# Split and merge

Tom Daguerre/Zoé Coudray

Octobre 2023

## 1 Liens

lien tableur excel

## 2 Journal de bord

### 2.1 1ère Partie : la modélisation

Début du projet le 27 septembre 2023.

— 27 septembre, 1ère séance et 2ème séance :
  — Réunion avec notre encadrant qui nous détaille notre sujet
  — On recherche des informations sur le fonctionnement de l'algorithme Split and Merge
  — On choisit le langage Python pour notre code et on recherche une librairie à utiliser pour ouvrir les fichiers .Nifti mais nous ne pouvons pas la tester immédiatement car nous ne possédons pas d'image de ce type.
  — Discussions et argumentations afin de structurer notre projet et futur code
  — Finalité : deux projets distinct imaginés, une version utilisant uniquement un tableau et l'autre avec un arbre et un graphe.
  — TO-DO : Essayer d'extraire un tableau depuis une image .Nifti

— 03 octobre, 3ème séance :
  — Après concertation avec notre encadrant, changement de langage de programmation vers Java
  — Recherches pour apprendre à utiliser github et à implémenter la librairie SimpleITK sur notre environnement
  — Nous avons réussi à lire puis réécrire une image au format nii.gz
  — TO-DO : Nous n'avons pas encore pus discuter des deux méthodes vu précédemment, il faut également réussir à extraire les informations de l'image dans un tableau ou autre format exploitable.
— 4 octobre, 4ème séance :
  — Meilleure compréhension de simpleITK, nous avons réussi à extraire le principal des informations de l'image, sa taille et la couleur des pixels
  — Nous ne savons pas comment obtenir l'encodage de couleur (est-ce qu'il se situe entre 0 et 255 ou bien entre 0 et 1 ...)
  — Nous avons pris ces deux heures pour modéliser entièrement notre projet, nous avons créé les classes, prévus les différentes fonctions, détaillé parfois leur fonctionnement et surtout réfléchis à leurs arguments en entrée et sortis en lien avec leur fonctionnement.
— 9 octobre, 5ème séance :

  - Correction de notre modèle et plus particulièrement de la méthode Merge qui ne marchait pas correctement
  - TO-DO : Réunion avec l'encadrant pour vérifier nos dernière modifications et plus généralement notre modélisation complète
- 10 octobre, 6ème séance :
  - Relecture du modèle avec notre encadrant différents points d'améliorations évoqués
  - Nouvelle rédactions des spécifications de manière plus clair avec les différentes modifications apportées au modèle afin de l'améliorer
  - Création d'un diagramme de classe UML

## 2.2 2ème Partie : la programmation

Début de programmation

- 18 octobre, 7ème séance :
  - Création de toutes nos classes
  - Création d'un main avec la lecture puis écriture d'une image
  - Début de la classe Split
  - TO-DO : finaliser la méthode IsNeighbourWith() de la classe cube / Avancer sur le split et tenter un affichage en couleur pour voir son efficacité
- 19 octobre, 8ème séance :
  - Codage de la méthode IsNeighbourWith de la classe Cube
  - Codage des fonctions Split et Merge +leurs methodes associées
  - TO-DO : tests et fonction main fonctionnelle
- 26 octobre, 9ème et 10ème séance :
  - Fonction Main fonctionnelle avec des mesures de temps on voit très vite que le plus long est la création du graphe de voisinage. Exemple sur un test : 1 seconde pour le Split,160 secondes pour le graphe et 5 seconde pour le Merge. Ce qui est normal car la complexité de notre algorithme de création de graphe est aux alentours de o($n^2$) contre o($n$) pour les deux autres, n étant le nombre de groupes en fin de Split.
  - Ecriture de la javadoc pour toutes les classes sauf le main
  - TO-DO : Le main n'as pas été commenté car il va probablement devoir être restructuré à des fins de lisibilité. Nous aimerions également pouvoir créer quelques tests automatique mais nous heurtons à des difficultés pour l'instant.

Quelques images obtenues :

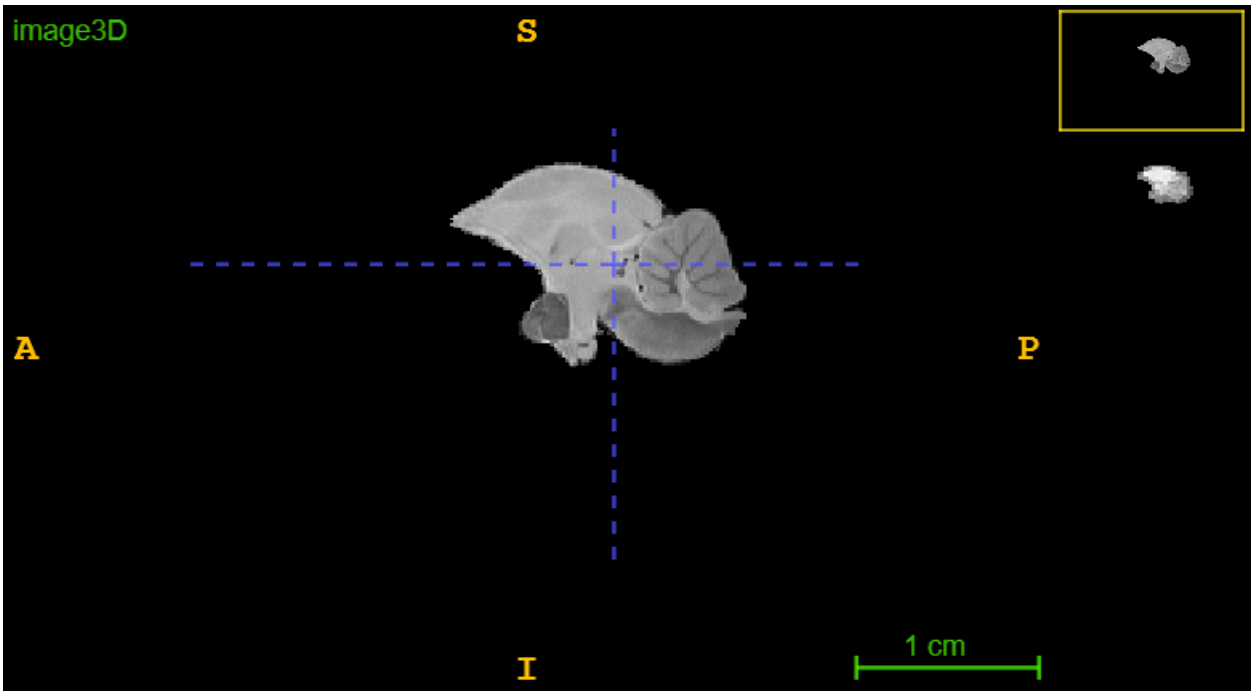


FIGURE 1 – Image avant toute opération

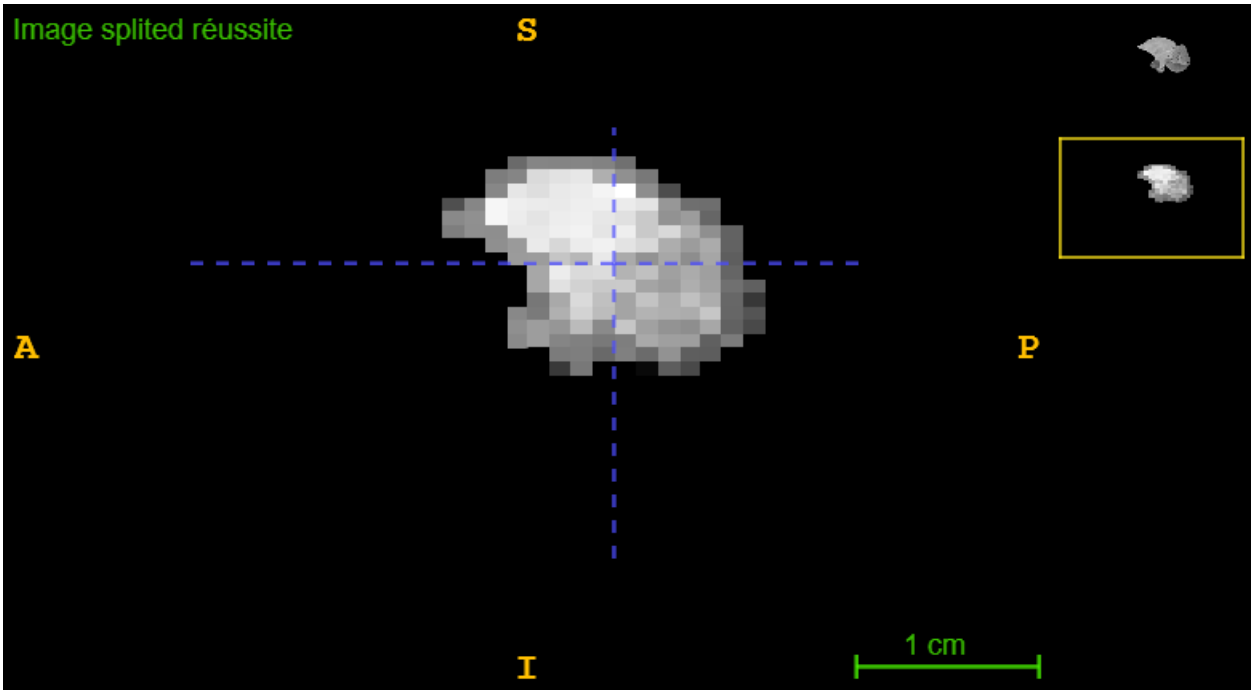


FIGURE 2 – Image après le split

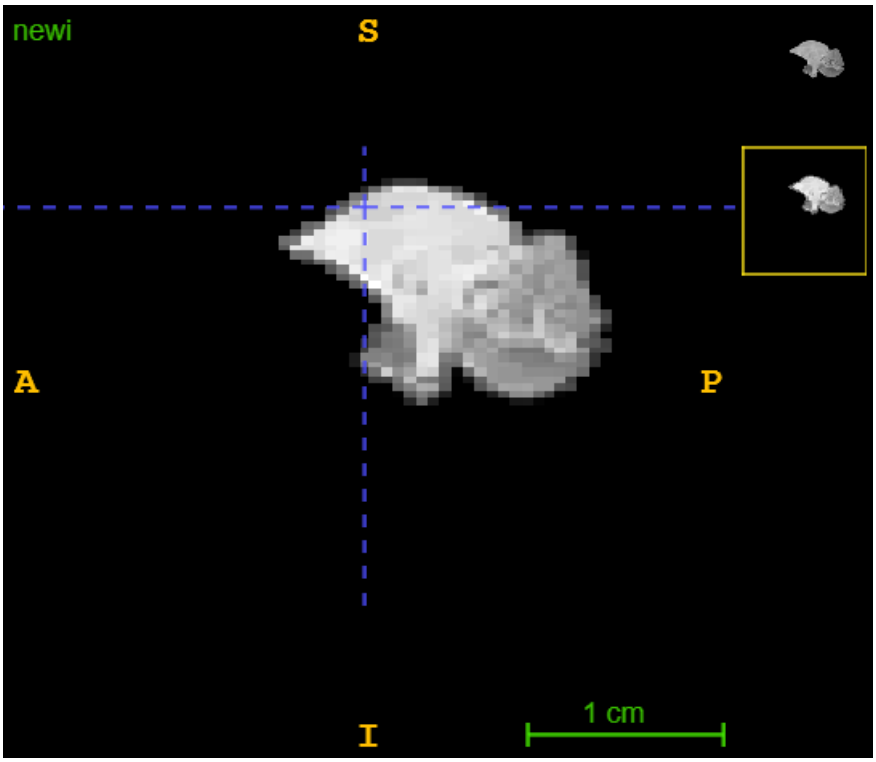


FIGURE 3 – Image après le merge

— 9 novembre, 11ème séance :
  — Création de trois classes de tests sur les classes Cube, Groupe et Split
  — Organisation du projet en deux packages distincts : Test contenant les fichiers des classes de test et Main contenant le reste du projet
  — TO-DO : Restructurer le main et essayer avec plus d'images
— 14 novembre, 12ème séance :
  — Ajout de la classe Graphe récupérant la méthode makeNeighbourGraph et ajout d'une deuxième méthode de calcul de ce graphe à l'aide de test sur les pixels proches
  — Premiers test de nouvelles images et correction du main qui ne permettait pas de travailler sur ces images
  — Ajout d'un excel au tout début de ce document pour entrer les performances de notre programme avec l'une où l'autre des deux méthodes de graphe
  — TO-DO : Continuer les tests avec différentes tailles d'images/ Créer des tests pour la nouvelle classe Graph et essayer cette méthode sur des images déjà traités
— 16 novembre, 13ème séance :
  — Tests sur la classe Graph
  — Ajout de la méthode permettant de supprimer les arcs identiques
  — TO-DO : Continuer les test avec différentes tailles d'images
— 20 novembre, 14ème séance :
  — Tests sur la classe Graph
  — Réorganisation et fragmentation de la classe main
  — Diapo pour la présentation commencé
  — TO-DO : Continuer les test avec différentes tailles d'images, finir la réorganisation du code
— 22 novembre, 15ème séance :

  - Réorganisation de la classe main (ajout d'attributs permettant de décrire l'image avec ajout de la classe files qui gère la création et lecture d'image
  - Commentaire sur la classe Main renommé ImageProcessing
  - Début de tests prometteurs sur la nouvelle méthode
  - Correction d'un bug qui semblait toucher la première méthode lors de la création du graphe mais qui était causé par les changements dans le main
  - TO-DO : Continuer les test avec la nouvelle méthode, empêcher le plantage si le nom de l'image à lire est incorrecte
- 24 novembre, 16ème séance :
  - Fin des test de performance
  - Ajout du traitement de l'exception fichier non trouvé
  - Début du rapport
  - TO-DO : Commenter la classe Files, Ecrire un read me
- 27 novembre, 17ème séance :
  - Début du rapport
  - Ecriture du readme
  - Edition de la javadoc

### 2.3 3ème Partie : Rédaction

Une fois le projet quasiment terminé nous avons commencer la rédaction du rapport et travaillé sur tous nos documents à rendre. Tous les cours étaient dédiés aux mêmes objectifs listés ci-après.

- Rédaction du rapport
- Correction de notre fichier de specifications
- Ajustement de notre diagramme de classe
- Edition de la javadoc finale
- Création d'un diapo de présentation