# Projet de fin d'étude: Analyse d'images 3D de peaux d'agrumes

CHAVEROUX Julien, KLIBI Salim, RIVERAIN Olivier

27 Mars 2024


Clients : Anne Vialard, Fabien Baldacci, Louise Le Barbenchon

# Table des matières

# Introduction

La bio-inspiration est un domaine qui existe depuis longtemps. Elle consiste à s'inspirer de la nature dans le but de trouver des solutions à des problèmes et à créer des nouvelles technologies. L'être humain s'est toujours inspiré de la nature pour créer de nouvelles technologies. Léonard de Vinci (1452 - 1519) disait à ses élèves « Scrute la nature, c'est là qu'est ton futur. ». Ce domaine de recherche prend une place de plus en plus importante. La structure en nid d'abeille a permis de créer des matériaux légers et solides. La structure du fil d'araignée a permis de réaliser des fils résistants et légers. Louise Le Barbenchon, chercheuse à l'I2M (Institut de mécanique et d'ingénierie) à Talence, a effectué sa thèse sur les propriétés absorbantes du liège. Aujourd'hui, ses travaux de recherche concernent les propriétés d'absorption d'énergie lors d'un impact sur la peau d'un fruit appelé Citrus Maxima ou encore connu sous le nom de pamplemousse chinois. Elle cherche à comprendre quelle est l'influence de la structure de la peau de ce fruit dans cette absorption. Elle a donc soumis plusieurs de ces pamplemousses à des rayons X pour pouvoir visualiser la densité de leurs peaux. Cette expérience a été réalisée dans le synchrotron de l'ESFR (European Synchrotron Radiation Facility) situé à Grenoble.

Louise souhaite avoir une modélisation de l'évolution de la densité de la peau de l'extérieur vers le centre du fruit, mesurer les réseaux vasculaires présents au sein de la peau des agrumes et caractériser la morphologie des cellules de la peau. Dans le cadre de l'UE "Projet de Fin d'étude", nous allons travailler sur le projet intitulé "Analyse d'images 3D de peaux d'agrumes". En effet, nous allons devoir mettre en place un modèle pour visualiser des images 3D d'agrumes, extraire automatiquement la densité de la peau le long de normales à la surface de l'agrume. La segmentation des régions d'intérêt devrait se faire à différentes échelles pour les réseaux vasculaires de la peau, et les surfaces des cellules afin de calculer les caractéristiques géométriques des régions d'intérêt.

Dans un premier temps, nous allons présenter une analyse de l'existant comprenant les différentes bibliothèques et logiciels. Dans un deuxième temps, nous allons définir les problématiques liées aux données fournies. Après avoir énoncé les User Stories et l'organisation du travail, nous enchaînerons sur l'architecture logicielle du projet et son implémentation. Nous finirons par discuter des différents tests que nous avons menés.

# 1 État de l'art

## 1.1 Bibliothèques

### 1.1.1 ITK

ITK (Insight Segmentation and Registration Toolkit) [McC+14] est une bibliothèque en C++ utilisée pour la manipulation et traitement d'images médicales 2D et 3D comme la segmentation par exemple.

### 1.1.2 VTK

VTK (Visualization Toolkit) est une bibliothèque de visualisation 3D accessible en C++ et Python. Principalement utilisée dans des domaines scientifiques et médicaux, elle est utile pour visualiser et manipuler des données en temps réel de manière interactive.

### 1.1.3 DGTal

DGTal (Digital Geometry Tools and Algorithms) est une bibliothèque en C++ qui, comme son nom l'indique, permet la manipulation et l'analyse de données géométriques discrètes et comprend de nombreux outils tels que la segmentation, la détection de contours, l'extraction de caractéristiques et plus encore.

### 1.1.4 QT

QT est une bibliothèque utilisée pour des interfaces graphiques codées en C++, Python ou encore JavaScript. Facile d'utilisation et multi-plateforme, elle permet la visualisation 2D de données pour plusieurs types de logiciels.

## 1.2 Les logiciels

### 1.2.1 QT creator

Qt Creator est un environnement de développement intégré (IDE) multiplateforme conçu pour maximiser la productivité des développeurs. Il prend en charge l'utilisation d'assistants de codage tels que GitHub Copilot

pendant la programmation. Il aide les développeurs à créer des logiciels pour les plates-formes de bureau, mobiles et embarquées [1].

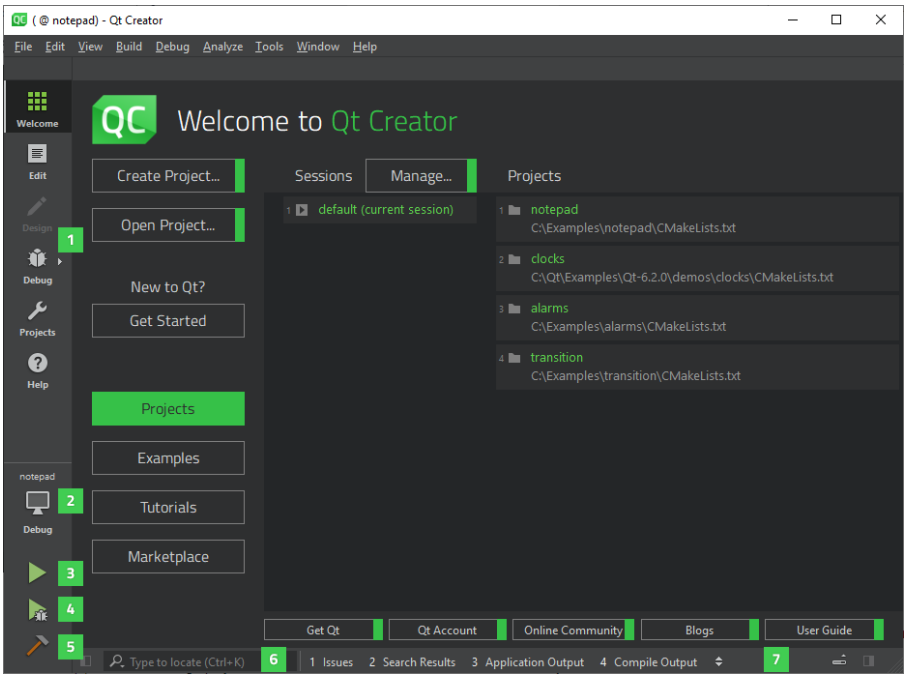


FIGURE 1 – Interface du logiciel QT Creator

### 1.2.2 Fiji/ImageJ

Le logiciel ImageJ et son extension Fiji est un logiciel basé sur Java qui permet d'analyser et de traiter des images dans plusieurs domaines scientifiques et plus particulièrement dans celui de la biologie et médical.
Il permet de visualiser les images 2D dans plusieurs formats dont le JPEG2000. Il permet de filtrer et de faire de la segmentation. Il est modulaire puisque des plugins peuvent être ajoutés.
L'utilisateur peut aussi afficher des images en 3D et intéragir avec elles.

### 1.2.3 3DSlicer

L'outil de visualisation 3D le plus utilisé est 3DSlicer [PHK04]. Il est beaucoup utilisé dans le domaine médical. Il gère beaucoup de formats comme les dicom, nrrd et tiff.
Il 2 s'appuie sur les bibliothèques TCL/Tk, ITK et VTK. TCL/Tk sert à créer l'interface graphique. Itk permet de gérer différents formats, de les convertir, d'empiler des images 2D pour créer une images 3D et de réaliser des segmentations. VTK, quant à elle, a pour utilité d'afficher une représentation 3D.
3DSlicer est bien adapté pour les tailles de fichiers peu importantes. Par contre, nous avons essayé de prendre en entrée un fichier de taille 1.4 Go et il a nécessité une quantité de RAM supérieure à 32 Go pour pouvoir afficher une représentation 3D. Or nos fichiers .nrrd ont une taille d'environ 15 Go.

1. https ://www.qt.io/product/development-tools

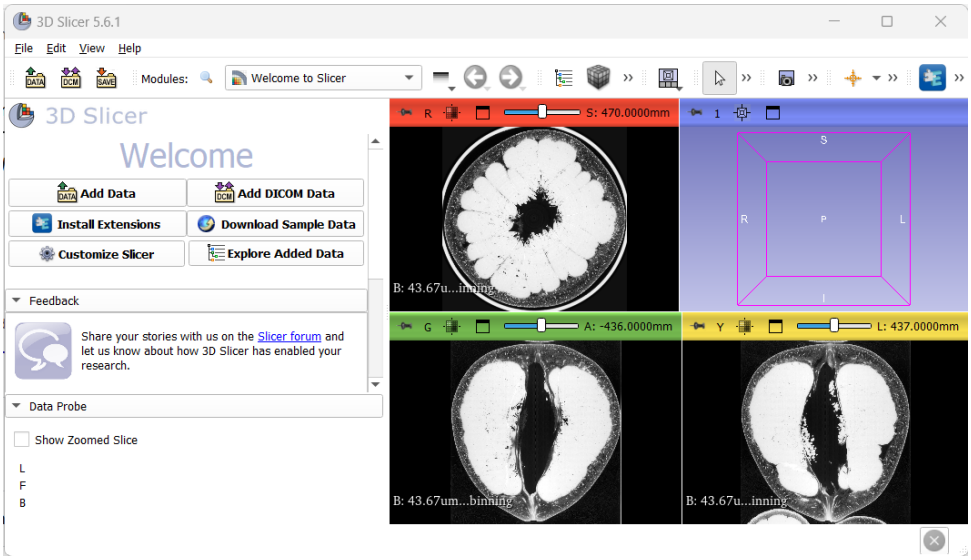


FIGURE 2 – Interface du logiciel 3DSlicer

### 1.2.4 ITK-SNAP

Le logiciel ITK-SNAP comme une partie de son nom l'indique est basé sur la bibliothèque ITK. il s'agit d'un logiciel intéractif qui permet à son utilisateur de naviguer dans des images 3D provenant d'IRM ou de tomographie. Son point fort est qu'il permet de faire des segmentations précises.

## 1.3 L'extraction de la densité

Pour extraire la densité d'un objet 3D, il est nécessaire de réaliser plusieurs étapes.

La première est d'obtenir une surface 3D. Il existe plusieurs méthodes pour obtenir une surface 3D d'un objet 3D conçu à partir d'un ensemble d'images 2D. La plus efficace a été développée par Lorensen et ses collègues en 1987. Il s'agit de l'algorithme Marching Cubes [LC87]. Cet algorithme est vraiment adapté pour les images provenant de la tomographie. Wheeler et al. [WSI98] utilise cet algorithme dans le cas d'images de taille importante en y ajoutant un octree pour pouvoir diminuer les temps de calcul et les besoins mémoire.

La deuxième est de calculer la normale en chacun des voxels de la surface. Pour cela, il suffit de calculer le produit vectoriel des vecteurs vers les voxels voisins.

La troisième est de partir d'un voxel et de se déplacer suivant la normale et de récupérer le niveau de la densité par le niveau de gris du voxel.

## 1.4 La segmentation : articles scientifiques

Dans cette section, nous citons quelques travaux qui ont été basés sur la segmentation.

Une étude de Vijayarekha K. [Vij12] a comparé quatre méthodes de segmentation des défauts sur des images de mandarines, en utilisant notamment l'amélioration itérative de l'intensité et la segmentation par couleur RVB. Les résultats ont montré que l'amélioration de l'intensité était préférable en raison de l'impact de l'éclairage sur les images, mais la segmentation par couleur était plus rapide et pourrait être utile dans les applications nécessitant uniquement la détection des défauts.

Wang et al. [WL17] ont présenté un cadre général pour la segmentation et la quantification automatique de divers types de cellules, introduisant des méthodes de calibration du seuil, de filtrage des blobs de bruit et de lissage des frontières. Leur approche offre une segmentation cellulaire plus convergente et précise, surpassant les méthodes antérieures de quantification des cellules.

Zhou et al. [Zho+08] ont développé un système d'analyse automatisé pour segmenter, suivre et quantifier les comportements du cycle cellulaire sur une grande population de noyaux cellulaires. Leur méthode combine la binarisation adaptative, l'algorithme des lignes de partage des eaux et un modèle de Markov pour identifier précisément les phases du cycle cellulaire. Les résultats expérimentaux ont confirmé l'efficacité du système, offrant un potentiel prometteur pour l'étude des processus cellulaires dynamiques à grande échelle.

Yin et al. [YMX08] ont présenté quatre méthodes de segmentation pour les images de fruits, démontrant que la méthode de seuillage dynamique est plus performante et rapide que les autres, y compris l'Otsu étendu et

l'arithmétique génétique. Cette méthode fonctionne bien même sous un éclairage naturel et utilise la caractéristique d'aberration chromatique R-G pour simplifier la segmentation en niveaux de gris. Elle offre de meilleures performances que la segmentation adaptative basée sur le réseau LVQ, moins adaptée aux applications en temps réel.

Roy et al. [Roy+19] ont développé une approche de segmentation pour détecter les légumes pourris, offrant des résultats prometteurs grâce à trois techniques différentes : marqueurs, couleurs et contours. Ces méthodes sont efficaces pour séparer les parties saines des parties pourries des légumes, facilitant ainsi le tri automatisé dans les usines alimentaires. Les résultats expérimentaux ont confirmé l'efficacité de cette approche, en particulier l'utilisation du filtre de Canny pour la détection des contours, offrant une différenciation nette et indépendante des valeurs de pixels et de la position de la caméra.

Dimopoulos et al. [Dim+14] ont présenté une méthode de segmentation cellulaire automatique qui garantit une détection précise des limites cellulaires individuelles en tenant compte des informations de la membrane cellulaire. Leur approche, basée sur des coupes graphiques avec des corrélations croisées directionnelles et des contraintes spatiales intégrées automatiquement, a démontré une précision remarquable sur des images de divers types de cellules cultivées en cultures denses, malgré l'irrégularité de la forme cellulaire et le bruit d'image. Cette méthode représente une amélioration significative par rapport aux approches établies, soulignant l'importance d'une adaptation spécifique des algorithmes de segmentation pour résoudre efficacement ce problème.

Chalfoun et al. [Cha+14] ont introduit une méthode de segmentation automatisée appelée FogBank, qui sépare précisément les cellules lorsqu'elles sont confluentes. Cette technique est efficace pour divers types d'images, y compris les microscopies en contraste de phase, en lumière vive, en fluorescence et les images binaires. FogBank utilise la segmentation morphologique avec deux nouvelles fonctionnalités : un regroupement d'histogrammes pour réduire le bruit d'image et un masque de distance géodésique pour détecter les formes des cellules individuelles. L'évaluation de la précision de la segmentation par rapport à des données manuellement segmentées a montré une précision d'environ 75%.

# 2 Problématique liée aux données

Après avoir soumis les pamplemousses au synchrotron, Louise a obtenu, pour chaque pamplemousse et chaque échelle de grossissement, un dossier contenant environ 3000 images au format compressé JPEG2000 d'une taille de 5 058 x 5 058 pixels et de 5 Mo. Ceci représente 15 Go de données. Chaque image représente une coupe, c'est-à-dire une image 2D. Une fois décompressée, la taille de l'image passe à 50 Mo. La taille globale devient 150 Go.

Le premier problème est lié au format JPEG2000 qui est peu supporté par des logiciels et les interfaces graphiques dont QT par exemple. Il va, donc, falloir convertir les images dans un format plus répandu comme le tiff par exemple.

En plus de pouvoir naviguer dans les coupes 2D, il va falloir convertir toutes ces images 2D en une image 3D pour obtenir une meilleure visualisation du pamplemousse et permettre des calculs géométriques en 3D.

Lors du passage aux rayons x, les pamplemousses sont empilés dans un tube et calés avec des matériaux type papier ou mousse. Sur une image 2D, on obtient une image en niveau de gris où le niveau des pixels reflète la densité du matériau. Or, le tube et les matériaux de calage apparaissent en plus du pamplemousse. Une segmentation pour parvenir à éliminer les pixels liés au tube et au calage est nécessaire.

Enfin, la gestion de l'énorme taille des données et du grand besoin en mémoire RAM est primordiale. Louise a fait une demande de financement d'un matériel informatique disposant de 160 Go de RAM. Le laboratoire, qui possède le synchrotron, lui a même conseillé d'aller jusqu'à 1 To de mémoire RAM. Le montant pour financer un tel investissement pour 160 Go s'élève à 40 000 €. Ne possédant pas de matériel aussi puissant, il peut-être envisageable soit de gérer la mémoire en fractionnant la mise en mémoire soit en réduisant la taille des données.

# 3 User Stories

## 3.1 La visualisation d'images 2D d'agrumes

Louise, notre cliente chercheuse, nous a fourni des coupes du synchrotron convertit au format JPEG2000 pour des raisons de taille de données. Cependant, le format .jp2 est mal supporté par les bibliothèques graphiques comme par exemple QT. Pour pouvoir les visualiser, il est nécessaire de les convertir dans un format mieux supporté tout en veillant à ne pas avoir de pertes dues à de la compression. Nous avons choisi le format TIFF, mais l'utilisateur pourra choisir PNG ou JPG. Même sur un grand écran, il est difficile d'afficher une image de taille 5058 x 5058 pixels. Par exemple, l'image au format TIFF a une taille de 50 Mo. Il est nécessaire de procéder à une réduction de la taille de l'image afin d'obtenir une prévisualisation. L'utilisateur doit pouvoir aussi naviguer dans les milliers de coupes de manière simple et rapide. Il doit pouvoir prévisualiser l'image souhaitée. L'utilisateur peut accéder à une fenêtre de visualisation 2D d'images qu'il aura choisies.

### 3.2 Création d'un volume à partir de segmentations de coupes

L'utilisateur peut charger des coupes 2D correspondant à un volume scanné qui seront segmentées afin de les "empiler" pour créer le volume prêt à être visualisé ensuite.

### 3.3 La visualisation d'images 3D d'agrumes

L'utilisateur peut visualiser un volume 3D depuis un fichier chargé dans différents formats (.tif, .tiff, .nrrd et .dcm) via une fenêtre de rendu. Cette fenêtre permet une interaction avec le volume grâce à la souris.

### 3.4 L'extraction automatique de la densité de la peau le long de normales à la surface de l'agrume

L'utilisateur doit pouvoir obtenir un profil de densité de la peau de l'agrume depuis l'extérieur du fruit vers le centre le long de la normale au point extérieur. Il doit pouvoir choisir les coordonnées du voxel de départ, soit en les tapant directement soit en cliquant avec la souris sur l'image 3D.
L'utilisateur doit pouvoir choisir deux modes d'extraction de la densité. La première est de partir du point de départ et de récupérer toutes les densités en se dirigeant vers le centre de l'image. La seconde est toujours de partir du point de départ, mais en suivant la direction opposée à la normale de ce même point.

### 3.5 La segmentation des régions d'intérêt à différentes échelles

L'utilisateur a la possibilité de choisir des régions d'intérêt selon la densité de la peau.

## 4 Organisation du travail

### 4.1 Planification

Voici les deux diagrammes de Gantt de notre projet réalisé grâce au logiciel GanttProject :

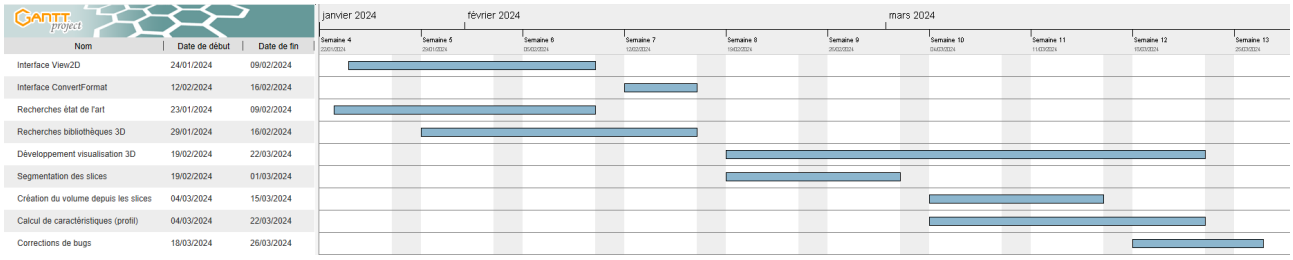


FIGURE 3 – Diagramme de Gantt prévisionnel

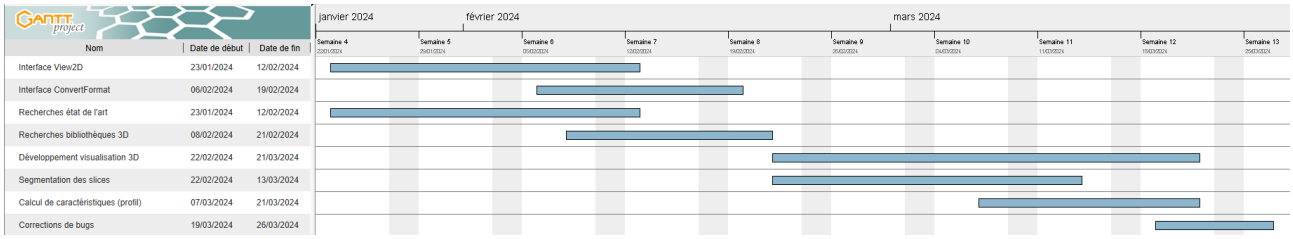


FIGURE 4 – Diagramme de Gantt effectif

En comparant les deux diagrammes, on remarque que l'essentiel de la planification a été relativement respecté. On retrouve quelques écarts de programmation pour les recherches, view2D, convertFormat et View3D mais ceux-ci sont acceptables puisque nous avons réussi à les implémenté sans perdre trop de temps. La partie segmentation de coupes a été commencée mais ensuite abandonnée au profit des calculs de densité et de profil afin de terminer ces deux dernier à temps. De plus, certains points comme les calculs de caractéristiques dans des zones d'intérêt ou la création du volume à partir de coupes n'ont pas été implémenté à temps.

## 4.2 Méthode de travail

Pour réaliser notre projet, nous avons adopté la méthode Scrum. En effet, c'est le cas puisqu'il nous a été demandé de travailler de cette manière en raison de l'efficacité, la flexibilité et la popularité de cette méthode.

## 4.3 Outils de collaboration et de suivi

Étant donné que nous avons travaillé avec la méthode Scrum, il était nécessaire de lister les tâches à réaliser lors de chaque "Sprint". Pour cela, nous avons utilisé la plateforme Trello permettant de créer son propre Kanban (Figure 5). Ce dernier nous a permis de lister les différentes User Stories séparées par type, de les décomposer sous forme de tâches d'implémentation à réaliser ou à tester ainsi que de permettre aux autres membres d'être tenu au courant de l'avancée du projet.

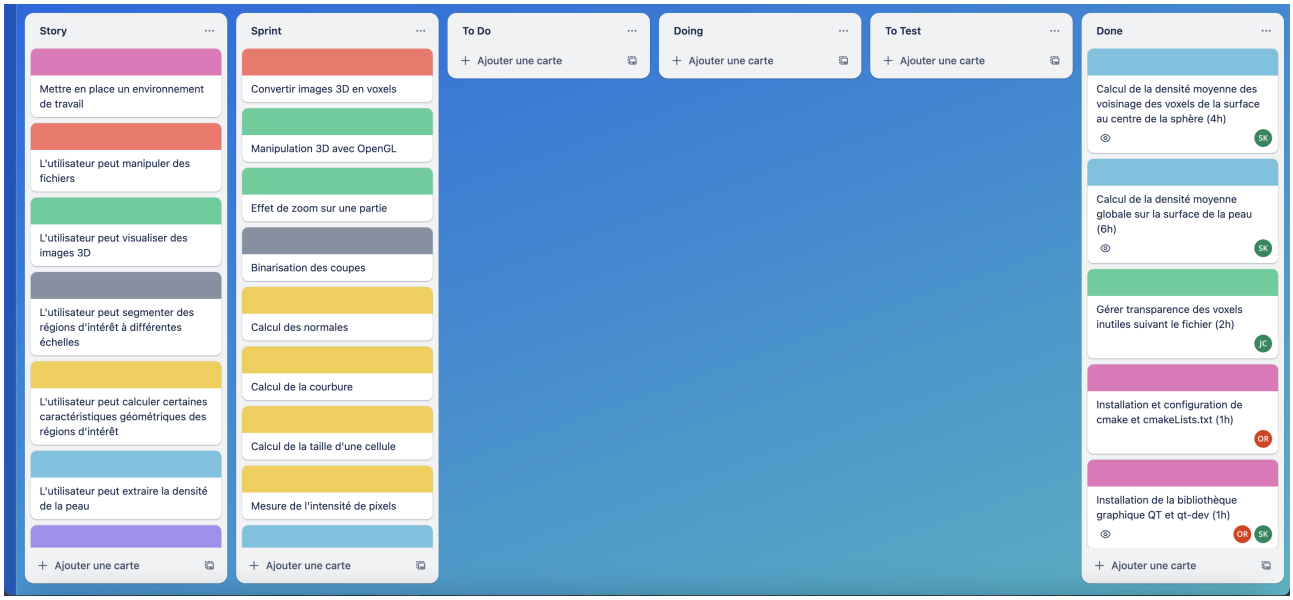


FIGURE 5 – Kanban utilisé via Trello

Nous avons également utilisé la plateforme de versioning Gitlab du CREMI afin de faciliter la collaboration et de permettre de travailler de chacun de son côté tout en étant dans le même environnement de travail.

De plus, nous avons utilisé les GOTCHA Keywords permettant à tous les membres d'être au courant de l'auteur de chaque bout de code et savoir l'état dans lequel il est (commentaire d'information basique, test, terminé, buggé, etc...).

# 5 Architecture du projet

Nous présentons dans la figure 6 le diagramme de classe de notre projet qui est composé de onze classes.

Le premier groupe de classes regroupe les classes MainWindow, View2D, View3D, ViewProfile, ToolImage, ToolImage3D et ToolGeom.

MainWindow correspond à l'interface graphique principale qui contient les différents menus.

View2D gère les menus View2D et Convert Format. La classe View2D gère l'interface graphique liée à la visualisation des images 2D et les différentes opérations réalisables sur celles-ci. Ces opérations telles que l'afffichage, la conversion, la détection de contours et la segmentation sont gérées par la classe ToolImage.

Les opérations permettant d'obtenir une image 3D à partir d'un ensemble de coupes est réalisée par la classe ToolImage3D. La classe View3D gère l'affichage des volumes 3D.

La classe ViewProfile gère l'interface graphique liée à l'obtention d'un profil de densité des pixels. Pour calculer et afficher le profil voulu, elle fait appel aux méthodes de la classe ToolGeom. Elle permet d'obtenir un profil calculé entre un point choisi par l'utilisateur et le centre de l'image et aussi le profil obtenu en choisissant de tenir compte de la normale.

TestMain, TestView2D et TestView3D font partie du second groupe. La classe TestMain gère l'ensemble des tests unitaires. La classe TestView2D contient tous les tests unitaires relatifs à la classe View2D, de même pour TestView3D.

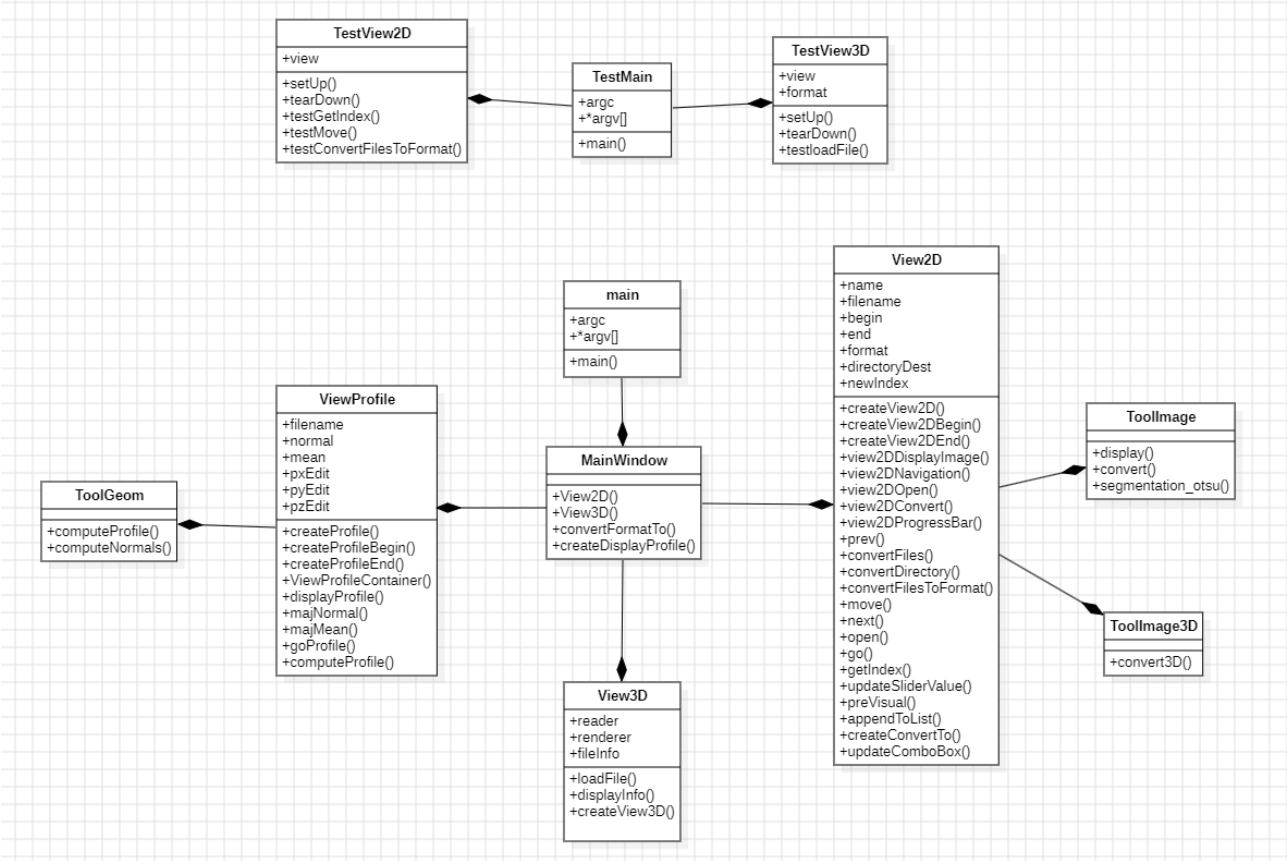


FIGURE 6 – Diagramme de classe

# 6 Implémentation

## 6.1 Pourquoi développer notre propre logiciel ?

— Ce projet de fin d'étude s'adresse à une chercheuse dans le domaine des matériaux et de l'ingénierie. Elle doit pouvoir utiliser un logiciel sans avoir à passer beaucoup de temps dans l'apprentissage de celui-ci. Elle doit pouvoir réaliser le maximum de tâches avec le même logiciel. Nous avons testé les logiciels 3DSlicer et Fiji. Ces deux logiciels sont très complets. Ils comportent beaucoup de fonctionnalités. Néanmoins, il nous est apparu qu'il est compliqué de retrouver où se trouvent le bon menu et la bonne fonction à utiliser pour réaliser une tâche précise. Ces deux logiciels ne permettent pas pour l'instant de créer des profils de densité allant d'un point précis vers le centre ou selon les normales. Ces deux logiciels sont plus adaptés aux images médicales et à leurs traitements. Nous avons, donc, chercher à construire un logiciel simple à utiliser, sur mesure, tout en un.

— Les différents ordinateurs utilisés par les participants à ce projet nous incitent à avoir un logiciel qui fonctionne aussi bien sur Windows, GNU Linux ou encore sur MacOS. Au cremi, les ordinateurs fonctionnent avec Debian Gnu Linux. Louise Le Barbenchon, la chercheuse à l'IMM, a à sa disposition un portable avec le système d'exploitation MacOS et des ordinateurs fixes équipés de Windows.

— Un argument important est le problème de la grande taille des données fournies. En effet, un exemple type de données se présente sous la forme d'un dossier commprenant environ 3000 images au format JPEG2000 de taille 5 Mo chacune. Le format JPEG2000 est un format compressé. Quand l'image est décompressée elle a une taille de 50 Mo. Pour obtenir une image 3D à partir de l'ensemble des images 2D, on obtient une taille de 3000 * 50 Mo, soit 150 Go. Nous avons créé un fichier 3D au format .nrrd de taille 1.4 Go. Nous avons utilisé 3DSlicer pour afficher ce fichier sur un ordinateur du Cremi disposant de 64 Go de mémoire RAM. L'utilisation mémoire pour ce fichier a été supérieure à 32 Go de mémoire RAM. Nous avons essayé aussi sur un ordinateur personnel disposant de 16 Go de RAM. 3DSlicer n'a jamais réussi à afficher le fichier en 3D. Il ne répondait plus et a planté. Ceci nous conduit à conclure que ces logiciels n'ont pas été conçus pour les fichiers de grande taille comme les nôtres.

— Enfin, il nous apparaît nécessaire d'envisager la possibilité de pouvoir ajouter facilement des fonctionnalités ultérieures. 3DSlicer et Fiji permettent d'ajouter des modules au logiciel de base. Nous avons choisi une

architecture logicielle qui permet d'associer une fonctionnalité principale à une classe. L'ajout se résume juste à créer une entrée dans les menus de l'interface graphique principale et à insérer la nouvelle classe dans le bon dossier.

## 6.2 Langage de programmation

Nous avons choisi le langage C++. Ce choix s'explique par le fait que toutes les bibliothèques disponibles sont codées dans ce langage. Le premier point important est que le langage C++ est libre et portable sous tous les systèmes d'exploitation existants comme Windows, Gnu Linux et MacOS. Le second point est que ce langage est compilé et non interprété. Il pourra permettre de réaliser des optimisations et de gérer la taille importante des données.

## 6.3 L'interface graphique

Comme interface graphique, notre choix s'est porté sur QT qui fournit une documentation complète et une interface simple et intuitive.
Comme seule la version 5 de QT est installée sur les ordinateurs du Cremi, nous l'avons utilisée, mais nous avons aussi essayé avec QT6 sur nos ordinateurs personnels.
Nous avons construit une interface graphique principale (Figure 7) qui contient plusieurs menus. Le premier est le menu File (Figure 8) qui contient le sous-menu "Quit". Le second est le menu "Tools" (Figure 9) qui contient les sous-menus "View2D", "View3D", "Convert Format" et "View Profile".

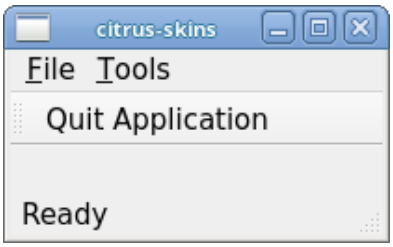


FIGURE 7 – Fenêtre principale

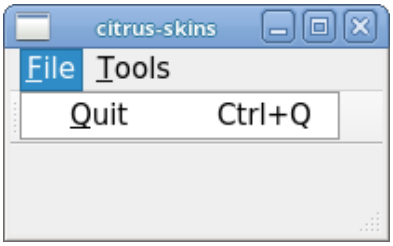


FIGURE 8 – Menu File

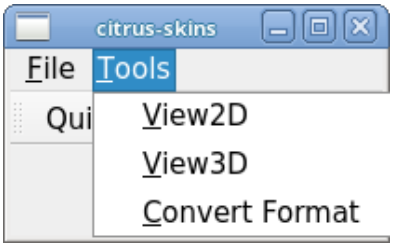


FIGURE 9 – Menu Tools

## 6.4 La visualisation d'images 2D d'agrumes

La visualisation 2D est accessible depuis le menu "Tools". Après avoir cliqué sur "View2D" ou utilisé le raccourcis clavier , une fenêtre de dialogue (Figure 10) s'ouvre dans laquelle l'utilisateur peut sélectionner un fichier. Ensuite, le widget view2D (Figure 11) s'ouvre. Cette fenêtre affiche une prévisualisation de l'image sélectionnée. L'utilisateur peut choisir une autre taille de prévisualisation en écrivant la taille dans la case et en

cliquant sur le bouton "Preview". Il peut naviguer dans les coupes en cliquant sur les boutons "Prev" et "Next". Il peut utiliser le glisseur. Le numéro de l'image s'affiche dans une case. Il peut sélectionner directement l'image voulue en écrivant le numéro de l'image et valider sur le bouton "Go". La fenêtre affiche le nom du fichier en bas. S'il veut ouvrir l'image dans sa taille d'origine, il lui suffit de cliquer sur le bouton "Open".

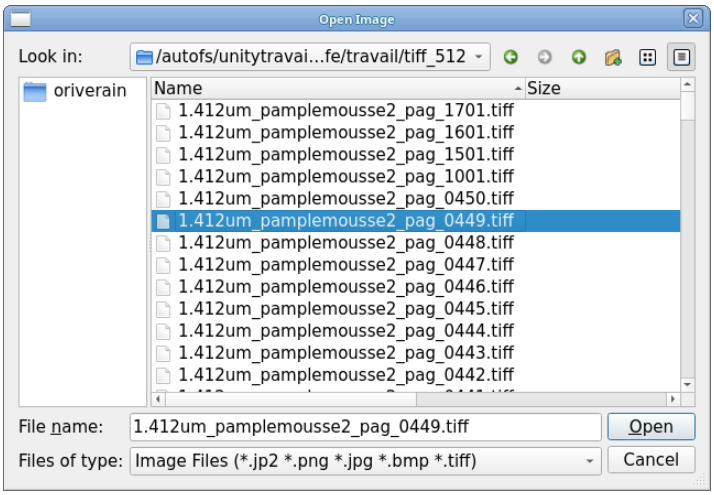


FIGURE 10 – view2D : Fenêtre de dialogue ouverture fichier

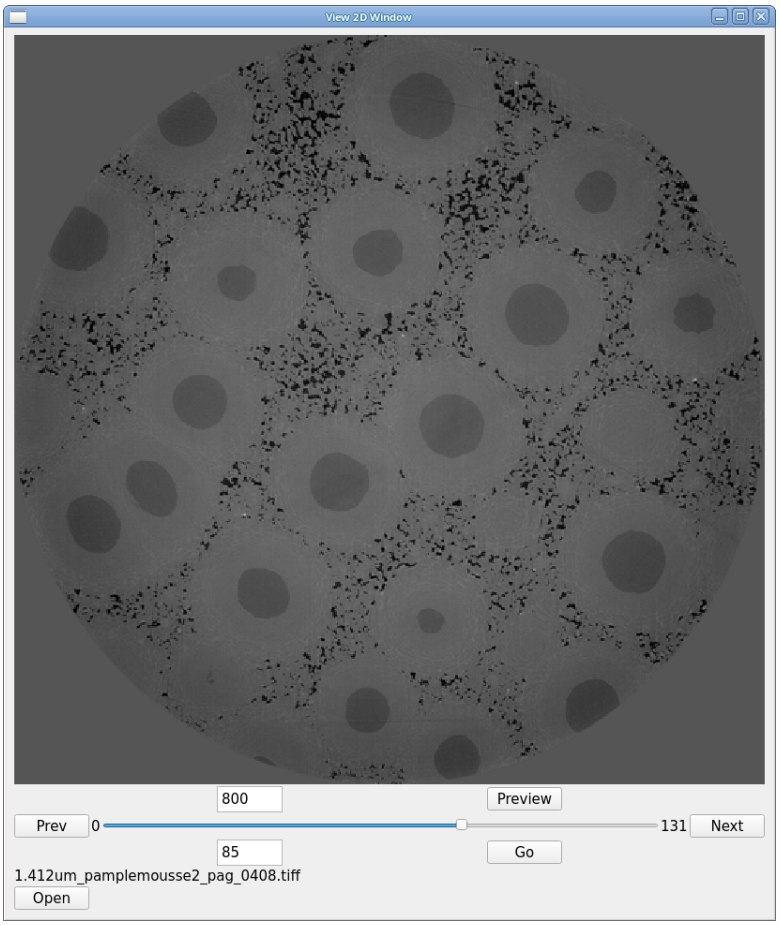


FIGURE 11 – view2D : Fenêtre de visualisation d'images 2D

Nous avons rajouté un menu "Convert Format" qui va permettre de convertir un format d'images en tiff, jpg, png ou encore nrrd. Ceci va surtout être utilisé pour convertir les fichiers au format JPEG2000. Pour la conversion d'images 2D en un format 2D, nous avons choisi d'utiliser la bibliothèque ImageMagick qui permet de convertir un grand nombre de formats d'images. Nous avons implémenté la conversion d'un ensemble d'images 2D en une image 3D. Nous avons utilisé les fonctions fournies par la bibliothèque ITK.

Après avoir cliqué sur "Convert Format" ou utilisé le raccourcis clavier, une fenêtre de dialogue 12 s'ouvre dans laquelle l'utilisateur peut sélectionner un fichier dans un répertoire. Ensuite, le widget "Convert" 13 s'ouvre. Une prévisualisation de l'image s'affiche. Il peut sélectionner le numéro de la première image et de la dernière image, le format voulu, la taille des images et le répertoire de destination. Enfin, il clique sur le bouton "Convert".

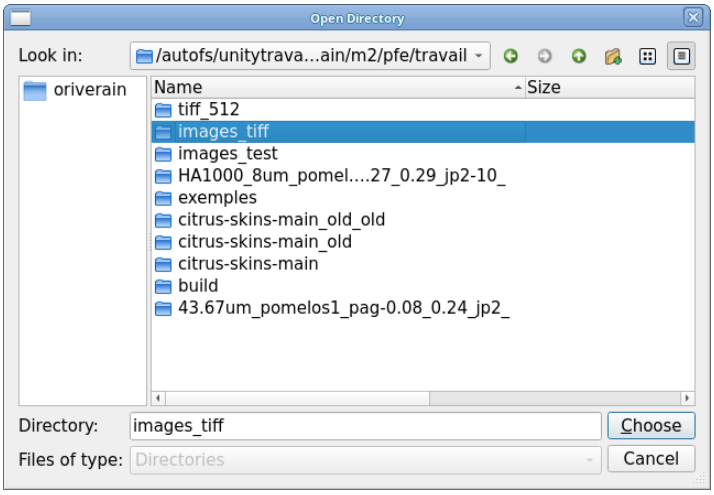


FIGURE 12 – convert : Fenêtre de dialogue ouverture fichier

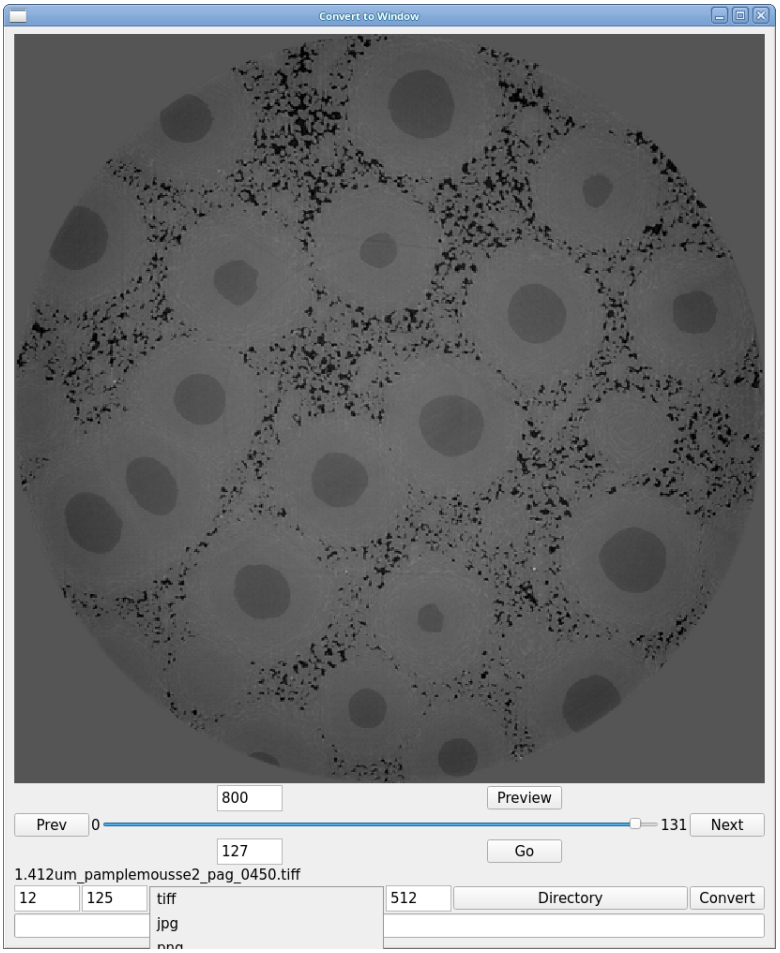


FIGURE 13 – convert : Fenêtre de visualisation d'une image 2D + Paramètres conversion

## 6.5 La visualisation d'images 3D d'agrumes

Nous avons choisi la bibliothèque VTK puisqu'elle est relativement simple d'utilisation et que 3DSlicer l'utilise donc nous avons des exemples d'utilisation. La visualisation 3D est accessible depuis le menu principal

via l'onglet "Tools" puis "View3D". Comme pour la visualisation 2D, un explorateur de fichiers s'ouvre afin de pouvoir choisir le fichier que l'on souhaite visualiser.

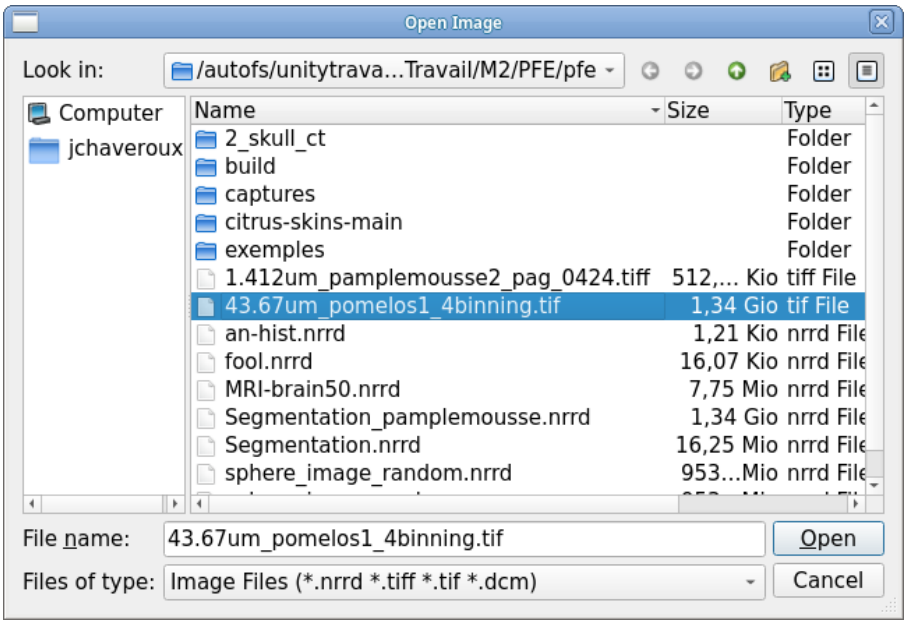


FIGURE 14 – Sélection du fichier à charger

Les formats de fichiers compatibles avec cette visualisation sont les formats suivants : .nrrd, .tiff, .tif et .dcm. La fenêtre de visualisation s'ouvre ensuite affichant le volume 3D sur un fond blanc, celui-ci peut être manipulé dans l'espace en utilisant la souris. Ces deux fonctionnalités sont directement intégrées grâce à VTK, c'est également le cas de quelques raccourcis utilisables depuis cette fenêtre.

Des informations sont également affichées en bas à gauche de la fenêtre de rendu telles que les dimensions 3D du volume (Dimensions) et l'intervalle de valeurs des voxels (Scalar Range).

Le volume présenté dans la figure 15 correspond au résultat d'une segmentation d'un volume de pamplemousse réalisé grâce au logiciel 3DSlicer. On peut observer l'intérieur du pamplemousse tranché ainsi que le tube autour qui a permis de le tenir lors des scans.

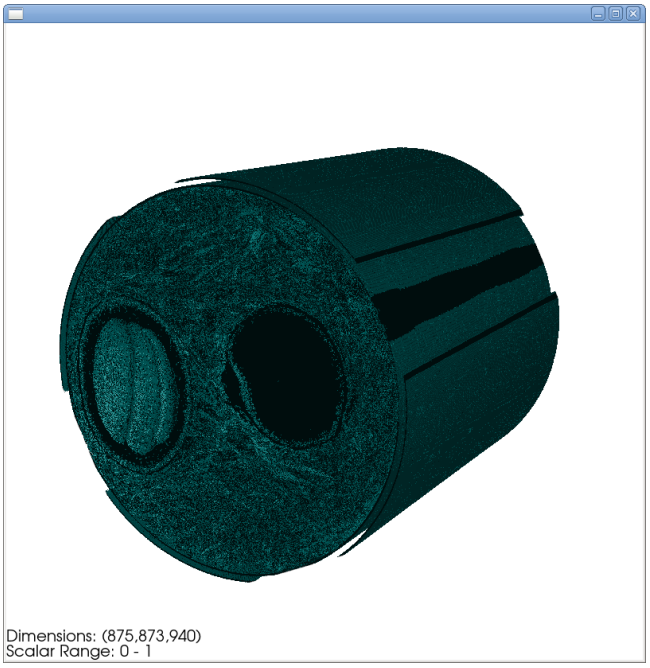


FIGURE 15 – Visualisation d'un volume segmenté de pamplemousse

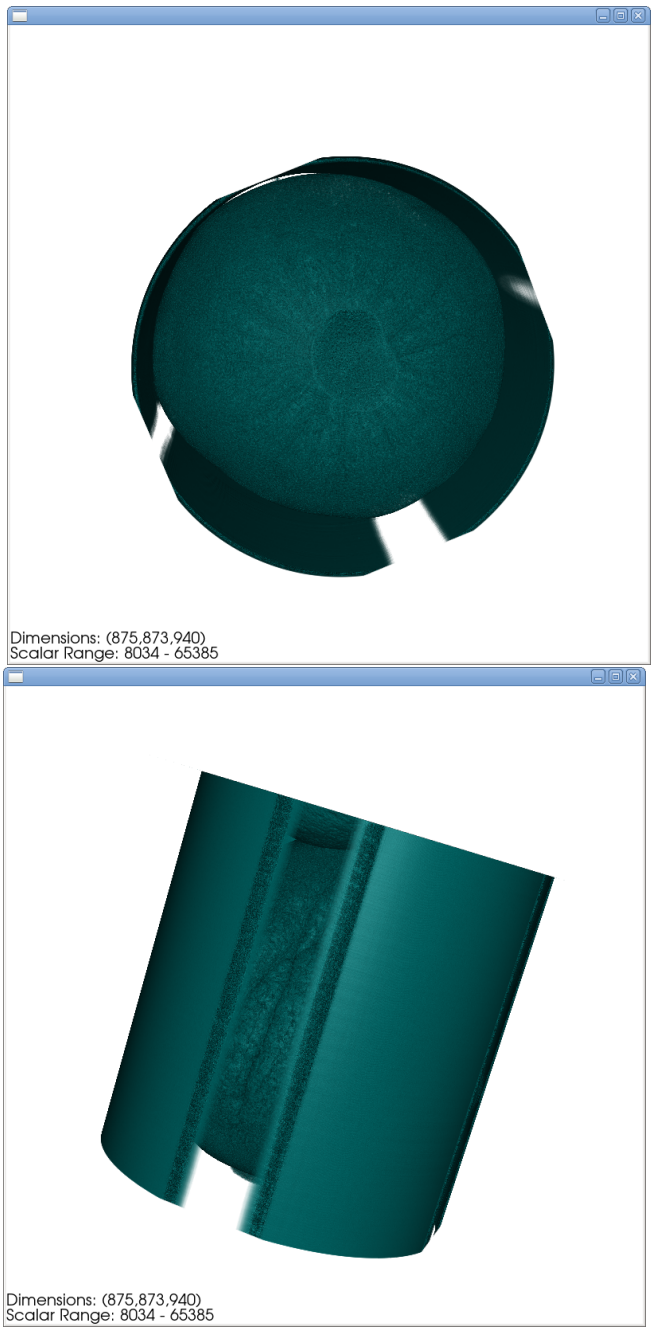


FIGURE 16 – Visualisation d'un volume de pamplemousse

Ces deux volumes présentés dans la figure 16 correspondent directement à un fichier .tif représentant le pamplemousse complet, on discerne16 bien mieux le tube autour comparé au volume précédent. Il s'agit du même fichier mais vu de deux angles différents. Étant donné qu'aucune segmentation n'ait été réalisé sur ce volume, on peut distinguer le pamplemousse de forme circulaire au milieu.

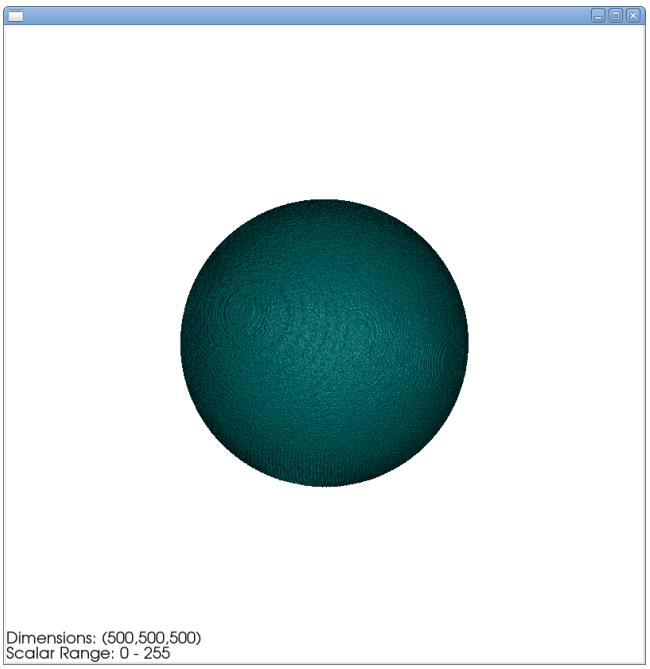


FIGURE 17 – Visualisation d'une sphère

Ce volume sphérique de la figure 17 correspond au modèle de test que nous avons utilisé pour le calcul de profil. On remarque bien avec la visualisation qu'on peut discerner les "escaliers" sur la surface de la sphère résultants de l'algorithme Marching Cubes.

## 6.6 L'extraction automatique de la densité de la peau le long de normales à la surface de l'agrume

### 6.6.1 Densité moyenne

Pour tout traitements 3D, nous avons porté notre choix sur Dgtal qui permet de calculer les normales sur des objets 3D aux formats supportés par Itk.

Après avoir converti nos images JPEG2000 en format Nrrd pour obtenir un volume 3D, nous avons entrepris d'analyser la densité des voxels dans la région du pamplemousse. Pour ce faire, nous avons d'abord calculé la densité moyenne des voxels dans la surface du pamplemousse. Cette mesure nous donne une indication de la densité globale des tissus à la surface du fruit.

Ensuite, pour explorer plus en détail la répartition de la densité à l'intérieur du pamplemousse, nous avons généré une liste des densités moyennes des voxels du voisinage, allant de la surface vers le centre du fruit. Cette liste nous permet d'appréhender comment la densité varie en fonction de la distance par rapport à la surface, donnant ainsi un aperçu de la structure interne du pamplemousse.

Les résultats de ces analyses sont illustrés dans la figure 18 pour la densité moyenne à la surface du pamplemousse et dans la figure 19 pour la liste des densités moyennes du voisinage de voxels de la surface vers le centre. Ces visualisations fournissent des informations précieuses sur la répartition spatiale de la densité des tissus dans le fruit, ce qui peut être essentiel pour diverses applications, telles que l'évaluation de la maturité du fruit ou la détection de défauts internes.

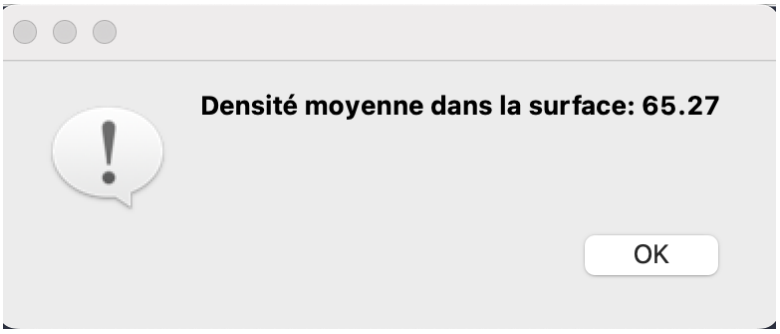


FIGURE 18 – Densité moyenne des voxels dans la surface du pamplemousse

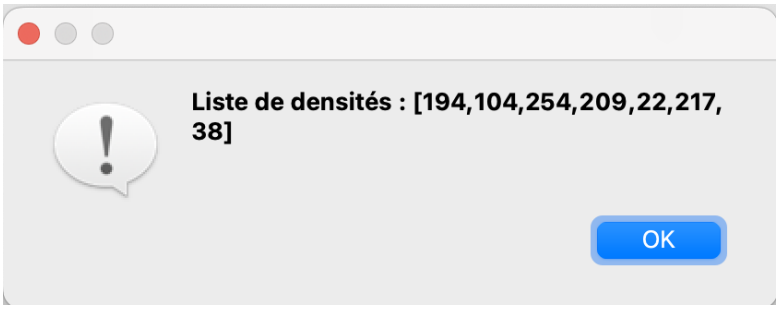


FIGURE 19 – Liste des densités moyennes du voisinage des voxels de la surface vers le centre du pamplemousse

### 6.6.2 Affichage d'un profil des densités

Lors de l'ouverture de la fenêtre "Profile" (Figure 20) un explorateur de fichiers permet de sélectionner un fichier 3D au format .nrrd ou un fichier .txt. Ce fichier .txt permet d'afficher un profil déjà calculé précédemment et donc d'éviter de recalculer un profil. Pour calculer et afficher le profil des densités, nous avons créé une fenêtre qui permet à l'utilisateur de renseigner manuellement le point de départ du calcul de densité. La possibilité de cliquer sur l'image 3D avec la souris n'a pas pu être réalisée par manque de temps. L'utilisateur peut sélectionner s'il veut que le calcul prenne en compte la normale et s'il veut la densité moyenne. Le bouton "Profile" lance le calcul du profil et son affichage.

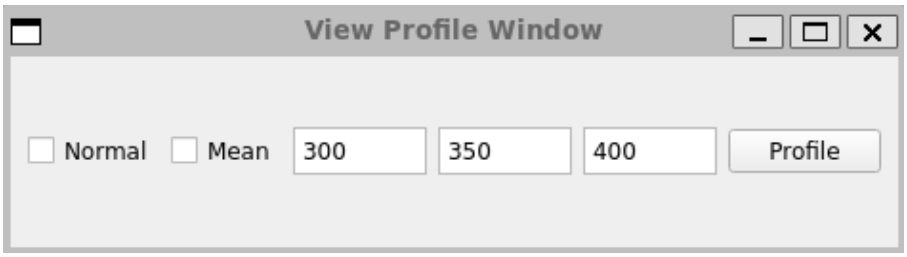


FIGURE 20 – Fenêtre Profile

Le calcul du profil et de la normale sont réalisés par la classe ToolGeom.

**Algorithme 1 :** Calcul du profil

**Entrées :** nom du fichier image 3D, booléen normal, point
**Sorties :** fichier texte
initialisation ;
chargement du fichier au format .nrrd avec DGTal et Itk ;
calcul du centre de l'image ;
**si** *normal* = *vrai* **alors**
  calcul de la normale normalisée au point et du nombre de pas ;
**sinon**
  calcul de la pente normalisée de la droite entre le point et le centre et du nombre de pas;
**fin**
calcul des coordonnées x,y,z des points par la droite paramétrique ;
récupération de la densité du pixel correspondant aux coordonnées x,y,z ;
écriture dans un fichier au format txt des coordonnées x,y,z et de la valeur du pixel ;

**Algorithme 2 :** Calcul de la normale

**Entrées :** image 3D, point
**Sorties :** vecteur normalisé de la normale
initialisation ;
calcul de la surface avec DGTal ;
calcul des coordonnées x,y,z de la normale au point ;

L'affichage du profil est effectué dans la classe ViewProfile en prenant en entrée un fichier .txt créé par le calcul du profil. L'affichage utilise les modules internes à QT pour afficher une courbe avec le niveau du pixel en ordonnée et les points se situant entre le point de départ et celui d'arrivée.

Comme exemple, nous avons créé une sphère (Figure 21) en essayant de reproduire l'évolution de la densité du pamplemousse. Nous affichons le résultat du profil (Figure 22).

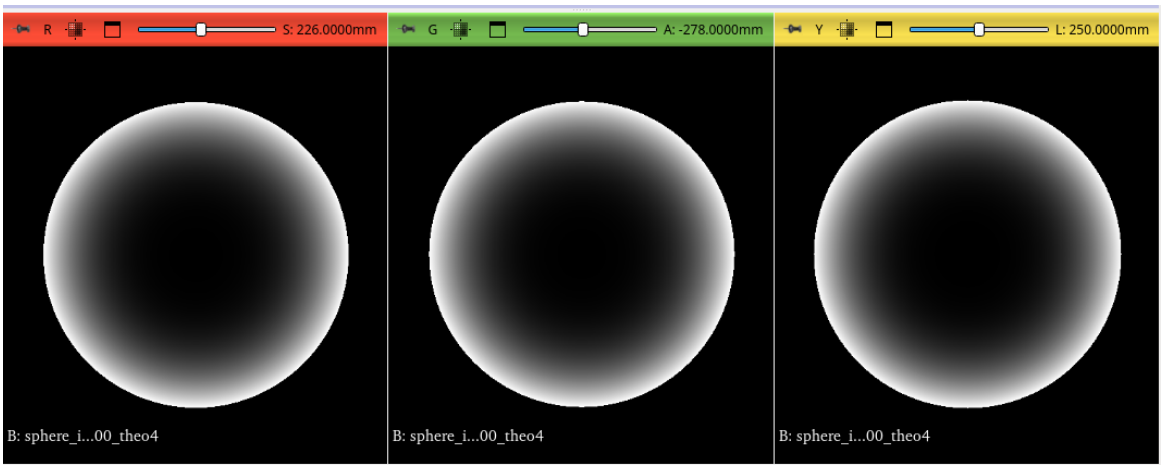


FIGURE 21 – Vue de la sphère

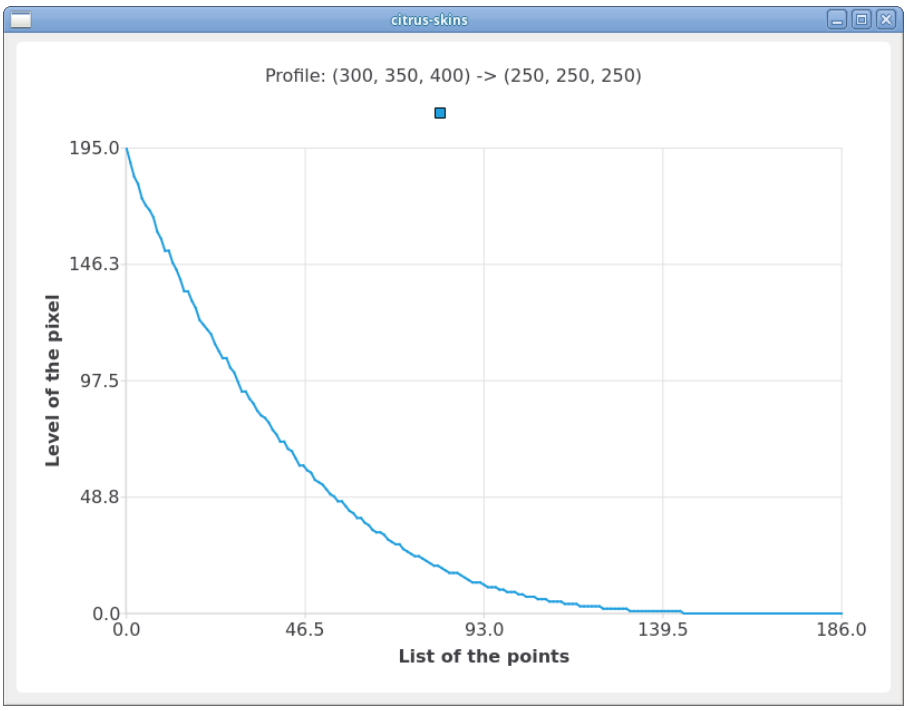


FIGURE 22 – Vue du profil obtenu

## 6.7 La segmentation des régions d'intérêt à différentes échelles

Nous avons sélectionné la même bibliothèque que celle utilisée par 3DSlicer. Il s'agit de ITK. Cette partie n'a pas été implémentée.

## 6.8 Binarisation des images par la méthode d'Otsu

Afin de pouvoir créer les volumes 3D à partir de coupes, il faut d'abord segmenter les coupes afin de garder uniquement les pixels correspondant au pamplemousse. Il faut donc enlever les pixels tout autour afin d'obtenir un volume 3D correct à la fin. Cependant, nous avons commencé mais nous n'avons pas eu le temps de finir la segmentation par 3DSlicer ou bien par la méthode d'Otsu afin de binariser les images.

Dans la fonction "segmentation_otsu" de ToolImage, nous avons d'abord visualisé l'histogramme de l'image en niveaux de gris avec la valeur du seuil optimal indiquée (Figure 23). Cette représentation graphique nous offre un aperçu de la distribution des intensités des pixels dans l'image (Figure 24). En se basant sur cet histogramme, nous avons ensuite appliqué la méthode de segmentation d'Otsu pour déterminer un seuil optimal de binarisation. Cette méthode recherche le seuil qui minimise la variance intra-classe et maximise la variance inter-classe, ce qui permet de séparer les pixels en deux classes distinctes : le premier correspondant au fond et le second aux objets d'intérêt.

Après avoir calculé le seuil optimal, nous avons utilisé cette valeur pour segmenter l'image en niveaux de gris, produisant ainsi une image binaire où les pixels sont classés comme appartenant soit au fond, soit aux objets d'intérêt. Cependant, notre objectif était de segmenter spécifiquement le pamplemousse dans l'image. Pour cela, nous avons créé un masque circulaire autour du pamplemousse, en utilisant le plus petit cercle englobant trouvé à partir des contours détectés par la méthode de contour externe. Ce masque circulaire a été appliqué à l'image binaire pour conserver uniquement les pixels à l'intérieur du pamplemousse, éliminant ainsi tout le reste de l'image.

Enfin, l'image segmentée résultante (Figure 25), représentant uniquement le pamplemousse sur un fond noir, a été affichée pour une visualisation facile. Cette approche de segmentation permet de focaliser l'analyse sur la région d'intérêt spécifique de l'image, en éliminant les distractions potentielles du fond.

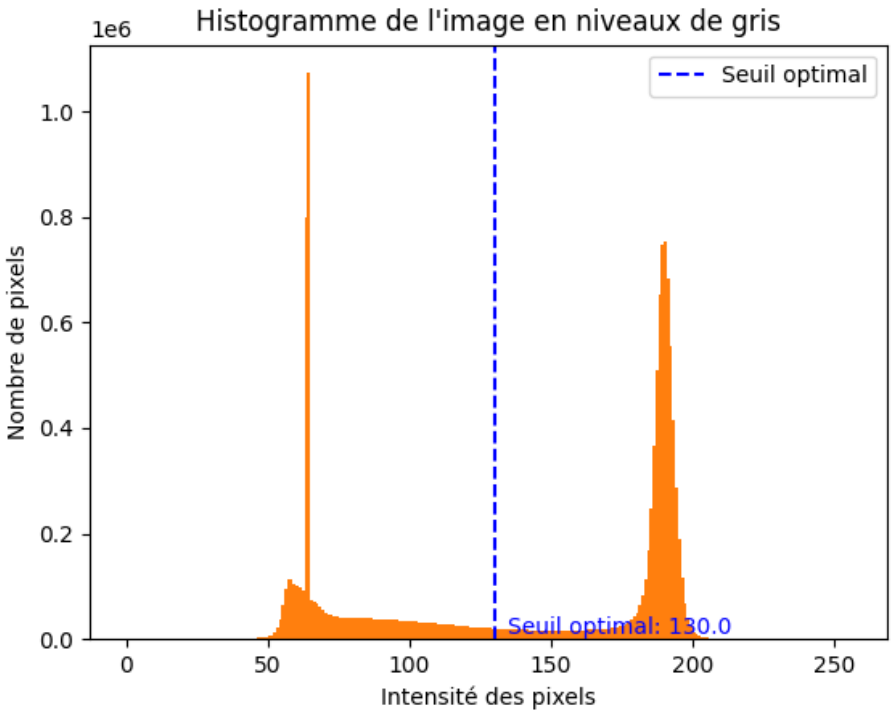


FIGURE 23 – Histogramme de l'image en niveaux de gris

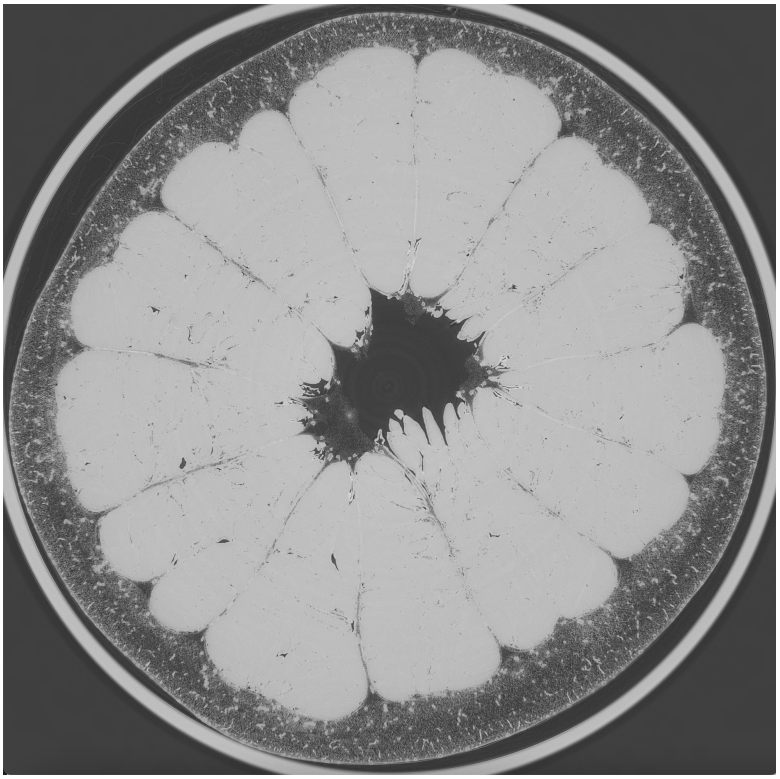

FIGURE 24 – Visualisation de l'image en 2D

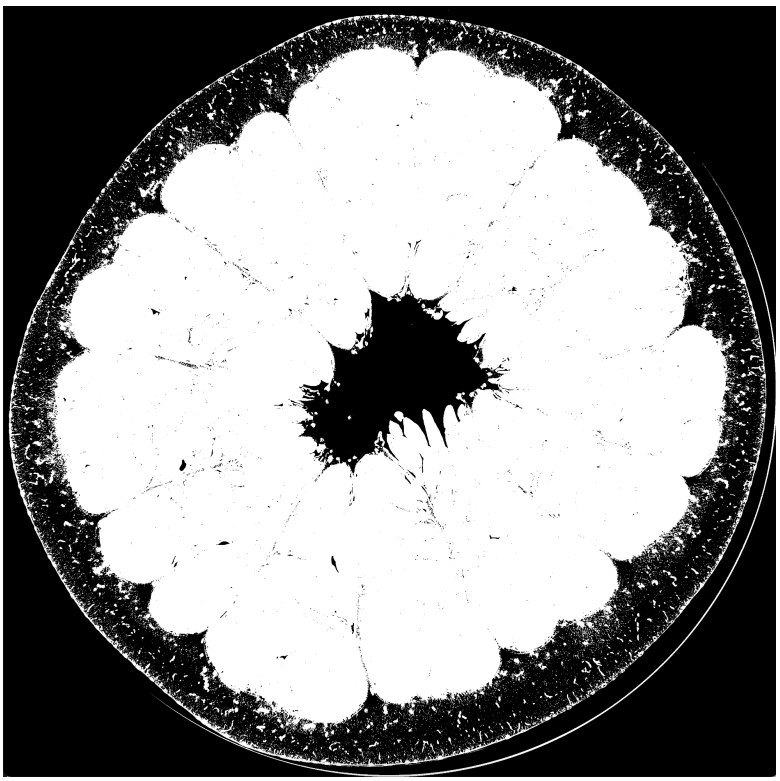

FIGURE 25 – Visualisation de l'image segmentée

# 7 Tests

Pour les tests, nous utiliserons la bibliothèque CppUnit étant donné que nous codons en C++ dans ce projet et qu'il s'agit d'une bibliothèque très utilisée.

## 7.1 Tests unitaires

### 7.1.1 View 2D

**int getIndex(QString filename)**

**Objectif :** vérifier que la fonction renvoie l'index correct.
**Donnée de base :** créer une liste de fichiers.
**Scénario négatif :**

— choisir un filename qui n'existe pas, la fonction renvoie -1

**Scénario positif :**

— choisir un filename, la fonction renvoie le bon index.

**int move(int new_index)**

**Objectif :** vérifier que la fonction sélectionne bien la nouvelle image.
**Donnée de base :** créer une liste de fichiers.
**Scénario négatif :**

— choisir un index négatif, la fonction renvoie -1
— choisir un index supérieur à l'index maximum possible, la fonction renvoie -1

**Scénario positif :**

— choisir un index correct, la fonction renvoie le nouvel index

**int convertFilesToFormat(int begin, int end, QString format, QString directoryDest)**

**Objectif :** vérifier que la fonction sélectionne bien le bon intervalle.
**Donnée de base :** créer une liste de fichiers et plusieurs possibilités d'intervalles.
**Scénario négatif :**

— choisir un begin négatif et un end corrrect, la fonction renvoie -1
— choisir un begin correct et un end négatif, la fonction renvoie -1
— choisir un begin supérieur au max et un end correct, la fonction renvoie -1
— choisir un begin correct et un end supérieur au max, la fonction renvoie -1
— choisir un begin supérieur au max et un end supérieur au max, la fonction renvoie -1

**Scénario positif :**

— choisir un begin correct et un end correct, la fonction renvoie 0

### 7.1.2 View 3D

**int loadFile(string format)**

**Objectif :** vérifier que la fonction traite uniquement les formats compatibles.
**Donnée de base :** format récupéré du fichier chargé.
**Scénario négatif :**

— choisir un format non compatible, la fonction renvoie 1.

**Scénario positif :**

— choisir un format compatible, la fonction renvoie 0.

## 7.2 Tests interface graphique

Étant donné que notre logiciel comporte diverses interfaces graphiques, il est difficile de réaliser certains tests unitaires habituels pour vérifier le bon fonctionnement de celles-ci. Pour cette raison, il faut réaliser ce qu'on appelle du "Monkey Testing" qui correspond à faire tout ce qui est possible de faire lors de l'exécution du programme pour vérifier que le logiciel ne se comporte pas d'une manière non souhaitable. Cela concerne donc les interactions possibles à la souris et au clavier.

### 7.2.1 View 2D

Pour la case texte relative à la taille de prévisualisation, nous avons écrit plusieurs valeurs et nous avons vérifié que l'image obtenue était bien l'image voulue et à la taille correcte.

Pour la case texte définissant le numéro de la coupe à afficher, nous avons écrit une valeur et vérifiée que l'image affichée était correcte. Nous avons, aussi, écrit des valeurs en dehors de l'intervalle et avons vérifié que l'image affichée ne changeait pas.

Nous avons cliqué sur les boutons "Prev" et "Next" et nous avons vérifié que l'image affichée était correcte et que la case texte affichait le bon numéro de l'image.

Nous avons manipulé le curseur et vérifié que la case texte affichait le bon numéro de l'image.

Nous avons actionné le bouton "Open" et vérifié que l'image correcte s'affichait dans sa taille originale.

A chaque changement d'images, nous avons vérifié que le nom de l'image affichée en bas était correct.

### 7.2.2 Convert format

Cette fenêtre se sert en partie de view2D, donc nous avons fait les mêmes tests que précédemment.

Nous avons fait plusieurs tests en changeant les valeurs de begin, end, du format voulu, de la taille et du répertoire de destination. Nous avons vérifié que l'image ou les images résultantes étaient bien celles attendues au bon format, de la bonne taille et enregistrées dans le bon répertoire. Nous avons aussi donné des valeurs aberrantes pour begin et end et vérifié la stabilité de notre logiciel.

### 7.2.3 Profile

Nous avons écrit des coordonnées d'un point n'existant pas dans l'image et avons vérifié que le programme ne plantait pas et avertissait l'utilisateur dans le terminal que ce point n'existait pas. Nous avons testé plusieurs points et testé le résultat obtenu.

### 7.2.4 View 3D

Étant donné que la fenêtre graphique de View3D utilise la bibliothèque VTK, il y a déjà des interactions clavier et souris intégrées. Par exemple, il est directement possible de manipuler le volume en maintenant le clic souris. De plus, il existe aussi des raccourcis claviers comme la touche 'E', entre autres, qui ferme la fenêtre graphique. En testant tout types d'interactions à la souris et au clavier et n'étant pas pré-intégrées par VTK, le programme se comporte correctement et ne cause aucun problème à l'exécution du logiciel.

## 7.3 Tests de performance

Les tests ont été effectués sur un ordinateur du Cremi équipé du système d'exploitation Debian Gnu Linux. Il est équipé d'un CPU Xeon(R) E3-1240 v6 à 3.70GHz avec 4 coeurs. Il possède une carte graphique GeForce GTX 1060 6GB et 32 Go de mémoire RAM.
Pour information, l'espace de stockage au Cremi est limité. Nous ne pouvons pas faire la conversion de 3000 images au format JPEG2000 en 3000 images Tiff. Ceci représente un besoin de stockage de 3000 * 50, soit 150 Go. Nous nous limiterons à 500 images pour un stockage de 25 Go pour une taille d'image de 5058 x 5058 pixels et 262 Mo pour une taille de 512 x 512 pixels.

| Fonction | Paramètres | Taille en pixels | Durée en s |
|---|---|---|---|
| View2D move() | tiff | 5058 x 5058 | 0.074 |
| View2D move() | tiff | 512 x 512 | 0.013 |
| View2D open() | tiff | 5058 x 5058 | 0.84 |
| View2D open() | tiff | 512 x 512 | 0.046 |
| View2D convertFilesToFormat() 2D | JPEG2000 -> Tiff 500 images | 5058 x 5058 -> 5058 x 5058 | 19 min |
| View2D convertFilesToFormat() 2D | JPEG2000 -> Tiff 500 images | 5058 x 5058 -> 512 x 512 | 20 min |
| View2D convertFilesToFormat() 3D | entrée = 500 images | 512 x 512 x 500 | 1.62 |
| View Profile computeProfile() | image sphere.nrrd | 512 x 512 x 512 | 135 |

Nous constatons bien une influence du nombre et de la taille des données sur la durée des traitements.

# Conclusion

Nous avons mis en place une interface graphique qui permet de naviguer dans les coupes 2D. L'utilisateur a aussi la possibilité de convertir un ensemble d'images dans un autre format. Ceci permet de ne plus avoir à manipuler des fichiers JPEG2000. La conversion d'un ensemble d'images 2D en un objet 3D au format nrrd nous permet de l'afficher grâce à la bibliothèque VTK. Nous avons ajouté l'extraction automatique de la densité depuis un point vers le centre et le calcul des normales à la surface de l'objet.

Nous n'avons pas réussi à éliminer le tube et les matériaux de calage des images 2D. Plusieurs points n'ont pas du tout étaient considérés comme la segmentation des régions d'intérêt à différentes échelles et la gestion de la mémoire.

Plusieurs perspectives sont à envisager. La première est de pouvoir interagir avec la vue 3D pour sélectionner une zone avec la souris. Ceci permettrait soit d'effectuer des calculs soit zoomer sur cette zone pour accéder aux données fournies avec une plus forte résolution. Une perspective serait d'obtenir une représentation 3D de la structure interne de la peau et de se déplacer à l'intérieur en immersion.

# Références


[LC87] William E. Lorensen et Harvey E. Cline. « Marching cubes : A high resolution 3D surface construction algorithm ». In : *SIGGRAPH Comput. Graph.* 21.4 (août 1987), p. 163-169. issn : 0097-8930. doi : 10.1145/37402.37422. url : https://doi.org/10.1145/37402.37422.

[WSI98] Mark D Wheeler, Yoichi Sato et Katsushi Ikeuchi. « Consensus surfaces for modeling 3D objects from multiple range images ». In : *Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271)*. IEEE. 1998, p. 917-924.

[PHK04] Steve Pieper, Michael Halle et Ron Kikinis. « 3D Slicer ». In : *2004 2nd IEEE international symposium on biomedical imaging : nano to macro (IEEE Cat No. 04EX821)*. IEEE. 2004, p. 632-635.

[YMX08] Jianjun Yin, Hanping Mao et Yongliang Xie. « Segmentation methods of fruit image and comparative experiments ». In : *2008 International Conference on Computer Science and Software Engineering*. T. 1. IEEE. 2008, p. 1098-1102.

[Zho+08] Xiaobo Zhou et al. « A novel cell segmentation method and cell phase identification using Markov model ». In : *IEEE Transactions on Information Technology in Biomedicine* 13.2 (2008), p. 152-157.

[Vij12] K Vijayarekha. « Segmentation techniques applied to citrus fruit images for external defect identification ». In : *Research Journal of Applied Sciences, Engineering and Technology* 4.24 (2012), p. 5313-5319.

[Cha+14] Joe Chalfoun et al. « FogBank : a single cell segmentation across multiple cell lines and image modalities ». In : *Bmc Bioinformatics* 15 (2014), p. 1-12.

[Dim+14] Sotiris Dimopoulos et al. « Accurate cell segmentation in microscopy images using membrane patterns ». In : *Bioinformatics* 30.18 (2014), p. 2644-2651.

[McC+14] Matthew McCormick et al. « ITK : enabling reproducible research and open science ». In : *Frontiers in neuroinformatics* 8 (2014), p. 13.

[WL17] Zhenzhou Wang et Haixing Li. « Generalizing cell segmentation and quantification ». In : *BMC bioinformatics* 18.1 (2017), p. 1-16.

[Roy+19] Kyamelia Roy et al. « Segmentation techniques for rotten fruit detection ». In : *2019 International Conference on Opto-Electronics and Applied Optics (Optronix)*. IEEE. 2019, p. 1-4.