



Ecole Polytechnique de l'Université 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 2021-2022

Analyse, visualisation 3D et comparaison de cerveaux et connectomes



POLYTECH[®]
TOURS

Tuteur académique
Jean-Yves RAMEL

Étudiant
Maxime LANDRIN (DI5)

3 avril 2022



Liste des intervenants

Nom	Email	Qualité
Maxime LANDRIN	maxime.landrin@univ-tours.fr	Étudiant DI5
Jean-Yves RAMEL	jean-yves.ramel@univ-tours.fr	Tuteur académique, Département Informatique



Avertissement

Ce document a été rédigé par Maxime LANDRIN susnommé l'auteur.

L'Ecole Polytechnique de l'Université de Tours est représentée par Jean-Yves RAMEL susnommé le tuteur académique.

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 assume 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 du tuteur académique 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

Maxime LANDRIN, *Analyse, visualisation 3D et comparaison de cerveaux et connectomes* ,
Projet Recherche & Développement, Ecole Polytechnique de l'Université de Tours, Tours,
France, 2021-2022.

```
@mastersthesis{
  author={LANDRIN, Maxime},
  title={Analyse, visualisation 3D et comparaison de cerveaux et connectomes: },
  type={Projet Recherche \& Développement},
  school={Ecole Polytechnique de l'Université de Tours},
  address={Tours, France},
  year={2021-2022}
}
```

Table des matières

Liste des intervenants	a
Avertissement	b
Pour citer ce document	c
Table des matières	i
Table des figures	iv
1 Introduction	1
1 Contexte et enjeux.....	1
2 Acteurs.....	3
3 Objectifs.....	3
3.1 Représentation du modèle.....	3
3.2 Externalisation des paramètres	4
3.3 Documentation de SILA	5
4 Hypothèses	5
4.1 Représentation du modèle.....	5
4.2 Externalisation des paramètres	6
4.3 Documentation de SILA3D	6
5 Bases méthodologiques.....	6
2 Description générale	8
1 Environnement du projet	8
1.1 Interface Utilisateur	9
1.2 Extension SILA3D.....	9

2	Caractéristiques des utilisateurs	10
3	Fonctionnalités du système	10
3.1	Représentation du modèle	10
3.2	Externalisation des paramètres	11
3	État de l'art / Veille technologique	12
1	Analyse de l'existant	12
1.1	Slicer, ITK et VTK	12
1.2	Comment se sert-on de SILA3D ?	13
2	Etat de l'art	17
2.1	Slicer - ITK & VTK	17
2.1.1	ITK - Insight Toolkit	17
2.1.2	VTK - Visualization Toolkit	17
2.2	SILA3D	18
2.2.1	Les classes externes	19
2.2.2	Les classes "simples"	20
2.2.3	Les classes Initialization	21
2.2.4	Les classes "Method Parameter"	22
2.2.5	Les classes pour l'invite de commande	23
2.2.6	Les classes pour la segmentation	23
2.2.7	Les classes pour le recalage	24
2.2.8	Les classes finales	25
4	Mise en œuvre	27
1	Documentation des classes	27
2	La classe "ConfigurationFileManager"	28
2.1	Outils et librairie utilisés	29
2.2	Éléments d'implémentation, choix techniques	29
2.3	Analyse des résultats, évaluation, qualité	31
3	Afficher les régions/atlas	31
3.1	Outils et librairie utilisés	32
3.2	Éléments d'implémentation, choix techniques	32
3.3	Analyse des résultats, évaluation, qualité	32
4	Afficher les liens entre régions	35
5	L'interface	35
5.1	La Maquette	36
5	Bilan et conclusion	39
1	Bilan du semestre 9	39
2	Bilan du semestre 10	39
3	Bilan sur la qualité	40
4	Bilan auto-critique	40

Annexes	41
A Planification, gestion de projet	42
1 Evolution du projet	42
2 Description des tâches.....	43
Tâche 1 : Extraction des données	43
Tâche 2 : Visualisation Modèle	43
Tâche 3 : Modification paramètre région.....	43
B Cahier de Spécifications	44
1 spécifications Fonctionnelles.....	44
1.1 Externalisation des données - La classe ConfigurationFileManager	44
Description des attributs :	45
Description des méthodes :	45
1.2 Visualisation du Modèle - Les classes VisualiseModel	46
2 Spécifications non fonctionnelles	46
2.1 Contraintes de développement et conception	46
2.2 Contraintes de fonctionnement et d'exploitation.....	46
2.2.1 Performances	46
2.2.2 Capacités	46
2.2.3 Contrôlabilité	46
C Bibliographie	47
D Glossaire	49
E Acronymes	51

Table des figures

1 Introduction

1.1 Image segmentation with SILA3D, Video source : http://www.rfai.li.univ-tours.fr/PublicData/3D_Brain_Seg/Videos/SILA3D_demo_1_042021.mp4	2
1.2 Diagramme sur le fonctionnement de SILA3D	3
1.3 Représentation d'une image 3D sur slicer	4

2 Description générale

2.1 Environnement actuel de SILA3D	8
2.2 Environnement de SILA3D avec les nouvelles fonctionnalités.....	9
2.3 Fonctionnalité d'externalisation	11

3 État de l'art / Veille technologique

3.1 Visualization Toolkit (VTK) Tutorial, John T. Bell source : cs.uic.edu/~jbell/CS526/Tutorial/Tutorial.html	
3.2 SILA3D - Sélection du modèle	14
3.3 SILA3D - Modèle sélectionné	14
3.4 SILA3D - Modèle sélectionné et chargé	14
3.5 SILA3D - Sélection de la zone à segmenter.....	15
3.6 SILA3D - Positionner le zone d'intérêt	15
3.7 SILA3D - Diagramme de classe UML.....	18
3.8 SILA3D - UML Dépendance & Simplification - Classes Externes.....	19
3.9 SILA3D - UML Dépendance & Simplification - Classes Simples	21
3.10 SILA3D - UML Dépendance & Simplification - Classes Initialization.....	22
3.11 SILA3D - UML Dépendance & Simplification - Classes MethodParameters	22
3.12 SILA3D - UML Dépendance & Simplification - Classes Interactives	23

3.13	SILA3D - UML Dépendance & Simplification - Classes Segmentations	24
3.14	SILA3D - UML Dépendance & Simplification - Classes Registration	25
3.15	SILA3D - UML Dépendance & Simplification - Classes Finales	26
4	Mise en œuvre	
4.1	ConfigurationFileManager - Paramètres A Extraire.....	28
4.2	ConfigurationFileManager - Diagramme UML.....	29
4.3	ConfigurationFileManager - Les méthodes modifiées dans les différentes classes.....	31
4.4	Afficher les régions/atlas - Test 1	33
4.5	Afficher les régions/atlas - Test 2	34
4.6	Afficher les régions/atlas - Résultat Test 1	34
4.7	Afficher les régions/atlas - Résultat Test 2	35
4.8	Interface - Maquette	36
4.9	Interface - Rendu Final.....	37
4.10	Interface - Rendu Final Connecté.....	38
A	Planification, gestion de projet	
A.1	Prévision diagramme de GANTT.....	42
A.2	Diagramme de GANTT - Global.....	42
A.3	Diagramme de GANTT - Développement.....	43
B	Cahier de Spécifications	
B.1	Diagramme de classe - ConfigurationFileManager	44

1

Introduction

1 Contexte et enjeux

Ce projet est réalisé lors de ma 3e année du cycle ingénieur. Le projet s'inscrit dans un cours nommé “**Projet de Recherche & Développement (PRD)**”. Il est découpé en deux parties, une par semestre. Durant ce premier semestre, j'ai pour but d'analyser le projet et de me préparer pour le développement de la fonctionnalité.

NeuroGéo[6] est un projet proposé par la région Centre-Val de Loire. La problématique que décrit le projet est qu'il existe divers outils d'analyse d'images **Imagerie par résonance magnétique (IRM)**, mais qu'ils sont très spécifiques. Le premier objectif de ce projet est de proposer un outil adaptable sur plusieurs espèces. Il s'agirait d'un outil interactif, simple et intuitif pour des néophytes. L'outil devra être interactif au niveau de la **segmentation** des images IRM. Si l'objectif est atteint alors un second objectif est proposé, créer un module de consultation et d'aide à la décision.

C'est dans cette optique qu'une **extension** à un logiciel déjà existant a été créée. SILA3D[14] est une extension au logiciel Slicer[15]. Slicer est un logiciel **open source**, développé lors d'une thèse et a pour but de simplifier la visualisation des images d'IRM. SILA3D est une extension qui va permettre de consulter des images IRM de manière simple en utilisant Slicer. Mais en plus de cela, SILA3D permet l'utilisation d'un modèle qui va permettre de découper des zones de l'anatomie.

SILA3D, s'inscrivant dans le projet NeuroGéo, permet de visualiser et de découper des zones d'anatomie de différentes espèces et de différentes zones. Comme par exemple un cerveau humain ou un cerveau de brebis. SILA3D est fourni avec, de base, des modèles de cerveau humain, de brebis et de caille. Cependant, du moment que SILA3D possède un modèle, il pourra faire une segmentation de la zone. Cela implique donc que le logiciel peut théoriquement fonctionner avec d'autres zones anatomiques comme le foie, les poumons, etc. . .

Comme indiqué plus haut, SILA3D est une extension de Slicer. En effet, lorsque l'on lance l'application, Slicer est lancé. Slicer étant un outil très visuel, SILA3D profite donc aussi de son interface. Mais en plus de cela, SILA3D reste tout de même très intuitif, mais pas simple pour autant. Sur le site, se trouve en première page des tutoriels qui montrent l'interface et comment l'utiliser.

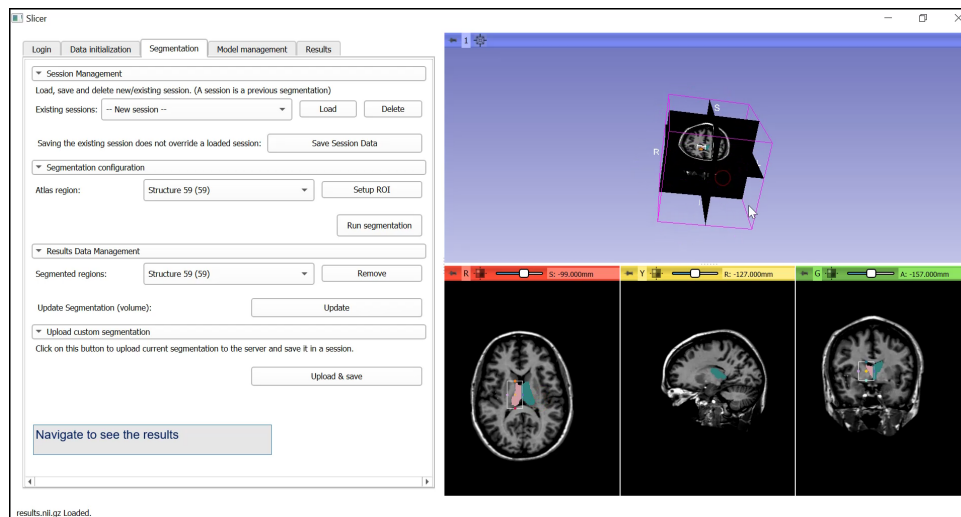


Figure 1.1 – Image segmentation with SILA3D, Video
 source : http://www.rfai.li.univ-tours.fr/PublicData/3D_Brain_Seg/Videos/SILA3D_demo_1_042021.mp4

Dans l'interface, on note que toute la partie droite est l'interface de Slicer. La partie gauche est ce que rajoute SILA3D.

SILA3D fonctionne en s'appuyant sur la thèse de Gaëtan GALISOT[1]. L'extension SILA3D, utilise des atlas locaux. “Les atlas sont une forme d'information a priori spatiale permettant de guider la localisation des structures anatomiques au sein d'images médicales.”[Gaëtan Galisot][2]. Les atlas locaux sont donc des représentations de sous-parties d'anatomie, là où un atlas est une partie de l'anatomie. Par exemple, le **cervelet** ou l'**hypothalamus** dans le cerveau. Le cerveau est ici un atlas, tant dis que le cervelet et l'hypothalamus représentent ici des atlas locaux. L'utilisation de plusieurs atlas locaux au lieu d'un seul atlas est très pratique. Cela permet de faire une segmentation rapide pour de bons résultats.

En utilisant ces atlas locaux, on fait un graphe où chaque atlas local est représenté par un nœud et où chaque arc représente la position relative de l'atlas local où l'on souhaite aller. Par exemple, imaginons un “atlas1” et un “atlas2”. L'arc qui relie “atlas1” à “atlas2” indique la position relative de “atlas2” par rapport à “atlas1”. On peut imaginer quelque chose comme [10, 10, 10]. Si on prend alors l'arc qui va de “atlas2” à “atlas1” on aura alors comme information sur l'arc [-10, -10, -10]. En utilisant ces informations sur les arcs, on est capable, lorsque l'on a déjà placé un atlas, de trouver assez facilement les autres atlas.

Chaque nœud du graphe se compose de deux fichiers. “ProbaMap.nii” qui est une **carte de probabilité**. On y répertorie les probabilités de trouver la zone que l'on recherche à tel ou tel voxel. Un voxel est simplement un pixel en 3D. On remplace le “pi” de “pixel” par “vo” de “volume”, et on obtient “un voxel”. Le deuxième fichier se nomme “Template.nii”. Ce fichier est similaire à la carte de probabilité, mais à la place d'avoir des probabilités, on a des voxels noirs et blancs. L'utilisation des templates est expliquée plus loin.

Finalement, pour décrire tout notre graphe, il y a un fichier nommé “Brain_Model.lgf”. On trouve dans ce fichier l'ensemble des arcs qui relient les nœuds entre eux. On trouve aussi à l'intérieur combien de fois, on a appris pour créer les différents atlas locaux. Ce que ça signifie, c'est combien de “image 3D” on a utilisé pour former la carte de probabilité et le template de l'atlas local. C'est donc sur ce principe que fonctionne SILA3D. Voici les différentes fonctionnalités de SILA3D :

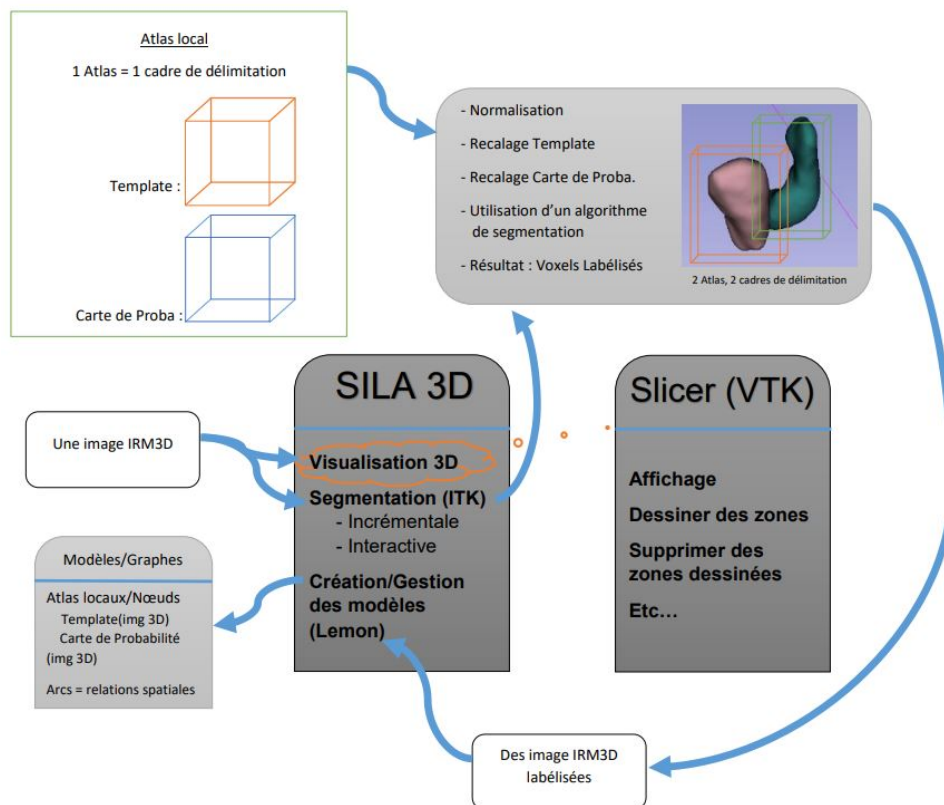


Figure 1.2 – Diagramme sur le fonctionnement de SILA3D

SILA3D utilise un modèle pré-entraîné pour pouvoir faire de la segmentation. L'avantage est que ce modèle peut être amélioré au fil du temps et des utilisations. De plus, le modèle n'a pas besoin de beaucoup d'image d'apprentissage pour commencer à être performant.

2 Acteurs

Acteurs :

- Client : Le client est mon encadrant, M. RAMEL.
- MOA : LANDRIN Maxime
- MOE : LANDRIN Maxime

3 Objectifs

Les objectifs sont de créer de nouvelles fonctionnalités sur SILA3D. Ces fonctionnalités ont pour but de faire une représentation du modèle, **externaliser des paramètres** et concevoir une documentation des classes.

3.1 Représentation du modèle

Les modèles qui sont utilisés dans SILA3D représentent des graphes. Ces graphes possèdent sur leurs nœuds des parties de l'anatomie et sur leurs arcs, on trouve les positions relatives de chaque nœuds les uns par rapport aux autres. Il s'agit donc de trouver un moyen d'afficher le graphe qui représente le modèle.

Après discussions, la solution choisie est d'afficher chaque nœud sur l'image. Sur Slicer, une image en 3D est visualisée en utilisant 3 plans :

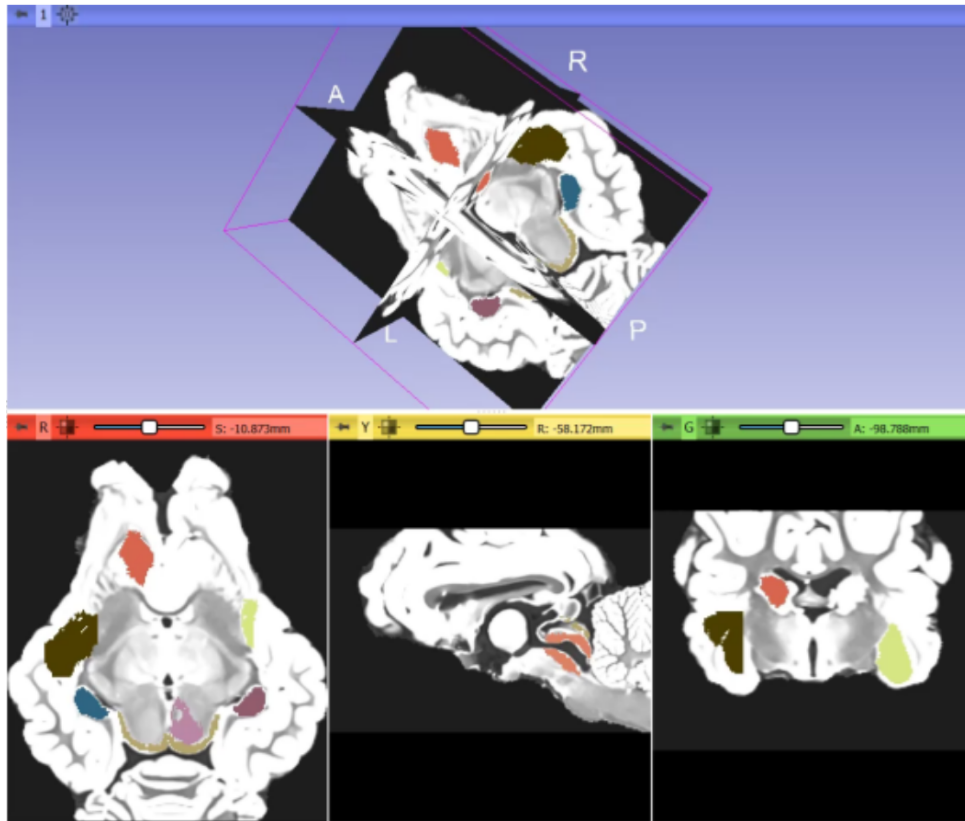


Figure 1.3 – Représentation d'une image 3D sur slicer

Comme on peut le voir dans la “figure 1.3”, il y a des zones colorées. Ces zones ont dû être segmentées une par une. Le but de ma fonctionnalité est de pouvoir afficher toutes les zones sans avoir à utiliser l'outil de segmentation. La segmentation utilise des outils comme le **recalage** et la carte de probabilité pour placer, au mieux, les zones et les colorier. En se passant de la segmentation, on souhaite simplement se servir des cartes de probabilité et de les afficher. De plus, une interaction et des manipulations doivent pouvoir être faite sur l'image.

L'affichage de toutes les régions est une des fonctionnalités. Une second fonctionnalité est de pouvoir afficher quelques régions choisies par l'utilisateur. Une interface peut être utilisée pour choisir les régions à colorier.

De plus, il est demandé à pouvoir afficher les arcs entre les régions. Lorsque cette option est choisie, il n'est pas demandé d'afficher les régions. Il sera donc possible de voir les liens entre les différentes zones du cerveau sélectionnées.

Plusieurs hypothèses ont été posées pour répondre à ces fonctionnalités.

3.2 Externalisation des paramètres

Lorsqu'une segmentation est utilisée dans SILA3D, une multitude de fonctions est alors utilisée. Dans ces fonctions, se trouvent des paramètres. Ces paramètres sont des valeurs qui permettent de régler la segmentation. L'objectif ici est de trouver les classes qui utilisent des fonctions qui

utilisent ces paramètres et de pouvoir modifier les paramètres.

Une fois les paramètres trouvés, on souhaite les extraire dans un fichier de configuration. De cette manière, on pourrait modifier les paramètres dans le fichier de configuration. Utiliser ce système implique la création de classe qui va avoir comme objectif de lire ce fichier config et de modifier les paramètres dans les diverses fonctions.

Plusieurs hypothèses ont été posées concernant l'utilisation et la lecture du fichier de configuration.

3.3 Documentation de SILA

SILA3D est une extension de Slicer où il y a beaucoup de classe et de fonctions. Malheureusement, une très faible partie de ce code est commentée ou documentée. L'objectif est donc de passer en revue chaque classe et de regarder chaque fonction pour les inscrire dans un document où l'on pourra trouver chaque classe.

Le document sera écrit au format texte. Il y aura une description des librairies externes utilisées dans les classes. De même, un diagramme de classe **Unified Modeling Language (UML)** sera fait pour voir la liaison entre chaque classe.

4 Hypothèses

Dans cette partie, j'explorerai les solutions proposées ainsi que les problèmes potentiels qui pourront être constatés. Cette partie sera séparée pour chaque objectif.

4.1 Représentation du modèle

Un premier problème est la taille du modèle. Le modèle est très grand et il se peut que tout charger ne soit pas possible. Plusieurs solutions sont de faire un "zoom" sur un nœud et de n'afficher que ses voisins. On peut afficher les nœuds principaux qui sont liés à lui. On peut utiliser des paramètres configurables dans l'interface pour indiquer le nombre de voisins que l'on souhaite afficher ou la distance maximale des voisins à afficher.

Dans tous les cas, cela implique de choisir la région qui sera au "centre". Le but de cette fonctionnalité étant de pouvoir tout afficher, le choix d'une région ne doit normalement pas se faire, car on souhaite simplement toutes les afficher. Cependant, dans le cas où il serait impossible de tout afficher, il faudra choisir une région pour pouvoir se concentrer sur ces voisins. Or, cette solution est la deuxième fonctionnalité demandée.

Un second problème est le fait de ne pas utiliser la segmentation. Omettre l'utilisation de la segmentation fait gagner beaucoup de temps. Mais ne pas l'utiliser implique une potentielle mauvaise position des régions. Comme on se base sur les cartes de probabilité de chaque région, on peut utiliser un paramètre qui permettrait de choisir un seuil. Ce seuil permettrait de n'afficher un voxel que si ce dernier possède une probabilité d'appartenir à la région au minimum valant la valeur du seuil. De cette manière, on évite d'avoir de trop grandes erreurs de placement.

De plus, sans segmentation, il n'est pas possible de créer une bordure à une région. Il y aura donc un bord flou. Un paramètre peut être utilisé pour régler la quantité de flous que l'on souhaite voir.

4.2 Externalisation des paramètres

Un premier problème va être de trouver les paramètres. Il y a de nombreuses classes dans SILA3D. De plus, s'il est demandé d'extraire ces paramètres, cela implique qu'ils ne sont pas forcément mis en avant. Pour remédier à ce problème, la recherche de paramètre sera faite en même temps que la documentation de SILA3D. De cette manière, les classes et les fonctions précises pourront être extraites.

Un second problème est l'écriture du fichier de configuration. Dans un premier temps, il était souhaité de modifier les variables directement dans le fichier. Après discussion, il a été proposé de faire une interface où chaque variable pourra être modifiée par l'interface. Cela rendra la manipulation des paramètres plus simple et moins archaïque.

Un troisième problème est la lecture du fichier de configuration. Dans un premier temps, il a été imaginé une importation du fichier via un bouton. La solution n'a pas plut et d'autre, on été proposé.

Une solution a été de lire le fichier lorsqu'on lance le logiciel de SILA3D. Mais la solution n'est pas pratique. En effet, si le logiciel est déjà lancé et que l'on modifie les paramètres, il faut alors fermer le logiciel puis le rouvrir pour charger les nouveaux paramètres. Une deuxième solution est de lire le fichier à chaque fois qu'une segmentation est faite. Cette solution est un peu plus coûteuse mais bien plus pratique. A chaque segmentation, le fichier est lu ce qui prend un peu de temps. L'avantage est qu'il n'y a plus besoin de redémarrer le logiciel SILA3D. C'est cette deuxième solution qui a été choisie.

4.3 Documentation de SILA3D

Il n'y a pas de réel problème quant à la documentation de SILA3D. Pour le diagramme de classe, gitmind[8] sera utilisé. Concernant la documentation des classes et des fonctions, la documentation sera faite dans un Google Doc[9].

5 Bases méthodologiques

Pour ce projet, une méthodologie agile sera utilisée avec la **méthode de KANBAN**. Ce qui implique beaucoup de communication avec l'encadrant, chose qui se fait déjà. Pour gérer l'évolution, l'utilisation d'un **diagramme de GANTT** sur un tableur excel est mis en place. Ce diagramme de **diagramme de GANTT** étant détaillé, il permet de suivre l'évolution du projet ainsi qu'à l'organiser. De plus, pour être en accord avec la méthode de KANBAN, les tâches décrites dans le **diagramme de GANTT** peuvent être mises à jour en fonction de si la tâche reste à faire, est faite, ou est en cours.

Lors de ma programmation, j'ai établi quelques règles :

- Convention de nommage
- Un code commenté
- Une indentation claire
- Si j'ai le temps, des tests d'intégrations et unitaire

Pour la convention de nommage, elle m'a été "imposée" puisqu'il faut reprendre un travail précédemment effectué. On a donc :

- Nom des fichiers : Pascal Case, mots commençant par une majuscule et liés sans espace.
Exemple : `MyFileName`.
- Nom des classes : Pascal case, mots commençant par une majuscule et liés sans espace.
Exemple : `MyClassName`
- Nom des méthodes : Camel case, mots commençant par une minuscule et liés sans espace.
Exemple : `myMehtod`
- Nom des constructeurs et destructeurs : Pascal case, mots commençant par une majuscule et liés sans espace. Exemple : `MyClassConstructor`
- Nom des variables : Camel case, mots commençant par une minuscule et liés sans espace.
Exemple : `myVariable`
- Nom des variables globales : Cobra case majuscule, tout écrit en majuscule et liés par des “_”. Exemple : `GLOBAL_VARIABLE`

En plus de ces outils, un git existe déjà pour SILA3D. Les modifications apportées au code seront vérifiées avant d’être envoyées sur le git par les responsables du git.

2

Description générale

1 Environnement du projet

Comme précédemment expliqué, ce projet de recherche et développement se base sur un logiciel déjà existant, le logiciel étant Slicer. Slicer requiert une installation sur un ordinateur, ce n'est pas un outil web. C'est un logiciel open source qui a pour but de simplifier la visualisation des images d'IRM ou d'image en 3D. Slicer peut, en plus de les visualiser, Slicer permet de modifier les images en les coloriant, découpant, etc. . .

Mais en plus, on ajoute à ce logiciel une extension qui est SILA3D. Ce dernier permet l'utilisation d'un modèle qui va permettre de segmenter des zones de l'anatomie en procédant à des segmentations. Pour se faire, SILA3D utilise un graphe qui permet de représenter l'anatomie dont il est question.

L'utilisateur utilise l'interface de SILA3D pour procéder aux diverses fonctionnalités. L'information de ce que souhaite faire l'utilisateur est ensuite envoyée au serveur qui va lancer les diverses fonctions.

Voici un diagramme correspondant au logiciel actuel :

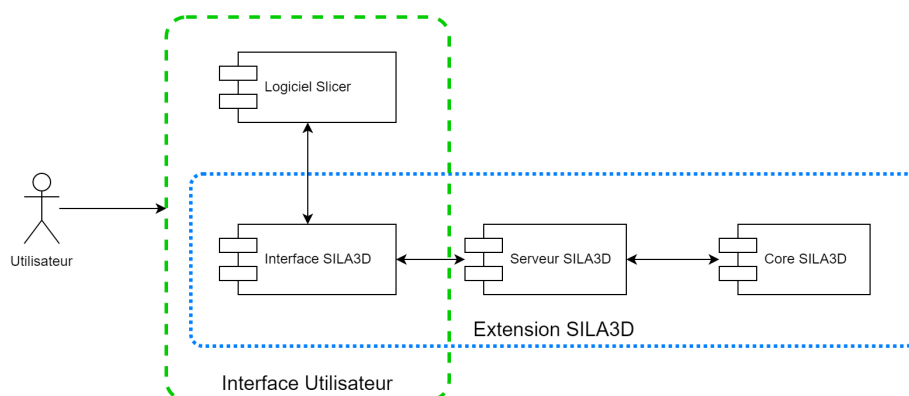


Figure 2.1 – Environnement actuel de SILA3D

Comme on peut le voir, notre environnement se découpe en deux parties, la verte et la bleue. La partie verte est la partie visible par l'utilisateur, c'est l'interface. La partie bleue est quant à elle "cachée". Elle permet d'effectuer les différentes fonctionnalités proposées dans SILA3D.

1.1 Interface Utilisateur

L'interface utilisateur est ce qui est visible par l'utilisateur. L'utilisateur peut interagir avec cet environnement et utiliser les diverses fonctionnalités présentes.

Sur l'image, on peut voir qu'il y a deux parties. Sur la gauche, il y a l'interface de SILA3D qui permet, entre autres, de faire des segmentations. Sur la droite, l'interface de Slicer qui permet de visualiser l'image 3D et de la parcourir.

1.2 Extension SILA3D

Pour mieux comprendre cette partie, il faut en expliquer chaque section. Comme on l'a précédemment vu, la première section "Interface SILA3D" se trouve sur la partie gauche du logiciel.

Les deux autres sections se font plus discrètes. La section "Serveur SILA3D" est légèrement visible. En effet, avant de pouvoir utiliser les diverses fonctionnalités de l'extension, il vous est demandé de vous connecter. Cette connexion permet d'entrer dans le serveur et d'utiliser les fonctionnalités qui permettent, entre autres, la segmentation.

Finalement, la section "Core SILA3D" est la partie la plus cachée. C'est pourtant elle la plus importante. C'est elle qui contient toutes les opérations mathématiques qui s'appliquent. Grâce à cette section, on peut réaliser la segmentation, le recalage ou encore l'utilisation de la chaîne de Markov. Mais pas seulement, c'est aussi elle qui permet de charger le modèle, de le modifier et de le sauvegarder. Comme vous l'avez compris, il s'agit du cœur de l'extension et qui s'occupe de faire toutes les fonctionnalités.

En ce qui concerne mon projet, il faut développer une nouvelle fonctionnalité. Cela signifie qu'il faudra majoritairement intervenir dans la partie "Core SILA3D". Mais en plus, il me faudra ajouter une nouvelle page dans l'interface. Voici mon diagramme :

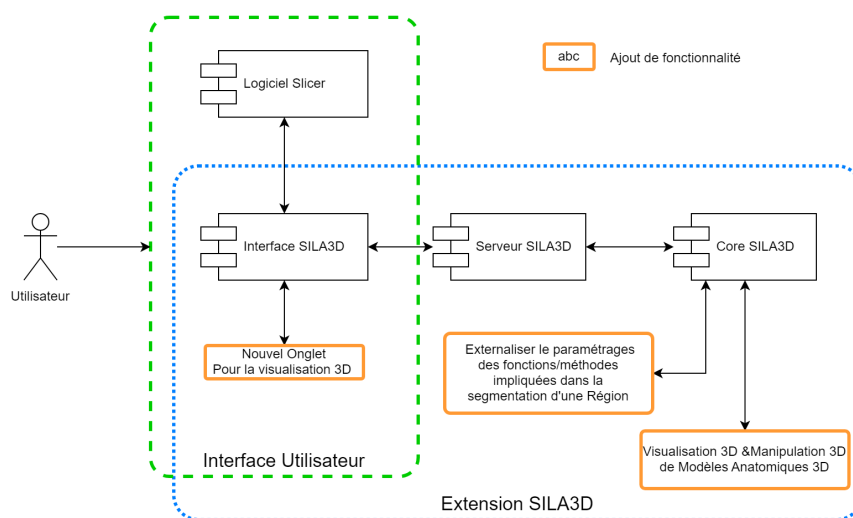


Figure 2.2 – Environnement de SILA3D avec les nouvelles fonctionnalités

La nouvelle page/nouvel onglet sur l'interface de SILA3D se situe au niveau de l'interface. Pour gérer cette interface, il faut ajouter les diverses fonctionnalités qui vont permettre sa visualisation dans la section “Core SILA3D” qui est, pour rappel, le cœur qui fait tout.

Pour **externaliser des paramètres**, il va falloir modifier les classes existantes et en ajouter des nouvelles. Les nouvelles classes permettront de modifier le fichier de configuration et de le lire pour en extraire les données. Ces données seront ensuite envoyées lors de la segmentation.

2 Caractéristiques des utilisateurs

Les personnes qui utilisent slicer peuvent être représentés sous 2 types d'utilisateurs. Quelque soit l'utilisateur, l'interface doit être simple à prendre en main et intuitive. Les utilisateurs seront souvent des professionnels de santé qui peuvent ou non avoir des connaissances en informatique. Mon projet étant de créer une nouvelle fonctionnalité, il est prévu de fournir un manuel utilisateur montrant comment utiliser mon implémentation.

Les différents statuts :

- Admin : L'admin a accès au logiciel et possède tous les droits. Les admins sont les propriétaires de l'extension de SILA3D. Un Admin peut rendre un modèle “public”. Rendre “public” un modèle permet de le rendre disponible à tous le monde.
- Utilisateur (User) : L'utilisateur a accès au logiciel mais est restreint sur certaines fonctionnalités. Les utilisateurs peuvent être des médecins ou des néophytes.

3 Fonctionnalités du système

On parle ici des fonctionnalités qu'il va falloir développer. Pour mieux comprendre les fonctionnalités déjà présentes dans SILA3D, merci d'aller voir mon état de l'art.

Mon système se compose de deux parties. La première partie est l'extraction et l'externalisation de paramétrage des fonctions/méthodes qui sont impliquées dans la segmentation d'une région. La seconde partie consiste à créer un nouvel onglet à SILA où l'on pourrait visualiser le graphe qui représente le modèle chargé.

3.1 Représentation du modèle

N'étant pas habitué aux bibliothèques **Insight Segmentation and Registration Toolkit (ITK)** et **Visualization ToolKit (VTK)**, je n'ai pour l'instant pas d'idée sur la méthode qu'il va falloir utiliser pour atteindre mon objectif. Mon encadrant, M. RAMEL, m'a mis en contact avec M. Mostafa DARWICHE. M. DARWICHE est une des personnes ayant participé au développement de SILA3D. C'est ce dernier qui octroie les droits pour avoir accès au code. Il a aussi aidé pour l'installation de l'extension sur la machine, a aussi expliqué le fonctionnement de SILA3D et, ce qu'il a déjà essayé de faire.

Une première piste, qui m'a été donnée par M. DARWICHE, est de regarder la classe `AnaModelManagement`. En effet, cette classe possède un objectif similaire à celui qui m'a été fixé. Cette classe sera donc un bon premier point de départ. De plus, M. DARWICHE a proposé de le contacter si une aide est nécessaire pour concevoir la modélisation. Mon objectif est donc de continuer le travail effectué dans la classe `AnaModelManagement` et de rendre cette partie fonctionnelle.

3.2 Externalisation des paramètres

La première partie repose sur une bonne compréhension du code existant. En effet, le but est de retrouver les fonctions/méthodes, de les comprendre et de trouver les différents paramètres qui permettent la segmentation d'une région. Aucune documentation n'étant fournie, il m'a fallu passer en revue la plupart des classes de SILA et de les documenter une par une.

Une fois les différentes méthodes trouvées, il faut penser à l'interface. Dans l'interface, qui sera détaillée plus loin, il sera possible de modifier ces paramètres :

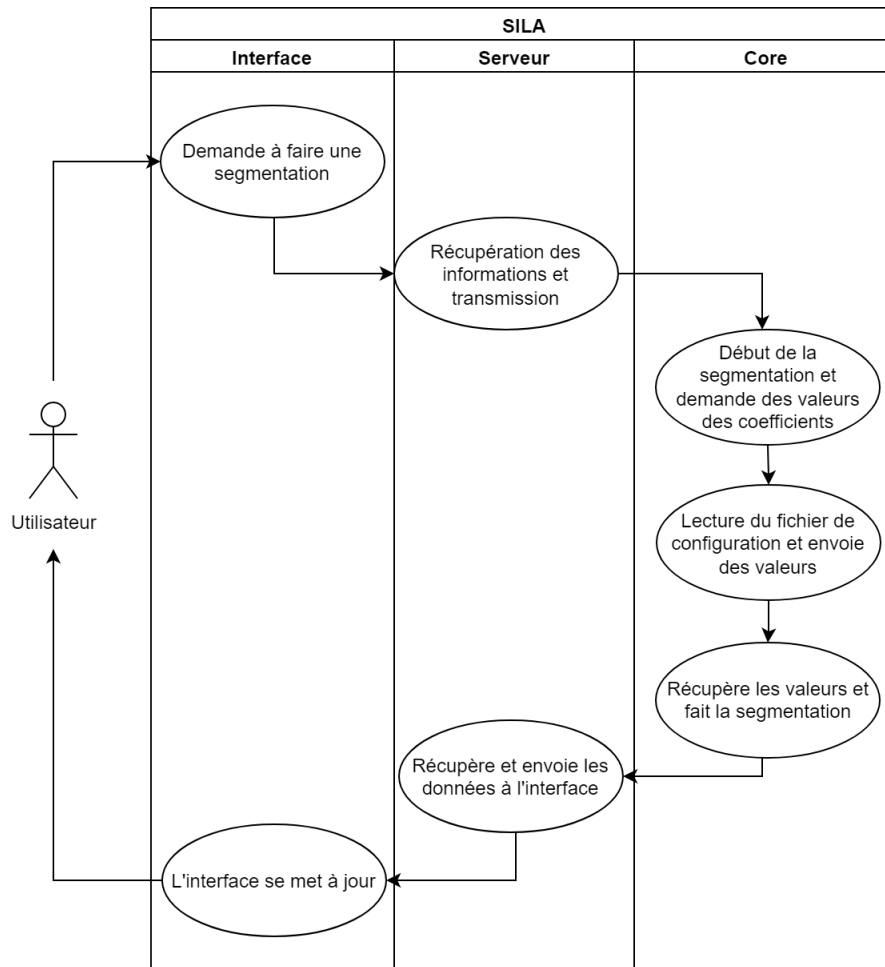


Figure 2.3 – Fonctionnalité d'externalisation

3

État de l'art / Veille technologique

1 Analyse de l'existant

On parle ici du fonctionnement de Slicer et de SILA3D. Concernant le fonctionnement de Slicer, il se base sur deux technologies **ITK** et **VTK**.

Concernant le fonctionnement de SILA3D, l'extension s'utilise d'une certaine manière et son utilisation est relativement simple. En revanche, malgré une utilisation simple, elle requiert des connaissances en anatomie pour utiliser les fonctionnalités.

1.1 Slicer, ITK et VTK

Ici, un petit rappel de ce qui a été dit dans l'introduction de ce document ainsi quelque précision supplémentaire pour mieux comprendre la provenance de ces technologies.

SILA3D s'inscrit dans un projet proposé par la région Centre-Val de Loire[6]. La région Centre-Val de Loire propose plusieurs projets où chaque sujet possède une thématique ainsi qu'une problématique et un ou plusieurs objectifs. Parmi ces sujets, il est proposé le projet NeuroGéo qui a pour thème "Nutrition, santé, bien-être". Le projet se déroule sur 3 ans et a un budget estimé à 200 k€ par an.

La problématique que décrit le projet est qu'il existe divers outils d'analyse d'images IRM, mais qu'ils sont très spécifiques. *"La majorité des outils actuellement disponibles sont spécifiques à certaines espèces (Humain, rongeurs) et/ou nécessitent des connaissances poussées en analyse d'images. L'ambition du projet est de développer une application de segmentation"*[7]

Le premier objectif de ce projet est de proposer un outil adaptable sur plusieurs espèces. Il s'agirait d'un outil interactif, simple et intuitif pour des néophytes. L'outil devra être interactif au niveau de la segmentation des images IRM. Si l'objectif est atteint alors un second objectif est proposé, créer un module de consultation et d'aide à la décision. Dans le cas de SILA3D, la consultation consiste à voir les images IRM de manière simple, j'expliquerai en détail plus tard. En ce qui concerne la décision, c'est dans le cas de SILA3D, de proposer à l'utilisateur de sélectionner une zone de l'anatomie humaine. Par exemple, le cervelet, le logiciel mettra cette

zone en avant aidant alors l’utilisateur. L’utilisateur pourra alors corriger la proposition en la modifiant ou la valider.

C’est donc dans ce cadre que SILA3D a été créé. SILA3D se base avant tout sur une ancienne technologie open source, 3D Slicer ou plus simplement Slicer[15]. Slicer est un ancien logiciel créé lors d’une thèse en 1998 au laboratoire d’intelligence artificielle du MIT. Il a pour but de simplifier la visualisation des images d’IRM.

Slicer sera mis à jour au fil des années, le rendant de plus en plus pratique, populaire et esthétique. C’est lors de sa deuxième version qu’il commence à faire parler de lui et où il commence à être téléchargé. Une troisième version sort en 2007 et c’est en 2009, lorsqu’un nouvel outil nommé “Qt”[13] apparaît que la mise en place d’une quatrième version débute. Il faudra attendre deux ans pour pouvoir voir Slicer dans sa quatrième version, en 2011.

Le logiciel est un logiciel open source ce qui lui a permis de se voir greffer des nouvelles technologies au fil du temps, tel que SILA3D. Slicer utilise notamment deux bibliothèques nommées **VTK**[16] et **ITK**[5].

Les bibliothèques **VTK** et **ITK** semblent similaires, mais sont bien deux choses différentes. **VTK** est une bibliothèque de visualisation. Avec cette dernière, on peut créer des images en partant de rien. Elle peut donc générer des images avec des données.

La bibliothèque **ITK** est utilisée pour traiter des images déjà existantes. Ainsi, cette bibliothèque ne crée pas d’image, mais les modifie. Un exemple simple. Avec la bibliothèque **VTK**, vous pouvez créer une montagne en 3D. Et en utilisant la bibliothèque **ITK**, vous pouvez modifier cette image pour, par exemple, marquer les altitudes en utilisant des lignes de couleurs.

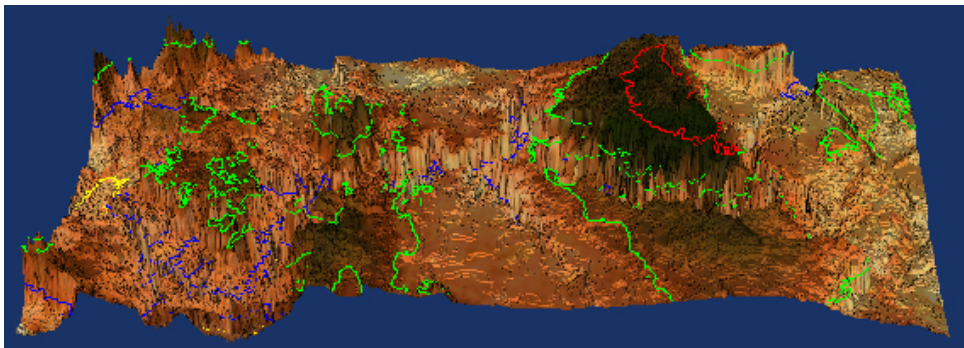


Figure 3.1 – *Visualization Toolkit (VTK) Tutorial, John T. Bell*
source : cs.uic.edu/~jbell/CS526/Tutorial/Tutorial.html

1.2 Comment se sert-on de SILA3D ?

Lorsque l’on ouvre SILA3D et que l’on se connecte, on demande à charger une image IRM.

L’image est alors affichée à l’écran grâce à Slicer et l’utilisation de la bibliothèque **VTK**. Une fois l’image chargée, on demande à l’utilisateur de choisir un modèle correspondant à l’image. Si on met une image du cerveau humain, on choisit un modèle du cerveau humain. Suite à ça, on peut alors sélectionner une région/un atlas local, que l’on souhaite mettre en avant.

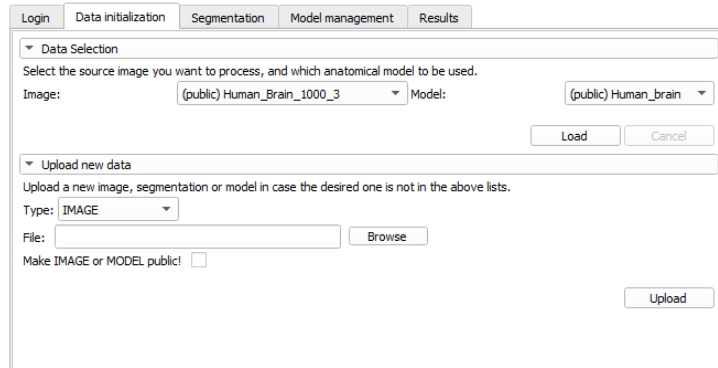


Figure 3.2 – SILA3D - Sélection du modèle

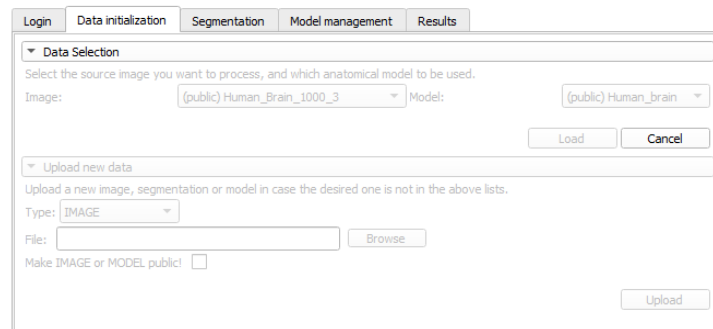


Figure 3.3 – SILA3D - Modèle sélectionné

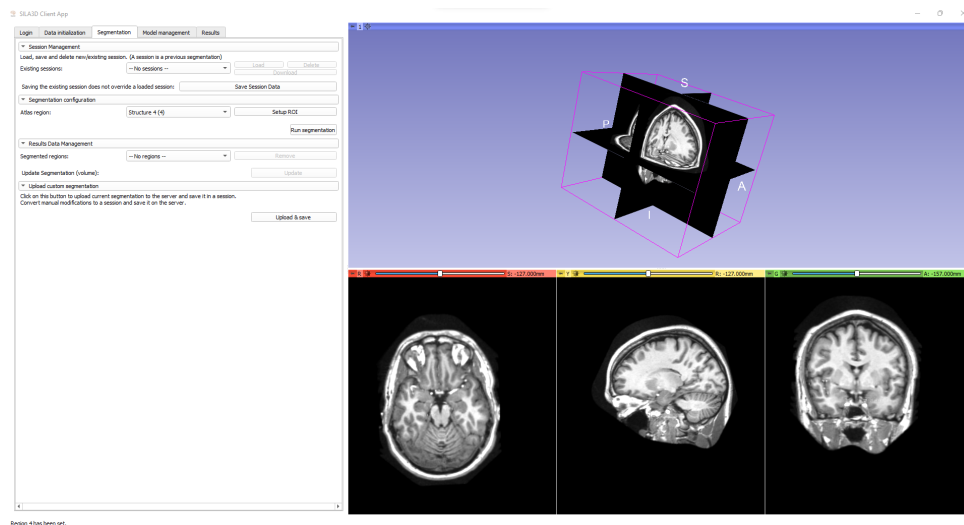


Figure 3.4 – SILA3D - Modèle sélectionné et chargé

L'utilisateur va donc choisir un atlas local à segmenter. En utilisant le modèle pré-entraîné présent dans SILA3D, le logiciel va essayer de délimiter la zone et de la mettre en évidence. Pour délimiter la zone, il va utiliser le template et la carte de probabilité présent dans chaque atlas.

Lorsque l'on va, pour la première fois, choisir un atlas local, on demande à placer une "ROI" ou "Region Of Interest". C'est une boîte que l'utilisateur doit placer pour indiquer au logiciel où se trouve l'atlas local. Cette opération n'est à faire que pour le premier atlas local. C'est à cette étape que des connaissances en anatomie sont requises.

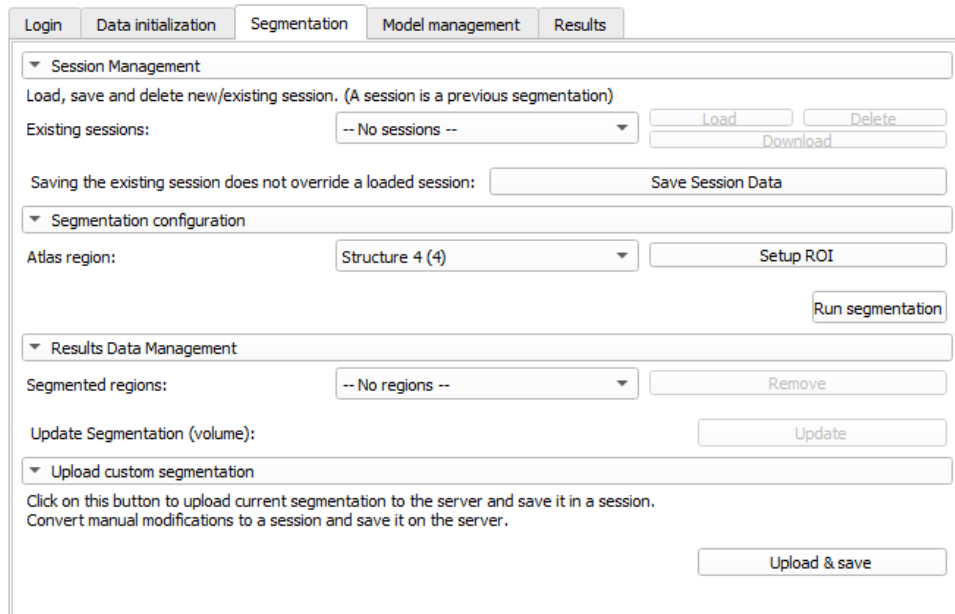


Figure 3.5 – SILA3D - Sélection de la zone à segmenter

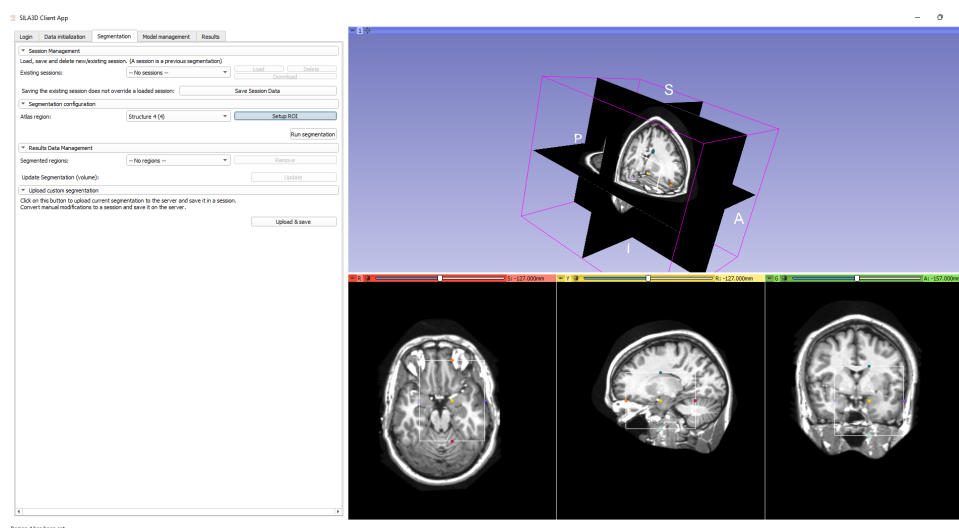


Figure 3.6 – SILA3D - Positionner le zone d'intérêt

En effet, sur SILA3D, les atlas possèdent des noms comme “structure 46”. Seulement, pour des néophytes, cela n’indique en aucun cas le nom ou la position de l’atlas local. Il faut donc faire des recherches sur Internet pour savoir à quoi chaque numéro de structure correspond. Par exemple, la structure 33 de SILA3D correspond en fait à “l’aire 33” du cerveau, c’est-à-dire le “prégénuaire”. Une fois cette information trouvée, il faut savoir la positionner.

C’est quelque chose d’assez fastidieux car chaque structure ou aire du cerveau n’est pas forcément définie par un numéro. Il est recommandé d’aller voir ce site qui est malheureusement payant : imaio.com[3]. Ce site internet permet de retrouver les aires du cerveau, le nom ainsi que sa position.

Une fois la région d’intérêt positionnée, on peut alors effectuer la segmentation. Ce qu’il faut comprendre, c’est que dans la région d’intérêt, se trouve notre atlas local ainsi que d’autre région. La segmentation va chercher dans la région d’intérêt, la forme du template de l’atlas local. Une

fois que la segmentation a trouvé une forme similaire, elle va déplacer, tourner et redimensionner le template pour essayer de le placer au mieux avec la zone qu'elle a trouvé.

Le template mis en place, la segmentation va placer par-dessus la carte de probabilité présente dans l'atlas local. De cette manière, elle arrive à déterminer quel voxel fait partie ou non de la structure que l'on recherche. La fonction va donc labelliser chaque voxel présent et dire si oui ou non il fait partie de l'atlas local sélectionné. Finalement, on affiche les voxels labellisés en utilisant Slicer.

On a alors une visualisation 3D de la structure sélectionnée. Avec cette structure nouvellement affichée, on peut en utilisant les outils de Slicer, redessiner la forme de la structure pour l'affiner et la faire encore mieux coller à l'image. Et c'est cette opération qui va faire apprendre à notre modèle. En modifiant l'affichage de la structure, on va pouvoir sauvegarder nos changements et mettre à jour la carte de probabilité et le template de la zone dont il est question.

Maintenant, on peut recommencer l'opération. En effet on a, ici, d'afficher qu'une seule structure. Mais le logiciel ne s'arrête pas là, car il permet de charger d'autres atlas locaux et de les afficher eux aussi. On pourra ainsi petit à petit rajouter des structures sur notre IRM. Et lorsque l'on va sélectionner une nouvelle région à segmenter, SILA3D va proposer une position pour la zone d'intérêt.

Pour rappel, SILA3D se base sur un modèle qui est un graphe. Ce graphe a pour nœud les atlas locaux et il a pour arc les distances relatives. C'est donc en utilisant les arcs qui relient chaque atlas local entre eux que SILA3D arrive à proposer une position pour la région d'intérêt.

2 Etat de l'art

Dans cette section, on va plus entrer dans les détails. Tout comme précédemment, cette section sera partagée en deux. Tout d'abord, des détails concernant **ITK** et **VTK**. Puis, on parlera plus en détail des fonctions qui permettent la segmentation et des paramètres utilisés.

2.1 Slicer - ITK & VTK

2.1.1 ITK - Insight Toolkit

Les informations qui suivent se trouvent sur leur site internet[12].

ITK est l'un des plus grands et anciens projets open source dans la communauté scientifique. ITK a été construit par des centaines de membres de la communauté à travers le monde entier où chacun possède diverses connaissances, personnalités et expériences travaillant pour des recherches ouvertes et reproductible dans l'analyse d'image.

La librairie ITK est une librairie open source et multiplateforme qui fournit aux développeurs une grande quantité d'outil pour l'analyse d'image. Développé avec une méthodologie de programmation extrême, ITK s'appuie sur une architecture approuvée de traitement orientée sur le spatial. Elle comprend de la segmentation et du recalage d'image scientifique en deux, trois ou plus dimensions.

Leur objectif est de :

- Établir une fondation pour la recherche future et reproductible.
- Créer un dépôt d'algorithmes fondamentaux.
- Développer une plateforme pour le développement de produits avancés
- Soutenir les applications commerciales de la technologie.
- Créer une convention pour les travaux futurs.
- Faire grandir une communauté autonome d'utilisateurs logiciels et de développeurs.

2.1.2 VTK - Visualization Toolkit

Les informations qui suivent se trouvent sur leur site internet[17].

VTK est un ensemble de logiciels open source utilisée pour faire de l'infographie 3D, de la modélisation, du traitement d'image, du rendu de volume, de la visualisation scientifique, et du traçage en 2D.

VTK contient une grande variété d'algorithmes de visualisation et des techniques de modélisation avancées. **VTK** tire parti du traitement parallèle à mémoire distribuée et à threads pour respectivement la vitesse et l'évolutivité.

VTK est conçu pour être agnostique en termes de plateforme. Cela signifie qu'il fonctionne globalement partout, y compris sur Linux, Windows et Mac, sur le Web et sur les appareils mobiles.

VTK utilise le processus logiciel de qualité de Kitware, qui comprend CMake, CTest, CDash et

CPack pour construire, tester et emballer le système. Deplus, **VTK** possède une forte communauté de développeurs distribués, le résultat est un code robuste et de très haute qualité. La fonctionnalité de base de VTK est écrite en C++ pour maximiser l'efficacité. Cette fonctionnalité est enveloppée dans d'autres langages afin de l'exposer à un public plus large. L'interopérabilité avec Python est particulièrement bien définie.

En tant que logiciel open source, **VTK** est libre d'être utilisé à toutes fins. Techniquement, **VTK** a une licence de type BSD, qui impose des restrictions minimales pour les applications à code source ouvert et fermé.

2.2 SILA3D

SILA3D est une extension complexe et très travaillée. Pour s'y retrouver, un diagramme de classe **UML** a été conçu.

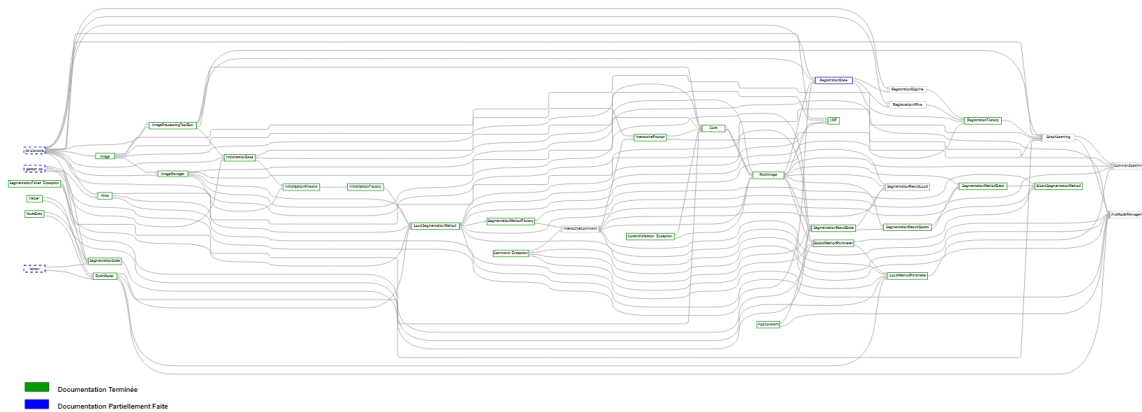


Figure 3.7 – *SILA3D - Diagramme de classe UML*

Comme on peut le constater, SILA3D se compose de nombreuses classes, 36 au total. Chacune participe au bon fonctionnement de SILA3D.

2.2.1 Les classes externes

SILA3D s'appuie sur l'utilisation de classe externes. Ces classes ont pour but d'aider à implémenter les diverses fonctionnalités. Tel que la gestion de graphe pour le modèle, la gestion des lignes de commandes, la gestion de la segmentation et du recalage.

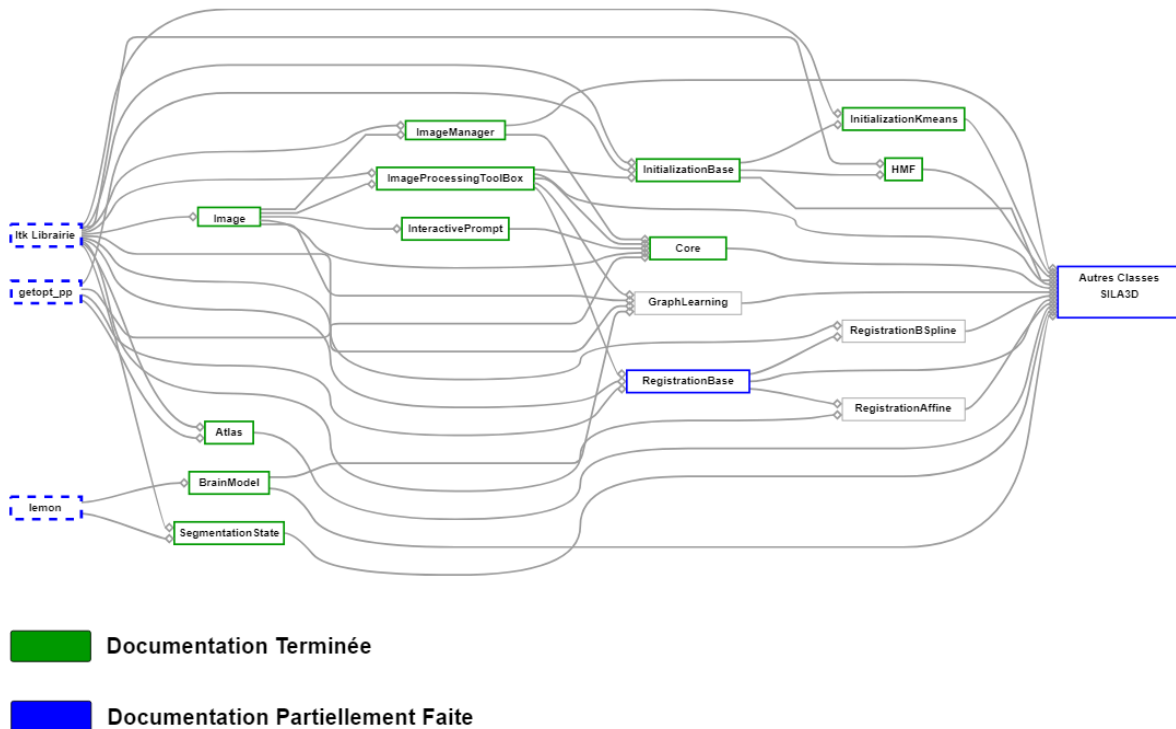


Figure 3.8 – SILA3D - UML Dépendance & Simplification - Classes Externes

"Library for **E**fficient **M**odeling and **O**ptimization in **N**etworks" ou plus simplement, lemon[11] ; Une librairie qui permet de gérer l'utilisation, la modification et la modélisation de graphe.

getopt_pp[10] ; Une librairie qui sert à décomposer de manière simple les lignes de commande et de récupérer les valeurs associées aux paramètres. Elle se dit simple d'utilisation et facile à apprendre.

Et finalement, la librairie ITK[5]. Cette librairie se compose d'une multitude de classe, voici une liste exhaustive des classes utilisées dans SILA3D :

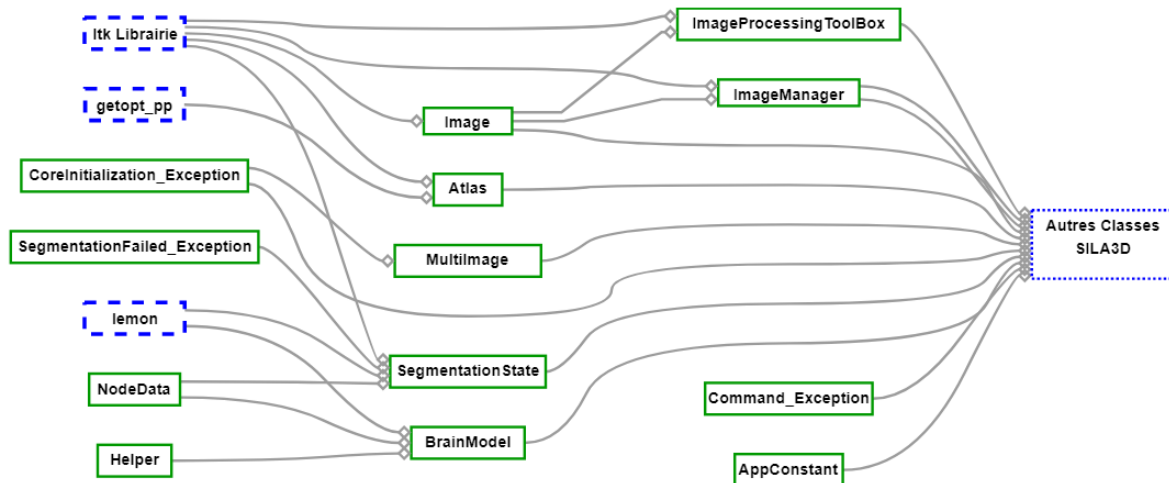
- itkAffineTransform
- itkBSplineTransform
- itkBSplineTransformInitializer
- itkCenteredTransformInitializer
- itkCorrelationImageToImageMetricv4
- itkDecisionRule
- itkDistanceToCentroidMembershipFunction
- itkEuclideanDistanceMetric
- itkExtractImageFilter
- itkGaussianDistribution
- itkHistogramMatchingImageFilter
- itkImage
- itkImageFileReader

- itkImageFileWriter
- itkImageRegistrationMethodv4
- itkKdTreeBasedKmeansEstimator
- itkKdTree
- itkLabelMap
- itkLBFGSOptimizerv4
- itkListSample
- itkMattesMutualInformationImageToImageMetricv4
- itkMeanSquaresImageToImageMetricv4
- itkMinimumDecisionRule
- itkMinimumMaximumImageCalculator
- itkNearestNeighborInterpolateImageFunction
- itkNormalVariateGenerator
- itkRegularStepGradientDescentOptimizer
- itkRegularStepGradientDescentOptimizerv4
- itkRescaleIntensityImageFilter
- itkResampleImageFilter
- itkSampleClassifierFilter
- itkScaleTransform
- itkVariableLengthVector
- itkVariableSizeMatrix
- itkWeightedCentroidKdTreeGenerator

Ces 35 différentes classes permettent le bon fonctionnement de SILA3D.

2.2.2 Les classes "simples"

Parmi les classes "simples", on trouve des classes utilitaires, des classes qui permettent de gérer des images et des modalités, des classes qui gèrent les diverses exceptions et les classes qui représentent le modèle.



Documentation Terminée

Documentation Partiellement Faite

Figure 3.9 – SILA3D - UML Dépendance & Simplification - Classes Simples

Les classes qui représentent le modèle. Elles sont au nombre de trois ; NodeData qui représente les données des nœuds du graphe comme les coordonnées d'origine et la dimension ; Atlas qui représente les atlas du modèle et BrainModel qui représente le modèle.

On trouve également les classes qui gèrent les exceptions qui sont au nombre de 3 ; SegmentationFailed_Exception qui gère les erreurs lors de la segmentation, Command_Exception qui gère les erreurs de commande et CoreInitialization_Exception et gère tous les autres problèmes qui peuvent survenir.

Les classes utilitaires nommées Helper et AppConstant. La première permet de calculer des durées, de découper des strings, de créer des fichiers temporaires ou de récupérer le nom d'un fichier en donnant le chemin vers ce dernier. La seconde permet de stocker des variables globales de l'application.

Les classes qui permettent la représentation et la gestion d'image ; Image qui représente les images, ImageManager qui permet de gérer les images, ImageProcessingToolBox qui contient des outils pour modifier les images et MultiImage qui gère les images multi-modales.

2.2.3 Les classes Initialization

Ces classes vont s'appuyer sur des classes ITK pour initialiser certaines fonctions comme le calcul de KMEANS. Comme on peut le voir dans la figure qui suit, InitializationBase va être inclus dans InitializationKmeans qui sera elle-même incluse dans InitializationFactory.

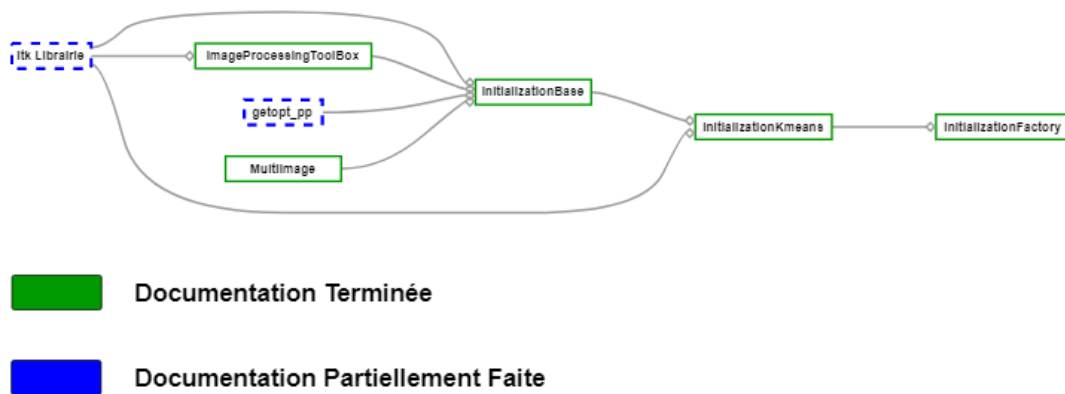


Figure 3.10 – SILA3D - UML Dépendance & Simplification - Classes Initialization

On trouve dans InitializationBase l'inclusion de 4 classes de librairie "ITK". La classe possède comme attribut le nombre de modalité, le nombre d'itération, le nombre de cluster, un ensemble d'image, un pointer vers les données contenu dans les cartes de probabilités et une matrice qui contient des paramètres statistiques.

2.2.4 Les classes "Method Parameter"

Il n'y a que deux classes.

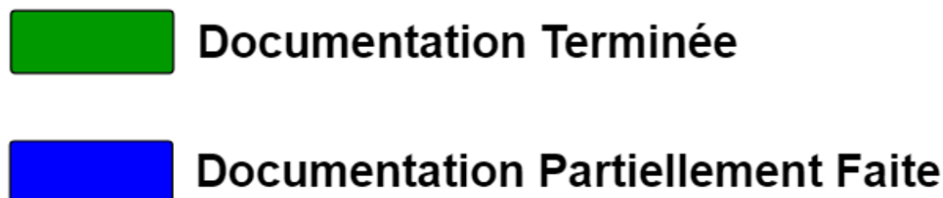
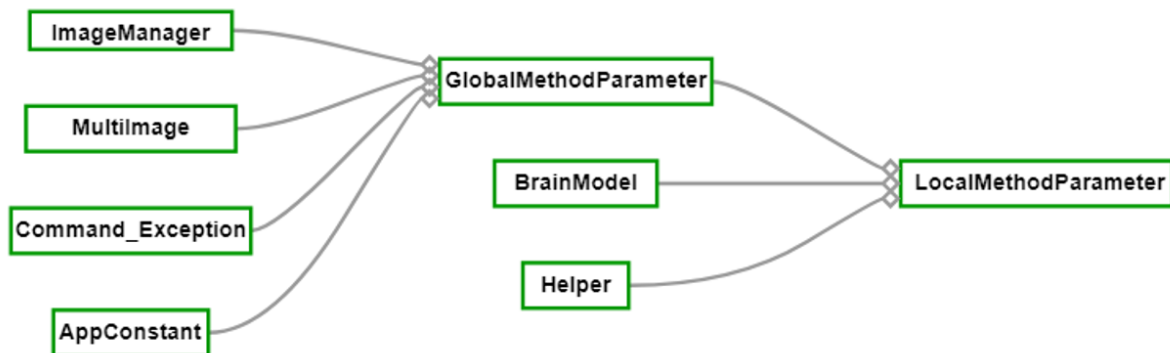


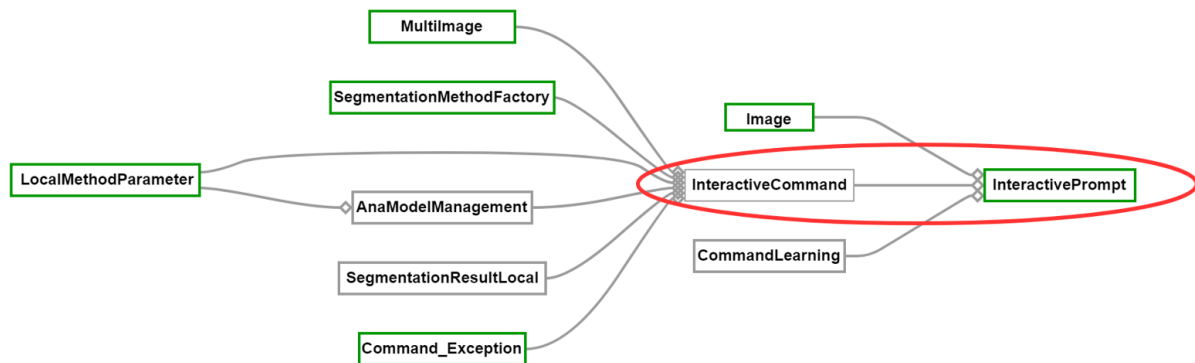
Figure 3.11 – SILA3D - UML Dépendance & Simplification - Classes MethodParameters

La première est GlobalMethodParameter. Cette classe permet de choisir le mode (Segmentation ou Apprentissage), de choisir l'image à segmenter, le mode de segmentation, la modalité principale, l'image labellisée, l'endroit où l'on souhaite avoir le résultat final et l'identifiant de l'utilisateur. Cette classe se compose de 20 méthodes qui sont principalement des "Getters" et "Setters".

La seconde est LocalMethodParameter. Cette classe permet de choisir et de définir l'identifiant de la région à segmenter, la méthode de recalage, le modèle, la position de la boîte de délimitation de la région ainsi que sa dimension. Cette classe se compose de 30 méthodes.

2.2.5 Les classes pour l'invite de commande

Il n'y a que deux classes dans cette catégorie. Ces classes permettent de récupérer les commandes envoyées dans l'invite de commande et d'appeler les fonctions qui doivent être appelées.



Documentation Terminée

Documentation Partiellement Faite

Figure 3.12 – SILA3D - UML Dépendance & Simplification - Classes Interactives

La première est InteractiveCommand. Cette classe permet de gérer les commandes utilisées dans l'invite de commande. Elle récupère les commandes dans un dictionnaire qui stocke en clef une chaîne de caractère et en valeur la fonction à appeler. Pour pouvoir appeler les diverses fonctions, elle possède en attribut la classe "LocalMethodParameter" ainsi que la classe "SegmentationResultBase". Cette classe se compose de 14 méthodes

La seconde classe est InteractivePrompt. Cette classe récupère en attribut la classe précédemment citée, "InteractiveCommand". Elle possède un deuxième attribut du type "CommandLearning". Avec ces deux attributs, la classe peut lire des lignes de commandes et exécuter la fonction voulu. Cette classe se compose de 5 méthodes.

2.2.6 Les classes pour la segmentation

C'est l'une des plus grandes familles de classe de SILA3D, il y a 9 différentes classes. Elles permettent d'implémenter la segmentation du modèle sur tous les niveaux.

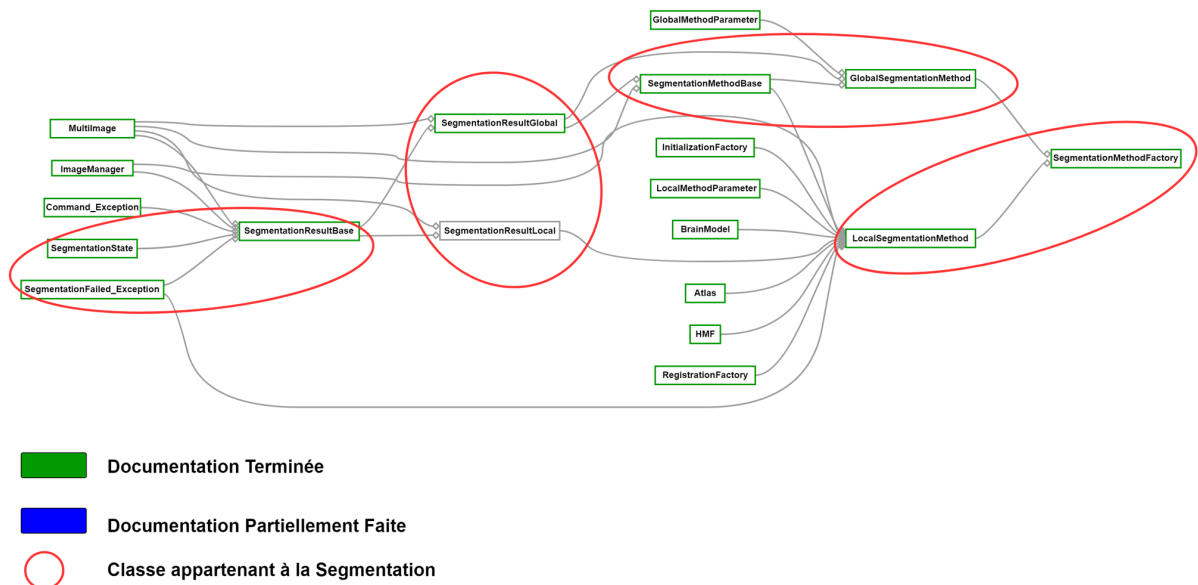


Figure 3.13 – *SILA3D - UML Dépendance & Simplification - Classes Segmentations*

Trois de ces classes permettent de récupérer l'état actuel de la segmentation, de gérer les erreurs qui peuvent provenir durant la segmentation et de créer une base pour récupérer les résultats de la segmentation. Ces classes sont respectivement `SegmentationState`, `SegmentationFailed_Exception` et `SegmentationResultBase`.

Deux classes de la famille permettent de récupérer les résultats des segmentations. L'une va récupérer les résultats provenant d'une segmentation globale et l'autre va récupérer les résultats provenant d'une segmentation locale. Ces classes sont respectivement `SegmentationResultGlobal` et `SegmentationResultLocal`.

La classe `SegmentationMethodBase` représente la base des méthodes pour la segmentation. Cette classe sera utilisée pour les segmentations locales et globales.

La classe `GlobalSegmentationMethod` hérite de la classe `SegmentationMethodBase` et va permettre de définir les méthodes pour la segmentation globale.

La classe `LocalSegmentationMethod`, hérite de la classe `SegmentationMethodBase` et va permettre de définir les méthodes pour la segmentation localr.

Finalement, la classe `SegmentationMethodFactory` qui possède dans ses attributs un type `SegmentationMethodBase` qui est la classe mère des deux classes `GlobalSegmentaionMethod` et `LocalSegmentationMethod`. Cette classe est celle qui est utilisée dans *SILA3D*. C'est elle qui va appeler les diverses fonctions qui sont présentent dans les classes utilisées pour la segmentation.

2.2.7 Les classes pour le recalage

Cette famille de classe contient 4 différentes classes. Elles permettent d'implémenter le recalage du modèle en utilisant plusieurs méthodes.

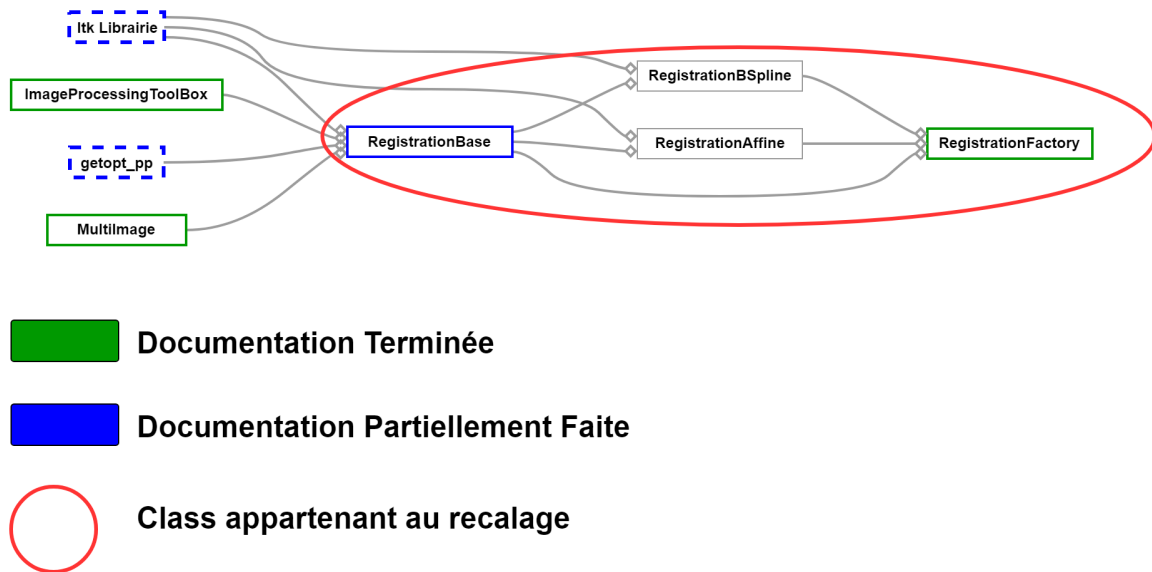


Figure 3.14 – SILA3D - UML Dépendance & Simplification - Classes Registration

La classe RegistrationBase est la classe qui va poser les bases pour le recalage.

Il y a ensuite deux méthodes de recalage qui sont représentées dans deux classes, respectivement RegistrationBSpline et RegistrationAffine. Ces deux classes héritent de la classe RegistrationBase. Ces classes s'appuient sur l'utilisation de ITK et de ces outils pour le recalage[4]

La première classe utilise une méthode pour le recalage qui est nommé B-Spline. *"En mathématiques, une B-spline est une combinaison linéaire de splines positives à support compact minimal. Les B-splines sont la généralisation des courbes de Bézier."*[19].

La second classe utilise la méthode affine. *"En mathématique une fonction affine est une fonction obtenue par addition et multiplication de la variable par des constantes."*[18].

Finalement, la classe RegistrationFactory qui possède dans ses attributs un type Registrationbase qui est la classe mère des deux classes RegistrationBSpline et RegistrationAffine. Cette classe est celle qui est utilisée dans SILA3D. C'est elle qui va appeler les diverses fonctions qui sont présentent dans les classes utilisées pour le recalage.

2.2.8 Les classes finales

Cette famille de classe contient 5 différentes classes. Elles permettent de gérer SILA3D et sont généralement celles qui font appel aux autres classes de SILA3D.

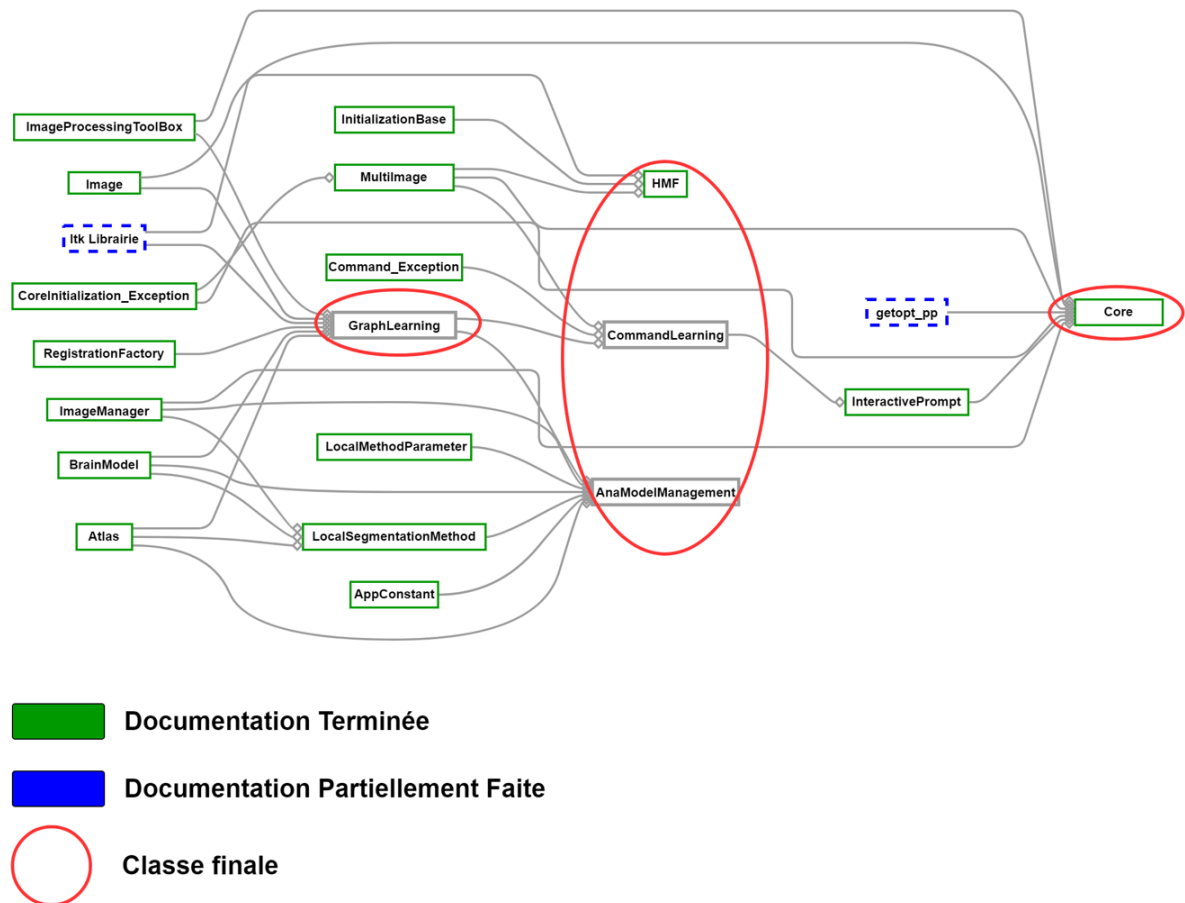


Figure 3.15 – SILA3D - UML Dépendance & Simplification - Classes Finales

La classe HMF représente un modèle de markov caché. La classe utilise des outils présents dans la librairie ITK. Elle utilise notamment la librairie "itkGaussianDistribution" qui permet de faire des calculs de distribution gaussien.

La classe GraphLearning permet de faire apprendre notre modèle. Elle m'est à jour notre modèle en utilisant les nouvelles données précédemment segmentées.

La classe CommandLearning utilise la précédente classe "GraphLearning". C'est elle qui est utilisée par SILA3D et c'est elle qui va utiliser "GraphLearning" pour faire apprendre au modèle.

La classe Core est celle qui gère SILA3D. Elle contient, entre autres, dans ses attributs les classes "SegmentationResultBase" ainsi que "InteractivePrompt". Ses attributs vont lui permettre de lire les commandes et de les exécuter.

La classe AnaModelManagement est une nouvelle classe en cours de développement. Cette classe est un début de ce qui m'est demandé de faire. En effet, elle contient des identifiants de régions fixes et son objectif est d'afficher ces régions. Cette classe n'est pas appelée et n'est pas directement utilisable dans l'interface. Cette classe retourne un fichier ".nii" qu'il faut ensuite importer dans Slicer pour voir le résultat.

4

Mise en œuvre

Cinq tâches ont été mises en œuvre avec plus ou moins de succès. La première tâche était de créer la documentation des diverses classes qui sont présentées dans l'état de l'art. La deuxième tâche était de créer la classe "ConfigurationFileManager". La troisième tâche était de pouvoir afficher les diverses régions/atlas sans faire appel à la segmentation. La quatrième tâche était de pouvoir afficher les liens entre les régions/atlas. Finalement, la cinquième tâche était de faire l'interface du logiciel.

1 Documentation des classes

L'objectif de cette tâche est de récupérer chaque classe de l'existant, les comprendre et les documenter. Pour se faire, j'ai utilisé Google Doc[9].

Cette tâche s'est principalement faite durant le premier semestre. Une très grande partie des classes a été documentée et cette documentation a été complétée avec les classes manquantes. De plus, deux documentations ont été faites.

La première documentation contient toutes les classes qui existaient avant que soit ajoutée les nouvelles durant ce **PRD**. Une second documentation contient toutes les classes, compris les nouvelles et leur impacte sur les anciennes classes.

Pour voir quelle classe a été documenté ou pas, un diagramme de classe a été tracé et contient toutes les classes et les relie entre-elles. Les classes ont été coloriées en vert pour indiquer qu'elles avaient été documentées et, en bleu pour indiquer que la classe est en cours de documentation. Finalement, toutes les classes grises signifiaient qu'elles n'avaient pas été documentées.

2 La classe "ConfigurationFileManager"

Une classe nommée "ConfigurationFileManager" a été créée. Pour rappel, cette classe a pour but de lire un fichier et de récupérer les différents paramètres qui sont présents dans le fichier. Cette classe a été réalisée en C++ en utilisant Visual Studio 2017.

Comme indiqué précédemment, la classe a pour but de lire des paramètres qui sont indiqués dans un fichier JSON. Dans un premier temps, il faut donc de savoir quels sont les paramètres ainsi que les classes dans lesquels se trouvent les paramètres.

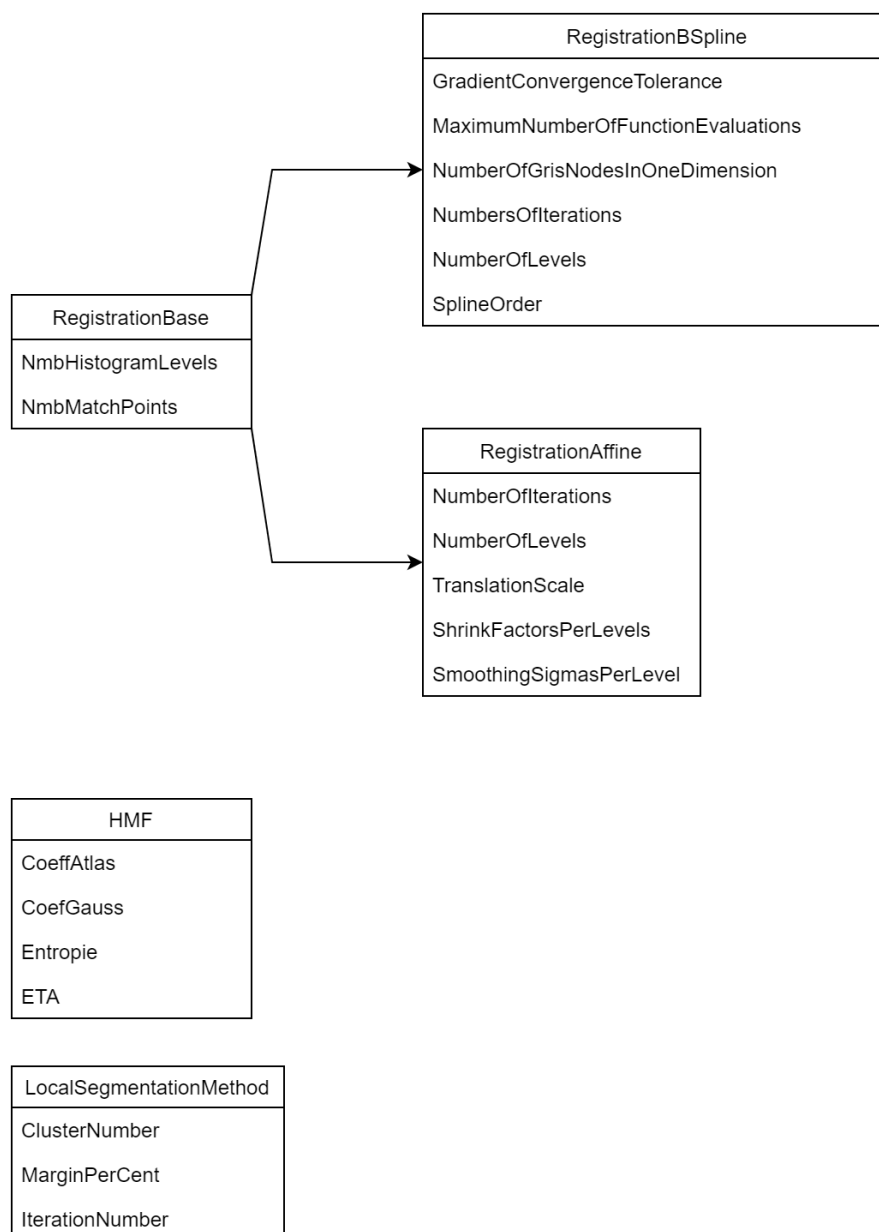


Figure 4.1 – *ConfigurationFileManager* - Paramètres A Extraire

Dans un deuxième temps, il faut voir comment va communiquer notre nouvelle classe avec les précédentes. Elle est pensée pour être comme une librairie qu'on importe simplement dans les

diverses classes qui en ont besoin. On peut représenter ce lien avec un diagramme UML :

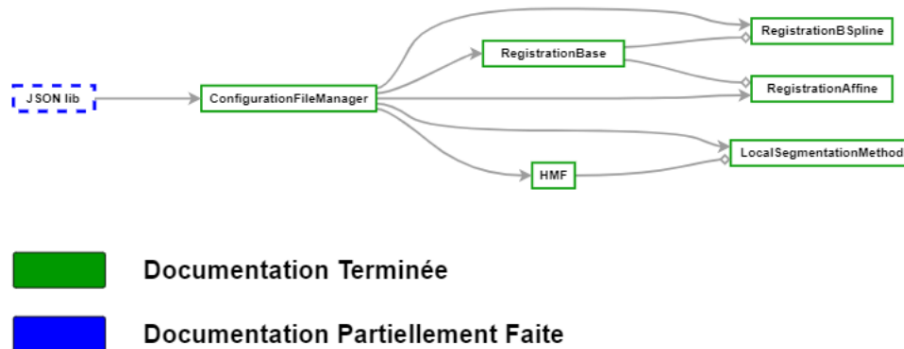


Figure 4.2 – *ConfigurationFileManager* - Diagramme UML

2.1 Outils et librairie utilisés

Une librairie en particulière, qui a été déjà présente dans l'état de l'art, a été utilisée. Cette librairie est [nlohmann/json.hpp](https://nlohmann.github.io/json.hpp). Cette librairie est utilisée pour lire de manière simple les fichiers JSON.

Une seconde librairie, qui est de base incluse dans C++, a été utilisée. Elle se nomme [fstream](https://en.cppreference.com/w/cpp/string/basic/basicfstream). Cette classe permet d'ouvrir des fichiers et de les manipuler. On s'en sert pour ouvrir notre fichier et le donner à la librairie précédemment citée pour lire les fichiers JSON.

2.2 Éléments d'implémentation, choix techniques

Voici les fichiers de la classe "ConfigurationFileManager". **Pour des raisons d'affichage dans le document, les commentaires dans les fichiers ont été retirés.**

Le fichier "ConfigurationFileManager.hpp" :

```

1 #include "nlohmann/json.hpp"
2 #include <fstream>
3 using namespace std;
4
5 class ConfigurationFileManager
6 {
7 private:
8     string pathToFile;
9     nlohmann::json parameters;
10
11 public:
12     ConfigurationFileManager();
13     ConfigurationFileManager(string pathToFile);
14     void setPathToFile(string &pathToFile);
15     inline string getPathToFile(void) { return pathToFile; }
16     inline nlohmann::json getParameters(void) { return parameters; };
17
18 private:

```

```

19         inline void setParameters(istream &file) { file >> parameters; };
20     };

```

Le fichier "ConfigurationFileManager.cpp" :

```

1  #include "ConfigurationFileManager.hpp"
2
3  ConfigurationFileManager::ConfigurationFileManager()
4  {
5      setPathToFile("config.json");
6  }
7
8  ConfigurationFileManager::ConfigurationFileManager(string pathToFile)
9  {
10     setPathToFile(pathToFile);
11 }
12
13 void ConfigurationFileManager::setPathToFile(string &pathToFile)
14 {
15     ifstream file(pathToFile);
16     if (file.good())
17     {
18         this->pathToFile = pathToFile;
19         setParameters(file);
20     }
21     else
22         this->pathToFile = "";
23 }

```

Les différentes classes indiquées précédemment à la figure 4.1 ont été modifiées pour accueillir cette nouvelle classe. De plus, la classe a été conçue pour pouvoir changer simplement et facilement la position du fichier JSON. En effet, les différentes classes, qui ont besoin de la classe "ConfigurationFileManager", utilisent toutes le constructeur par défaut. Ainsi, si on souhaite changer de place le fichier JSON, il suffit de modifier la valeur inscrite dans le constructeur qui par défaut est "config.json".

Voici les différentes fonctions qui ont été modifiées :

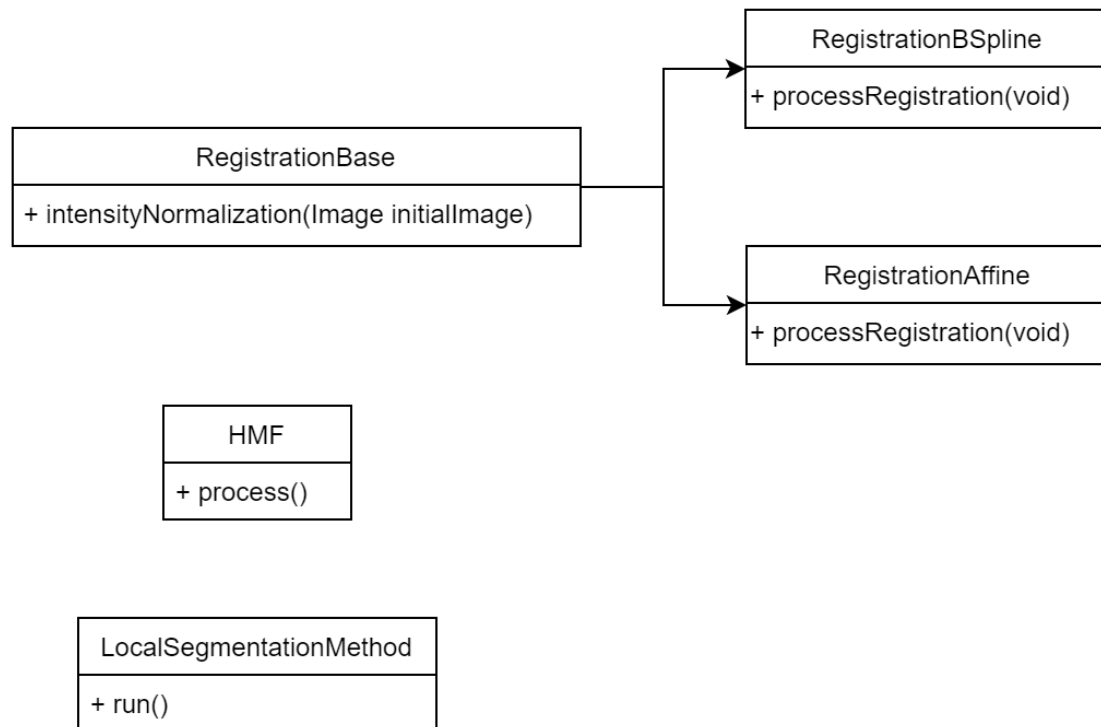


Figure 4.3 – *ConfigurationFileManager* - Les méthodes modifiées dans les différentes classes

2.3 Analyse des résultats, évaluation, qualité

Pour vérifier si l'implémentation de la classe a fonctionné, il a simplement fallu lancer l'application et voir constater des résultats similaires au précédent. En effet, les anciens coefficients n'ont pas changé. Ils ont simplement été déplacé et inscrit dans un fichier JSON et son désormais modifiable.

Après avoir lancé l'application avec la nouvelle classe, l'application fonctionnait normalement et les résultats étaient similaires aux précédents. Ce qui indique que la classe a bien été implémentée et fonctionne.

3 Afficher les régions/atlas

Cette tâche a précédemment essayé d'être réalisée. Il y a donc une classe nommée "AnaModelManagement" qui existe et contient déjà des essais pour afficher les différentes régions. Finalement, en s'appuyant sur ce qui avait déjà été fait et en le modifiant, il est désormais possible d'afficher les régions/atlas.

Un seul problème reste. En effet, le but de cette classe est de pouvoir voir les différentes régions/atlas sans avoir à faire la segmentation des images. Malheureusement, il faut forcément placer une première région pour pouvoir placer les autres. Et c'est cette sélection de bases qui pose problème. Il aurait été préférable de n'avoir à rien placer. Mais ce n'est actuellement pas possible.

3.1 Outils et librairie utilisés

Des librairies internes à SILA3D ont été utilisées :

- BraiModel
- ImageProcessingToolBox
- Atlas
- LocalSegmentationMethod
- LocalMethodParameter
- ConfigurationFileManager

3.2 Éléments d'implémentation, choix techniques

Voici les fichiers de la classe "AnaModelManagement. **Pour des raisons d'affichage dans le document, les commentaires dans les fichiers ont été retirés.**

Le fichier "AnaModelManagement.h" :

```

1  #include "BrainModel.h"
2  #include "ImageProcessingToolBox.h"
3  #include "Atlas.h"
4  #include "LocalSegmentationMethod.h"
5  #include "LocalMethodParameter.h"
6  #include "ConfigurationFileManager.hpp"
7
8  class AnaModelManagement
9  {
10 private:
11     BrainModel* model;
12     LocalMethodParameter* params;
13     SegmentationResultBase* result;
14
15 public:
16     AnaModelManagement(LocalMethodParameter* params,
17         SegmentationResultBase* result);
18     void visualizeModel();
19     ~AnaModelManagement(void);
20 };

```

Le fichier "AnaModelManagement.cpp" est un fichier assez grand et conséquent. De ce fait, il ne sera pas mis ici.

3.3 Analyse des résultats, évaluation, qualité

Pour vérifier le correct fonctionnement de la classe, des tests ont été mis en place. Pour être plus précis, précisément, deux tests. Ces tests font quasiment la même chose. Ils récupèrent le modèle, l'image et la sortie. On définit la région que l'on souhaite afficher, pour la première région, on la positionne. Puis on exécute la fonction. On répète ce procédé sur plusieurs régions. Finalement, on regarde si les régions se sont bien affichées.

```

setModelPath D:/Ecole/PRD/SILA/SILA3D_SERVER-master/public/Data/Models/Humain_proba/Brain_Model
setInputT1Image D:/Ecole/PRD/SILA/SILA3D_SERVER-master/public/Data/Images/1003_3.nii
setOutputPath C:/Users/Administrateur/Desktop

setCurrentRegion 60
setBoundingBox 112.0 85.0 127.0 30.0 37.0 37.0

visualizeModel

setModelPath D:/Ecole/PRD/SILA/SILA3D_SERVER-master/public/Data/Models/Humain_proba/Brain_Model
setInputT1Image D:/Ecole/PRD/SILA/SILA3D_SERVER-master/public/Data/Images/1003_3.nii
setOutputPath C:/Users/Administrateur/Desktop

setCurrentRegion 59

visualizeModel

setModelPath D:/Ecole/PRD/SILA/SILA3D_SERVER-master/public/Data/Models/Humain_proba/Brain_Model
setInputT1Image D:/Ecole/PRD/SILA/SILA3D_SERVER-master/public/Data/Images/1003_3.nii
setOutputPath C:/Users/Administrateur/Desktop

setCurrentRegion 153

visualizeModel

setModelPath D:/Ecole/PRD/SILA/SILA3D_SERVER-master/public/Data/Models/Humain_proba/Brain_Model
setInputT1Image D:/Ecole/PRD/SILA/SILA3D_SERVER-master/public/Data/Images/1003_3.nii
setOutputPath C:/Users/Administrateur/Desktop

setCurrentRegion 74

visualizeModel

setModelPath D:/Ecole/PRD/SILA/SILA3D_SERVER-master/public/Data/Models/Humain_proba/Brain_Model
setInputT1Image D:/Ecole/PRD/SILA/SILA3D_SERVER-master/public/Data/Images/1003_3.nii
setOutputPath C:/Users/Administrateur/Desktop

setCurrentRegion 101

visualizeModel

```

Figure 4.4 – *Afficher les régions/atlas - Test 1*

Concernant le deuxième test, comme on peut le voir, il fait exactement la même chose. À la différence près qu'il n'affiche pas les régions dans le même ordre. C'est en faisant ce test qu'on s'est aperçu que cela changé beaucoup de choses.

Après avoir exécuté les deux tests, on regarde les images 3D que les tests ont donné et on remarque quelque chose d'inattendu.

```

setModelPath D:/Ecole/PRD/SILA/SILA3D_SERVER-master/public/Data/Models/Humain_proba/Brain_Model
setInputT1Image D:/Ecole/PRD/SILA/SILA3D_SERVER-master/public/Data/Images/1003_3.nii
setOutputPath C:/Users/Administrateur/Desktop

setCurrentRegion 60
setBoundingBox 112.0 85.0 127.0 30.0 37.0 37.0

visualizeModel

setModelPath D:/Ecole/PRD/SILA/SILA3D_SERVER-master/public/Data/Models/Humain_proba/Brain_Model
setInputT1Image D:/Ecole/PRD/SILA/SILA3D_SERVER-master/public/Data/Images/1003_3.nii
setOutputPath C:/Users/Administrateur/Desktop

setCurrentRegion 153

visualizeModel

setModelPath D:/Ecole/PRD/SILA/SILA3D_SERVER-master/public/Data/Models/Humain_proba/Brain_Model
setInputT1Image D:/Ecole/PRD/SILA/SILA3D_SERVER-master/public/Data/Images/1003_3.nii
setOutputPath C:/Users/Administrateur/Desktop

setCurrentRegion 74

visualizeModel

setModelPath D:/Ecole/PRD/SILA/SILA3D_SERVER-master/public/Data/Models/Humain_proba/Brain_Model
setInputT1Image D:/Ecole/PRD/SILA/SILA3D_SERVER-master/public/Data/Images/1003_3.nii
setOutputPath C:/Users/Administrateur/Desktop

setCurrentRegion 101

visualizeModel

setModelPath D:/Ecole/PRD/SILA/SILA3D_SERVER-master/public/Data/Models/Humain_proba/Brain_Model
setInputT1Image D:/Ecole/PRD/SILA/SILA3D_SERVER-master/public/Data/Images/1003_3.nii
setOutputPath C:/Users/Administrateur/Desktop

setCurrentRegion 59

visualizeModel

```

Figure 4.5 – Afficher les régions/atlas - Test 2

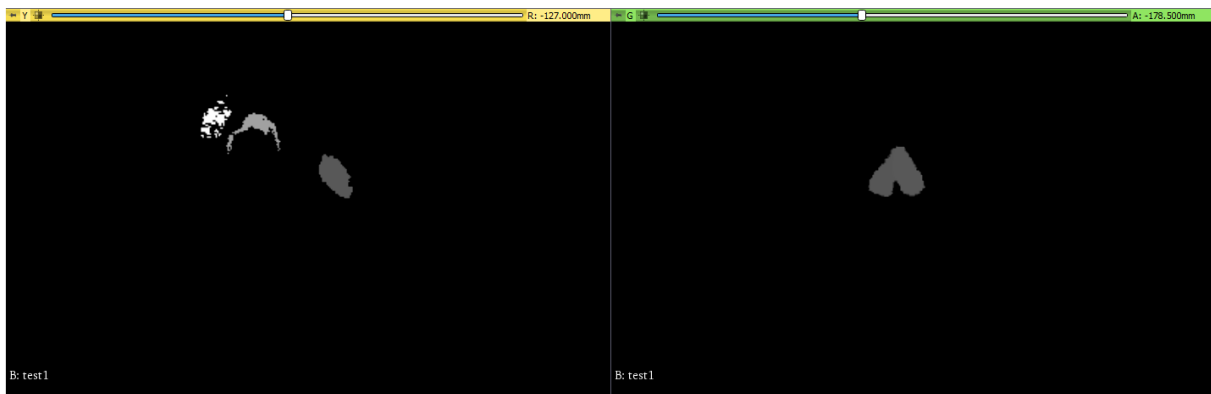


Figure 4.6 – Afficher les régions/atlas - Résultat Test 1

On remarque que les images sont différentes. Ce qui n'est pas censé être le cas. En effet, l'ordre dans lequel les régions sont affichées ne devrait pas changer leurs positions dans l'image. Et pourtant, comme on peut le voir à la suite des tests, c'est le cas.

Ce test nous a permis de voir qu'il y a un problème dans ce qui est présent dans le code. Après analyse, l'erreur vient non pas de l'affichage des régions, mais du positionnement du cadre de délimitation. Ce cadre permet de positionner de façon automatique, sauf pour la première région qui est placée à la main, les régions. Et son positionnement automatique semble ne pas fonctionner ou du moins pas dans ce cas.

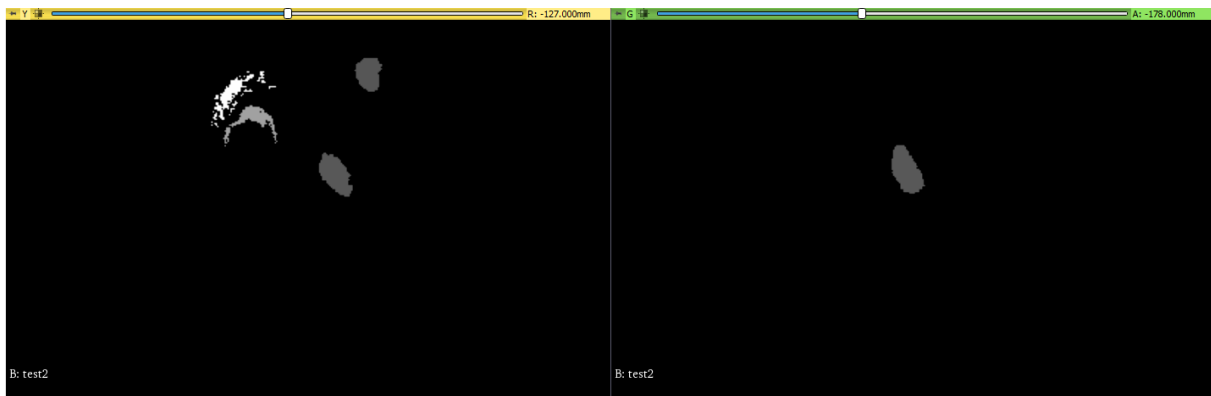


Figure 4.7 – Afficher les régions/atlas - Résultat Test 2

4 Afficher les liens entre régions

Cette fonctionnalité n'a pas eu le temps d'être pensée ou réfléchi. Ainsi, rien n'était prévu sur la manière d'implémenter cette fonctionnalité. De plus, aucun essai n'a été fait par de précédentes personnes. En conséquence, cette fonctionnalité n'a pas eu le temps d'être implémenté.

Ainsi, la fonctionnalité a été écartée par manque de temps. L'interface a été priorisée par rapport à cette fonctionnalité. Malheureusement, comme on le verra après, peu de chose ont pu être faites.

5 L'interface

Une fois les fonctionnalités mises en place, il fallait faire l'interface. Deux problèmes se sont présentés. Le premier est que rien n'a été prévu durant la phase de spécification. Mais une maquette a été faite en prévention de ce problème. Le deuxième problème est la manière dont l'interface va être mise en œuvre. Il fut découvert que le langage sera du python pour coder l'interface. De plus, les technologies m'étaient inconnues.

5.1 La Maquette

Voici la maquette prévue :

Login	Visualisation	Segmentation	
<div> <div>Image Selection</div> <div> <div>Image :</div> <div></div> <div>Modèle :</div> <div></div> <div>Load</div> <div>Cancel</div> </div> </div>			
<div> <div>Visualisation Configuration</div> <div> <div>Atlas Region :</div> <div></div> <div>Setup ROI</div> <div>Visualize All</div> <div>Run Visualisation</div> </div> </div>			
<div> <div>Results Data Management</div> <div> <div>Visualized <u>Regions</u> :</div> <div></div> <div>Remove</div> <div>Update</div> </div> </div>			
<div> <div>Upload Custom Segmentation</div> <div> <div>Upload & Save</div> </div> </div>			

Figure 4.8 – Interface - Maquette

Voici le résultat obtenu :

The image shows a web interface with four main sections, each with a dropdown arrow on the left:

- Image Selection:** Contains the text "Select the source image you want to process, and which anatomical model to be used." Below this are two dropdown menus: "Image:" and "Model:" (with "NEW" selected). To the right are "Load" and "Cancel" buttons.
- Visualisation configuration:** Contains an "Atlas region:" dropdown menu. To its right is a "Setup ROI" button. Below these are "Visualize All" and "Run visualisation" buttons.
- Results Data Management:** Contains a "Segmented regions:" dropdown menu. To its right is a "Remove" button. Below these is an "Update Segmentation (volume):" label and an "Update" button.
- Upload custom segmentation:** Contains the text "Click on this button to upload current segmentation to the server and save it in a session. Convert manual modifications to a session and save it on the server." Below this text is an "Upload & save" button.

Figure 4.9 – *Interface - Rendu Final*

Pour comprendre comment faire l'interface, il a fallu regarder et comprendre ce qui avait déjà été fait dans l'interface. N'ayant pas d'expérience dans la création d'interface et encore moins avec les librairies utilisées, le procédé a été fastidieux et long.

Après avoir finalement compris comment faire, les boutons ont commencé à apparaître ainsi que le texte et les onglets. Finalement, il ne manquait plus qu'à faire les fonctions qui activent les différents boutons ainsi que l'onglet. Malheureusement, c'est aussi la partie la plus compliquée.

Les différentes classes qui forment l'interface sont emboîtées les unes dans les autres et ne connaissant pas les méthodes à appeler, ça a rendu la tâche très compliqué. L'interface est donc incomplète et pas finie. L'aspect visuel est donc fait, mais les boutons, champs et volets déroulant sont présents, mais ils n'appellent pas les fonctions qu'ils doivent appeler. De plus, lorsqu'on s'identifie dans le logiciel, l'onglet doit pouvoir s'activer et devenir interactif. Mais après beaucoup d'effort, certains boutons s'activent tant dis que d'autre restent inactif. Comme on peut le voir sur l'image qui suit :

The interface is titled 'Rendu Final Connecté' and features a navigation bar with the following tabs: Login, Visualization, Data initialization, Segmentation, Model management, and Results. The 'Visualization' tab is currently selected.

Image Selection
 Select the source image you want to process, and which anatomical model to be used.
 Image: Model:
 [Load] [Cancel]

Visualisation configuration
 Atlas region:
 [Setup ROI]
 [Visualize All] [Run visualisation]

Results Data Management
 Segmented regions:
 [Remove]
 Update Segmentation (volume):
 [Update]

Upload custom segmentation
 Click on this button to upload current segmentation to the server and save it in a session.
 Convert manual modifications to a session and save it on the server.
 [Upload & save]

Figure 4.10 – *Interface - Rendu Final Connecté*

Une fois l'interface terminée, il faut la relier au serveur de SILA. C'est un domaine où j'ai très peu de connaissance. Le langage JAVASCRIPT étant nouveau et par manque de temps, c'est quelque chose qui n'a pas pu se faire.

5

Bilan et conclusion

1 Bilan du semestre 9

Voici les tâches effectuées et restant à effectuer :

- L'analyse de l'existant et l'état de l'art sont terminés.
- Le planning prévu pour le semestre 10 est terminé.
- Le **diagramme de GANTT** actuel a été mis à jour.
- Le cahier des spécifications n'a pas été entièrement terminé.
- Le diagramme **UML** de l'existant est terminé.
- L'externalisation des données a été commencé. Les paramètres ont été trouvé mais rien de fonctionnel n'a été fait.
- La représentation des données n'a pas été commencé.
- La documentation demandée est elle en très grande partie faite, il ne reste que 6 classes à document qui sont : "AnaModelManagement", "CommandLearning", "GraphLearning", "InteractivCommand", "RegistrationAffine" et "RegistrationBSpline".

2 Bilan du semestre 10

Ce semestre a été, pour moi, assez mal accompli. Il était convenu de faire plus que ce qui a finalement été produit. Malgré le bon départ de ce semestre, par une avancé très rapide des points qui ont eu le temps d'être vu durant le premier semestre, le reste a été beaucoup plus long que prévu.

Durant le second semestre, il fallait terminer la documentation et mettre en œuvre les différentes fonctionnalités qui m'ont été demandé. Cependant, comme on peut le voir dans le **diagramme de GANTT** et dans le résultat de ce semestre beaucoup de chose n'ont pas pu être faite.

Voici les tâches effectuées et restant à effectuer :

- Le **diagramme de GANTT** actuel a été mis à jour.
- Le diagramme UML a été mis à jour pour correspondre aux changements.
- La documentation demandée est terminée et a été mise à jour avec les changements qui ont eu lieu durant l'externalisation des données.

- L'externalisation des données a conduit à la création d'une classe qui est terminée.
- La représentation des régions a été commencée et est terminée.
- La représentation des liens entre les régions n'a pas été commencée et n'est donc pas terminée.
- L'interface utilisateur a été commencée mais n'est pas terminée.
- La liaison entre l'interface et le serveur en utilisant NodeJs n'a pas été faite.

3 Bilan sur la qualité

Le code a été commenté et documenté. Chaque méthode et attributs possèdent une description qui permet de comprendre comment la classe fonctionne. La convention de nommage a suivi ce qui avait été précédemment fait pour éviter la confusion et gardé l'ensemble du code sous la même forme.

4 Bilan auto-critique

Le début du deuxième semestre était très prometteur. Durant le premier semestre, une étude a été faite sur comment j'allais mettre en place ma classe et il y avait une connaissance déjà acquise des technologies qu'il fallait utiliser. Comme on peut le constater sur le GANTT, j'avais prévu plusieurs jours et semaines pour faire les premières tâches qui au final ne m'auront pris qu'une seule et unique journée.

Lorsque cette partie fut terminée, il fallait passer sur l'interface. De par le manque de formation et de connaissance sur les technologies, il avait été prévu que cela prenne beaucoup de temps. Mais fur et à mesure que les jours avançaient, l'interface se construisait, mais lentement. Le manque de connaissance sur la technologie et de documentation sur le code déjà présent pour l'interface fût long à comprendre et à reprendre.

Malgré ces problèmes, l'organisation était bien présente. De multiples réunions et entrevues avec les encadrants de projet et encadrants extérieurs ont eu lieu comme prévu dans le GANTT. Les délais ont toujours été respectés. Tout n'a pas pu être fait, mais ce qui a été fait a été commenté et rédigé avec soin.

Annexes

A

Planification, gestion de projet

1 Evolution du projet

Le diagramme de **diagramme de GANTT** Initial pour la planification de ce projet.

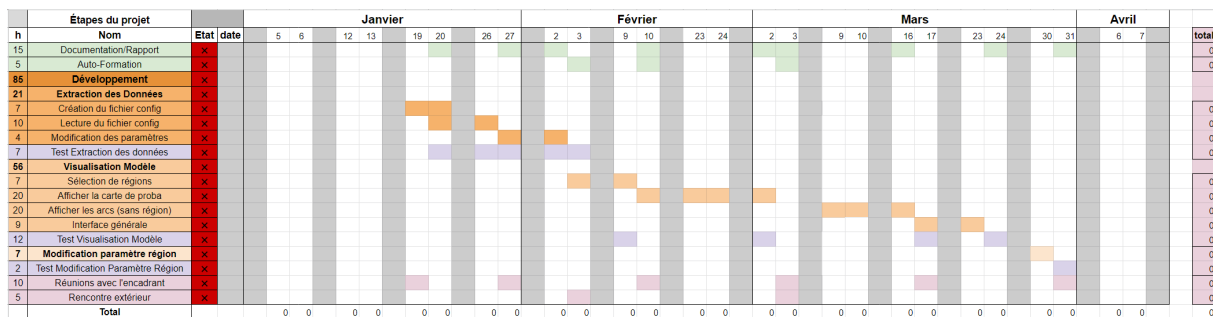


Figure A.1 – Prévission diagramme de GANTT

Le diagramme de **diagramme de GANTT** Final de ce projet se compose de deux images. La première représente le diagramme de **diagramme de GANTT** de tout ce qu'il s'est passé avec le nombre d'heures passé sur chaque étape. La deuxième image correspond à la partie développement, mais avec beaucoup plus de détail.

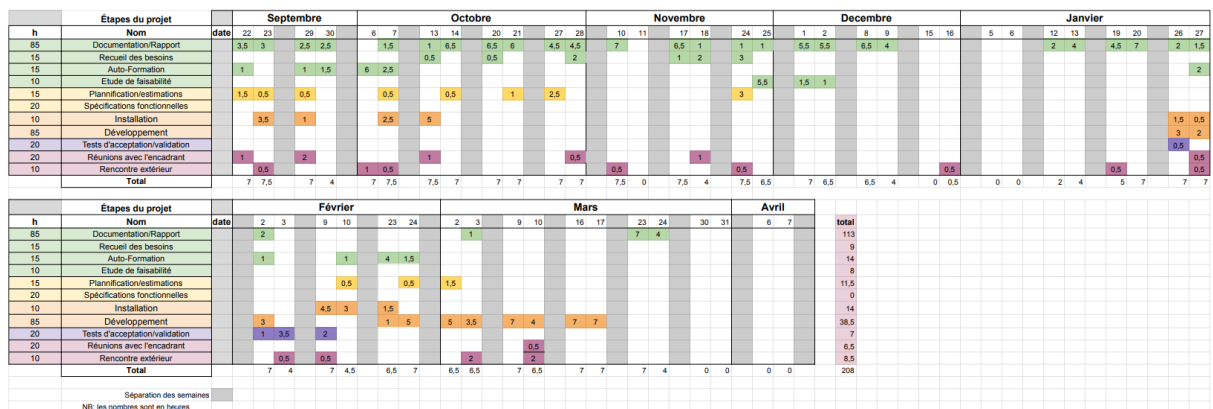


Figure A.2 – Diagramme de GANTT - Global

Dans cette seconde image, les cases colorées représentent les prévisions, ce qui était prévu. On peut voir qu'au final, rien ne s'est passé comme prévu !

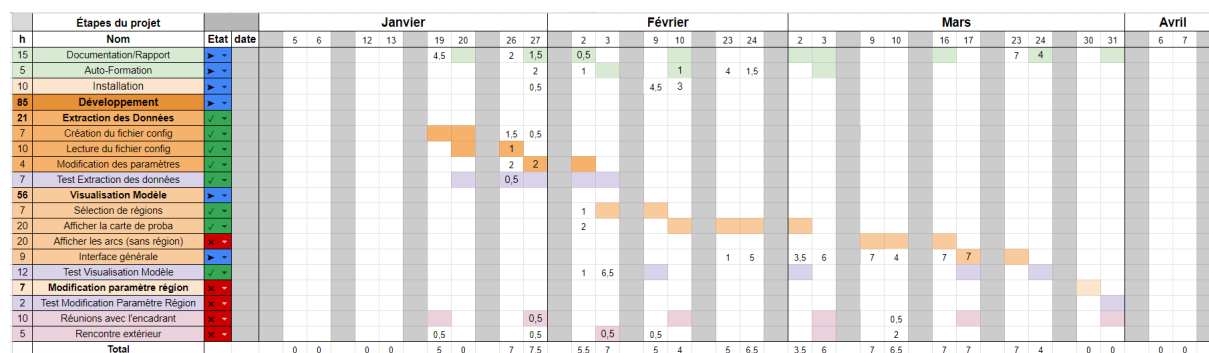


Figure A.3 – Diagramme de GANTT - Développement

2 Description des tâches

Tâche 1 : Extraction des données

- Date de début : 19 Janvier 2022
- Date de fin : 02 Février 2022
- Durée : 6 jours répartis sur 3 semaines
- Description : Création du fichier config, Lecture du fichier config, Modification des paramètres, Test Extraction des données
- Livrable : Un document qui explique comment utiliser la fonctionnalité. Il répertorie aussi comment les tests vont être effectués

Tâche 2 : Visualisation Modèle

- Date de début : 03 Février 2022
- Date de fin : 23 Mars 2022
- Durée : 13 jours sur 6 semaines
- Description : Sélection de régions, Afficher la carte de proba, Afficher les arcs (sans région), Interface générale, Test Visualisation Modèle
- Livrable : Un document qui explique comment utiliser la fonctionnalité. Il répertorie aussi comment les tests vont être effectués

Tâche 3 : Modification paramètre région

- Date de début : 30 Mars 2022
- Date de fin : 31 Mars 2022
- Durée : 2 jours sur 1 semaines
- Description : Modification paramètre région, Test Modification Paramètre Région
- Livrable : Un document qui explique comment utiliser la fonctionnalité. Il répertorie aussi comment les tests vont être effectués

B

Cahier de Spécifications

1 spécifications Fonctionnelles

1.1 Externalisation des données - La classe ConfigurationFileManager

Pour cette fonctionnalité, on crée une nouvelle classe qui s'insérera dans les classe déjà existante. Cette classe a pour objectif de lire le fichier de configuration et d'en extraire les divers paramètres.

ConfigurationFileManager
- pathToFile: std::string - parametersMap : std::map
+ ConfigurationFileManager (pathToFile : std::string) + void setPathToFile(pathToFile : std::string) + std::string getPathToFile(void) + std::map getParametersMap(void) + float getParameter(parameterName : std::string) + std::map readParameter(void)

Figure B.1 – Diagramme de classe - ConfigurationFileManager

Description des attributs :

La classe se compose de deux attributs.

- `string pathToFile` ; Cet attribut représente une chaîne de caractère qui contient le chemin pour accéder au fichier de configuration.
- `map parametersMap` ; Cet attribut représente un dictionnaire. En élément clef on a le nom du paramètre. En valeur, on a la valeur du paramètre en float.

Description des méthodes :**ConfigurationFileManager(string pathToFile)**

Entrée : `string pathToFile` ; Une chaîne de caractère qui désigne le chemin vers le fichier de configuration.

Sortie : `void`

Postconditions : Si le fichier est introuvable, retourne une erreur.

void setPathToFile(pathToFile : string)

Entrée : `string pathToFile` ; Une chaîne de caractère qui désigne le chemin vers le fichier de configuration.

Sortie : `void`

Postconditions : Si le fichier est introuvable, retourne une erreur.

string getPathToFile(void)

Entrée : `void`

Sortie : `string` ; Retourne la chaîne de caractère contenu dans l'attribut `"pathToFile"`.

map getParametersMap(void)

Entrée : `void`

Sortie : `map` ; Retourne le dictionnaire contenu dans l'attribut `"parametersMap"`.

float getParameter(parameterName : string)

Entrée : `string parameterName` ; Une chaîne de caractère qui désigne le nom du paramètre dont on souhaite récupérer la valeur.

Sortie : `float` ; Retourne la valeur associée au paramètre.

Postconditions : Si le paramètre n'existe pas, retourne une erreur.

map readParameter(void)

Entrée : `void`

Sortie : `map` ; Retourne le dictionnaire contenu dans l'attribut `"parametersMap"`.

Description : Cette méthode est appelée dès que l'on crée la classe ou quand on change le chemin vers le fichier. C'est cette méthode qui va initialiser l'attribut `parametersMap`.

Préconditions : Le chemin vers le fichier a déjà été vérifié et est stocké dans l'attribut `"pathToFile"`.

1.2 Visualisation du Modèle - Les classes VisualiseModel

2 Spécifications non fonctionnelles

2.1 Contraintes de développement et conception

Le langage de programmation est C++. SILA3D a été construit dans ce langage et les ajouts qui seront fait seront aussi dans ce langage. Visual Studio 2017 sera utilisé.

2.2 Contraintes de fonctionnement et d'exploitation

2.2.1 Performances

Aucune contrainte sur la performance n'a été abordé. Mis à part le fait que ça doit pouvoir se lancer et s'exécuter.

2.2.2 Capacités

Aucune contrainte sur la capacité n'a été abordé.

2.2.3 Contrôlabilité

Pour vérifier le bon déroulement, des affichages sur la console seront fait.

- [1] Gaëtan GALISOT. « Segmentation incrémentale et interactive d'images médicales 3D ». In : (2018). URL : http://rfai.li.univ-tours.fr/PublicData/PhD/Manuscrit_GaetanGalisot.pdf.
- [2] Gaëtan GALISOT. « Segmentation incrémentale et interactive d'images médicales 3D ». In : (2018). URL : http://rfai.li.univ-tours.fr/PublicData/PhD/Manuscrit_GaetanGalisot.pdf#page=37.
- [3] « imaios.com - Site sur l'anatomie, imagerie médicale et e-learning pour les professionnels de santé, Cerveau, atlas d'anatomie humaine en IRM ». In : (). URL : <https://www.imaios.com/fr/e-Anatomy/Cerveau/Cerveau-IRM-3D>.
- [4] « ITK : Registration ». In : (). URL : <https://itk.org/ITKSoftwareGuide/html/Book2/ITKSoftwareGuide-Book2ch3.html>.
- [5] McCormick M; Liu X; Jomier J; Marion C; Ibanez L. « ITK : enabling reproducible research and open science ». In : *Front Neuroinform* (Published 2014 Feb 20). URL : [doi:10.3389/fninf.2014.00013](https://doi.org/10.3389/fninf.2014.00013).
- [6] « La région Centre-Val de Loire & Les Chercheurs inventent un nouvel avenir ». In : (2014). URL : <https://www.centre-valdeloire.fr/sites/default/files/media/document/2020-06/APR-2014-bdef.pdf>.
- [7] « La région Centre-Val de Loire & Les Chercheurs inventent un nouvel avenir, Projet Neurogé ». In : (2014). URL : <https://www.centre-valdeloire.fr/sites/default/files/media/document/2020-06/APR-2014-bdef.pdf#page=34>.
- [8] « Site internet : gitmind ». In : (). URL : <https://gitmind.com/>.
- [9] « Site internet : Google Doc ». In : (). URL : <https://www.google.fr/intl/fr/docs/about/>.
- [10] « Site internet : <https://code.google.com/archive/p/getoptpp/> ». In : (). URL : <https://code.google.com/archive/p/getoptpp/>.
- [11] « Site internet : <https://lemon.cs.elte.hu/trac/lemon> ». In : (). URL : <https://lemon.cs.elte.hu/trac/lemon>.
- [12] « Site internet : [itk.org/about](https://itk.org/about/#overview) ». In : (). URL : <https://itk.org/about/#overview>.
- [13] « Site internet : qt.io ». In : (). URL : <https://www.qt.io/>.

- [14] « Site internet : SILA3D ». In : (). URL : http://www.rfai.li.univ-tours.fr/PublicData/3D_Brain_Seg/home.html.
- [15] « Site internet : Slicer.org ». In : (). URL : <https://www.slicer.org/>.
- [16] « Site internet : vtk.org ». In : (). URL : <https://vtk.org/>.
- [17] « Site internet : vtk.org/about ». In : (). URL : <https://vtk.org/about/#overview>.
- [18] « Wikipédia : Affine ». In : (). URL : https://fr.wikipedia.org/wiki/Fonction_affine.
- [19] « Wikipédia : B-spline ». In : (). URL : <https://fr.wikipedia.org/wiki/B-spline>.

D

Glossaire

carte de probabilité C'est une image où sur chaque pixel de l'image, on indique la probabilité que le pixel possède d'appartenir à la région. On peut par exemple dire qu'au plus c'est clair, au plus c'est probable que le pixel appartienne à la région dont il est question.. [2](#)

cervelet Partie de l'encéphale située sous le cerveau et en arrière du tronc cérébral, intervenant dans le tonus musculaire, le maintien de l'équilibre, les mouvements automatiques et la coordination des mouvements volontaires.. [2](#)

diagramme de GANTT Le diagramme de [diagramme de GANTT](#) est un outil permettant de visualiser dans le temps les diverses tâches composant un projet. Il s'agit d'une représentation qui permet de représenter graphiquement l'avancement du projet.. [6](#), [39](#), [42](#), [49](#)

extension Un ensemble de fonctionnalité ajouté aux fonctionnalités déjà présentes. Le logiciel de base peut fonctionner sans une extension mais pas l'inverse.. [1](#)

externaliser des paramètres Le fait d'externaliser signifie "de sortir". Dans une fonction, on utilise parfois des paramètres qui permettent de régler certaines choses dans une fonction. Ces paramètres sont généralement fixés ce qui empêche de les changer. Externaliser des paramètres signifie retrouver les paramètres utilisés et les sortir de la fonction pour pouvoir les modifier.. [3](#), [10](#)

hypothalamus Partie du cerveau, située sous le thalamus, qui joue un rôle capital dans la régulation des fonctions vitales (sommeil, activité sexuelle...).. [2](#)

méthode de KANBAN La méthode de KANBAN consiste à grouper ses tâches en 4 parties. Le but est de faire naviguer toutes les tâches de la première partie à la dernière partie. La première partie se nomme "tâche" et contient toutes les tâches. La deuxième partie se nomme "À Faire" et contient les tâches qui sont à faire. La troisième partie "En Cours" contient les tâches qui sont entrain d'être faites. Finalement, la dernière partie est "Terminé" et contient toutes les tâches qui ont été entièrement terminées.. [6](#)

open source Libre de droits, dont les codes sources sont disponibles et modifiables librement. [1](#)

recalage Le recalage ou "Registration" en anglais est une technique qui consiste à comparer deux images entrées. Pour ce faire, le but est de prendre une première image et de la superposer à la seconde. Lorsqu'on va superposer l'image, on va la tourner, la redimensionner, la déplacer pour faire en sorte qu'elle soit parfaitement superposable à la seconde image.. [4](#)

segmentation Le fait de segmenter une image. Segmenter une image revient à découper une image. Le but de la segmentation est de "découper" des formes ou des zones que l'on souhaite identifier. Comme par exemple retrouver un chien dans une image.. [1](#)

E

Acronymes

IRM Imagerie par résonance magnétique. [1](#)

ITK Insight Segmentation and Registration Toolkit. [10](#), [12](#), [13](#), [17](#)

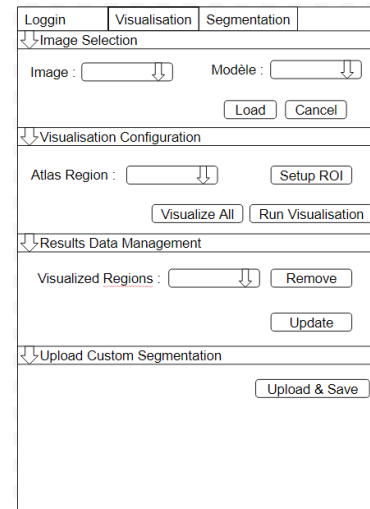
PRD Projet de Recherche & Développement. [1](#), [27](#), [54](#)

UML Unified Modeling Language. [5](#), [18](#), [29](#), [39](#)

VTk Visualization ToolKit. [10](#), [12](#), [13](#), [17](#), [18](#)

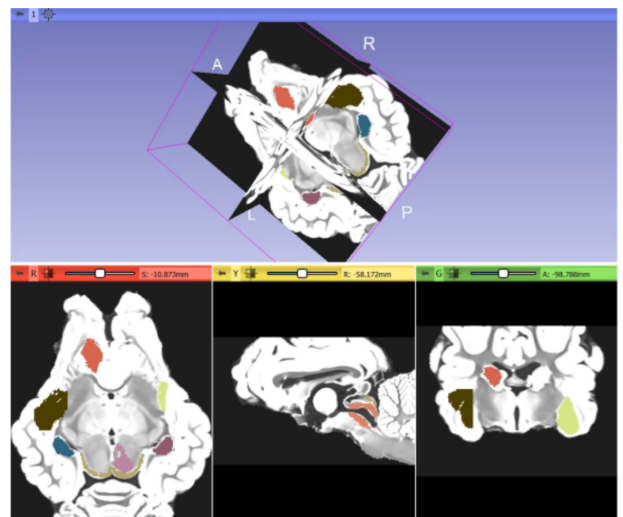
Objectifs

- Représenter un graphe/modèle dans Slicer
- Externalisation des paramètres utilisés pour la segmentation
- Faire une documentation des classes présentes dans SILA3D



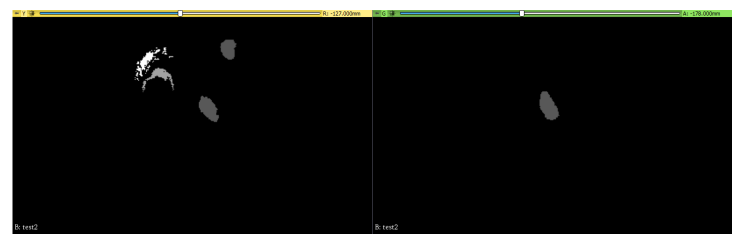
Mise en œuvre

1. Documentation des classes
2. Externalisation des données
3. Visualisation des régions du cerveau
4. Interface utilisateur



Résultats attendus

La documentation doit être claire et organisée. L'externalisation des paramètres doit permettre la modification de ces derniers. La modification doit être simple et fonctionnelle. Lors de la représentation du graphe, on doit pouvoir afficher les nœuds du graphe sans avoir à faire la segmentation. On doit pouvoir afficher quelques arcs ou quelques régions ou l'entièreté du graphe.



Analyse, visualisation 3D et comparaison de cerveaux et connectomes :

Maxime LANDRIN

Encadrement : Jean-Yves RAMEL

Objectifs

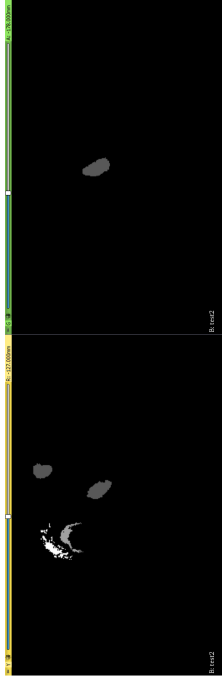
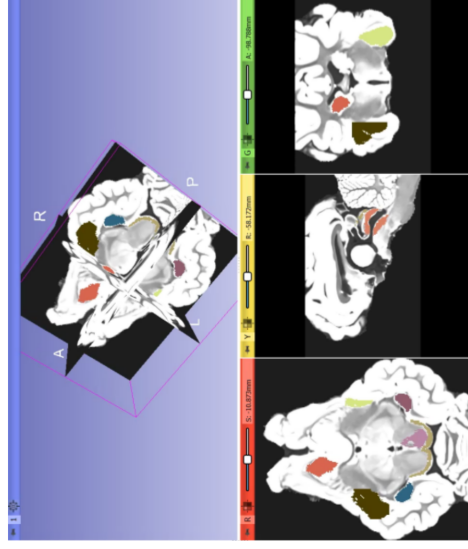
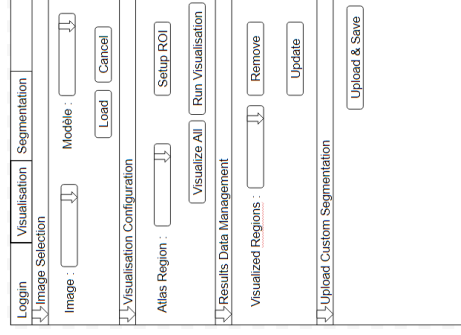
- Représenter un graphe/modèle dans Slicer
- Externalisation des paramètres utilisés pour la segmentation
- Faire une documentation des classes présentes dans SILA3D

Mise en œuvre

1. Documentation des classes
2. Externalisation des données
3. Visualisation des régions du cerveau
4. Interface utilisateur

Résultats attendus

La documentation doit être claire et organisée. L'externalisation des paramètres doit permettre la modification de ces derniers. La modification doit être simple et fonctionnelle. Lors de la représentation du graphe, on doit pouvoir afficher les nœuds du graphe sans avoir à faire la segmentation. On doit pouvoir afficher quelques arcs ou quelques régions ou l'entièreté du graphe.



Analyse, visualisation 3D et comparaison de cerveaux et connectomes

Résumé

Durant mon projet de recherche et développement, j'ai été amené à utiliser un logiciel nommé Slicer ainsi que son extension SILA3D. Slicer est logiciel open source. Il a pour but de simplifier la visualisation des images d'IRM.

J'ai appris beaucoup de choses grâce à ces logiciels et j'ai dû créer une nouvelle fonctionnalité à SILA3D. Ces dernières m'ont permis d'apprendre l'utilisation d'outils de visualisation ainsi que l'utilisation de graphe.

Mon travail a été découpé en deux parties. Durant le premier semestre, mon travail consistait à faire de la recherche et d'étudier l'existant. C'est aussi dans cette période que j'ai écrit mon cahier des spécifications et que j'ai prévu mon programme pour la phase de développement.

Au second semestre, mon travail consistait à développer les diverses fonctionnalités qui m'ont été demandées. Il a fallu pour cela utiliser ce que j'avais appris durant mon premier semestre.

Pour terminer, j'ai trouvé ce PRD très enrichissant, quoi qu'un peu long sur le premier semestre. Malgré cela, j'ai quand même découvert beaucoup de technologie et j'ai aussi pu apprendre à les implémenter dans un logiciel.

Mots-clés

Atlas, Graphe, NeuroGéo, Recalage, Segmentation, SILA3D, Slicer

Abstract

During my research and development project, I have been led to use software named Slicer with his extension SILA3D. Slicer is open-source software. His purpose is to simplify the visualization of MRI images.

I learned a lot thanks to that two software. I also had to create new functionalities in SILA3D. Those allowed me to learn how to use visualization tools as well as graph usage.

My work was split into two parts. During the first semester, my work consisted of doing research and studying the existing. It is also during this period that I have written my specification book and that I prepared my planning for the development phase.

During the second semester, my work consisted of developing multiple functionalities that have been asked of me. For that, I had to use what I have learned during the first semester.

Finally, I found this RDP very rewarding yet, a bit longer during the first semester. But, I still have discovered a lot of technologies and I also had the chance to learn how to implement them in the software.

Keywords

Atlas, Graph, NeuroGéo, Registration, Segmentation, SILA3D, Slicer

Tuteur académique

Jean-Yves RAMEL

Étudiant

Maxime LANDRIN (DI5)