

Projet ingénieur | TI Santé 1A



Maillage personnalisé d’une image médicale pour la modélisation et la simulation en biomécanique

Tuteur

BAUMGARTNER Daniel

Superviseur

BAYLE Bernard

Membres du groupe

ARRAHMANE Manele (cheffe de projet)

CHOYE Awa

DURAND-TERRASSON Louis

LARQUIER Amaury 2024-2025

Sommaire

1. Présentation du projet………………………..…….…………….………………..……………2
2. Automatisation du processus de segmentation………………………………….…...…..3
3. Automatisation du processus de maillage……..………………………………….…..…..8
4. Conclusion………………………………………………………………………………………..13

Bibliographie……………....……………………………………………………………………………..15

Annexe..………….…….…………………….…………………………………………………………….16

Notice d’utilisation…..…………………….…………………………………………………………….17

1. **Présentation du projet**

Contexte

La modélisation numérique de segments anatomiques permet de simuler leur comportement sous l’effet de contraintes et de mieux appréhender leurs interactions avec divers matériaux et dispositifs. Ayant de nombreuses applications pratiques, elle se retrouve notamment dans la chirurgie assistée par ordinateur, qu’il s’agisse de diagnostic, de planification opératoire via des jumeaux numériques ou encore d’utilisation de la réalité augmentée de manière peropératoire. Dans le cas du crâne humain, par exemple, les modèles numériques peuvent fournir des informations essentielles sur les conditions d’impact au sol de la tête lors d’un choc, et permettent notamment de modéliser des contraintes lors de mouvements brusques.

Afin de mieux prendre en compte les spécificités anatomiques de chaque individu, il est préférable d’utiliser un modèle spécifique au patient. La construction de modèles individualisés, bien qu’elle confère des précisions et des avantages considérables, peut néanmoins représenter une étape chronophage.

Objectifs

Le projet vise ainsi à établir une méthode permettant l’obtention d’un jumeau numérique d’un segment anatomique (modèle par éléments finis), en particulier le crâne humain, pour modéliser et simuler les interactions entre le tissu biologique et des dispositifs médicaux (ex : instruments chirurgicaux). L'objectif final est de fournir un maillage adaptable à chaque individu pour un segment anatomique donné, en s’appuyant d’une part sur des images médicales spécifiques au patient, et d’autre part sur un maillage de référence. L’un des enjeux du projet est également d’automatiser le processus autant que possible afin que l’obtention du modèle personnalisé soit moins laborieuse que si l’on devait adapter manuellement un maillage générique.

La solution proposée, qui doit ainsi permettre d’obtenir un maillage personnalisé du crâne humain à partir d’images médicales (IRM, par exemple), s’appuiera sur l’outil ITK-SNAP pour l’acquisition et le traitement d'images médicales, et sur Altair HyperMesh pour la modification et la personnalisation du modèle par éléments finis.

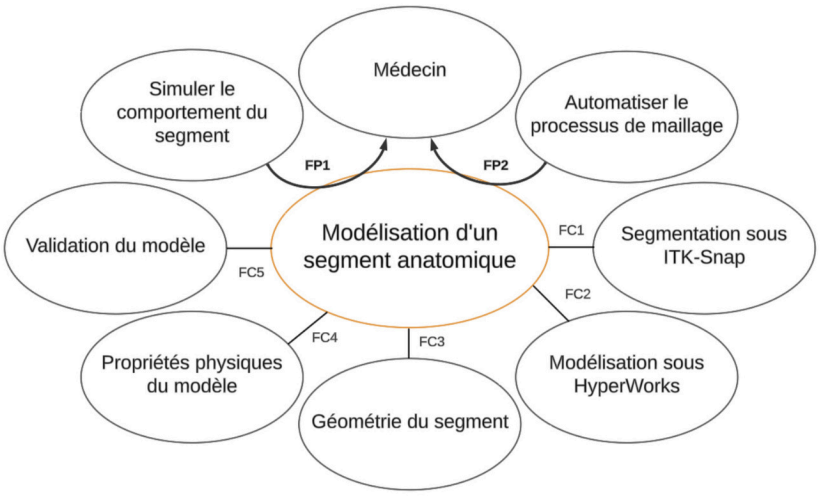
Cahier des charges



Figure 1. Cahier des charges pour la modélisation d’un segment anatomique

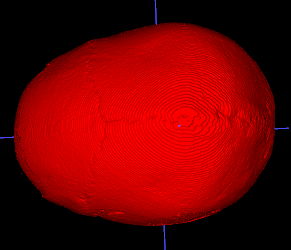
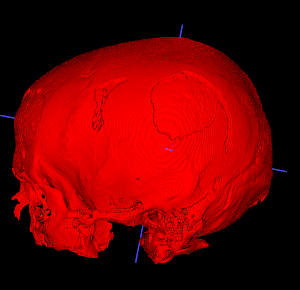
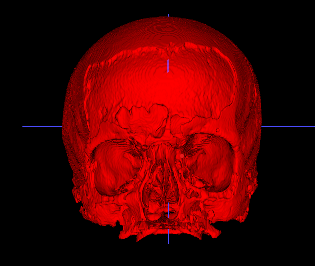
Pourquoi avoir choisi un crâne humain ?

Le crâne humain a été choisi comme fragment anatomique en raison de son importance clinique dans de nombreuses spécialités médicales, notamment la neurochirurgie, la chirurgie maxillo-faciale et la traumatologie. Son maillage personnalisé permet une visualisation précise de la structure osseuse, facilitant la planification d'interventions chirurgicales, la fabrication d'implants sur mesure ou encore l'entraînement médical. De plus, la rigidité et les contours bien définis du crâne en font une structure particulièrement adaptée aux techniques de segmentation et de modélisation 3D.

1. **Automatisation du processus de segmentation**

Premiers résultats : segmentation sur ITK-SNAP

ITK-SNAP est un logiciel open source permettant la segmentation semi-automatique d’images médicales à partir de fichiers DICOM. Après reconstruction du volume 3D, l'utilisateur initialise la segmentation en plaçant des marqueurs (graines) dans la région d'intérêt. Le logiciel utilise alors un algorithme basé sur les contours actifs (level sets), qui fait croître la région segmentée autour des graines selon des critères d’intensité et de forme. Ce processus peut être affiné manuellement, et le résultat final exporté sous forme de masque binaire ou de modèle 3D.



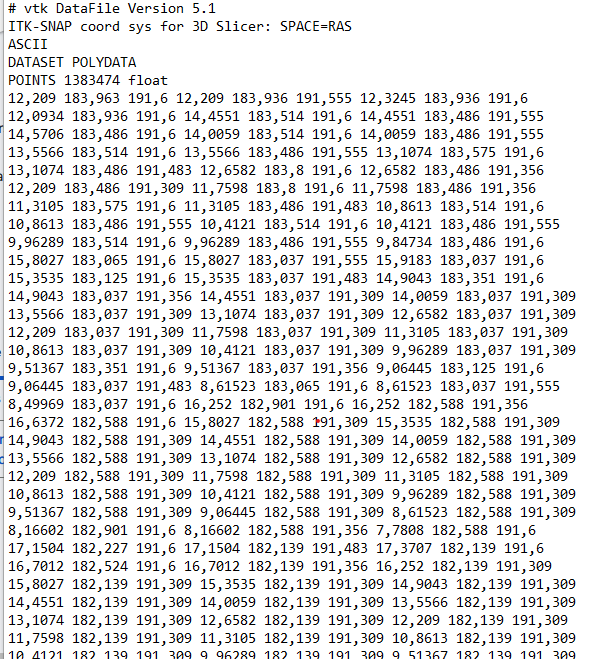
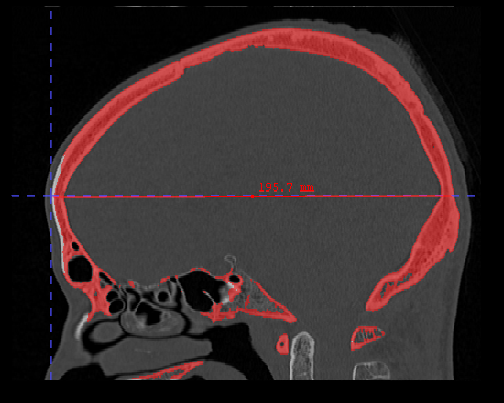
 

Figure 2. Segmentation manuelle réalisée sur ITK-SNAP et exportation de points.

Afin de rendre le processus de modélisation d’un segment plus efficace, nous cherchons, dans le cadre de notre projet, à l’automatiser partiellement. Il s’agit de mettre en œuvre un code qui exécute la plupart des étapes du processus de modélisation afin de réduire au minimum l’intervention humaine dans le processus. Pour la segmentation via ITK-SNAP, bien qu’elle soit déjà semi-automatique, elle nécessite encore l’intervention du médecin, ce que nous ne voulons pas. C’est pourquoi, pour une automatisation complète, nous avons décidé de travailler exclusivement sur Python pour obtenir le fichier STL que nous souhaitons.

Transformation DICOM en Numpy

Pour le code de segmentation, nous avons fait le choix de nous affranchir du logiciel ITK-SNAP. La première étape du code consiste à transformer les images DICOM en un objet facilement manipulable via Python. La fonction implémentée fait appel à diverses bibliothèques : Pydicom (manipulation fichiers DICOM), NumPy (tableaux) et Matplotlib (affichage). Elle convertit une image DICOM en une matrice (tableau numpy) remplie de niveaux de gris entre 0 (noir) et 255 (blanc).

Segmentation par croissance de région

Une technique efficace pour extraire le crâne à partir d’images DICOM est la segmentation par croissance de région. Cette méthode commence avec un ou plusieurs points de départ, qui sont sélectionnés manuellement ou automatiquement dans la région cible.

Dans notre code, les points de départ pour l’algorithme de croissance sont sélectionnés en utilisant la détection des bords. Au lieu de choisir manuellement un point, nous utilisons le détecteur de bord Canny pour identifier les contours du crâne. Ces points de contour sélectionnés sont ensuite utilisés comme points de départ pour la segmentation, garantissant que l’algorithme commence à partir d’emplacements significatifs près de la limite du crâne. Cela permet d’améliorer la précision et d’éviter une segmentation incorrecte.

En combinant la détection des bords et la croissance de région, nous créons une méthode plus fiable pour segmenter le crâne à partir de l’image DICOM.

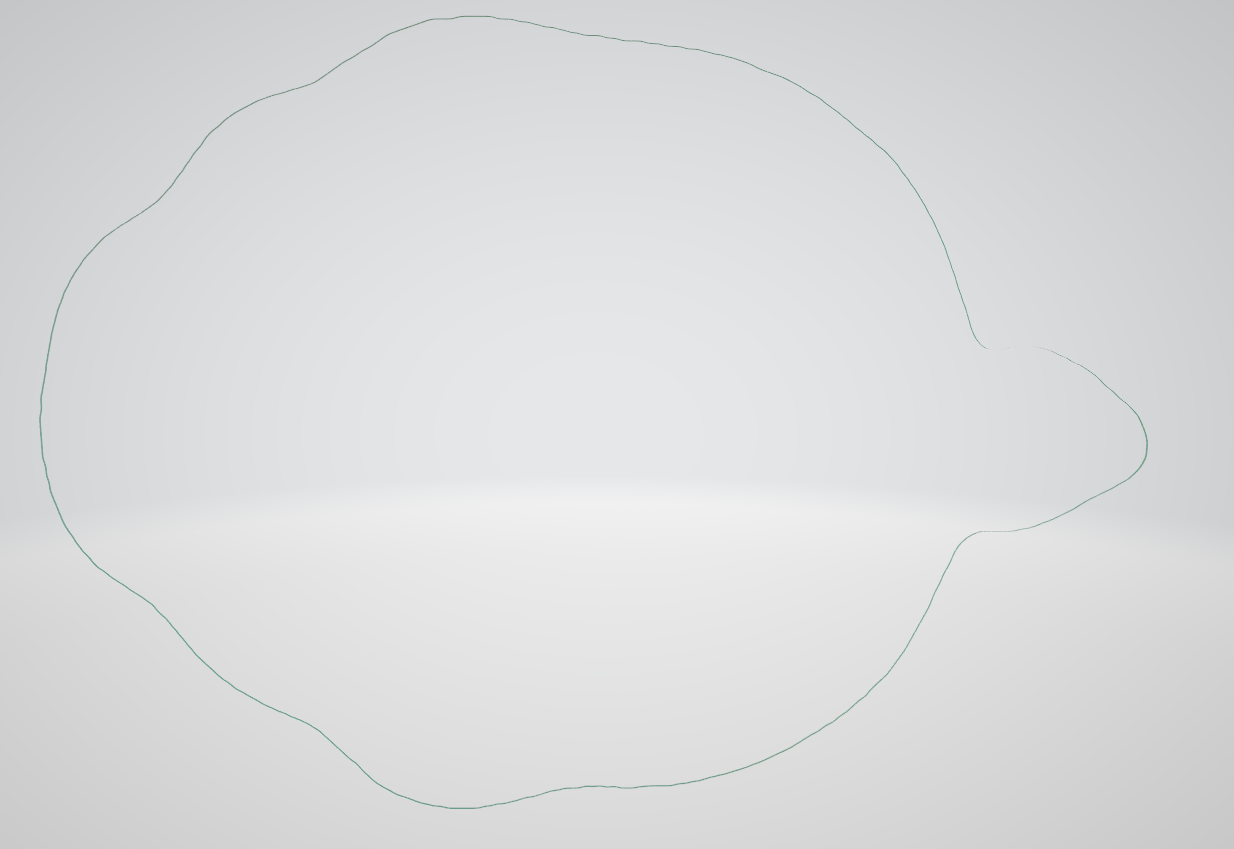
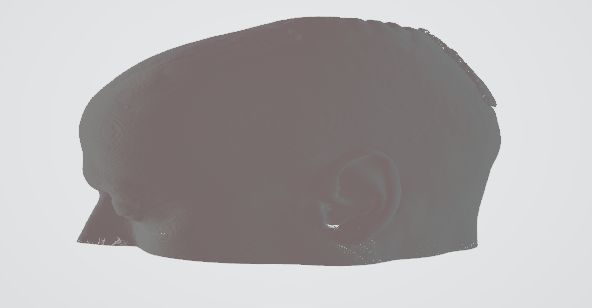


Figure 3. Segmentation d’une couche de crâne humain

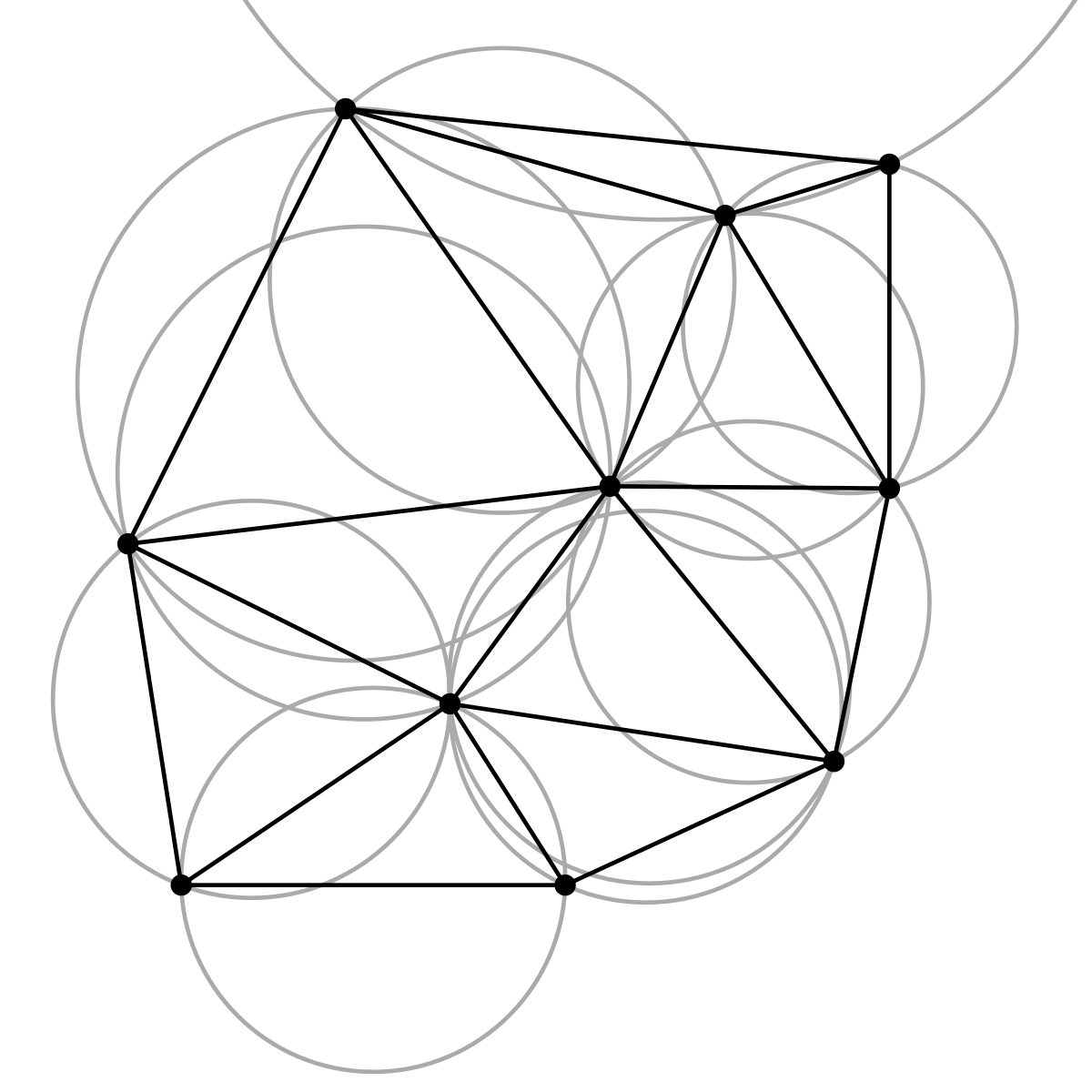
L’algorithme élargit ensuite la région en incluant les pixels voisins qui répondent à un critère de similarité, pour nous, il s’agit des valeurs d’intensité. En Python, cela peut être implémenté en utilisant des bibliothèques comme NumPy, Matplotlib et Pydicom.

L’avantage de cette approche est qu’elle est intuitive et efficace pour segmenter des structures avec des différences d’intensité claires, comme les os. Toutefois, il faut sélectionner soigneusement les points de départ et choisir une bonne tolérance pour éviter d’inclure des zones indésirables.

Une fois la segmentation obtenue pour les 262 images, il nous a fallu reconstruire l’image du crâne. Pour cela nous avons utilisé un algorithme qui, pour chaque tranche, calcule son épaisseur puis va superposer les différentes tranches en faisant en sorte qu'elles ne se chevauchent pas.

Figure 4. Assemblage du crâne 

Après avoir fait cette étape de reconstruction, il nous faut échantillonner ce fichier. En effet, il possède un grand nombre de sommets et d'arêtes et cela peut rendre la manipulation compliquée. Pour cela, nous avons donc échantillonné en utilisant le processus de triangulation de Delaunay qui permet de créer un maillage triangulaire de façon optimale. Il consiste à faire en sorte que, lors de la reconstruction des arêtes après avoir échantillonné de manière uniforme les points, il n’y ait aucun point présent dans le cercle circonscrit de chaque triangle.

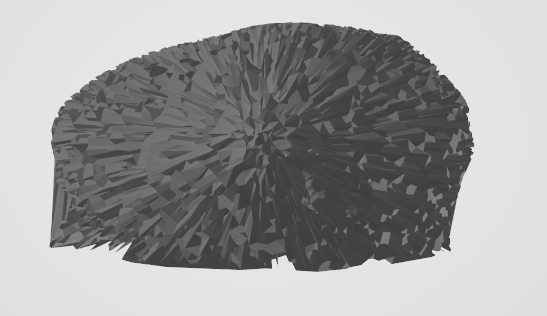
Figure 5. Illustration de la triangulation de Delaunay 

Cette méthode permet de ne pas avoir de triangles trop plats, ce qui pourrait perturber les prochaines étapes. Cependant, une fois échantillonnée, notre structure a perdu une partie de sa forme, il va donc nous falloir la lisser.

Le processus de lissage consiste à considérer la structure externe de notre crâne dans le but d’en extraire uniquement les contours, parties qui nous intéressent le plus.

La raison du choix de cette méthode plutôt qu’un lissage “normal” est que la structure reste désordonnée même avec l’utilisation de la triangulation de Delaunay, ce qui empêche le lissage de se faire correctement.

Pour cela nous avons écrit un programme qui va considérer que ces contours sont en fait l’enveloppe convexe de la structure. Dans ce but, nous avons utilisé l’algorithme Quickhull qui consiste à créer le plus grand tétraèdre possible, diviser les points hors du tétraèdre en catégories reliées à chaque face, trouver le point le plus éloigné pour toutes les catégories et l’ajouter à l’enveloppe convexe afin de créer de nouvelles faces à partir des ces points, enfin réitérer le processus jusqu’à ce qu’il n’y ait plus de points à l’extérieur de l’enveloppe puis assembler les faces ensemble.



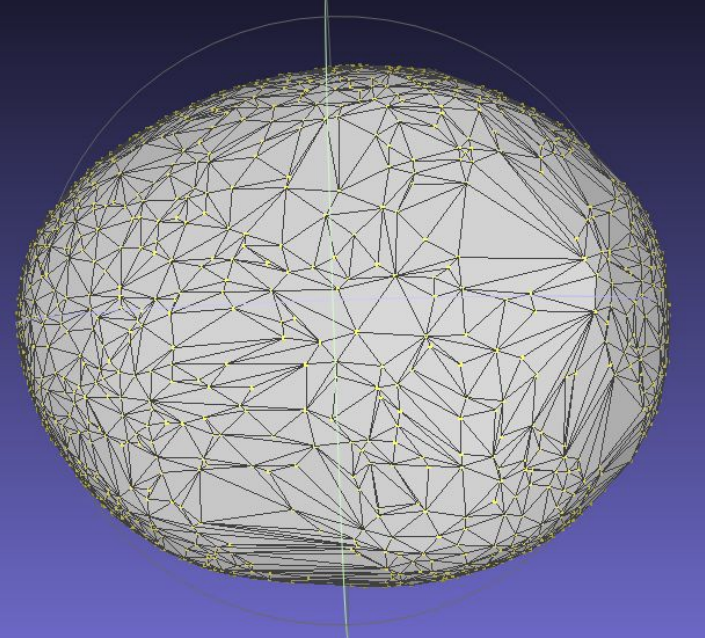


Figure 6. Image du crâne après avoir été échantillonné, avant lissage (image 1) puis après (image 2)

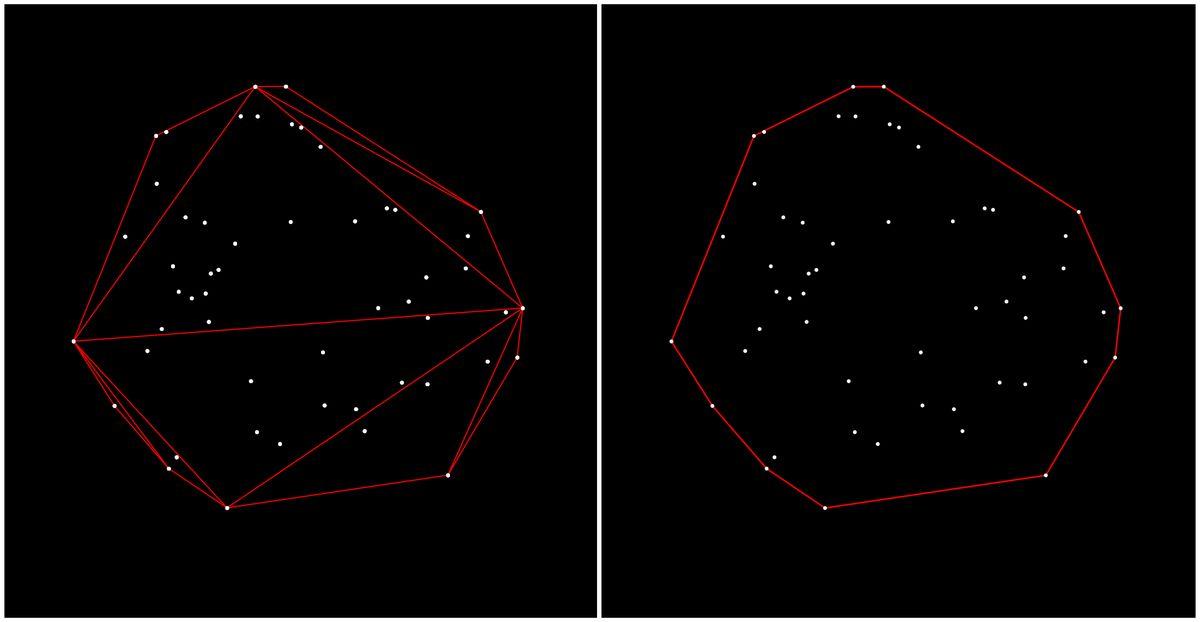
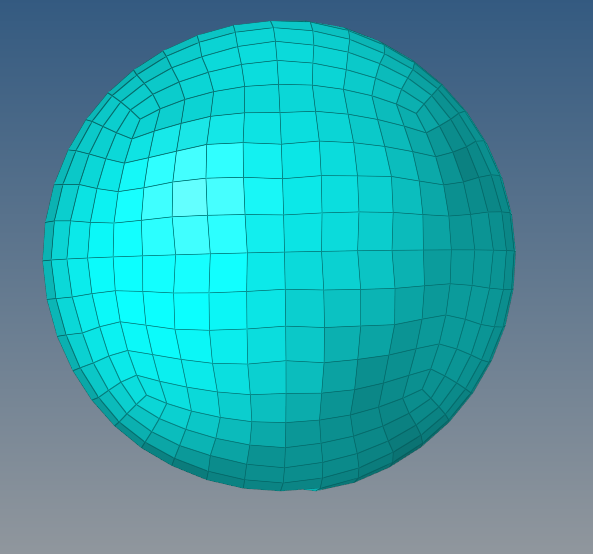
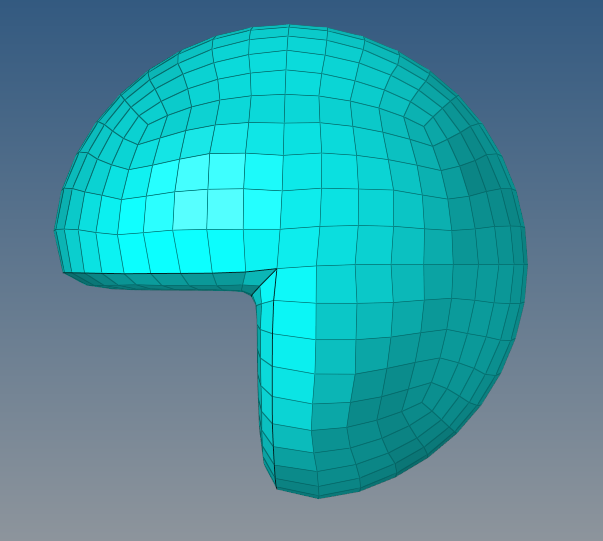


Figure 7. Algorithme Quickhull

1. **Automatisation du processus de maillage**

### **Création et modification du maillage générique**

Afin de pouvoir procéder à une automatisation de l’adaptation du maillage, il était nécessaire de disposer au préalable d’un maillage générique de crâne. Celui-ci a été élaboré dans le cadre d’un cours encadré par M. Baumgartner (voir figure 8 ci-après). Ce modèle représente une base structurée et cohérente sur laquelle les étapes de personnalisation peuvent être appliquées.

Figure 8. Modèle générique 

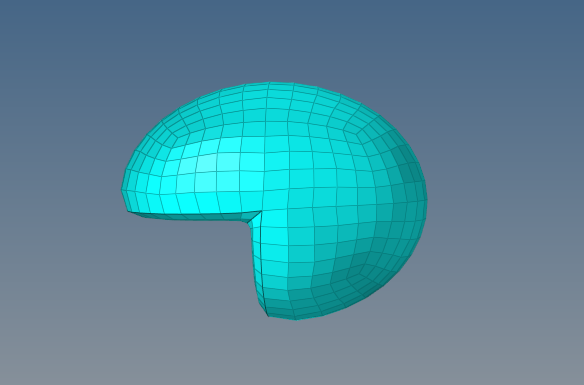
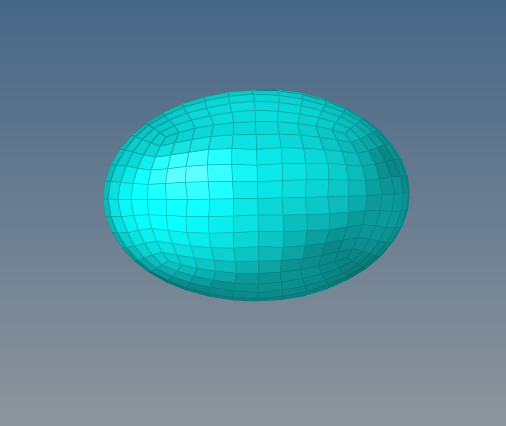
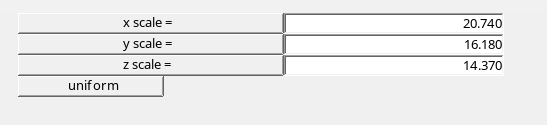
Dans un premier temps, nous avons entrepris de modifier manuellement ce maillage générique, en ajustant certaines propriétés physiques du modèle (largeur, hauteur…). Cette phase de test nous a permis de mieux comprendre la structure du fichier .rad et d'anticiper les contraintes de l'automatisation. (Voir illustration de la modification manuelle ci-dessous figure 9.)

Figure 9. Première modification 

1. **Automatisation du processus d’obtention d’un maillage personnalisé**

Afin d’automatiser le processus d’un maillage personnalisé à partir d’une image médicale et d’un maillage de référence, nous avons exploré deux méthodes : le krigeage ainsi qu’un algorithme de plus proche voisin.

**Première méthode : krigeage**

Le krigeage est une forme d’interpolation spatiale permettant de prédire des valeurs à des endroits non échantillonnés en se basant sur les valeurs connues à d’autres positions. Elle prend en compte la corrélation spatiale entre les données, affectant différentes pondérations aux valeurs connues selon leur distance au point dont on cherche à estimer la valeur. Dans notre cas, le krigeage est dit « dual » car utilisant les données des nœuds du modèle source et celles des nœuds patient.

L’idée est d’ajuster les positions des nœuds du maillage source afin qu’ils correspondent au crâne du patient. L’interpolation est nécessaire dans la mesure où le nombre de nœuds du modèle et du patient n’est pas nécessairement le même. Plutôt que de repositionner directement les nœuds selon leur plus proche voisin, de nouvelles positions pour les nœuds du modèle source sont alors calculées en prédisant leurs valeurs sur la base de la transformation fournie par le krigeage. Cette approche permet de créer une déformation plus lisse et d’obtenir un maillage plus fidèle à la géométrie du patient, tout en préservant l’intégrité structurelle du modèle original.

Étapes du processus

L’implémentation du processus se fait à travers un script Python exécutable dans l’API Python de HyperMesh. Elle nécessite l’installation préalable de librairies additionnelles : PyKrigeet Numpy-stl, afin de pouvoir mettre en place le krigeage et extraire les données d’un fichier STL contenant les coordonnées des nœuds du patient.

Le processus consiste à charger le fichier contenant le maillage générique dans HyperMesh, extraire les coordonnées des nœuds patient et de ceux du maillage, implémenter le krigeage, mettre à jour le maillage (sauvegardé dans un nouveau fichier), et finalement effectuer l’analyse de la qualité du maillage. Le critère que pris en compte pour celle-ci est la distance entre les nœuds source et les nœuds cible après krigeage, autrement dit, pour un nœud du modèle source mis à jour, il s’agit d’évaluer sa distance avec le nœud patient le plus proche.

Essais

À travers le script Python, nous avons réussi à charger le modèle générique et à extraire les coordonnées de ses nœuds ainsi que celles des nœuds du patient contenues dans un fichier STL issu du processus de segmentation détaillé dans la partie précédente.

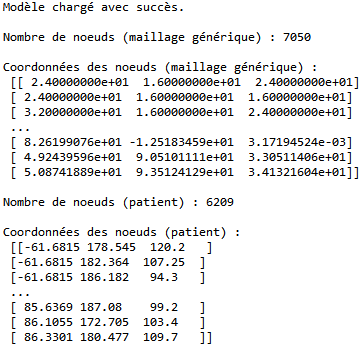


Figure 10. Sortie du script Python sur HyperMesh

Des problèmes sont néanmoins rencontrés lors de l’implémentation du krigeage à proprement parler. En effet, on constate que le temps d’exécution du script varie de manière exponentielle avec le nombre de nœuds. Aussi, là où le script est exécuté en moins de 2 secondes pour une dizaine de nœuds, il requiert plus de 22 minutes pour 3500 nœuds, et plante lorsque plus de 4000 nœuds sont impliqués.

On ne peut donc implémenter le krigeage sans, en contrepartie, réduire de manière conséquente le nombre de nœuds, ce qui ne représente pas une solution optimale. Nous avons alors décidé de nous intéresser à une seconde méthode.

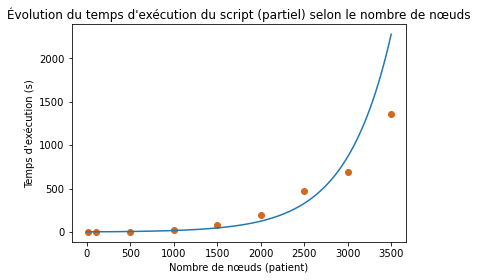


Figure 11. Évolution exponentielle du temps

d’exécution du script avec le nombre de nœuds

**Seconde méthode : algorithme de plus proche voisin**

Dans le cadre de notre projet, nous avons été amenés à choisir un langage de programmation permettant d'interagir efficacement avec le logiciel Altair HyperMesh.

**Exploration de TCL dans HyperMesh :**

Dans un premier temps, nous avons également étudié l'utilisation de TCL (Tool Command Language), un langage de script intégré à HyperMesh qui présente de nombreux atouts:

● Une intégration native avec HyperMesh : accès complet aux fonctionnalités internes

● Une automatisation simplifiée des tâches répétitives

● Une syntaxe simple et efficace : il dispose de nombreuses bibliothèques puissantes

● Un usage répandu dans l'industrie, donc une large communauté

Nous avons néanmoins conclu que l'apprentissage et la maîtrise de TCL demanderaient trop de temps au vu des exigences de notre projet. Nous avons donc décidé de nous orienter vers Python, qui reste un langage plus familier et accessible.

Ainsi, nous avons décidé d'utiliser Python pour plusieurs raisons fondamentales, notamment sa grande puissance et flexibilité dans le traitement de données, une syntaxe claire et intuitive ainsi qu’une vaste communauté et un large écosystème. L'API (*Application Programming Interface*) Python d'Altair HyperMesh offre la possibilité d'interagir avec le logiciel via des scripts Python. Cette intégration permet :

● L'automatisation des configurations et des modifications du modèle

● L'exécution de processus en lot : réduction des interventions manuelles

● L'extraction et l'analyse des données de simulation

● L'intégration avec des bibliothèques externes, ce qui facilite les analyses de données

**Limitations de HyperMesh 2022.3**

Après une analyse approfondie de la documentation officielle d'Altair, nous avons découvert que la version 2022.3 d'HyperMesh ne permettait pas l'exécution interne de scripts Python. Cette restriction s'explique par le fait que l'intégration complète de Python n'a été introduite que dans des versions plus récentes du logiciel.

Bien que la mise à jour du logiciel sur les ordinateurs de l’école ne soit pas possible dans l’immédiat, nous avons pu accéder à la version 2024.1 à travers nos ordinateurs personnels, version qui intègre un support Python plus avancé et nous permettra ainsi de travailler de manière plus efficace avec l’API Python d’HyperMesh.

C’est pour cela que dans le but d’adapter le maillage générique de crâne aux données spécifiques issues de la segmentation d’un patient(STL), nous avons opté pour une automatisation du processus en externe, en dehors d’Altair HyperMesh. Cette approche permet une plus grande flexibilité tout en conservant la compatibilité avec les formats de simulation utilisés dans notre workflow.

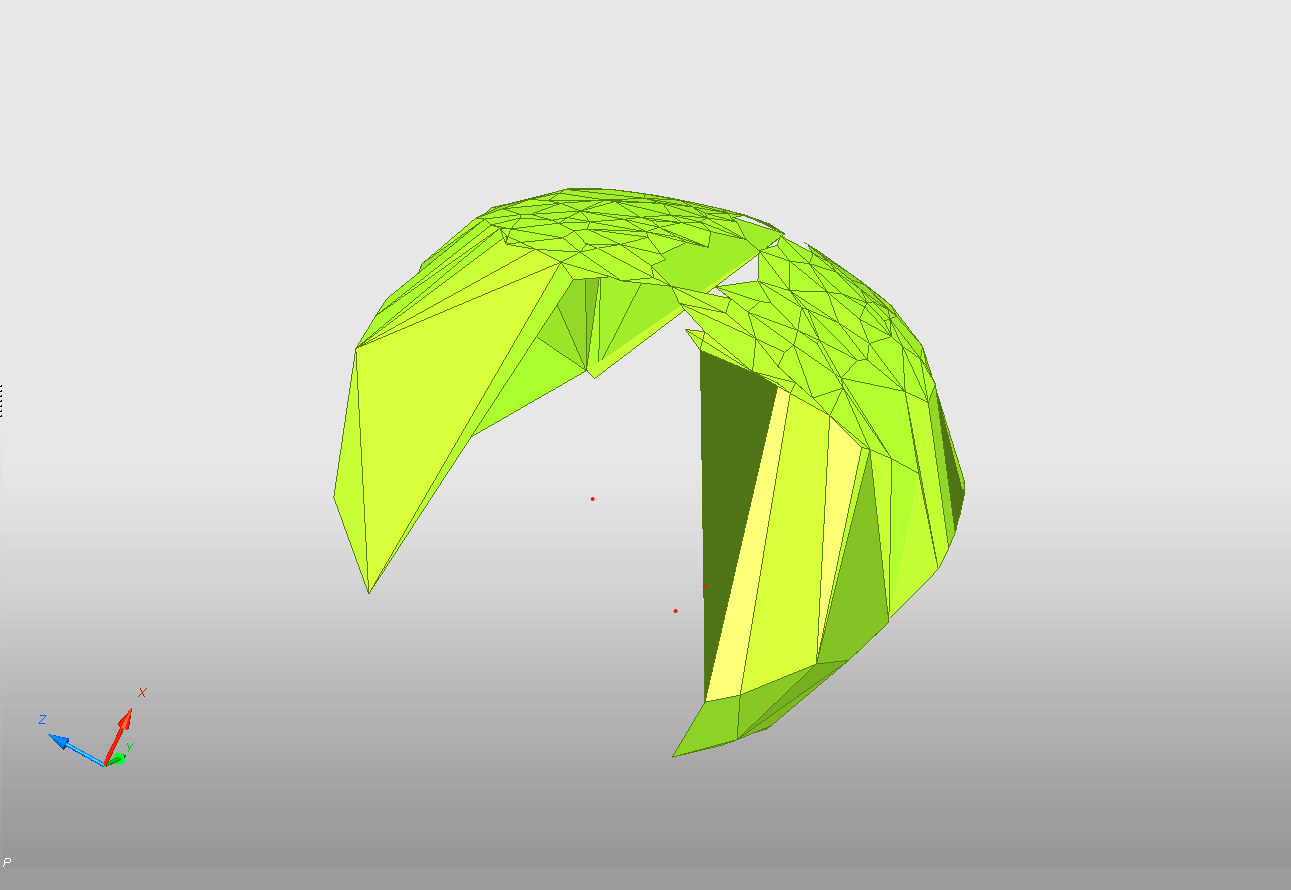
**Méthodologie**

1. Extraction du maillage générique :  
    Le maillage de référence est initialement généré dans HyperMesh puis exporté au format Radioss (.rad), un format compatible avec notre chaîne de simulation.
2. Projection sur le modèle patient :  
    Le modèle patient est issu d’une segmentation anatomique et exporté au format STL. L’objectif est de projeter les nœuds du maillage générique sur la géométrie du crâne patient de manière automatisée, en s’assurant de conserver la structure du maillage pour garantir la compatibilité avec les simulations ultérieures.
3. Approche algorithmique :  
    Nous avons développé un script Python utilisant les bibliothèques trimesh, scipy, numpy, et matplotlib. L’algorithme suit les étapes suivantes :
   * Lecture des nœuds (/NODE) et des éléments surfaciques (/ELEMENT/SHELL) depuis le fichier .rad.
   * Centrage des points du maillage et de la géométrie STL afin d’aligner les référentiels.
   * Utilisation d’une structure cKDTree (algorithme de plus proche voisin) pour projeter chaque nœud du maillage générique sur la surface STL.
   * Recentrage des points projetés vers le référentiel original de la STL.
   * Filtrage des éléments contenant des nœuds absents ou invalides.
   * Réécriture du fichier Radioss projeté, en conservant les éléments structuraux.
4. Visualisation et vérification :  
    Une visualisation 3D des points projetés est générée afin de vérifier la qualité de la projection et d’identifier d’éventuels décalages ou artefacts.

**Résultats**

Cette automatisation nous a permis d’obtenir un maillage personnalisé adapté à la morphologie du patient, tout en respectant les contraintes de format et de connectivité imposées par la simulation Radioss. Ce processus est reproductible, rapide, et offre une bonne robustesse, en particulier pour des géométries complexes telles que le crâne humain.

Néanmoins, le résultat que nous obtenons n’est pas satisfaisant vis-à-vis de la reconstruction des éléments structuraux (cf. figure 12).

Figure 12. Résultat final au format Radioss

1. **Conclusion**

Livrables :

1. Interface

Afin de faciliter l’utilisation de nos programmes par les utilisateurs nous avons développé un interface permettant de :

* Glisser déposer le dossier contenant les images dicom à segmenter
* Sélectionner les différents paramètres modifiables (nombre de points de départ, tolérance lors de la segmentation et taux d’échantillonnage)
* Voir la progression en temps réel via une barre de progression

De plus à l’exécution du programme un message d’avertissement apparaît afin de préciser à l’utilisateur les différentes bibliothèques Python nécessaires au bon fonctionnement du programme.

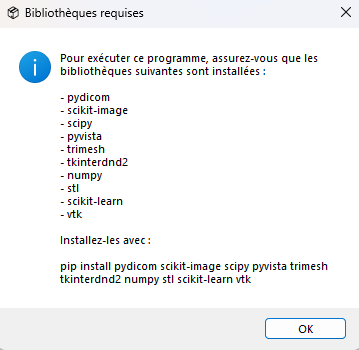
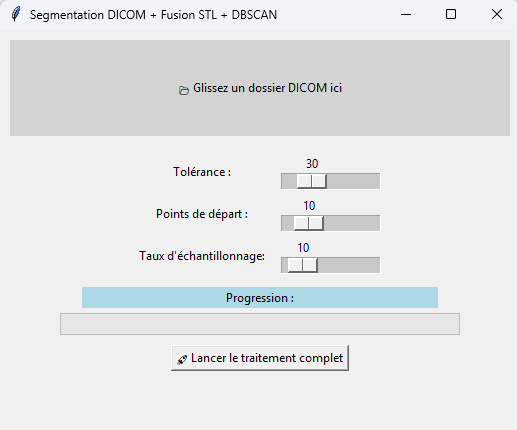


Figure 13. Interface utilisateur

1. Notice d’utilisation

Dans le cadre de ce projet, une notice d’utilisation a été rédigée afin de guider l'utilisateur dans l'installation et l'exécution des scripts Python développés. Cette documentation détaille l’ensemble des bibliothèques nécessaires à l’environnement d’exécution (telles que pydicom, numpy, matplotlib, trimesh, scipy, PyKrige, numpy-stl, etc.) ainsi que la procédure à suivre pour lancer les scripts de segmentation, de reconstruction 3D et d’adaptation du maillage.

Elle précise également la structure attendue des dossiers (ex. dossier DICOM en entrée, emplacement du modèle STL, fichiers .rad du maillage générique) ainsi que les paramètres configurables (seuils, tolérances, options d’affichage). L’objectif est de rendre l’ensemble de notre travail réutilisable et facilement déployable par un tiers, sans connaissance approfondie du code source.

La notice permet ainsi d’assurer la reproductibilité du processus et facilite sa diffusion dans un cadre médical ou de recherche. La notice d’utilisation se trouve en annexe.

Aboutissement du projet et respect du cahier des charges

Ce projet a permis de développer un processus automatique de génération de maillages personnalisés à partir d’images médicales, centré sur le crâne humain. En combinant la segmentation d’images DICOM, le traitement géométrique en Python et l’adaptation du maillage générique via projection et/ou krigeage, nous avons obtenu un modèle 3D adapté à la morphologie spécifique d’un patient. Bien que certaines limitations logicielles aient ralenti l’intégration complète avec HyperMesh, des solutions alternatives ont été explorées pour garantir la compatibilité et l’efficacité du script.

Le projet répond partiellement au cahier des charges initial : il permet de produire un maillage qui n’est certes pas fidèle mais qui reste convaincant, pas loin d’être réutilisable en simulation biomécanique, tout en réduisant la part d’intervention manuelle. Des perspectives d’amélioration restent ouvertes, notamment sur le temps de calcul du krigeage pour les maillages de grande taille, et l’intégration directe avec les versions récentes d’HyperMesh pour une automatisation plus fluide.

Bibliographie

[1] NOLWENN FOUGERON, PIERRE-YVES ROHAN, AURÉLIEN MACRON, CHRISTOPHE TRAVERT, HÉLÈNE PILLET & WAFA SKALLI : Subject specific finite element mesh generation of the pelvis from biplanar x-ray images: application to 120 clinical cases : *Computer Methods in Biomechanics and Biomedical Engineering (2018), p. 408-412*

*DOI: 10.1080/10255842.2018.1469624*

[2] CHRISTIANE CAOUETTE, NICOLE IKIN, ISABELLE VILLEMURE et al. : Geometry reconstruction method for patient-specific finite element models for the assessment of tibia fracture risk in osteogenesis imperfecta : *Medical & Biological Engineering & Computing, Vol. 55 (2017), p. 549–560*

*DOI:* *10.1007/s11517-016-1526-5*

[3] NADINE MICHÈLE LALONDE, YVAN PETIT, CARL-ERIC AUBIN, ERIC WAGNAC, PIERRE-JEAN ARNOUX : Method to Geometrically Personalize a Detailed Finite-Element Model of the Spine : *IEEE Transactions on Biomedical Engineering, Vol. 60, Issue 7 (2013), p. 2014-2021*

*DOI: 10.1109/TBME.2013.2246865*

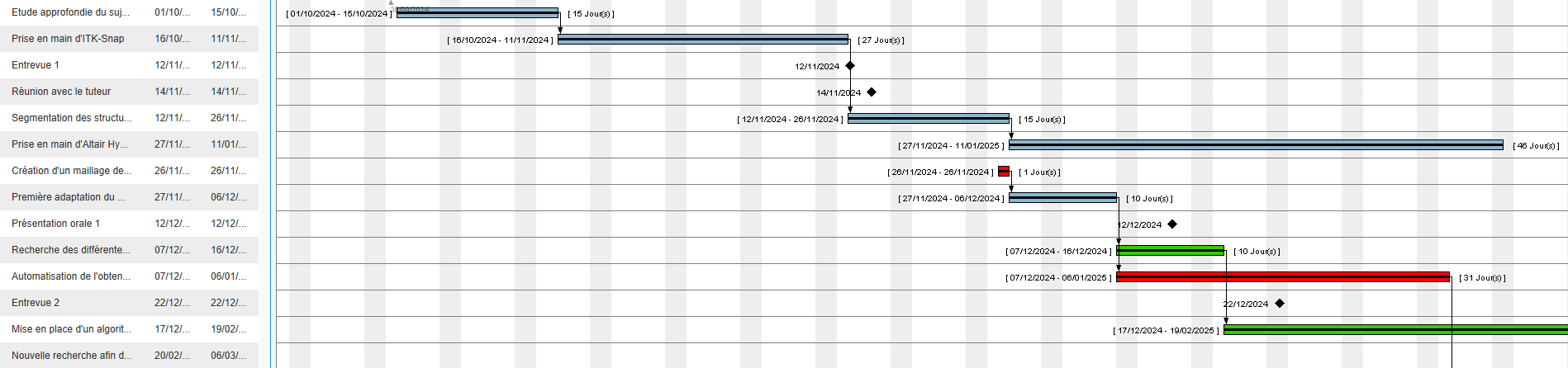
*ISSN: 0018-9294 (print), 1558-2531 (electronic)*

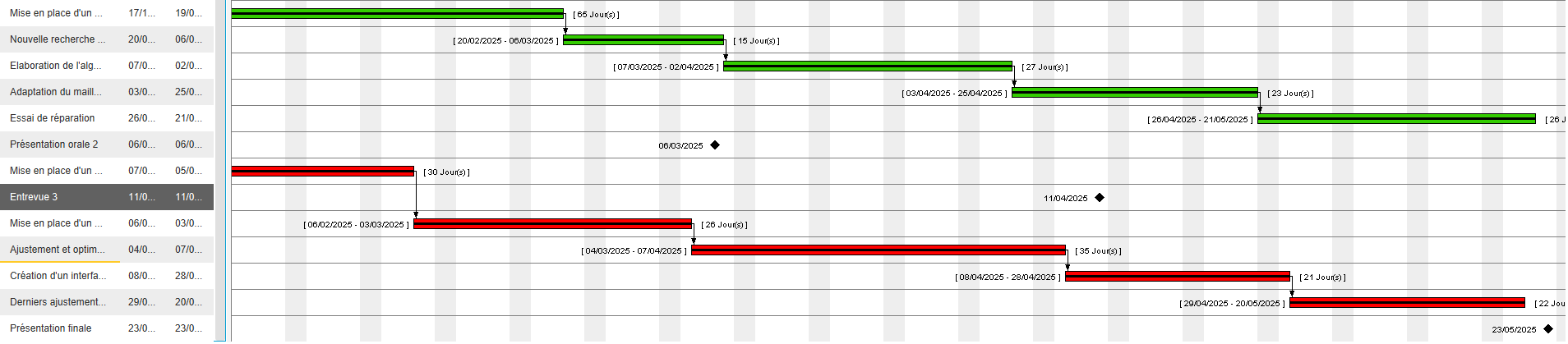
[4] HILMI RIFAI, ISABELLE BLOCH, SETH HUTCHINSON, JOE WIART, LINE GARNERO : Segmentation of the skull in MRI volumes using deformable model and taking the partial volume effect into account : *Medical Image Analysis, Vol. 4, Issue 3 (2000), p. 219-233*

*DOI: 10.1016/S1361-8415(00)00016-5*

Annexe

* Planning du travail sur l’année





## **Notice d'utilisation — Globale**

## **Version** : 1.0 **Auteur** : ARRAHMANE Manele (cheffe de projet), CHOYE Awa, DURAND-TERRASSON Louis, LARQUIER Amaury. **Contact** : [manele.arrahmane@etu.unistra.fr](mailto:manele.arrahmane@etu.unistra.fr)

## **Notice d'utilisation — Script de segmentation et de modification du maillage**

**Nom du script** : segmentation\_dicom\_vers\_stl.py

### **Objectif**

Ce programme permet de segmenter des images au format dicom afin d’obtenir un maillage au format stl (**stéréolithographie). Pour cela le programme va passer par une phase de segmentation, une phase d'échantillonnage et une phase de lissage.**

### **Prérequis**

#### **1. Python 3.8+ installé**

#### **2. Bibliothèques Python nécessaires : pydicom, scikit-image, scipy, pyvista, trimesh, tkinterdnd2, numpy, stl, scikit-learn, vtk.**

Installer via pip si besoin : “pip install pydicom scikit-image scipy pyvista trimesh tkinterdnd2 numpy stl scikit-learn vtk”

#### 3. Fichiers requis :

* Un dossier contenant les images dicom à segmenter

### **Utilisation**

1. Faire glisser le dossier contenant les images sur l’interface .
2. Sélectionner le nombre de points de départ, la tolérance pour la segmentation et le taux d’échantillonnage.
3. **Cliquer sur “Lancer le traitement complet”**

**Fichier de sortie**

* **Nom** : Libre (défini dans le paramètre de sortie).
* **Format** : .stl.
* **Contenu** : - Votre structure segmentée, échantillonnée selon votre choix puis lissée

**Notice d'utilisation — Script de projection de maillage Radioss sur surface STL**

**Nom du script** : projection\_radioss\_sur\_stl.py

### **Objectif**

Ce programme permet de **projeter automatiquement les nœuds d’un maillage Radioss (.rad)** sur une **surface anatomique issue d’un fichier STL** (par exemple, un crâne reconstruit par segmentation), afin d’adapter le maillage générique à une morphologie patient-spécifique. Le résultat est un **nouveau fichier .rad prêt pour la simulation**, avec la topologie d’origine mais une géométrie adaptée.

### **Prérequis**

#### **1. Python 3.8+ installé**

#### **2. Bibliothèques Python nécessaires : numpy, trimesh, matplotlib, scipy.**

Installer via pip si besoin : “pip install numpy trimesh matplotlib scipy”

#### 3. Fichiers requis :

* Un fichier Radioss contenant les nœuds et éléments : votre\_fichier.rad
* Un fichier STL de la géométrie cible : votre\_fichier.stl

### **Utilisation**

1. Faire attention à bien spécifier les chemins d’accès :

**'chemin/vers/fichier.rad' ; 'chemin/vers/fichier.stl' ; ‘chemin/vers/fichier\_de\_sortie.rad',**

1. **Lancer le script avec Python.**

### **Fichier de sortie**

* **Nom** : Libre (défini dans le paramètre de sortie).
* **Format** : .rad (compatible HyperMesh / Radioss).
* **Contenu** : -Section /NODE mise à jour avec les points projetés.  
   -Section /ELEMENT/SHELL.