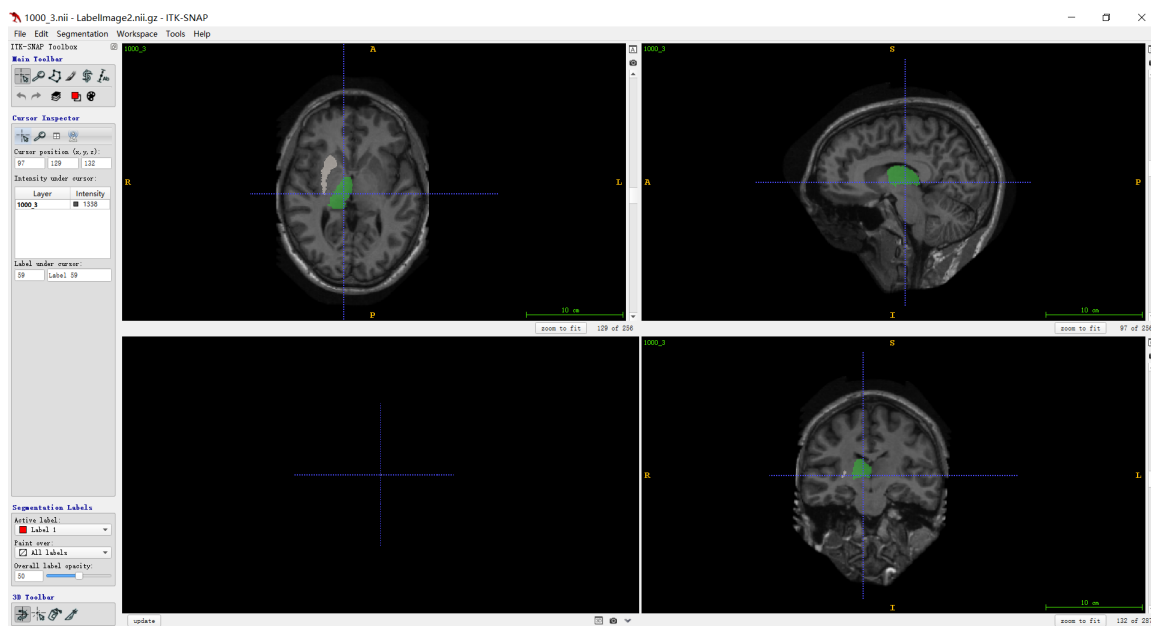


Figure 26 – La région 59 (le label avec la couleur vert) et 57 (le label avec la couleur beige)

On peut voir que les régions peuvent être segmentées au plusieurs niveaux, et les régions segmentées peuvent être sauvegardées dans plusieurs images.

F

Document de tests

1 Tests fonctionnels

J'utilise 'Microsoft : :VisualStudio : :CppUnitTestFramework' pour faire le test. C'est le Framework de tests unitaires de Microsoft pour C ++. Parce que, c'était plus facile et pratique pour configurer et programmer le test.

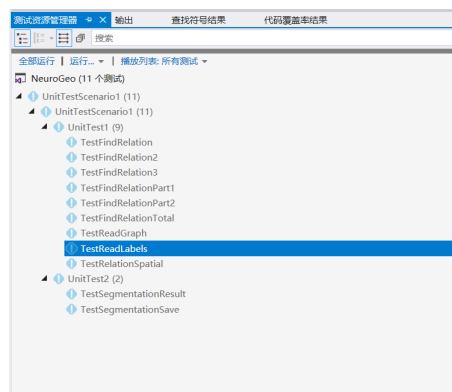
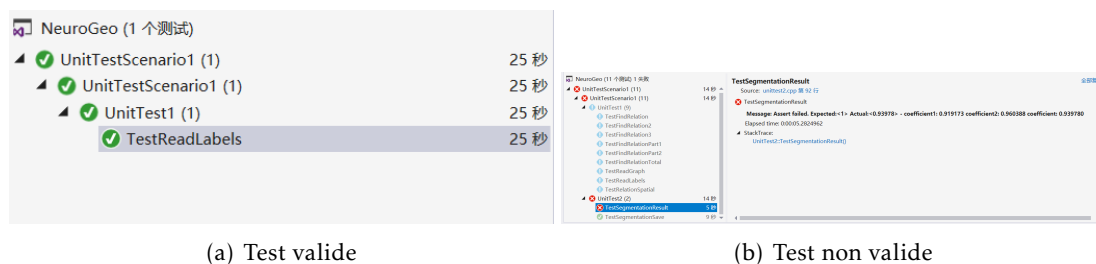


Figure 1 – La structure de tests

Si notre test est valide, on peut voir le résultat (alerte vert) comme Figure 2.



(a) Test valide

(b) Test non valide

Figure 2 – Test

Si notre test n'est pas valide, on peut voir le résultat (alerte rouge) comme Figure 2, et on peut utiliser 'Debug Test' pour trouver la cause d'erreur.

1.1 Test de créer et sauvegarder du graphe

Nom de test : TestReadLabels()

Date de test : 16/01/2019

Description de test : il lit le fichier ' Brain_Model.lgf 'du graphe G et il vérifie si toutes les régions sont enregistrées comme un nœud dans ce graphe.

Description de fonctionnalités à tester : il test trois méthodes :

- Le constructeur de la classe 'BrainModel' : cela permet de créer le graphe de connaissances a priori modifiée (avec l'attribut relation).

- La méthode 'save()' de la classe 'BrainModel' :cela permet de sauvegarder le graphe dans un fichier qui s'appelle 'Brain_Model.lgf'.

- La méthode 'load()' de la classe 'BrainModel' : cela permet de lire le fichier 'Brain_Model.lgf' et créer le graphe.

Entrée :

- Le fichier ' Brain_Model.lgf 'du graphe G.
- Deux images labellisées L_1 et L_2 qui sont pour l'apprentissage.

Résultats attendus :

La liste de nœuds du graphe G = la liste de régions dans l'image L_1 + la liste de régions dans l'image L_2

Résultats obtenus :

La liste de nœuds du graphe G = la liste de régions dans l'image L_1 + la liste de régions dans l'image L_2

Le temps d'exécution de test : moins de 1 minute.

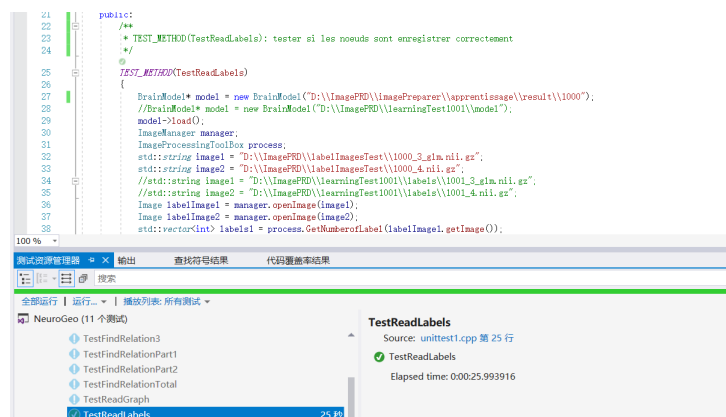


Figure 3 – Test de nœuds

1.2 Test de relation spatiale

Nom de test : TestRelationSpatial()

Date de test : 17/01/2019

Description de test : il lit le fichier ' Brain_Model.lgf 'du graphe G et il vérifie si toutes les relations spatiales sont enregistrées dans ce graphe.

Description de fonctionnalités à tester : il test trois méthodes :

- Le constructeur de la classe 'BrainModel' : cela permet de créer le graphe de connaissances a priori modifiée (avec l'attribut relation).
- La méthode 'save()' de la classe 'BrainModel' : cela permet de sauvegarder le graphe dans un fichier qui s'appelle 'Brain_Model.lgf'.
- La méthode 'load()' de la classe 'BrainModel' : cela permet de lire le fichier 'Brain_Model.lgf' et créer le graphe.

Entrée :

- Le fichier ' Brain_Model.lgf 'du graphe G_1 (après mes implémentations).
- Le fichier ' Brain_Model.lgf 'du graphe G_2 (avant mes implémentations).

Explication :

parce que le graphe G_1 contient plus d'arcs que le graphe G_2 , donc, je juste vérifier si le graphe G_1 contient tous les arcs du graphe G_2 .

Résultats attendus :

Toutes les relations spatiales du graphe G_2 = les relations spatiales du graphe G_1 (cette formule n'est pas inversée)

Résultats obtenus :

Toutes les relations spatiales du graphe G_2 = les relations spatiales du graphe G_1

Le temps d'exécution de test : environ 7 minutes.

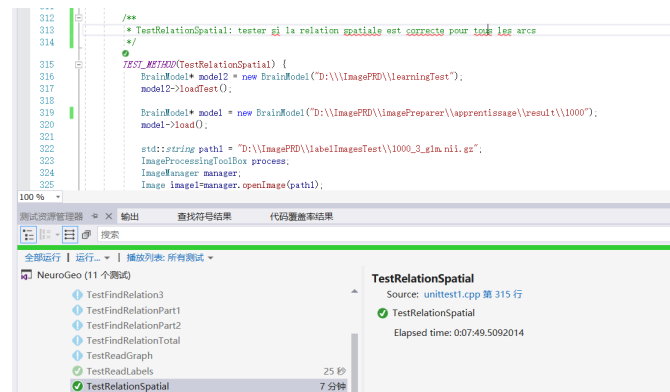


Figure 4 – Test de relation spatiale

1.3 Test 1 d'analyser relation

Nom de test : TestFindRelation3()

Date de test : 25/01/2019

Description de test : il lit le fichier ' Brain_Model.lgf 'du graphe G et il vérifie si la relations entre les deux régions (qui sont viennent de la même image labellisée) égale 0.

Description de fonctionnalités à tester

- La méthode 'getEdgeRelation ()' de la classe 'BrainModel' : cela permet de retourner la relation entre les deux régions.
- La méthode 'getPixelPairs ()' de la classe 'GraphLearning' : cela permet de créer le tableau de voxels complets.

- La méthode 'findRealtions ()' de la classe 'GraphLearning' : cela permet de déterminer la relation.
- La méthode 'updateRelation ()' de la classe 'GraphLearning' : cela permet de mettre à jour la relation du graphe.
- La méthode 'setEdgeRelationAll ()' de la classe 'BrainModel' : cela permet de mettre à jour la relation du graphe.

Entrée :

- Le fichier ' Brain_Model.lgf 'du graphe G.
- Deux images labellisées L1 et L2 qui sont pour l'apprentissage.

Résultats attendus :

La relation de régions (la région r_1 , la région r_2), si r_1 et r_2 viennent de la même image, la relation =0.

Résultats obtenus :

La relation de régions (la région r_1 , la région r_2), si r_1 et r_2 viennent de la même image, la relation =0.

Le temps d'exécution de test : moins de 1 minute.

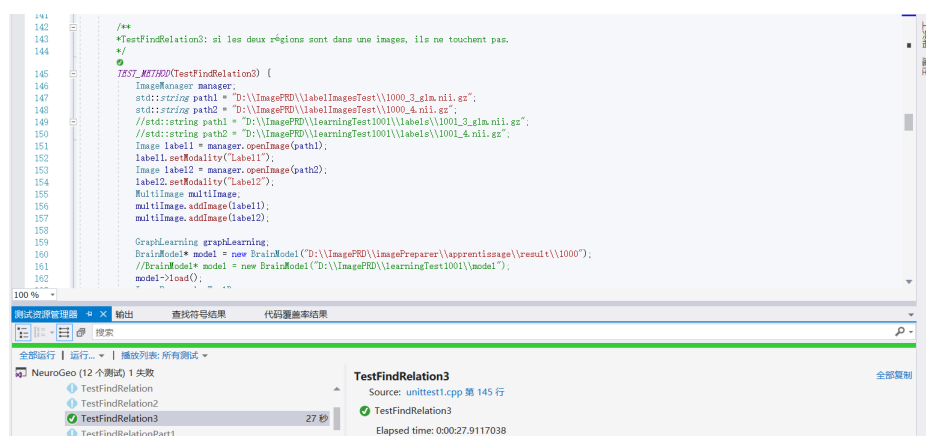


Figure 5 – Test relation séparée

1.4 Test 2 d'analyser relation

Nom de test : TestFindRelationPart1 () et TestFindRelationPart2 ()

Date de test : 07/02/2019

Description de test : il lit le fichier ' Brain_Model.lgf 'du graphe G. Il compare la relation (qui est obtenue par le tableau de voxels complets) et la relation (qui est obtenue par la méthode 'findRealtionsTest ()' de la classe 'GraphLearning'). A cause du grand nombre de régions, je divise toutes les régions à deux parties.

Description de fonctionnalités à tester

- La méthode 'getEdgeRelation ()' de la classe 'BrainModel' : cela permet de retourner la relation entre les deux régions.
- La méthode 'getPixelPairs ()' de la classe 'GraphLearning' : cela permet de créer le tableau de voxels complets.

- La méthode 'findRealtions ()' de la classe 'GraphLearning' : cela permet de déterminer la relation.
- La méthode 'updateRelation ()' de la classe 'GraphLearning' : cela permet de mettre à jour la relation du graphe.
- La méthode 'setEdgeRelationAll ()' de la classe 'BrainModel' : cela permet de mettre à jour la relation du graphe.
- La méthode 'findRealtionsTest ()' de la classe 'GraphLearning' : pour chaque région, il parcourt tous les coordonnées de cette région. Après, il compare les coordonnées de régions. Cette fonctionnalité est la première version de déterminer la relation entre les deux régions.

Entrée :

- Le fichier ' Brain_Model.lgf 'du graphe G.
- Deux images labellisées L_1 et L_2 qui sont pour l'apprentissage.

Résultats attendus :

La relation (qui est obtenue par le tableau de voxels complets) = la relation (qui est obtenue par la méthode 'findRealtionsTest ()' de la classe 'GraphLearning')

Résultats obtenus :

La relation (qui est obtenue par le tableau de voxels complets) = la relation (qui est obtenue par la méthode 'findRealtionsTest ()' de la classe 'GraphLearning')

Le temps d'exécution de test : environ 9 heures.

1.5 Test de sauvegarder la segmentation

Nom de test : TestSegmentationSave ()

Date de test : 25/02/2019

Description de test : il lit le fichier ' CurrentGraph.lgf '(le statu de la segmentation). Il compare la liste de noeuds de fichier ' CurrentGraph.lgf ' et la liste de régions dans l'images de résultat.

Description de fonctionnalités à tester

- La méthode 'saveState()' : cela permet de sauvegarder les résultats (un graphe et d'image(s) segmentation(s)) dans un dossier.

Entrée :

- Le fichier ' CurrentGraph.lgf ' qui enregistre le statu de segmentation.
- Des images 'LabelImage1.nii.gz' et 'LabelImage2.nii.gz' (qui sauvegardent les régions segmentées).

Résultats attendus :

- La liste de nœud du fichier ' CurrentGraph.lgf ' = la liste de régions d'image 'LabelImage1.nii.gz' + la liste de régions d'image 'LabelImage2.nii.gz'

Résultats obtenus :

- La liste de nœud du fichier ' CurrentGraph.lgf ' = la liste de régions d'image 'LabelImage1.nii.gz' + la liste de régions d'image 'LabelImage2.nii.gz'

Le temps d'exécution de test : moins de 1 minute.

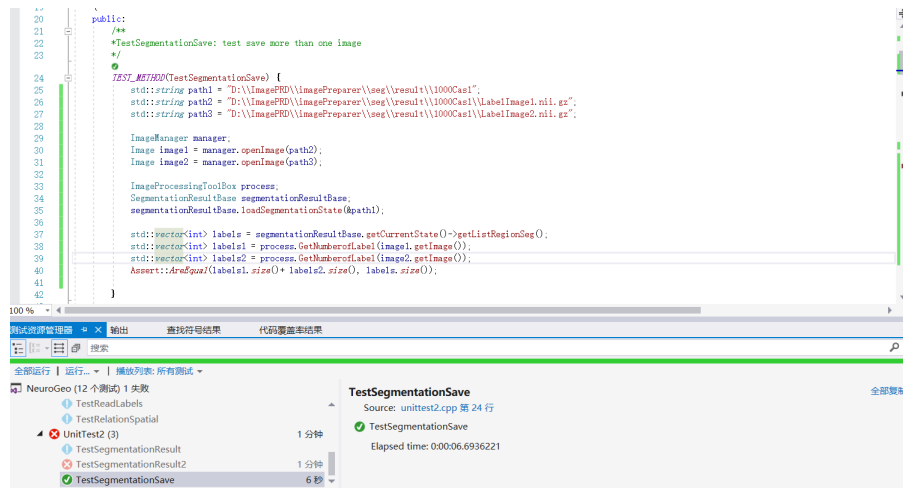


Figure 6 – Test le graphe de statut de segmentation

1.6 Test le résultat de segmentation

Nom de test : TestSegmentationResult ()

Date de test : 10/03/2019

Description de test : il calculer le coefficient Sørensen-Dice de résultat.

Description de fonctionnalités à tester

- La méthode 'localToGlobal ()' : cela permet d'étiqueter les régions avec le résultat de 'HMF' et la relation de régions.

Entrée :

- Des images 'LabelImage1.nii.gz' et 'LabelImage2.nii.gz' (qui sauvegardent les régions segmentées).
- Des images segmentées vérités '1000_3_glm.nii.gz' et '1000_4.nii.gz'.

Résultats attendus :

- Le coefficient de Sørensen-Dice > 90%

Résultats obtenus :

- Le coefficient de Sørensen-Dice > 90%

Le temps d'exécution de test : moins de 1 minute.

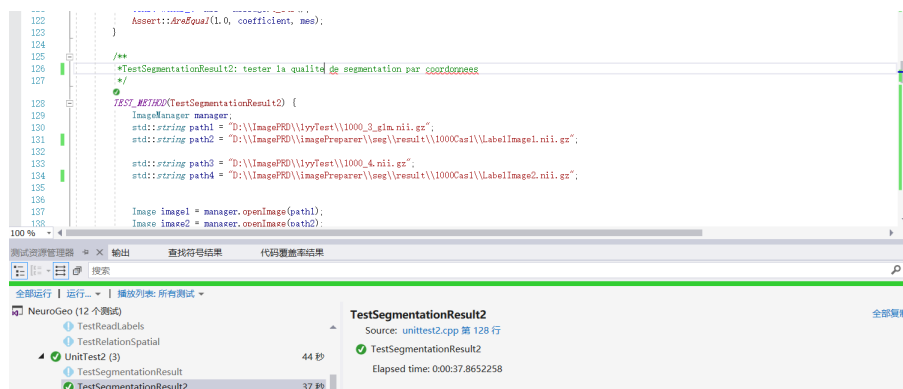


Figure 7 – Test la qualité de segmentation



Webographie

- [WWW1] *Affine transformation Wikipédia*. URL : https://en.wikipedia.org/wiki/Affine_transformation (visité le 17/09/2018).
- [WWW2] *k-means Wikipédia*. URL : <https://fr.wikipedia.org/wiki/K-moyennes>.
- [WWW3] *L'indice de Sørensen-Dice Wikipédia*. URL : https://fr.wikipedia.org/wiki/Indice_de_S%C3%B8rensen-Dice (visité le 21/05/2018).
- [WWW4] *Modèle de Markov caché Wikipédia*. URL : https://fr.wikipedia.org/wiki/Mod%C3%A8le_de_Markov_cach%C3%A9 (visité le 11/11/2018).
- [WWW5] *Rigid transformation Wikipédia*. URL : https://en.wikipedia.org/wiki/Rigid_transformation (visité le 21/09/2018).
- [WWW6] *Voting method Wikipédia*. URL : <https://en.wikipedia.org/wiki/Voting> (visité le 29/11/2018).

Bibliographie

- [1] Mariano CABEZAS, Arnau OLIVER, Xavier LLADÓ, Jordi FREIXENET et Meritxell Bach CUADRA. « A review of atlas-based segmentation for magnetic resonance brain images ». In : *Computer Methods and Programs in Biomedicine* 104 (2011), p. 158–177.
- [2] Sasank CHILAMKURTHY. *Brain Anatomy Segmentation*. 2016.
- [3] Alexandre DUPAS. « Opérations et Algorithmes pour la Segmentation Topologique d’Images 3D ». Thèse de doct. Université de Poitiers, 2009.
- [4] Gaëtan GALISOT. « Segmentation incrémentale et interactive d’images médicales 3D ». Thèse de doct. Université François - Rabelais de Tours, 2017.
- [5] Hrvoje KALINIC. *Atlas-based image segmentation : A Survey*. 2009.
- [6] Medical Image Segmentation using OBJECT ATLAS VERSUS OBJECT CLOUD MODELS. Renzo Phellana, Alexandre X. Falc aoa, Jayaram K. Udupa. 2015.
- [7] Clare B POYNTON, Kevin T CHEN, Daniel B CHONDE, David IzQUIERDO-GARCIA, Randy L GOLLUB, Elizabeth R GERSTNER, Tracy T BATCHELOR et Ciprian CATANA. « Probabilistic atlas-based segmentation of combined T1-weighted and DUTE MRI for calculation of head attenuation maps in integrated PET/MRI scanners ». In : *American Journal of Nuclear Medicine and Molecular Imaging* (2014), p. 160–171.
- [8] Multi-atlas Segmentation Enables Robust Multi-contrast MRI Spleen Segmentation for SPLENOMEGALY. *Brain Anatomy Segmentation*. 2017.

Implémentation de nouvelles fonctionnalités dans la plateforme NeuroBrainSeg de segmentation d'images médicales 3D

Résumé

Le sujet de Projet Recherche et Développement est « Implémentation de nouvelles fonctionnalités dans la plate-forme NeuroBrainSeg de segmentation d'images médicales 3D ». Le but de ce projet est d'implémenter des nouvelles fonctionnalités au sein du logiciel 'NeuroBrainSeg' qui permettent de réaliser la segmentation d'image médicale 3D basée sur Atlas probabiliste locale lorsque des régions sont incluses ou imbriquées les unes dans les autres. Pour faire ce projet, il contient deux parties principales, un est pour analyser et modéliser, l'autre est pour développer, tester et améliorer.

Mots-clés

Segmentation d'image médicale 3D, Atlas probabiliste locale, Graphe de connaissance a priori, NeuroGeo, C++, ITK

Abstract

The subject of Research and Development of Project is "Implementation of new functionalities in the NeuroBrainSeg platform for 3D medical image segmentation". The aim of this project is to implement some new functionalities within the software 'NeuroBrainSeg' that realizes segmentation of 3D medical image based on 'Atlas probabiliste locale' when regions are included or nested within others. To make this project, it contains two main parts, one is to analyse and model, the other is to develop, test and improve.

Keywords

Segmentation of 3D medical image, Atlas probabiliste locale, Graph a priori, NeuroGeo, C++, ITK

Tuteurs académiques

Gaëtan GALISOT
Jean-Yves RAMEL

Étudiant

Yuan Yuan LI (DI5)