ÉCOLE POLYTECHNIQUE DE L'UNIVERSITE FRANÇOIS RABELAIS DE TOURS
Spécialité Informatique
64 av. Jean Portalis
37200 TOURS, FRANCE
Tél +33 (0)2 47 36 14 31
www polytech univ-tours fr

CAHIER DE SPECIFICATIONS

Projet : RFAI14	Développement de techniques de segmentation automatique d'images IRM 3D de cerveaux animaux.	
Emetteur :	Léo Boulanger	Coordonnées : leo.boulanger-2@etu.univ-tours.fr
Encadrants :	Jean-Yves Ramel	Coordonnées : jean-yves.ramel@univ-tours.fr
	Antoine Bourlier	Coordonnées : antoine.bourlier@univ-tours.fr
Date d'émission :	Mardi 12 décembre 2023	

Validation

Nom	Date	Valide (O/N)	Commentaires
Antoine Bourlier	12/12/2023	N	

Historique des modifications

Version	Date	Description de la modification
00	12/12/2023	Version initiale
01	20/12/2023	Corrections & ajout des parties concernant les tests et les livrables

TABLE OF CONTENTS

CAHIER DE SPECIFICATIONS **3**

1. INTRODUCTION 3
2. CONTEXTE DE LA REALISATION 3
   - *2.1. Contexte* *3*
   - *2.2. Objectifs* *3*
3. DESCRIPTION GENERALE 4
   - *3.1. Environnement du projet* *4*
   - *3.2. Caractéristiques des utilisateurs* *4*
   - *3.3. Fonctionnalités du système* *4*
   - *3.4. Structure générale du système* *5*
   - *3.5. Définition des tests* *5*
4. DESCRIPTION DES INTERFACES EXTERNES DU LOGICIEL 5
   - *4.1. Interfaces matériel/logiciel* *5*
   - *4.2. Interfaces homme/machine* *5*
   - *4.3. Interfaces logiciel/logiciel* *5*
5. SPECIFICATIONS FONCTIONNELLES 6
   - *5.1. Fonctions de traitement de données* *6*
     - 5.1.1. Définition de la fonction 1.1 : import_mri_data 6
     - 5.1.2. Définition de la fonction 1.2 : load_settings 6
     - 5.1.3. Définition de la fonction 1.3 : setup 7
   - *5.2. Fonctions de clustering* *8*
     - 5.2.1. Définition de la fonction 2.1 : start_process 8
     - 5.2.2. Définition de la fonction 2.2 : local_membership 10
     - 5.2.3. Définition de la fonction 2.3 : global_membership 11
     - 5.2.4. Définition de la fonction 2.4 : combined_membership 12
     - 5.2.5. Définition de la fonction 2.5 : objective_function 13
     - 5.2.6. Définition de la fonction 2.6 : compute_new_cluster 14
     - 5.2.7. Définition de la fonction 2.7 : histogram_peak_analysis 15
     - 5.2.8. Définition de la fonction 2.8 : subsegment 15
   - *5.3. Fonctions utilitaires* *16*
     - 5.3.1. Définition de la fonction 3.1 : save_progress 16
     - 5.3.2. Définition de la fonction 3.2 : load_progress 16
     - 5.3.3. Définition de la fonction 3.3 : append_log_file 17
     - 5.3.4. Définition de la fonction 3.4 : update_log_key 17
     - 5.3.5. Définition de la fonction 3.5 : remove_log_key 17
6. SPECIFICATIONS NON FONCTIONNELLES 18
   - *6.1. Contraintes de développement et conception* *18*
   - *6.2. Contraintes de fonctionnement et d'exploitation* *18*
     - 6.2.1. Performances 18
     - 6.2.2. Contrôlabilité 18
     - 6.2.3. Sécurité 18
     - 6.2.4. Intégrité 18
7. LIVRABLES 19

GLOSSAIRE **20**

BIBLIOGRAPHIE **21**

CAHIER DE SPECIFICATIONS

1. Introduction

Ce document définit les spécifications concernant le projet de fin d'étude « Développement de techniques de segmentation automatique d'images IRM 3D de cerveaux animaux ». Ce projet s'intègre dans une collaboration entre le Laboratoire d'informatique fondamentale et appliquée de Tours (LIFAT) et l'Institut national de recherche pour l'agriculture, l'alimentation et l'environnement (INRAE).
Ce cahier des spécifications est rédigé par Léo Boulanger, étudiant à Polytech Tours, représentant le MOE. Le projet sera encadré par Antoine Bourlier et Jean-Yves Ramel, représentant le MOA.
Ce document comporte des *mots en italique*, correspondant à la première référence d'un mot difficile et peu commun, qui sont décris dans le glossaire en fin de document.

2. Contexte de la réalisation

2.1. Contexte

L'avancement des technologies d'imagerie médicale, en particulier l'*Imagerie par Résonance Magnétique (IRM)*, nous permet de comprendre de façon approfondie les structures cérébrales des animaux. La *segmentation* manuelle des structures cérébrales sur des images d'IRM d'*encéphales* animaux représente actuellement une tâche laborieuse et chronophage pour les chercheurs. La combinaison de la complexité morphologique des encéphales avec la variabilité inter-espèce et interindividuelle rend ce processus particulièrement exigent, et peut également introduire des biais, causés par les experts effectuant les analyses. Afin de palier ces défis, les chercheurs aspirent à mettre au point des méthodes de segmentation automatiques, permettant, non seulement de réduire significativement le temps de l'analyse des données, mais aussi d'améliorer la cohérence et la précision des résultats obtenus.

2.2. Objectifs

L'objectif de ce projet est de développer un programme informatique permettant à un utilisateur de segmenter automatiquement une image 3D d'IRM. Ce programme sera basé principalement sur la méthode « Unsupervised modified spatial fuzzy c-means » publiée par Kamarujjaman, et Mausumi Maitra[1].
Cette méthode permet de *segmenter* des données en un certain nombre de *clusters*, en calculant une valeur d'adhésion globale et locale pour chaque *voxel* à chaque cluster, c'est-à-dire calculer à quel point chaque voxel est proche de chaque cluster, en utilisant uniquement la valeur du voxel pour l'adhésion globale, et en utilisant celles des voxels voisins, en plus, pour l'adhésion locale. Ces valeurs d'adhésion sont ensuite combinées pour obtenir une segmentation et optimiser la *fonction objectif*. Si la fonction objectif est optimale, la segmentation calculée précédemment est optimale et la méthode est terminée, sinon, on calcule de nouveaux clusters puis on recommence la méthode avec ces nouveaux clusters.
Le nombre de clusters à obtenir est soit donné par l'utilisateur, soit défini automatiquement avec une méthode d'analyse d'histogramme incluse dans le programme.
Le programme devrait être utilisable sur tous les systèmes d'exploitation pouvant exécuter des scripts Python, mais la validation des fonctionnalités de ce projet sera faite uniquement sur les systèmes Windows et Linux. Il sera donc préférable d'utiliser Windows ou Linux afin de réduire les risques de disfonctionnement.

3. Description générale

3.1. Environnement du projet

Ce projet est lié à la thèse d'Antoine Bourlier[2], dont l'objectif est de proposer de nouveaux algorithmes d'analyse et de comparaison *anatomo-fonctionnel*. Certains de ces algorithmes utilisent la *théorie des graphes* pour modéliser les structures de l'encéphale, où les nœuds correspondent aux structures, et les arêtes aux connexions anatomiques ou fonctionnelles. Cependant, il est nécessaire d'avoir une segmentation pour modéliser les structures en nœuds.

3.2. Caractéristiques des utilisateurs

Les principaux utilisateurs sont les chercheurs spécialisés en *neurosciences*, mais l'utilisation du programme reste accessible pour les personnes travaillant dans le domaine de l'informatique et de la neuroscience.
Le programme est donc développé pour des utilisateurs qui :

- ont peu de connaissances en informatique
- n'ont pas d'expérience avec ce genre d'outils
- se serviront occasionnellement de cet outil
- ne nécessitent pas de privilèges spécifiques dans leur système

3.3. Fonctionnalités du système

Le programme peut être chargé de 3 façons :

- L'utilisateur exécute le programme en incluant des arguments, puis chaque configuration manquante lui est demandé via l'invite de commandes.
  Exemple : « python segmentation_tool.py path\to\mri_data.nii.gz -n 20 » lancera la segmentation sur l'image d'IRM située à « path\to\mri_data.nii.gz », pour 20 structures.
- L'utilisateur peut aussi inclure un fichier de configuration en argument, qui appliquera les configurations y étant définies. Les paramètres passés en argument écraseront ceux importés depuis le fichier de configuration.
- L'utilisateur peut aussi lancer la reprise de l'exécution en passant uniquement un fichier de sauvegarde au programme.

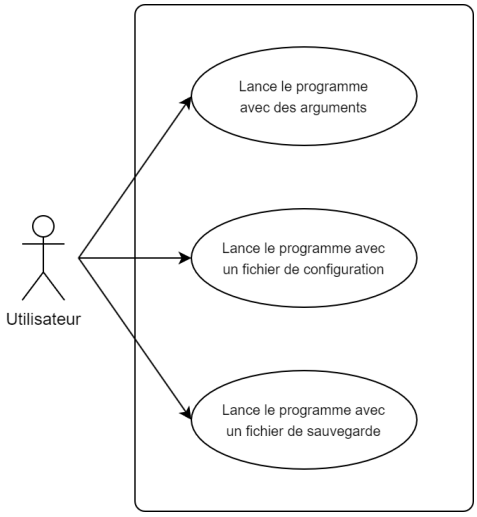


Figure 1 - Diagramme de cas d'utilisation général du programme

3.4. Structure générale du système

Le code du projet sera séparé en 3 fichiers Python :

- configuration.py
- 3D_umsfcm.py
- logger.py

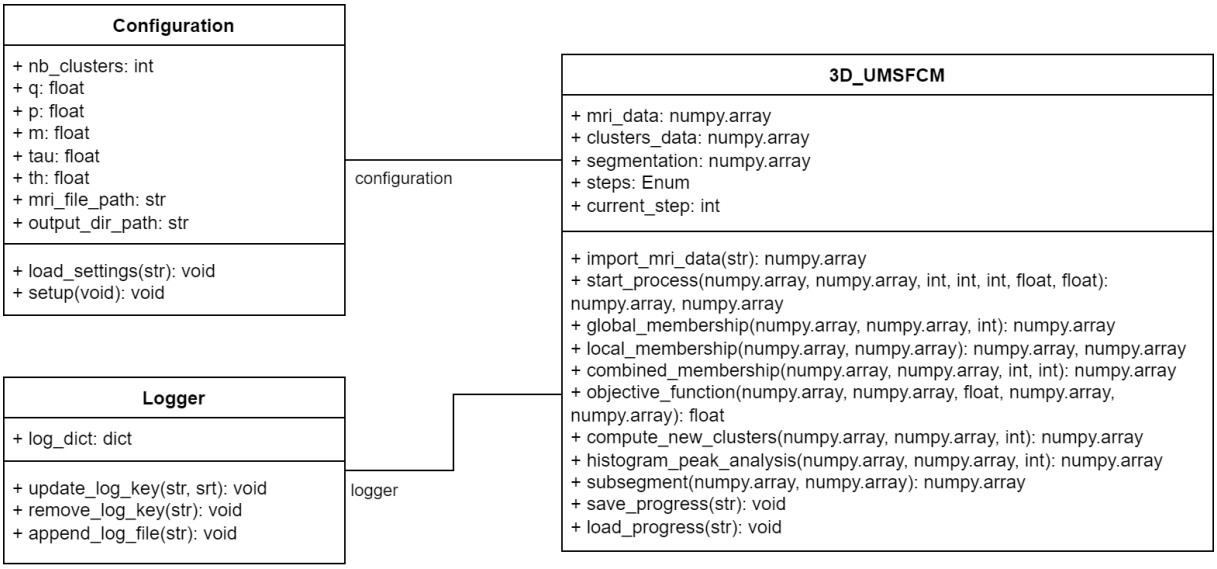


Figure 2 - Diagramme de classes

Les fonctions de ce diagramme seront détaillées dans la partie 5 : « Spécifications fonctionnelles ».

3.5. Définition des tests

Les tests seront définis en utilisant la librairie « unittest[3] ». Chaque fichier du programme principal aura un fichier de test associé, composé d'une classe où un test unitaire sera associé à chaque fonction. Un fichier « validation.py » vérifiera l'état de l'ensemble des tests, en affichant sur la console le pourcentage des fonctions validées, ainsi que le nom des fonctions ne l'étant pas.
La validation des fonctions pourra être effectuée en utilisant le programme principal avec le paramètre « --validation » ou « -v ». Par exemple : « python segmentation_tool.py --validation »

4. Description des interfaces externes du logiciel

4.1. Interfaces matériel/logiciel

Il n'y a pas d'interface matériel/logiciel spécifiquement défini pour ce projet.

4.2. Interfaces homme/machine

Le programme sera utilisé via console, aucune interface homme/machine ne sera fournie.

4.3. Interfaces logiciel/logiciel

Il n'y a pas d'interface logiciel/logiciel défini pour ce projet.

5. Spécifications fonctionnelles

Les fonctions ont 3 niveaux de priorité :

- **Primordiales** : le cœur du programme. Ces fonctions doivent impérativement être développées.
- **Secondaires** : fonctionnalités complémentaires, pour enrichir l'expérience utilisateur et augmenter la flexibilité du programme. Définies après le développement de l'ensemble des fonctions primordiales.
- **Optionnelles** : fonctionnalités souhaitables, mais non essentielles à la fonction de base du programme. Elles seront développées uniquement en fin de projet, après les fonctions secondaires.

5.1. Fonctions de traitement de données

5.1.1. Définition de la fonction 1.1 : import_mri_data

Identification :

- Nom : import_mri_data
- Cette fonction importe les données d'un fichier NIfTI pour les rendre exploitables par le reste du programme.
- Priorité : primordiale

Description :

- Entrée :
  - file_path (str) : Chemin vers le fichier NifTI à importer.
- Sortie :
  - mri_data (numpy.array) : Données exploitables de l'image d'IRM importée
- Gestion des erreurs :
  - Le fichier est introuvable : Lever une erreur indiquant le chemin du fichier pour que l'utilisateur puisse vérifier et corriger l'emplacement du fichier.
  - Une erreur lors de la lecture du fichier : Lever une erreur indiquant à l'utilisateur que le fichier est illisible ou corrompu.

5.1.2. Définition de la fonction 1.2 : load_settings

Identification :

- Nom : load_settings
- Cette fonction permettra de charger une configuration du programme à partir d'un fichier au format JSON[4].
- Priorité : optionnelle

Description :

- Entrée :
  - file_path (str) : Chemin vers le fichier de configuration.
- Sortie :
  - Les paramètres de la configuration seront appliqués à l'objet Configuration appelant cette fonction, il n'y a donc pas de données en sortie.
- Gestion des erreurs :
  - Le fichier est introuvable : Lever une erreur indiquant le chemin du fichier pour que l'utilisateur puisse vérifier et corriger l'emplacement du fichier.

- Une erreur lors de la lecture du fichier : Lever une erreur indiquant à l'utilisateur que le fichier est illisible ou corrompu.

5.1.3. Définition de la fonction 1.3 : setup

Identification :

- Nom : setup
- Cette fonction ajoute les configurations passées en paramètres de l'application, puis demande à l'utilisateur via la console d'ajouter les paramètres manquants.
- Priorité : secondaire

Description :

- Entrée :
  - Aucune donnée n'est nécessaire en entrée
- Sortie :
  - Les paramètres de la configuration seront appliqués à l'objet Configuration appelant cette fonction, il n'y a donc pas de données en sortie.
- Gestion des erreurs :
  - L'ensemble des paramètres nécessaires ne sont pas affectés à la fin de la fonction : Lever une erreur indiquant à l'utilisateur les paramètres nécessaires indéfinis.

5.2. Fonctions de clustering

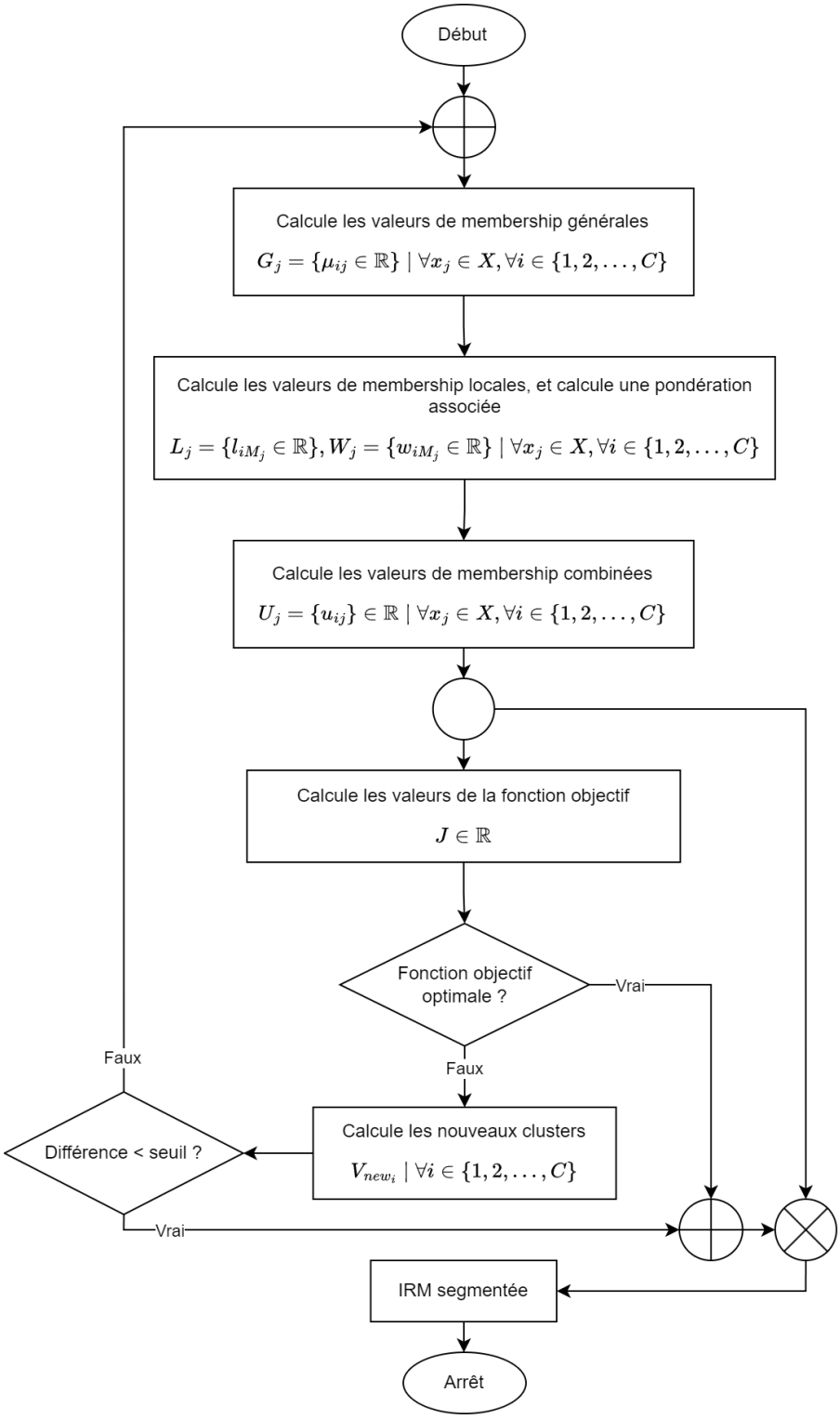
5.2.1. Définition de la fonction 2.1 : start_process

Identification :

- Nom : start_process
- Cette fonction initialise les éléments, puis utilise les fonctions nécessaires pour appliquer la méthode *3D unsupervised modified spatial fuzzy c-means* sur les données.
- Priorité : primordiale

Description :

- Entrées :
  - mri_data (numpy.array) : matrice 3D contenant les données à segmenter : $X = \left\{ x_j \in \mathbb{N} \,\middle|\, \forall j \in \{1,2,\dots,(nc * nr * ns)\} \right\}$
  - cluster_values (numpy.array) : liste des valeurs des centres de cluster : $V = \{V_i \in \mathbb{R} \mid \forall i \in \{1,2,\dots,C\}\}$
  - global_modifier (float) : paramètre de contrôle d'importance du membership global : p
  - local_modifier (float) : paramètre de contrôle d'importance du membership local : q
  - fuzzifier (float) : valeur utilisée pour définir un degré d'incertitude appliqué dans la fonction de calcul du membership local : m
  - threshold (float) : seuil à partir duquel la différence entre les valeurs des clusters, avant et après calcul des nouveaux clusters, est considéré comme trop faible, impliquant l'arrêt de la boucle d'entraînement : $seuil$
  - spatial_rate (float) : le taux d'inclusion des informations spatiales dans la fonction objectif : τ (« tau »)
- Sorties :
  - La liste des valeurs de chaque centre de cluster (numpy.array)
  - La matrice des voxels segmentés (numpy.array)
- Gestion des erreurs :
  - Une erreur est survenue lors de l'exécution : Lever une erreur indiquant à quelle étape elle a eu lieu, puis indiquer le chemin vers la dernière sauvegarde.
- Flowchart :


Début
Calcule les valeurs de membership générales
$G_j = \{\mu_{ij} \in \mathbb{R}\} \mid \forall x_j \in X, \forall i \in \{1, 2, \ldots, C\}$
Calcule les valeurs de membership locales, et calcule une pondération associée
$L_j = \{l_{iM_j} \in \mathbb{R}\}, W_j = \{w_{iM_j} \in \mathbb{R}\} \mid \forall x_j \in X, \forall i \in \{1, 2, \ldots, C\}$
Calcule les valeurs de membership combinées
$U_j = \{u_{ij}\} \in \mathbb{R} \mid \forall x_j \in X, \forall i \in \{1, 2, \ldots, C\}$
Calcule les valeurs de la fonction objectif
$J \in \mathbb{R}$
Fonction objectif optimale ?
Vrai
Faux
Calcule les nouveaux clusters
$V_{new_i} \mid \forall i \in \{1, 2, \ldots, C\}$
Différence < seuil ?
Faux
Vrai
IRM segmentée
Arrêt

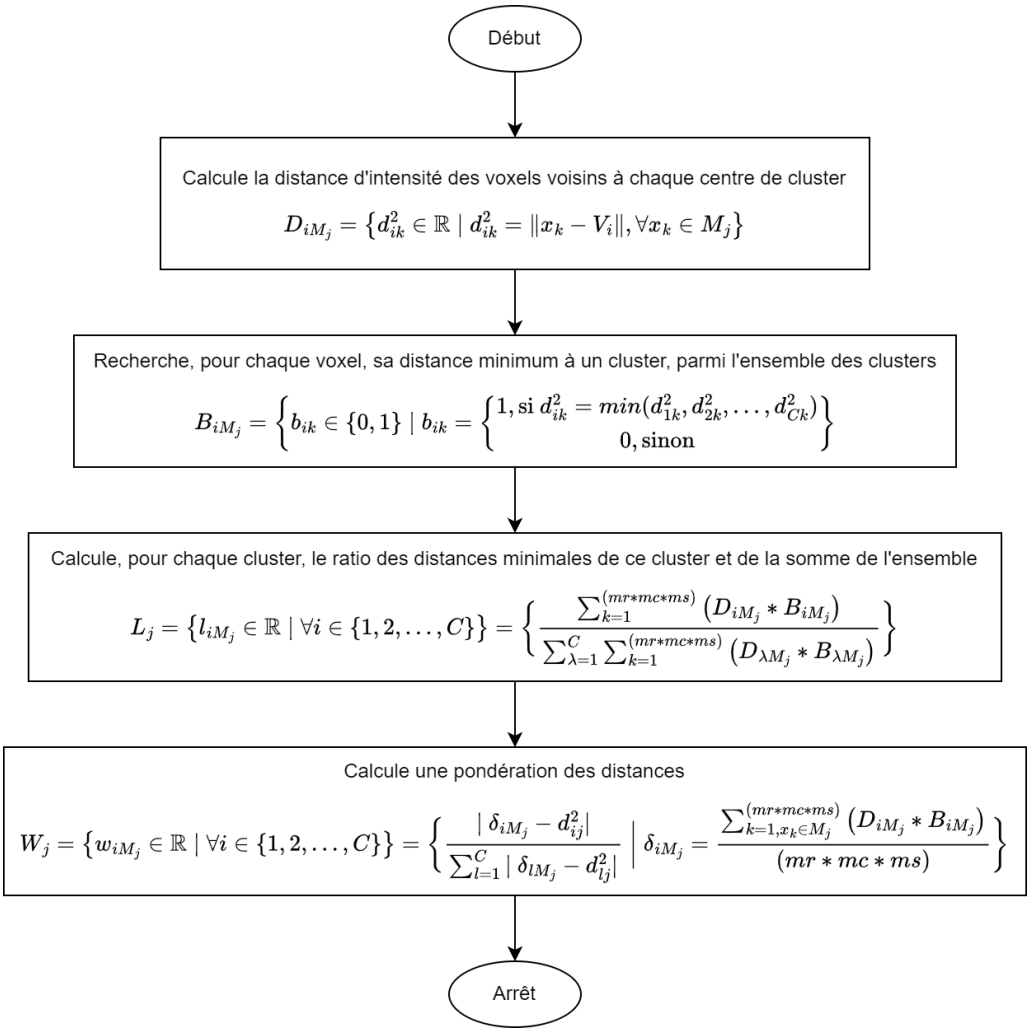
5.2.2. Définition de la fonction 2.2 : local_membership

Identification :

- Nom : local_membership
- Cette fonction calcule les valeurs de membership locales d'un voxel pour chaque centre de cluster, et calcule une pondération pour optimiser la fonction objectif.
- Priorité : primordiales

Description :

- Entrées :
  - mask_data (numpy.array) : matrice des données locales du voxel analysé
  - cluster_values (numpy.array) : valeurs des centres de clusters
- Sorties :
  - Une liste des valeurs de membership locales (numpy.array)
  - Une liste des valeurs de la pondération (numpy.array)
- Gestion des erreurs :
  - Erreur lors de l'exécution de la fonction : Lever une erreur indiquant qu'une erreur est survenue dans la fonction local_membership
- Flowchart :



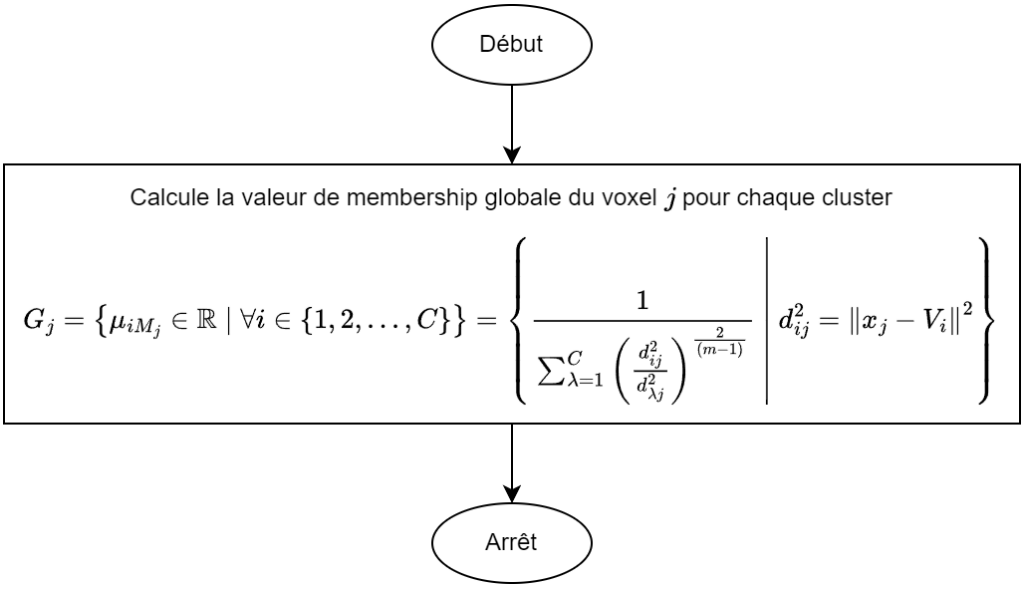
5.2.3. Définition de la fonction 2.3 : global_membership

Identification :

- Nom : global_membership
- Cette fonction calcule les valeurs de membership générale d'un voxel pour un centre de cluster.
- Priorité : primordiale

Description :

- Entrées :
  - mri_data (numpy.array) : matrice des données
  - cluster_values (numpy.array) : valeurs des centres de clusters
  - fuzzifier (float) : valeur du fuzzifier
- Sorties :
  - Une liste des valeurs de membership générales (numpy.array)
- Gestion des erreurs :
  - Erreur lors de l'exécution de la fonction : Lever une erreur indiquant qu'une erreur est survenue dans la fonction global_membership.
- Flowchart :



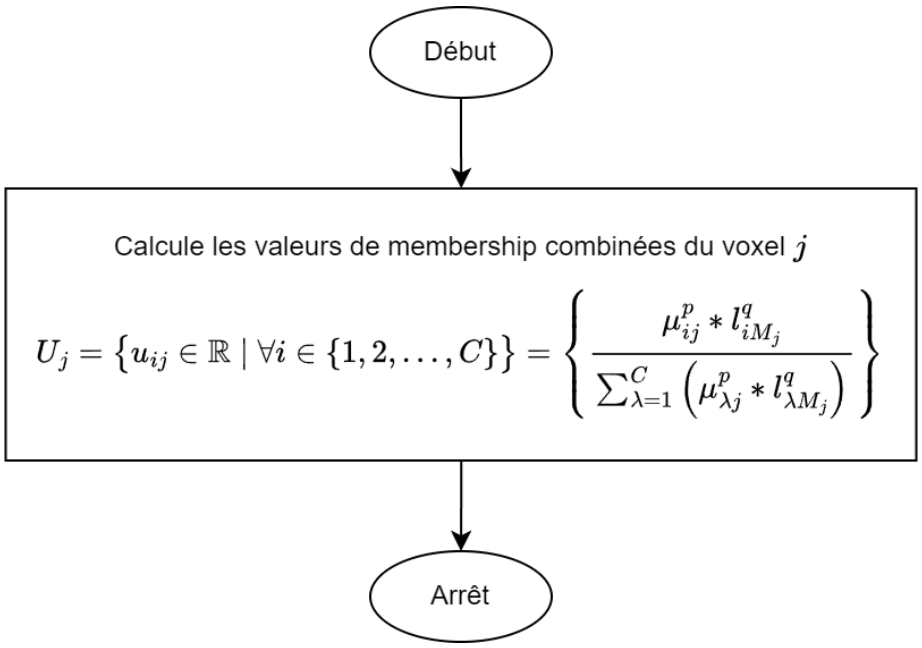
5.2.4. Définition de la fonction 2.4 : combined_membership

Identification :

- Nom : combined_membership
- Cette fonction calcule les valeurs de membership d'un voxel, obtenues en combinant les valeurs de membership locales et générales.
- Priorité : primordiale

Description :

- Entrées :
  - global_membership (numpy.array) : liste des valeurs du membership global
  - local_membership (numpy.array) : liste des valeurs du membership local
  - global_modifier (float) : paramètre de contrôle d'importance du membership global
  - local_modifier (float) : paramètre de contrôle d'importance du membership local
- Sorties :
  - Une liste des valeurs de membership combinées (numpy.array)
- Gestion des erreurs :
  - Erreur lors de l'exécution de la fonction : Lever une erreur indiquant qu'une erreur est survenue dans la fonction combined_membership.
- Flowchart :



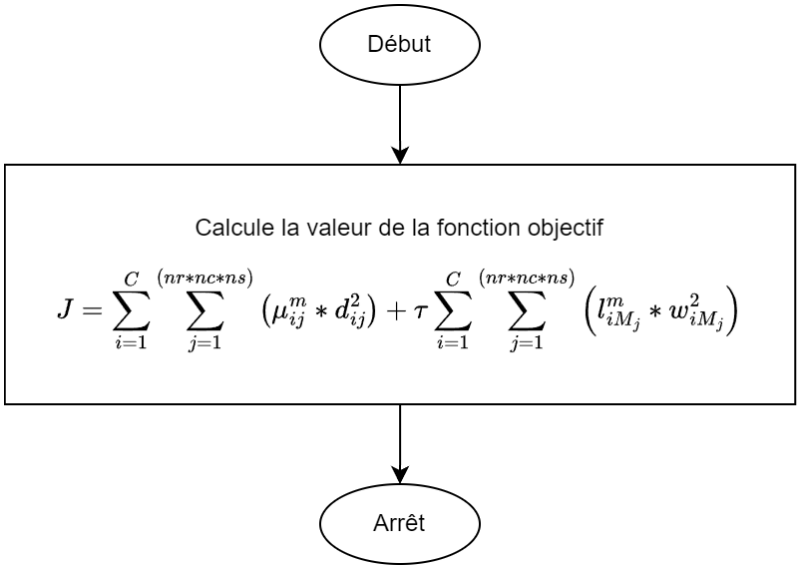
5.2.5. Définition de la fonction 2.5 : objective_function

Identification :

- Nom : objective_function
- Fonction objectif utilisée pour rechercher l'optimalité de la solution.
- Priorité : primordiale

Description :

- Entrées :
  - global_membership (numpy.array) : matrice des valeurs de membership générales
  - cluster_values (numpy.array) : liste des valeurs des clusters
  - spatial_rate (float) : le taux d'inclusion des informations spatiales dans la fonction objectif
  - local_membership (numpy.array) : matrice des valeurs de membership locales
  - weighted_data (numpy.array) : matrice des valeurs pondérées
- Sorties :
  - La valeur de la fonction objectif (float)
- Gestion des erreurs :
  - Erreur lors de l'exécution de la fonction : Lever une erreur indiquant qu'une erreur est survenue dans la fonction objective_function.
- Flowchart :



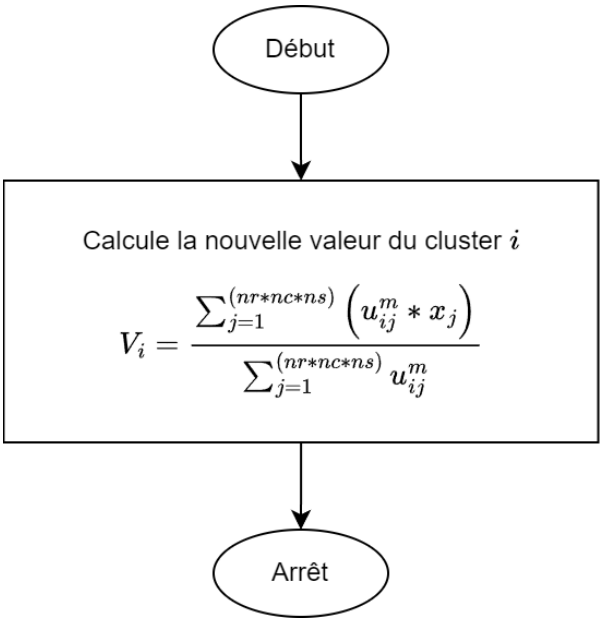
5.2.6. Définition de la fonction 2.6 : compute_new_cluster

Identification :

- Nom : compute_new_cluster
- Cette fonction calcule les valeurs des nouveaux clusters grâce à la dernière segmentation calculée.
- Priorité : primordiale

Description :

- Entrées :
  - combined_membership (numpy.array) : matrice des données de la nouvelle segmentation
  - mri_data (numpy.array) : matrice des données de l'image d'IRM
- Sorties :
  - La liste des nouvelles valeurs de clusters (numpy.array)
- Gestion des erreurs :
  - Erreur lors de l'exécution de la fonction : Lever une erreur indiquant qu'une erreur est survenue dans la fonction compute_new_cluster.
- Flowchart :



5.2.7. Définition de la fonction 2.7 : histogram_peak_analysis

Identification :

- Nom : histogram_peak_analysis
- Cette fonction utilisera une méthode d'analyse de pics d'histogramme[5] pour obtenir automatiquement un nombre prédéfini, ou non de clusters.
- Priorité : secondaire

Description :

- Entrées :
  - mri_data (numpy.array) : la matrice des données de l'image d'IRM
  - nb_clusters (int) : le nombre de clusters recherchés. Si « 0 », le nombre de clusters sera recherché automatiquement.
- Sorties :
  - Une liste contenant les valeurs des clusters trouvés (numpy.array)
- Gestion des erreurs :
  - Erreur lors de l'exécution de la fonction : Lever une erreur indiquant qu'une erreur est survenue dans la fonction histogram_peak_analysis.

5.2.8. Définition de la fonction 2.8 : subsegment

Identification :

- Nom : subsegment
- Cette fonction applique un masque binaire sur une image d'IRM, pour isoler une partie des données.
- Priorité : optionnelle

Description :

- Entrées :
  - mri_data (numpy.array) : matrice des données de l'image d'IRM
  - mask (numpy.array) : matrice binaire utilisée pour isoler une partie des données, doit être de la même taille que mri_data
- Sorties :
  - La matrice des données de l'image d'IRM associée au masque binaire fourni. (numpy.array)
- Gestion des erreurs :
  - Le masque n'a pas la même taille que l'image : Lever une erreur indiquant à l'utilisateur les tailles des 2 matrices.

5.3. Fonctions utilitaires

5.3.1. Définition de la fonction 3.1 : save_progress

Identification :

- Nom : save_progress
- Sauvegarde l'état et l'avancée de la segmentation.
- Priorité : secondaire

Description :

- Entrées :
  - save_path (str) : chemin vers le fichier où sauvegarder la progression
- Sorties :
  - Aucune sortie n'est nécessaire
- Gestion des erreurs :
  - Le fichier ne peut pas être créé ou modifié : Lever une erreur informant l'utilisateur qu'une erreur est survenue lors de l'interaction avec le fichier.

5.3.2. Définition de la fonction 3.2 : load_progress

Identification :

- Nom : load_progress
- Reprend l'exécution du programme à partir d'un fichier de sauvegarde.
- Priorité : secondaire

Description :

- Entrées :
  - load_path (str) : chemin vers le fichier de sauvegarde à lire
- Sorties :
  - Aucune sortie n'est nécessaire
- Gestion des erreurs :
  - Le fichier est introuvable : Lever une erreur informant l'utilisateur que le fichier est introuvable, et afficher le chemin vers le fichier attendu.
  - Le fichier est illisible : Lever une erreur indiquant à l'utilisateur que le fichier ne peut pas être lu.

5.3.3. Définition de la fonction 3.3 : append_log_file

Identification :

- Nom : append_log_file
- Ajoute une information dans le fichier de logs
- Priorité : secondaire

Description :

- Entrées :
  - log_info (str) : chaîne de caractères à ajouter dans le fichier de logs
- Sorties :
  - Aucune sortie n'est nécessaire
- Gestion des erreurs :
  - Le fichier de logs ne peut pas être créé ou modifié : Lever une erreur indiquant à l'utilisateur qu'il est impossible d'interagir avec le fichier de logs

5.3.4. Définition de la fonction 3.4 : update_log_key

Identification :

- Nom : update_log_key
- Ajoute ou modifie un couple clef / valeur dans le dictionnaire de logs.
- Priorité : secondaire

Description :

- Entrées :
  - key (str) : nom de la clef
  - value (str) : description de la clef
- Sorties :
  - Aucune sortie n'est nécessaire.
- Gestion des erreurs :
  - Erreur lors de l'exécution de la fonction : Lever une erreur indiquant qu'une erreur est survenue dans la fonction update_log_key.

5.3.5. Définition de la fonction 3.5 : remove_log_key

Identification :

- Nom : remove_log_key
- Supprime une clef du dictionnaire de logs
- Priorité : secondaire

Description :

- Entrées :
  - key (str) : nom de la clef à supprimer
- Sorties :
  - Aucune sortie n'est nécessaire
- Gestion des erreurs :
  - Erreur lors de l'exécution de la fonction : Lever une erreur indiquant qu'une erreur est survenue dans la fonction remove_log_key.

6. Spécifications non fonctionnelles

6.1. Contraintes de développement et conception

- Python est imposé comme langage de programmation.
- Le code, les commentaires et la documentation devront être rédigés en anglais.
- Les données des images d'IRM seront au format NIfTI (.nii / .nii.gz).
- Le programme doit être utilisable sur les systèmes d'exploitation Windows et Linux.
- Le programme doit être utilisable pour des utilisateurs ne maîtrisant potentiellement pas l'informatique. Un liste complète et simplifiée des étapes de l'installation sera donc fournie, ainsi qu'un fichier « requirements.txt », utilisé pour installer automatiquement les dépendances.

6.2. Contraintes de fonctionnement et d'exploitation

Le programme devant être exécutable sous Windows et Linux, la manipulation de fichiers devra être compatible pour les différentes syntaxes : séparateurs (« / » pour Linux, « \\ » pour Windows), et prise en compte des jokers (« * », « ? », « [] », etc…)).

6.2.1. Performances

L'exécution du programme doit être complété de préférence en moins de 10 minutes. Cependant, une durée d'exécution de quelques heures est acceptable pour l'analyse d'images d'IRM de très grandes dimensions.

6.2.2. Contrôlabilité

Un système de fichiers de log permettra de suivre l'état de l'algorithme le long de son exécution. De plus, à chaque itération de l'entraînement du clustering, l'état du programme est sauvegardé pour reprendre la progression en cas d'erreur.

6.2.3. Sécurité

Le système ne nécessitera pas d'élévation des privilèges pour fonctionner, ni de quelconque information d'identification.

6.2.4. Intégrité

Dans le cas de coupure lors de l'exécution, le système récupèrera les données sauvegardées lors de la dernière étape du processus. Les données de l'étape où l'erreur a eu lieu seront perdues.

7. Livrables

L'ensemble des livrables sera inclus dans une archive zip contenant :

- Documentation : dossier contenant la documentation du programme
- Source_code : dossier contenant le code du programme
    - 3D_umsfcm.py
    - configuration.py
    - logger.py
- Validation_tests : dossier contenant les fichiers de test
    - test_3D_umsfcm.py
    - test_configuration.py
    - test_logger.py
    - validation.py
- README.md : fichier décrivant les étapes d'installation et d'utilisation
- requirements.txt : fichier contenant la liste des dépendances à installer
- segmentation_tool.py : le fichier principal, exécuté pour lancer le programme ou valider les tests

GLOSSAIRE

Anatomo-fonctionnel : Relatif à l'anatomie et aux fonctions physiologiques.

Cluster : Groupe d'éléments, généralement homogènes. Dans le contexte de ce projet, un cluster possède une valeur et regroupe un ensemble de voxels étants plus proche (en intensité) de la valeur de ce cluster que celles des autres clusters.

Encéphale : Signifie littéralement « dans la tête » en grec ancien. En neurosciences, désigne le système nerveux central contenu dans la boîte crânienne. Le cerveau n'est qu'une grande partie de l'encéphale.

Fonction objectif : Fonction utilisée en tant que critère pour trouver une solution optimale à un problème d'optimisation. L'objectif est de minimiser ou maximiser cette fonction jusqu'à atteindre l'optimum ou l'approcher jusqu'à un certain seuil.

IRM : Imagerie par Résonnance Magnétique, examen effectué pour obtenir des images précises en deux ou trois dimensions de l'intérieur d'un corps.

Membership : Mot anglais signifiant « Adhésion ». Dans le contexte de ce projet, la valeur de membership d'un voxel à un centre de cluster correspond à quel point un voxel est proche d'un cluster.

Neuroscience : Discipline scientifique étudiant le système nerveux et la cognition.

Théorie des graphes : Discipline mathématique et informatique sur l'étude des graphes. Un graphe est un couple composé d'un ensemble de sommets, aussi appelés nœuds ou points, et d'un ensemble d'arêtes reliant chacune deux de ces sommets.

Segmentation : Séparation d'un ensemble de données en plusieurs sous-ensembles disjoints : n'ayant aucun élément en commun.

Voxel : Mot-valise contractant « pixel » et « volume », correspond à un élément d'un volume. Dans le contexte de ce projet, un voxel correspond à un point de l'image d'IRM 3D analysée.

BIBLIOGRAPHIE

[1] Kamarujjaman, Maitra, M. 3D unsupervised modified spatial fuzzy *c*-means method for segmentation of 3D brain MR image. *Pattern Anal Applic* **22**, 1561–1571 (2019). https://doi.org/10.1007/s10044-019-00806-2

[2] Antoine Bourlier, Apprentissage profond sur graphes pour l'analyse et la comparaison morphofonctionnelle d'encéphales. https://www.theses.fr/s350453

[3] unittest – Framework de tests unitaires. https://docs.python.org/fr/3/library/unittest.html

[4] Working with JSON. https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON

[5] Namburu, Anupama & Kumar, S & Edara, Sreenivasa. (2017). Generalized rough intuitionistic fuzzy c-means for MR brain image segmentation. IET Image Processing. 11. 10.1049/iet-ipr.2016.0891.