Projet de Fin d'Etudes

Master 2 Informatique parcours Informatique pour l'Image et le Son

université de BORDEAUX

Outil d'analyse 3D de l'appareil vasculaire de plantes dans un contexte de sécheresse sévère

,
Amandine CHAUVEAU,
Thomas GUESDON,
Delphine MARTIN-DELOZANNE,
Robin MONTFERME

Clients : Gaël GUENNEBAUD et Pascal BARLA
Utilisateurs : UMR BIOGECO à INRAE Bordeaux

Janvier 2021 — Mars 2021

Table des matières

1 Introduction

Dans un contexte de changement climatique, l'équipe de chercheurs en biologie de l'UMR BIOGECO à INRAE Bordeaux étudie le dysfonctionnement hydraulique de l'appareil vasculaire au sein de diverses plantes dont les conséquences peuvent être irréversibles et conduire au dépérissement de la plante. En cas de sécheresse, le stress hydrique (résultat d'une transpiration supérieure à l'absorption d'eau) de la plante augmente, ce qui entraîne un risque d'obstruction (embolie) des vaisseaux du xylème. Les parties activement étudiées dans le cadre de cette recherche sont les stomates de la plante, organes permettant la respiration de la plante et régulant par la même occasion l'hydratation de cette dernière.

Le but de la solution proposée ici est de faciliter l'exploitation des données acquises par l'équipe de biologistes en leur offrant un outil permettant la visualisation de ces données en 3D et également capable de segmenter les données obtenues de manière semi-automatique afin de ne conserver que les zones d'intérêt. L'outil doit également permettre la prise de mesures pour notamment obtenir le volume des différentes parties segmentées afin d'obtenir plus d'informations sur les répercussions de la sécheresse.

2 État de l'art / existant

Dans le domaine de la recherche, deux logiciels de traitement d'images sont souvent employés : **Gimp**[1] et **ImageJ**[2].

ImageJ est un logiciel de traitement d'images développé en 1987 par le *National Institute of Health*. Il est écrit en Java et fonctionne donc sur la machine virtuelle Java. Il propose de nombreuses opérations basiques en traitement d'images mais permet surtout de pouvoir traiter des piles d'images (ou *image stack*) ainsi que de prendre des mesures directement sur l'image.
Le logiciel supporte plusieurs formats d'image (TIFF, PNG, JPEG...), ainsi que certains formats standards d'imagerie scientifique (DICOM, FITS), mais aussi certains formats d'image bruts.
Son code étant **open source**, il est facile de développer des plug-ins capable d'étendre les fonctionnalités de ce dernier.
De par sa facilité d'utilisation, d'installation, sa portabilité et sa modularité, ImageJ est très utilisé dans la communauté scientifique, notamment dans le domaine de la biologie.

GIMP est un autre logiciel de traitement d'images, développé depuis 1995, faisant partie du projet **GNU** diffusé sous la licence GPLv3 [3] et est donc lui aussi un logiciel *open-source*. C'est un outil puissant, présentant de très nombreuses fonctionnalités et supportant nativement de nombreux formats d'image aussi bien matricielle (ex. PNG, JPEG) ou bien vectorielle [1] (ex. SVG). Gimp supporte aussi des fichiers peu conventionnels en entrée (ex. PDF) et en sortie (ex. fichiers sources C, HTML).
L'interface est bien plus complète et offre un grand nombre de traitements plus complexes que ImageJ. Le logiciel Gimp propose lui aussi la possibilité d'ajouter des plug-ins, mais permet en plus l'implémentation directe de fonctionnalités via une console et un langage de scripting. Contrairement à ImageJ, Gimp propose une utilisation plus générale et convient donc aussi bien à des fins artistiques que scientifiques, mais comme ce dernier, Gimp à pour avantage d'être portable, relativement simple et intuitif d'utilisation, ce qui en fait un logiciel apprécié par la communauté scientifique.

Inviwo [4] est une structure logicielle extensible permettant de prototyper facilement des applications interactives. Ce logiciel fournit un éditeur de réseau pour la conception de réseaux de flux de données, qui sont automatiquement évalués et exécutés pour produire des résultats sur un ou plusieurs processeurs de sortie (généralement un canvas).
Le flux de données dans un tel réseau va de haut en bas, et les nœuds sont appelés processeurs. Outre ces processeurs, il existe deux autres objets de première classe. Les ports, qui sont utilisés pour échanger des données entre les processeurs, et les propriétés, qui définissent l'état d'un processeur. Inviwo est écrit en C++ et est disponible sous licence BSD, ce qui permet une utilisation

1. extraction des tracés sous forme matricielle

libre dans n'importe quelle configuration, même commerciale.
Inviwo supporte donc en plus des formats d'images, des formats de données volumiques et des maillages de sommets. Il est possible d'afficher directement une pile d'images sous forme d'objet 3D. Tout comme Gimp et ImageJ, Inviwo est pensé pour permettre l'ajout de fonctionnalités via des plug-ins, et comme Gimp, il offre une console permettant l'ajout en direct de fonctionnalités via le langage de programmation Python.
Bien qu'il ne s'agisse pas d'un logiciel de traitement d'images, Inviwo offre de nombreux processeurs capables de réaliser ces diverses opérations, et de par sa modularité, il est possible d'implémenter des plug-ins utiles pour la recherche scientifique.

3 User stories

En tant qu'utilisateur, je souhaite pouvoir visualiser à partir d'images au format TIFF, la reconstruction 3D des stomates afin de savoir ce qu'il se passe à l'intérieur d'une feuille et de pouvoir quantifier les degrés d'embolie et de fuite stomatique lors du stress hydrique.

En tant qu'utilisateur, je veux pouvoir exploiter les données acquises. Je souhaiterais un outil capable de ne conserver que les parties intéressantes pour mes recherches. C'est-à-dire pouvoir segmenter mes données en plusieurs parties : une partie correspondant à l'intérieur de la feuille et une partie pour l'air extérieur.

En tant qu'utilisateur, je veux pouvoir obtenir les parties de matière végétale et d'air interne de manière automatique ou semi-automatique. Si elle est semi-automatique, je dois pouvoir définir des régions d'intérêt en sélectionnant et annotant de manière différente les données que je souhaite conserver et celles qui ne m'intéressent pas.

En tant qu'utilisateur, je souhaiterais pouvoir obtenir à partir des données volumiques, des mesures : du volume d'air à l'intérieur de la feuille, du volume d'air à l'extérieur de la feuille et du volume de la matière, ainsi que des ratios comme la quantité d'air intérieur par rapport au volume total de la feuille.

En tant qu'utilisateur, je veux obtenir les propriétés géométriques liées au stomate, comme le volume de l'ostiole (c.f 14.1) par exemple.

4 Besoins fonctionnels, non fonctionnels

4.1 Besoins fonctionnels

Les besoins fonctionnels correspondent aux fonctionnalités que notre module doit offrir :

- Le module doit pouvoir **charger** des piles d'images au format TIFF/TIF sous forme de volumes.
- L'utilisateur doit pouvoir **annoter** les régions d'intérêts et d'arrière-plan du volume depuis les slices 2D du volume.
- L'utilisateur doit pouvoir **modifier** les seuils conservatifs permettant de définir les différentes régions.
- L'utilisateur doit pouvoir **corriger** localement les seuils à l'aide d'un outil de peinture.
- L'utilisateur doit pouvoir **colorier** les zones d'intérêts selon les trois axes x, y et z.
- L'utilisateur doit pouvoir **gommer** les coups de peinture donnés pour corriger les erreurs et/ou réduire les éléments indésirables sélectionnés.
- Le module implémenté doit pouvoir **appliquer** un algorithme de segmentation sur le volume en entrée selon les zones d'intérêts définies.
- L'utilisateur doit pouvoir **visualiser** le résultat de la segmentation en 2D et en 3D.
- L'utilisateur doit pouvoir **distinguer** la feuille de l'air extérieur mais également l'air de la matière à l'intérieur de la feuille.
- Le module doit permettre de **quantifier** les volumes des parties segmentées afin d'obtenir le nombre de voxels de matière, d'air à l'intérieur et d'air à l'extérieur.

— L'utilisateur doit pouvoir **obtenir** un fichier texte contenant les mesures effectuées sur le volume.

4.2 Besoins non fonctionnels

Les besoins non fonctionnels précisent les qualités et exigences de qualité attendues par le module :

— Le module doit pouvoir être **compatible** avec différents systèmes d'exploitation (Linux, Windows).
— Le module doit être **simple d'utilisation** et doit permettre d'obtenir un résultat en quelques clics.
— Le module doit être **ergonomique** et pouvoir être utilisé par des utilisateurs travaillant dans des domaines autres que l'informatique.
— Le module doit être **performant** et fournir des réponses rapides à chaque interaction. Les temps de réponse et de chargement des données doivent être minimaux.
— Les résultats obtenus doivent être **fidèles** à la réalité.
— Le code de l'application doit pouvoir être **accessible** et **modifiable** librement.
— Le module doit être **robuste** et garantir un fonctionnement stable même en présence de données complexes.

5 Choix logiciels

5.1 Framework choisi

Le projet étant prévu sur une courte durée, il était nécessaire de pouvoir rapidement mettre en place un environnement de travail commun. Pour cela, les clients ont proposé de baser le travail sur des logiciels de visualisation volumique open-source permettant l'ajout de fonctionnalités sous forme de modules ou plug-ins. Ont ainsi été étudiés les frameworks suivants :
— 3DSlicer [5]
— Inviwo [6]
— Voreen [7]
— Volview [8]
— Ogles [9]
— ITK-Snap [10]
— ImageVis3D [11]
Parmi cela, **3DSlicer, Inviwo et Voreen** ont été retenus.
Le choix s'est fait sur le type de licence, la présence de documentation, la date de la dernière version, le statut de développement de l'application (en cours ou arrêté), la modularité, ainsi que le support du format de données utilisé par l'utilisateur, en l'occurrence le format **TIFF**.

Parmi les trois choix restants, **Inviwo** a été la solution retenue. Bien que Inviwo soit à la base un fork de Voreen, il offre une interface plus moderne, sous forme de réseau nodal pratique d'utilisation et offre pour les développeurs des fonctionnalités accélérant l'ajout de nouvelles fonctionnalités : génération de squelette de code, ajout automatique des nouveaux fichiers aux listes de compilation.

3DSlicer a été écarté par sa lente mise en place (la compilation était très longue, de l'ordre de l'heure, et le logiciel ne fonctionnait pas directement chez tous les membres travaillant sur le projet).

5.1.1 Type de licence

L'un des critères les plus importants était le type de licence du framework. Il fallait s'assurer qu'il s'agisse ici d'un logiciel open-source : l'utilisateur faisant parti d'une équipe de recherche, le caractère ouvert du logiciel permet de pouvoir aisément modifier et rajouter du code tout en s'assurant que l'utilisation du délivrable soit sans restriction et puisse être potentiellement partagée au sein de la communauté scientifique.

5.1.2 Documentation

La courte durée de ce projet a poussé l'équipe de développement à se tourner vers des frameworks munis d'une documentation de bonne qualité afin de réduire le temps consacré à la compréhension du code déjà présent et pouvoir rapidement commencer à travailler.

5.1.3 Date de la dernière version

Afin de d'assurer que les différentes dépendances et bibliothèques utilisées par le framework ne soient pas obsolètes, l'équipe a fait le choix de se concentrer sur des logiciels mis à jour récemment et/ou fréquemment mis à jour.

5.1.4 État du développement

Il ne s'agit pas ici d'un critère essentiel, cependant, utiliser une solution en cours de développement active a permis à l'équipe de ce projet de pouvoir discuter avec les développeurs du framework afin de trouver des solutions à certains bugs ou bien d'éclaircir certaines des fonctionnalités offertes par leur logiciel.

5.1.5 Modularité

La capacité à pouvoir ajouter aisément des fonctionnalités au framework était l'un des points les plus importants pour pouvoir se concentrer sur l'implémentation du code répondant aux besoins du client et ne pas perdre du temps à ce qu'il s'intègre à la base de code existante. Des frameworks offrant un support pour l'intégration de modules ou plug-ins sont donc bien plus intéressants.

5.1.6 Support de format TIFF

L'utilisateur de ce projet souhaite pouvoir manipuler des données (piles d'images) au format **TIFF**. Les frameworks ayant un support natif de ce format de fichier étaient donc à privilégier.

5.2 Langage et bibliothèques

Les modules **Inviwo** sont codés en C++ version 17. De ce fait, le module développé dans ce projet a été codé dans ce langage de programmation. **Inviwo** dépend de nombreuses bibliothèques dont **Eigen** [12], utilisée pour certaines des fonctionnalités développées lors de ce projet. Aucune autre bibliothèque n'a été ajoutée.

6 Architecture

6.1 Module développé

Le module Inviwo développé, appelé *PFE*, contient deux processeurs : le premier appelé *Frame Painter* et le second appelé *Segmentation*. Le module est contenu dans le fichier *pfemodule.cpp*. C'est dans ce fichier que les processeurs sont enregistrés afin de pouvoir être manipulés dans un workspace d'Inviwo. Chaque processeur correspond à une classe dérivée de la classe *Processor* d'Inviwo dans laquelle les fonctions *process* et *getProcessorInfo* à réimplémenter sont déclarées. Ces classes contiennent les ports permettant la connexion aux autres processeurs ainsi que les propriétés du processeur.
La classe *FramePainter* contient les fonctionnalités de peinture permettant d'obtenir des masques utiles à la segmentation.
La classe *Segmentation* implémente l'algorithme de "random walker" permettant de réaliser la segmentation.

6.2 Création de module

La création de module Inviwo peut se faire de plusieurs manières :
— Automatiquement via l'interface graphique du logiciel Inviwo (onglet *Tools* > *Create Sources* > *Add module*/*Add processor*).
— Automatiquement en ligne de commande.

— Manuellement en créant les fichiers nécessaires.

Bien qu'il soit possible de créer un module manuellement, en créant les fichiers sources et les fichiers de configuration *Cmake*, il reste plus facile de directement passer par les outils en ligne de commande ou par l'interface graphique. L'outil se charge de créer les fichiers nécessaires. Il suffit simplement de cocher dans le CMake la case correspondant au module créé pour pouvoir le compiler.

Il est à noter qu'il n'y a besoin de créer un module qu'une seule fois, un module peut avoir autant de processeurs que nécessaire. À l'ajout d'un nouveau processeur, le fichier enregistrant les modules est automatiquement mis à jour. La démarche pour charger un module dans Inviwo, est expliquée dans le manuel d'utilisation (Annexe 15.3).

7 Gantt prévisionnel

Le gantt prévisionnel suivant a été établi (voir annexe 15). Celui-ci ne contient pas la première semaine du projet durant laquelle les tâches 1.1 et 1.2 étaient déjà en cours puisque le gantt a été établi le 08/02/2021.

Une comparaison avec le gantt effectif est disponible dans la partie 10.

8 Réalisation

8.1 Outil de segmentation

La segmentation d'image est une opération de traitement d'images qui a pour but de partitionner une image en plusieurs régions (les "objets" de l'image). Pour cela, une étiquette est attribuée à chaque pixel de l'image de sorte que les pixels ayant la même étiquette partagent certaines caractéristiques et font partie de la même région. Le but de la segmentation est de simplifier ou modifier la représentation d'une image en quelque chose de plus significatif et de plus facile à analyser. Ici, la segmentation permet de définir des régions d'intérêt et de distinguer les cellules de la plante de l'air.

L'algorithme utilisé est l'algorithme de "random walker". Cet algorithme prend en entrée un certain nombre de pixels étiquetés par un utilisateur selon qu'ils appartiennent à l'arrière-plan ou au premier plan (les régions d'intérêt). Ici, l'arrière-plan correspond à l'air et les régions d'intérêt aux cellules de la feuille. Après avoir défini les graines d'arrière-plan et de premier plan, chaque pixel de l'image non étiqueté peut être classé par la probabilité qu'un marcheur aléatoire partant d'un pixel donné arrive d'abord à l'une des graines du premier plan. Ces probabilités peuvent être déterminées analytiquement en résolvant un système d'équations linéaires. L'image est modélisée sous forme de graphe, dans lequel chaque pixel correspond à un nœud qui est relié aux pixels voisins par des arêtes, et les arêtes sont pondérées pour refléter la similarité entre les pixels.

Ici, la segmentation va nous permettre de binariser les données : les voxels ayant obtenus une probabilité inférieure à 0,5 après la segmentation seront labellisés "air" et ceux ayant une probabilité supérieure à 0,5 auront le label "matière". L'interpolation est réalisée à partir des densités des données. Ainsi, si deux voxels voisins ont des densités très différentes, alors la segmentation leur attribuera un label différent à chacun.

L'implémentation de cet algorithme a été réalisée en s'inspirant de la section 2 de l'article [13]. L'équation linéaire à résoudre est la suivante :

$$L_u p_u = -R^T p_m \tag{1}$$

où p_m est le vecteur des probabilités des graines étiquetées (les graines d'arrière-plan et de premier plan), p_u le vecteur des probabilités des graines non étiquetées et L_u et R^T des sous matrices de la matrice Laplacienne L :

$$\begin{bmatrix} L_u & R \\ R^T & L_m \end{bmatrix} \tag{2}$$

La matrice Laplacienne du graphe représentant le volume est définie comme :

$$L_{ij} = \begin{cases} d_i & \text{si i = j} \\ -\omega_{ij} & \text{si les sommets } v_i \text{ et } v_j \text{ sont adjacents} \\ 0 & \text{sinon} \end{cases} \quad (3)$$

avec $-\omega_{ij}$ le poids de l'arête e_{ij} défini comme

$$-\omega_{ij} = exp(-\beta(g_i - g_j)^2) \quad (4)$$

où β est un facteur d'échelle, et

$$d_i = \sum_j w_{ij}$$

où g_i et g_j sont les niveaux de gris des arêtes v_i et v_j

Pour résoudre l'équation linéaire (1), la méthode du gradient conjugué implémentée dans Eigen [12] a été utilisée.

Le processeur implémentant cet algorithme contient deux ports d'entrée : un pour récupérer le volume sur lequel va être appliqué la segmentation et un autre volume de même dimension indiquant les voxels étiquetés (d'arrière-plan ou de premier plan) et les voxels non marqués. En sortie de ce processeur, on obtient un volume contenant les probabilités pour chacun des voxels. Les probabilités des graines d'arrière-plan sont initialisées à 0 et celles de premier plan à 1. Pour les graines non étiquetées, leur probabilité est initialement à -1 et est ensuite modifiée par interpolation selon la méthode décrite précédemment.

8.2 Outil de peinture

La segmentation est, au départ, automatisée par des seuils permettant de définir les graines d'arrière-plan et de premier plan. Avec cette méthode, moins les seuils sont conservatifs, plus on a de chance d'obtenir des faux positifs (pixels annotés comme appartenant au premier plan, alors qu'ils appartiennent en réalité à l'arrière-plan) ou inversement des faux négatifs. L'outil de peinture permet alors de corriger localement les seuils ou d'attribuer des labels à des pixels non annotés. Le retour de la segmentation sur des slices 2D permet de visualiser les labels attribués à chaque pixel et ainsi de corriger les incohérences sur les slices 2D de peinture. Plusieurs fonctionnalités sont proposées afin d'améliorer l'ergonomie de l'outil, notamment la possibilité de contrôler la taille de la brosse et la possibilité d'annuler la dernière action. L'utilisation de l'outil est décrite dans le manuel d'utilisation en annexe (section 15.3).

8.3 Outil de mesure

Les principales mesures attendues sont le nombre de voxels correspondant au volume des parties segmentées : c'est-à-dire le nombre de voxels correspondant à la feuille et le nombre de voxels correspondant à de l'air. Plus précisément, il faut séparer la feuille de l'air extérieur pour pouvoir obtenir les volumes de la matière, de l'air à l'intérieur de la feuille et de l'air à l'extérieur de la feuille. Pour ce faire, il faut donc séparer l'intérieur de l'extérieur de la plante et distinguer l'air de la feuille à l'intérieur. L'idée est de d'ignorer une composante connexe (celle correspondant à l'air extérieur) à partir d'une graine. En partant de cette graine, on peut alors réaliser une méthode de "region growing" implémentée ici comme un parcours en largeur sur les probabilités obtenues suite à la segmentation. Pour cela, on commence par enfiler (dans une structure FIFO) la graine de départ. Ensuite, on défile le sommet en tête de la liste et on enfile les sommets adjacents à celui-ci, seulement si ces sommets n'ont pas déjà été explorés et qu'ils correspondent bien à de l'air. On réalise ces deux dernières étapes tant que la file n'est pas vide et on compte ainsi le nombre de voxels correspondant à l'air extérieur.

Pour obtenir le nombre de voxels correspondant à de l'air à l'intérieur de la feuille et le nombre de voxels correspondant à de la matière, il suffit de récupérer la structure des sommets explorés utilisée lors du parcours en largeur et pour chaque sommet non exploré de compter ceux ayant une probabilité inférieure à 0,5 et ceux ayant une probabilité supérieure à 0,5. Sachant qu'une probabilité supérieure à 0,5 va correspondre à de la matière et une probabilité inférieure à 0,5 va correspondre à de l'air.

L'ensemble des valeurs obtenues sont ensuite écrites dans un fichier. Ces valeurs sont (en nombre de voxels) : le volume total, le volume de l'air extérieur, le volume de matière dans la feuille, le volume de l'air dans la feuille, le volume total de la feuille, la proportion d'air dans la feuille et respectivement la proportion de matière dans la feuille.

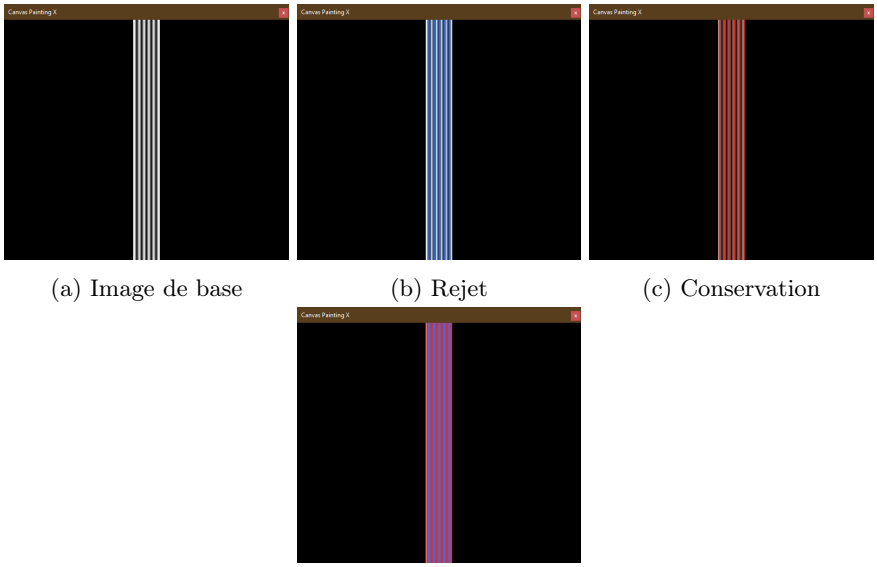
9 Tests fonctionnels

9.1 Test sur le seuillage automatique de volume

Vérification que le seuillage automatique du volume soit correct. À partir d'un volume où les voxels ont une intensité binaire (nulle ou maximale), on doit s'assurer que :

— pour un seuil minimal à 0.5, tous les voxels d'intensité nulle soient marqués comme rejetés
— pour un seuil maximal à 0.5, tous les voxels d'intensité maximum soient marqués comme conservés

Le volume testé possède 11 slices, 5 d'intensité nulle et 6 d'intensité maximale.



(a) Image de base (b) Rejet (c) Conservation

(d) Rejet + conservation

FIGURE 1: Seuillage automatique sur volume de test

Sur la figure (1b) on voit que les 5 bandes noires correspondant aux valeurs nulles ont été rejetées (en bleu), tandis que sur la figure (1c) les 6 bandes blanches de valeurs maximales sont conservées (en rouge) [2].

9.2 Test sur l'inter connexion des canvas

Vérification que les modifications effectuées sur le canvas d'un des axes soient bien reportées sur les deux autres axes. Pour cela, on peint d'une couleur une partie de la première slice de l'axe X, ce qui doit faire apparaître sur les slices de l'axe Y et Z, une ligne de la même couleur. La figure 2 montre que le résultat correspond bien à ce qui est attendu.

2. On notera que l'effet de "bavement" des couleurs est dû à la résolution des images ainsi qu'au filtrage automatique qui leur est appliqué

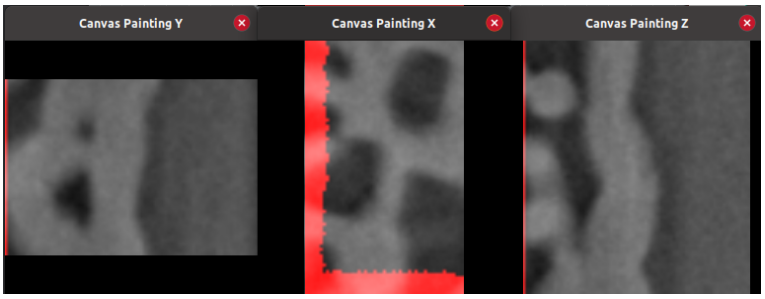


FIGURE 2: Retour peinture interconnexion des axes

9.3 Test sur l'annotation des voxels

Vérification que l'annotation des voxels à l'aide de l'outil de peinture modifie bien la sortie de la segmentation. On vérifie également que les modifications effectuées génèrent des résultats correspondant aux résultats attendus.

— Sans utilisation de seuil : une slice est peinte en partie en bleu (air), en partie en rouge (matière). Il n'y a pas de zone indéterminée entre les deux. Le résultat affiché dans le canvas de retour de la segmentation doit correspondre exactement à la peinture. Le résultat obtenu valide ce critère comme en témoigne la figure 3.

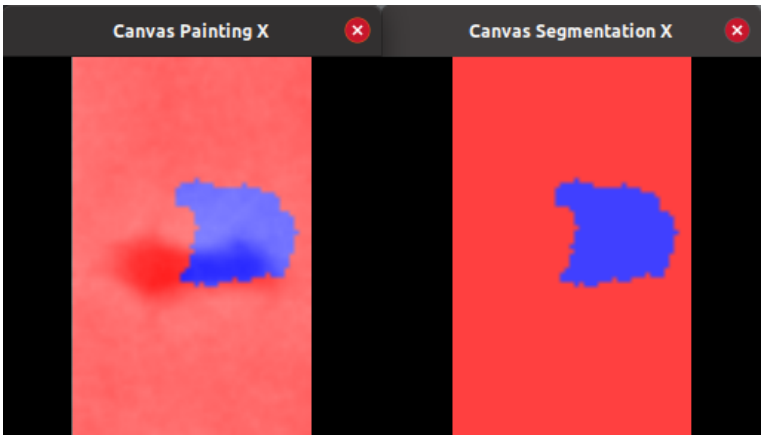


FIGURE 3: Retour de la segmentation sans zone indéterminée

— Sans utilisation de seuil : une slice est peinte en partie en bleu (air), en partie en rouge (matière), une zone indéterminée est laissée entre les deux. Le résultat affiché en retour de la segmentation doit voir la zone indéterminée répartie entre l'air et la matière. Le résultat obtenu valide ce critère comme en témoigne la figure 4.

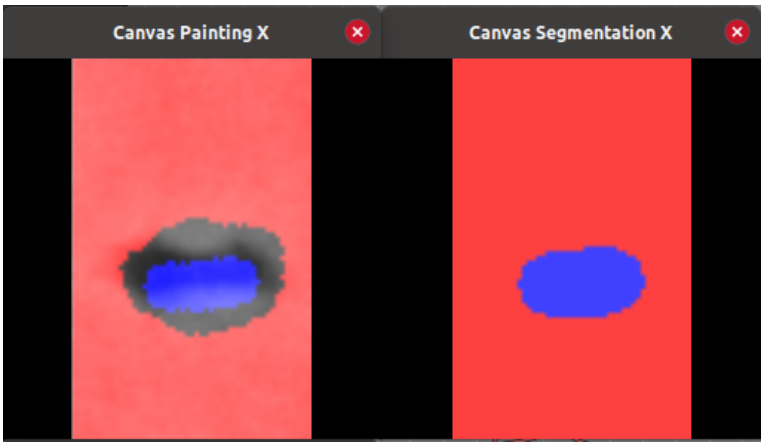


FIGURE 4: Retour de la segmentation avec une zone indéterminée

— Sans utilisation de seuil : un rectangle est peint sur une slide. 3 côtés sont peints d'une couleur, le dernier côté de l'autre couleur. L'intérieur du rectangle est indéterminé. Le résultat affiché dans le retour de la segmentation doit voir l'intérieur du rectangle majoritairement de la première couleur et à l'extérieur du rectangle, les pixels indéterminés doivent être de la couleur du côté du rectangle dont ils sont le plus proche. Le résultat obtenu valide ce critère comme en témoigne la figure 5.

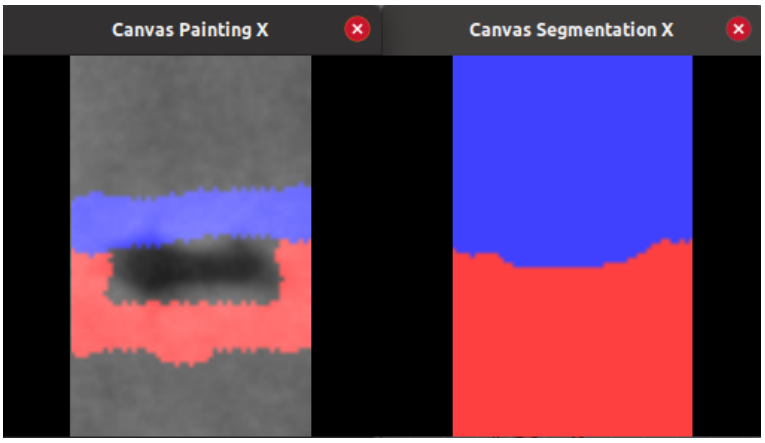


FIGURE 5: Retour de la segmentation en dessinant un rectangle dont les contours sont inégalement répartis en terme de couleur

9.4 Test sur la cohérence des mesures

Vérification que les mesures effectuées par la solution soient cohérentes avec les résultats pouvant être attendus.

— Tous les voxels sont peints en rouge. On s'attend à ce que la proportion de matière soit de 100%. Le résultat obtenu valide ce critère comme en témoigne la figure 6.

```
1 Dimensions : [       73,       81,       51]
2 Total volume : 301563 voxels
3 Outside air volume : 1 voxels
4 Volume of the matter : 301562 voxels
5 Interior air volume : 0 voxels
6 Total volume of the leaf : 301562 voxels
7 Proportion of air inside the leaf : 0%
8 Proportion of matter inside the leaf : 100%
```

FIGURE 6: Mesures obtenues sur un volume entièrement annoté comme étant de la matière

— Tous les voxels sont peints en bleu. On s'attend à ce que le proportion de matière soit de 0%. Le résultat obtenu valide ce critère comme en témoigne la figure 7.

```
1 Dimensions : [       73,       81,       51]
2 Total volume : 301563 voxels
3 Outside air volume : 301563 voxels
4 Volume of the matter : 0 voxels
5 Interior air volume : 0 voxels
6 Total volume of the leaf : 0 voxels
7 Proportion of air inside the leaf : 0%
8 Proportion of matter inside the leaf : 0%
```

FIGURE 7: Mesures obtenues sur un volume entièrement annoté comme étant de l'air

10 Projet : comparaison gantt prévisionnel/effectif

Le gantt effectif (voir annexe 16) diffère du gantt prévisionnel à partir de la semaine 4. Premièrement, les tâches *implémentation de l'algorithme de segmentation* et *intégration de l'algorithme au module* ont été regroupées en une seule tâche. Cela est dû au fait que l'implémentation de l'algorithme directement au sein d'Inviwo était envisageable au moment de commencer la tâche, contrairement à l'idée de départ qui était d'implémenter l'algorithme random walker en dehors d'un module d'Inviwo afin de le tester. Mais il a été préférable de l'implémenter directement dans Inviwo étant donné la complexité de compréhension de la manipulation des données dans Inviwo. Deuxièmement, la tâche *interaction/peinture* a demandé plus de temps que ce qui avait été prévu. Une réunion avec le client a été planifiée au mardi 9 mars 2021 afin de prioriser les tâches restantes en vue de la fin du projet. Il a été décidé de ne pas s'occuper de la partie orientation, grisée sur le gantt effectif, et de se concentrer sur la partie mesure : principalement sur les mesures volumiques des parties segmentées, mais également sur la préparation du rendu. La préparation du rendu n'avait pas été incluse dans le gantt prévisionnel. De plus, la correction de bugs et la préparation d'un environnement de travail et d'un manuel d'utilisation n'avaient pas été anticipées.

11 Difficultés rencontrées

11.1 Mise en place de l'environnement de développement

Pour ce projet, tous les membres de l'équipe de développement travaillent sous Linux, cependant les distributions diffèrent entre certains membres. Cela a pour impact un comportement différent du framework selon la distribution : les bugs présents chez certains membres ne sont pas apparus chez d'autres.

Afin que tous les membres puissent travailler de manière correcte, il a fallu s'assurer que tout le monde ait la même version du framework Inviwo. Or, certaines dépendances du logiciel étaient incompatibles avec la version du système d'exploitation installée chez certains. Il a donc fallu en premier lieu, mettre à jour les machines de ces membres afin qu'ils puissent par la suite installer et modifier le logiciel. On notera que dans le cas d'un des membres, le processus de mise à jour n'a pas pu être mené à bien, forçant ainsi une méthode de travail par *pair-programming* à distance (au vu de la situation sanitaire actuelle).

L'utilisateur de ce projet compte utiliser le logiciel et les modules implémentés sur des ordinateurs étant sous Windows. L'équipe développant sous Linux, il a fallu s'assurer que la compilation et l'exécution du logiciel ainsi que du module développé fonctionnent bien sous Windows. Le manque d'expérience dans l'équipe en terme de développement sous Windows a ralenti l'accomplissement de cette tâche.

11.2 Modification des données par un processeur

Dans le but d'implémenter l'outil de peinture permettant d'étiqueter les différentes zones des slices, il était nécessaire de modifier les valeurs obtenues en entrée du port. La première idée fut de copier les données du port d'entrée dans une image puis de donner cette image au port de sortie

dans la fonction appelée `process`. Une fonction `paint` est appelée en réponse à un événement de souris. Cependant la fonction `process` étant appelée à chaque fois que le processeur est recalculé, les modifications étaient écrasées systématiquement.

Il a alors été convenu de travailler avec un masque contenant les modifications appliquées aux images qui seraient ensuite superposées aux données en entrée. Ce masque est initialisé dans la fonction `process` uniquement s'il n'est pas encore créé ou si la dimension du port d'entrée a changé. Cela permet de ne pas l'écraser à chaque modification. Un processeur permettant de superposer deux images existant déjà dans Inviwo, il a été décidé de l'utiliser afin de ne pas ré-implémenter cette fonctionnalité. Il y a alors une sortie pour chaque masque de chaque axe. Le masque est ensuite combiné avec la slice correspondante via le processeur de mixage.

12 Résultats obtenus

Comme on peut le voir en comparant les figures 8 et 9, la segmentation permet de bien séparer les différentes régions de l'image et de facilement différencier l'air de la matière : la frontière entre ces deux parties est nettement visible. De plus, l'affichage du stomate après segmentation permet plus facilement de déterminer s'il est ouvert ou non. Une comparaison peut être faite avec la figure 10 qui montre le volume sans segmentation, avec uniquement l'application d'une fonction de transfert manuellement paramétrée pour séparer au mieux l'air de la matière. On constate que la séparation est moins nette et que l'on trouve des zones d'air à l'extérieur de la feuille qui sont labellisées comme de la matière.
De bons résultats peuvent être obtenus en modifiant uniquement les seuils. Les voxels dont la densité est au-dessus du seuil maximal sont considérés comme appartenant à la matière et ceux dont la densité est en dessous du seuil minimal sont considérés comme appartenant à l'air. Entre ces deux seuils, les pixels n'appartiennent à aucune classe et c'est la segmentation qui va permettre de les labelliser. Pour corriger le résultat obtenu à l'aide de ces seuils ou annoter des voxels sans labels, l'outil de peinture peut être utilisé. Sur la figure 11, on distingue clairement les pixels labellisés par le seuillage (rouge et bleu plus clairs) de ceux labellisés grâce à l'outil de peinture (rouge et bleu plus foncé).
Il est intéressant d'avoir un retour de la segmentation sur les slices 2D et de pouvoir les comparer avec les slices 2D originales sans peinture afin de savoir si la segmentation a mal labellisé certains des voxels ou non comme on peut le voir sur la figure 12. Si c'est le cas, il est alors possible de corriger ces erreurs avec l'outil de peinture.
Les figures 13 et 14 permettent de voir l'utilité de l'outil de peinture. Si les seuils originaux sont assez écartés (un seuil plus proche de 0 pour le minimum et plus proche de 1 pour le maximum), ou si les seuils ne fonctionnent pas très bien sur le jeu de données utilisé, il y a plus de risques d'obtenir des faux positifs ou des faux négatifs. C'est ce que nous montre la figure 13. Il faut dans ce cas corriger localement les seuils avec l'outil de peinture afin de réduire au maximum le nombre de voxels mal classés. C'est ce que nous montre la figure 14. On voit sur les slices 2D du haut que beaucoup plus de pixels ont été annotés. On peut voir l'effet produit sur la segmentation sur les slices 2D du bas et sur le volume affiché. Le résultat n'est certes pas parfait (il faudrait pouvoir annoter l'ensemble des slides pour obtenir un meilleur résultat), mais on peut constater que de nombreux voxels mal annotés précédemment ont été mieux classés. En effet, moins d'air extérieur a été confondu avec de la matière.

12.1 Screenshots

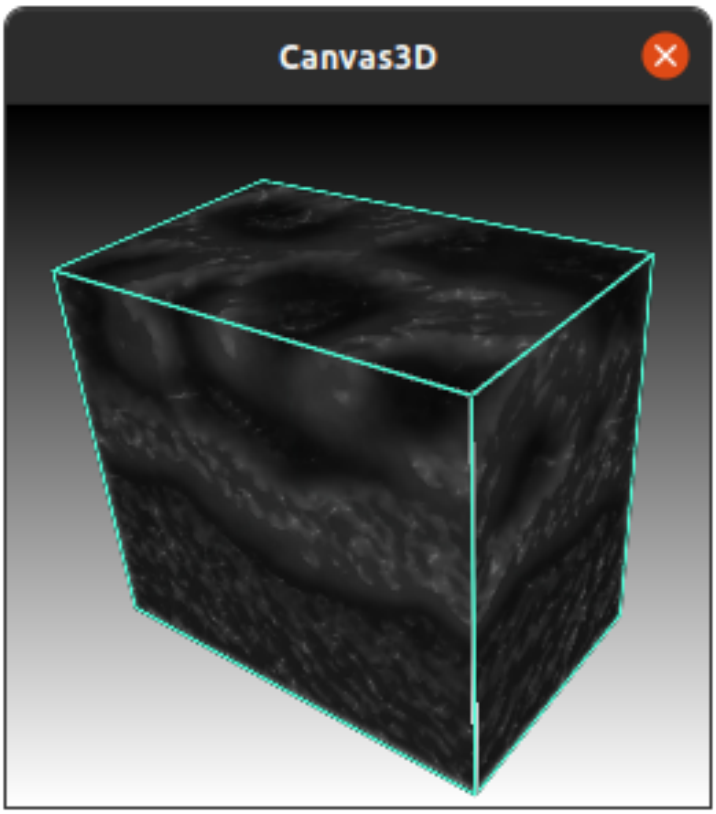


FIGURE 8: Résultat obtenu avant segmentation

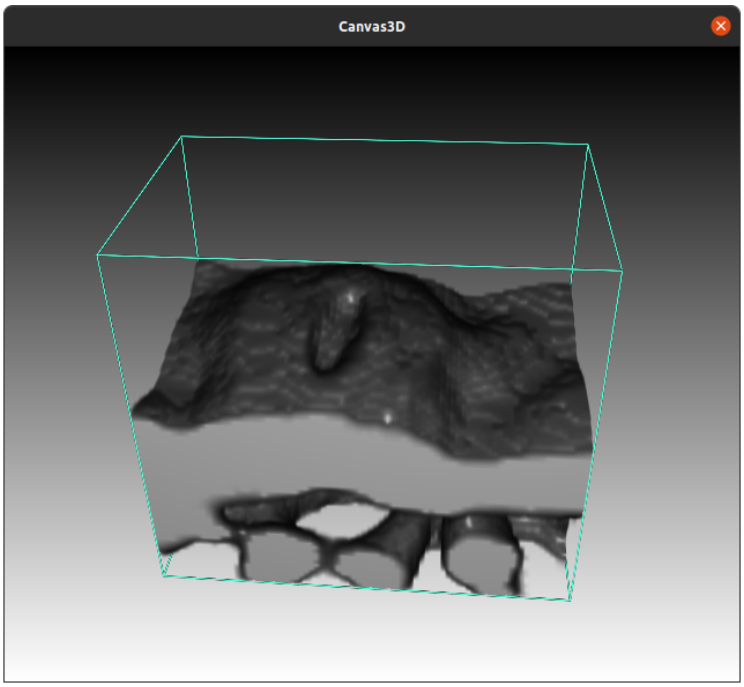


FIGURE 9: Résultat obtenu avec la segmentation

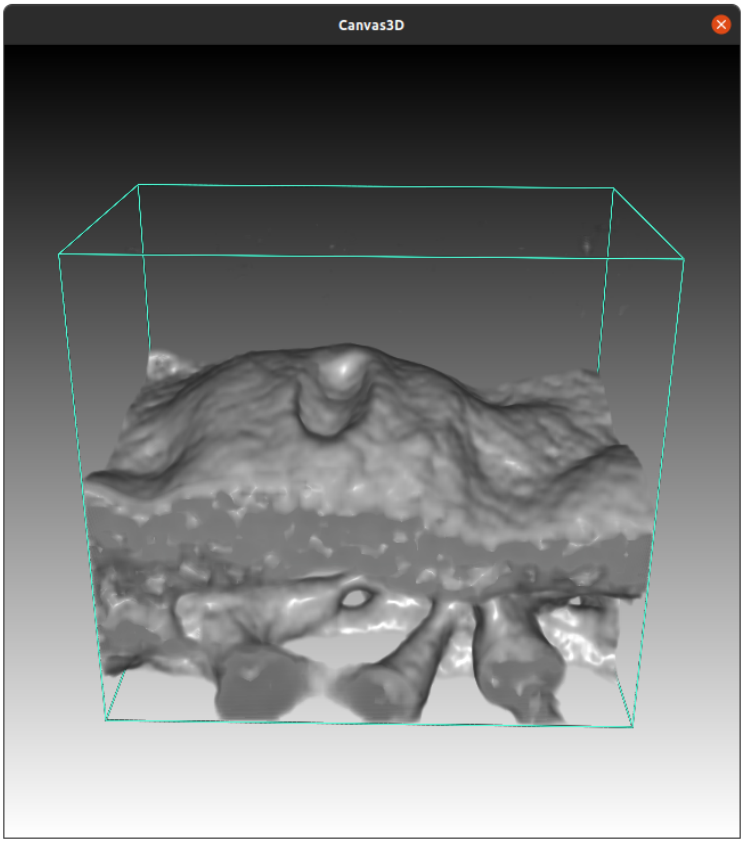


FIGURE 10: Résultat obtenu avec fonction de transfert uniquement

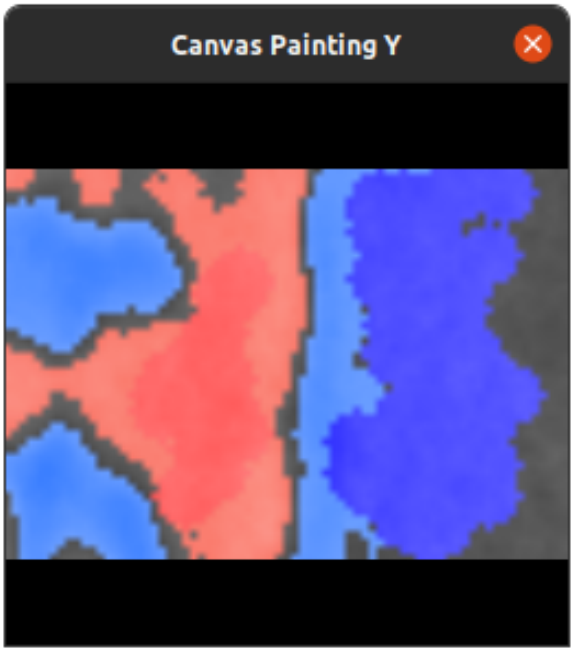


FIGURE 11: Résultat obtenu sur les canvas 2D en utilisant les seuils et la peinture pour la segmentation

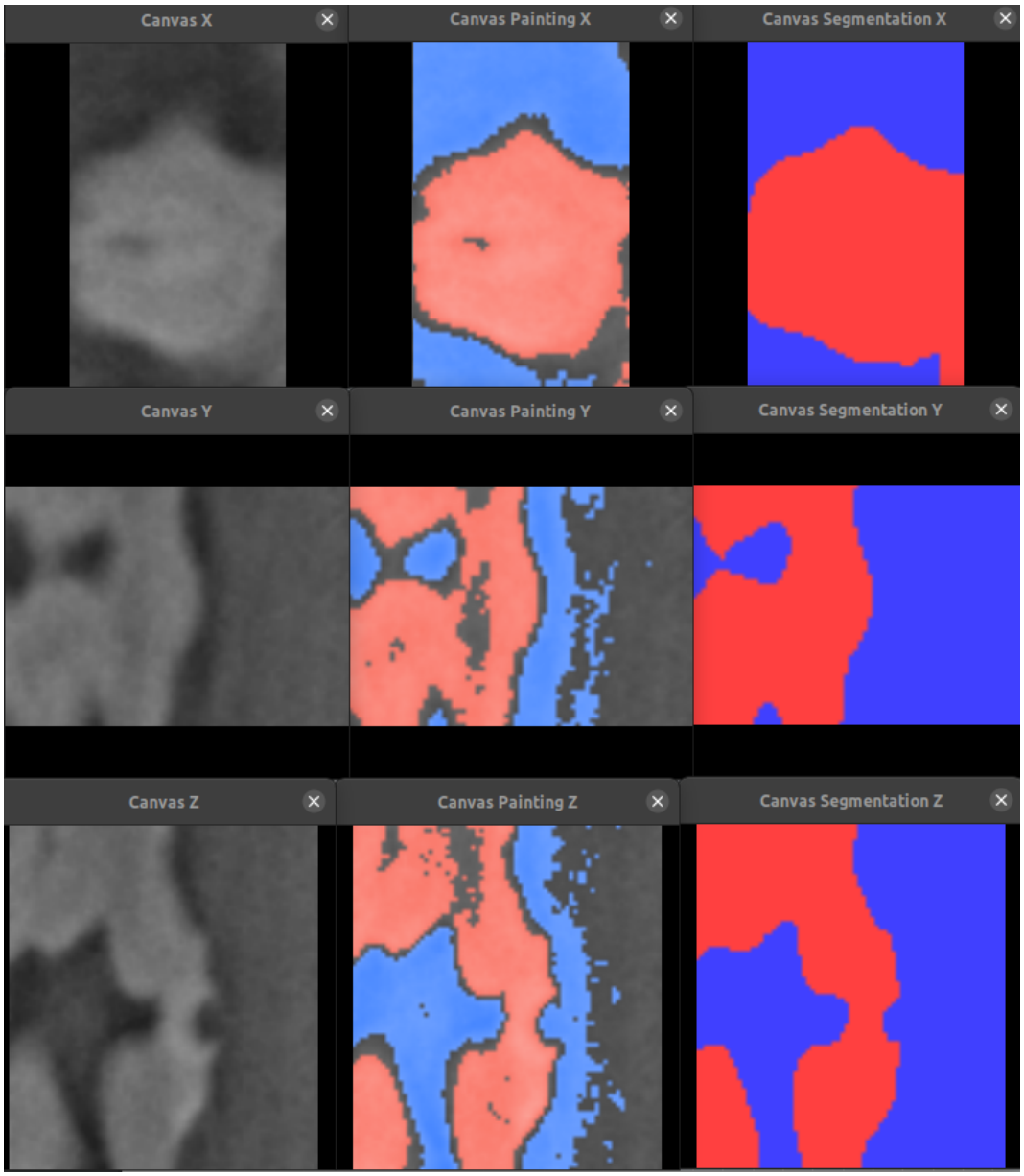


FIGURE 12: Différents canvas 2D : à gauche, extraction d'une slice du volume original. Au milieu : en rouge les pixels que l'on veut garder (les graines de premier plan) et en bleu les pixels que l'on ne souhaite pas garder (les graines d'arrière-plan). À droite : retour de la segmentation sur les slices 2D

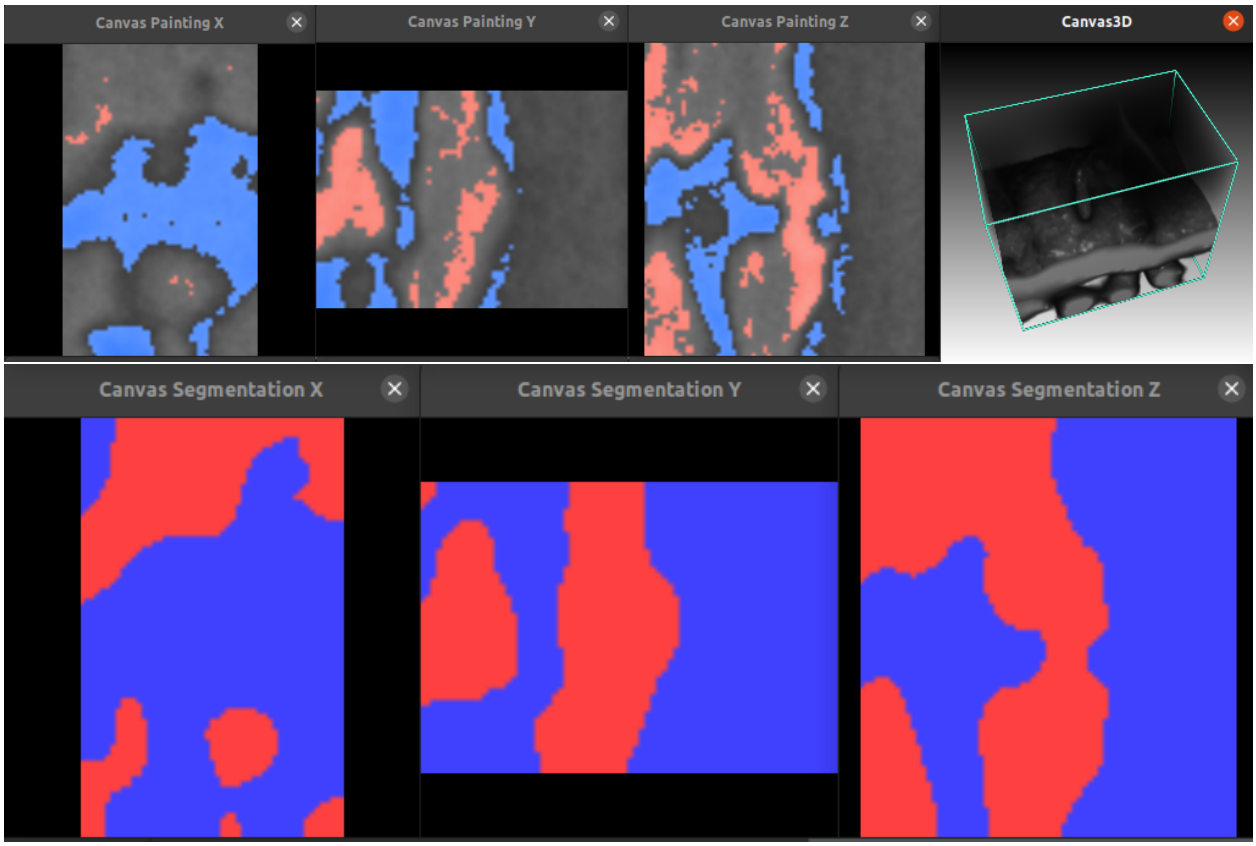


FIGURE 13: Résultat de la segmentation en utilisant uniquement les seuils

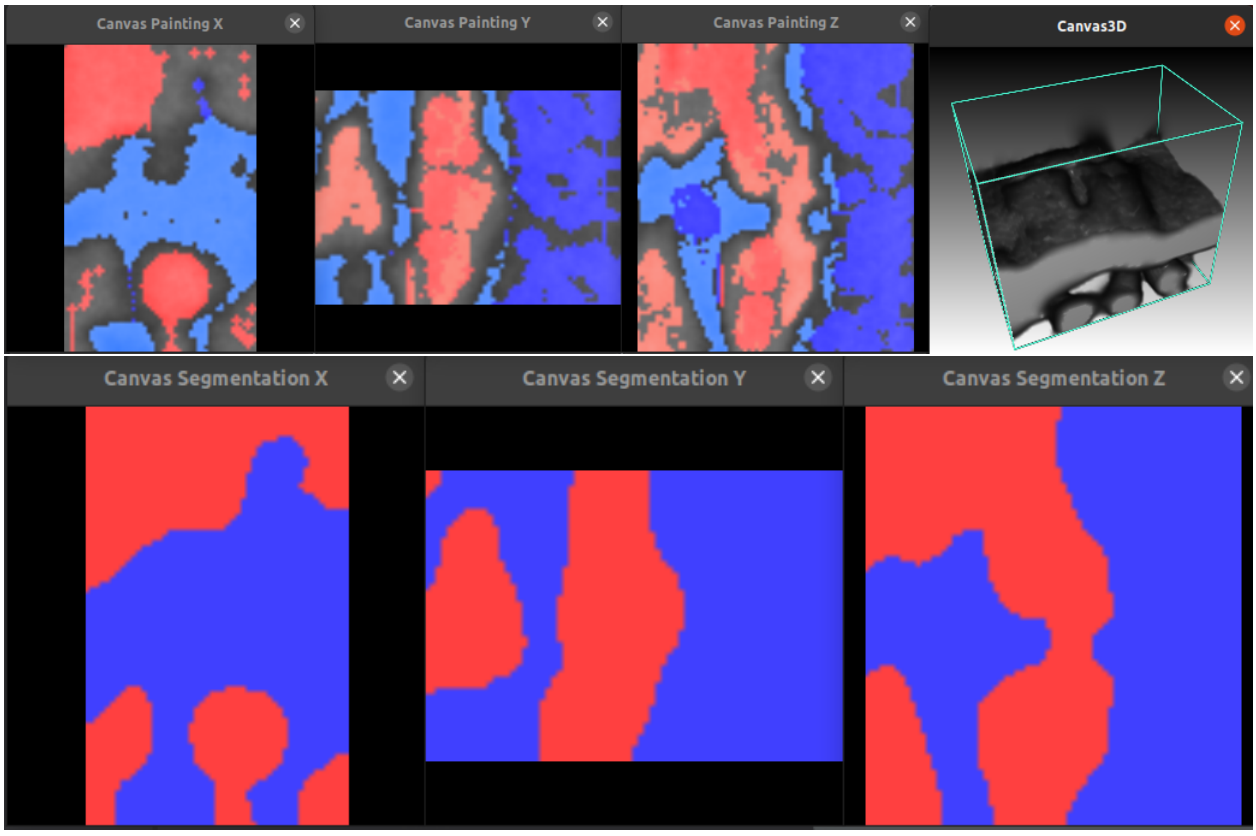


FIGURE 14: Résultat de la segmentation en utilisant les seuils et la peinture

13 Conclusion (Bilan / Perspectives)

L'outil de segmentation permet d'obtenir une visualisation des données pertinente qu'il n'était pas possible d'obtenir avec la seule modification des fonctions de transfert. De plus, l'annotation manuelle à l'aide de l'outil de peinture permet d'affiner le résultat obtenu uniquement avec les seuils. L'outil de mesure réalisé permet d'obtenir trois informations très importantes pour les utilisateurs : le nombre de voxels correspondant à l'air à l'extérieur de la feuille, le nombre de voxels correspondant à l'air à l'intérieur de la feuille et le nombre de voxels correspondant à de la matière. Ces valeurs, acquises suite à la segmentation, n'avaient jusqu'alors jamais pu être obtenues par les utilisateurs de INRAE.

Il existe de nombreuses pistes d'amélioration. Premièrement, en plus des mesures des volumes segmentés obtenues, il pourrait être intéressant de pouvoir distinguer les différents tissus à l'intérieur de la feuille et donc de réaliser différents types de segmentation. De plus, l'implémentation actuelle pour l'outil de mesure permet uniquement de travailler sur des feuilles dont le stomate est fermé. En effet, si le stomate est ouvert, il n'est plus possible de distinguer l'air intérieur de l'air extérieur. Il faudrait donc pouvoir refermer le stomate pour délimiter l'intérieur de l'extérieur.
Deuxièmement, des améliorations ergonomiques seraient profitables, notamment l'amélioration de la fonction d'annulation qui est limitée à 20 retours en arrière. Elle n'est actuellement pas optimisée par rapport à son coût en mémoire. Enfin, des retours visuels permettant d'afficher les slices 2D actuellement modifiées sur le volume 3D ont également été suggérés par les clients.

14 Glossaire

14.1 Ostiole

L'ostiole est une petite ouverture présente dans le stomate par laquelle peut passer l'air.

14.2 Processeur

Un processeur est un nœud dans le réseau nodal, il correspond à un ou plusieurs processus précis. Un processeur peut aussi être une source de données (agissant ainsi comme point d'entrée du réseau nodal) ou comme affichage de données (c'est donc une sortie de réseau). C'est la connexion entre plusieurs processeurs qui forment ainsi le réseau nodal et qui permet un traitement spécifique des données.

14.3 Segmentation

La segmentation est un processus par lequel on sépare un ensemble de données en classes distinctes. Dans le cadre de ce projet, la segmentation correspond principalement à séparer les données volumiques de sorte à ce que la matière végétale et l'air soient identifiés.

14.4 Slice

Une slice (ou coupe en français), correspond à une subdivision d'un ensemble de données volumétriques le long d'un axe. Le plus souvent l'un des axes définissant l'espace dans lequel se trouve les données. Dans le cadre de ce projet, les slices extraites sont traitées comme des données en deux dimensions. Ceci est possible ici, car la profondeur des données est implicite (les voxels étant des cubes).

14.5 Stomate

Le stomate est une structure qui assure les échanges gazeux au niveau des feuilles.

15 Annexes

15.1 Gantt Previsionnel

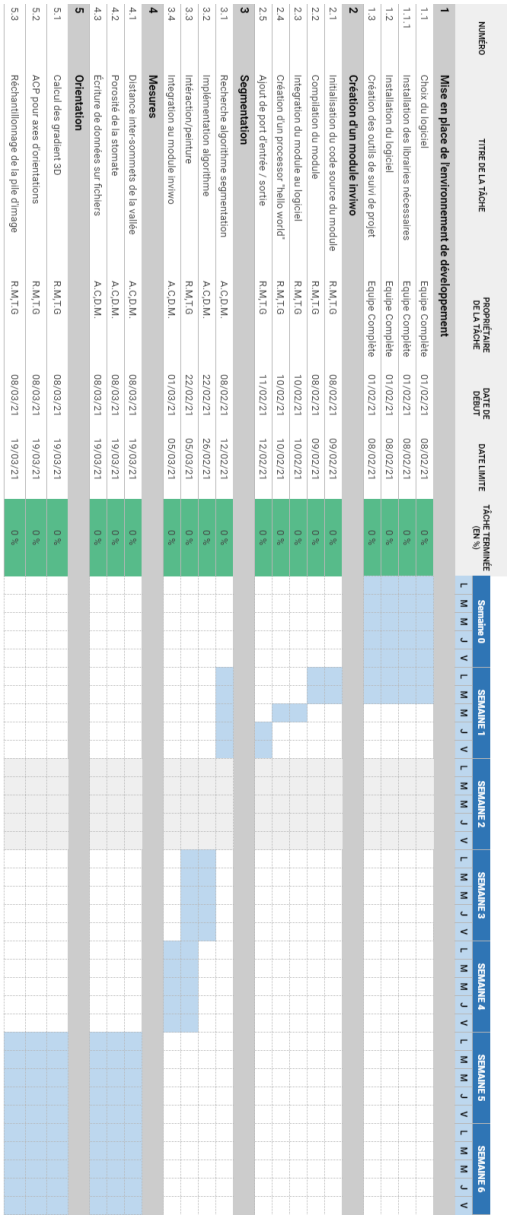


FIGURE 15: Gantt Prévisionnel

15.2 Gantt Effectif

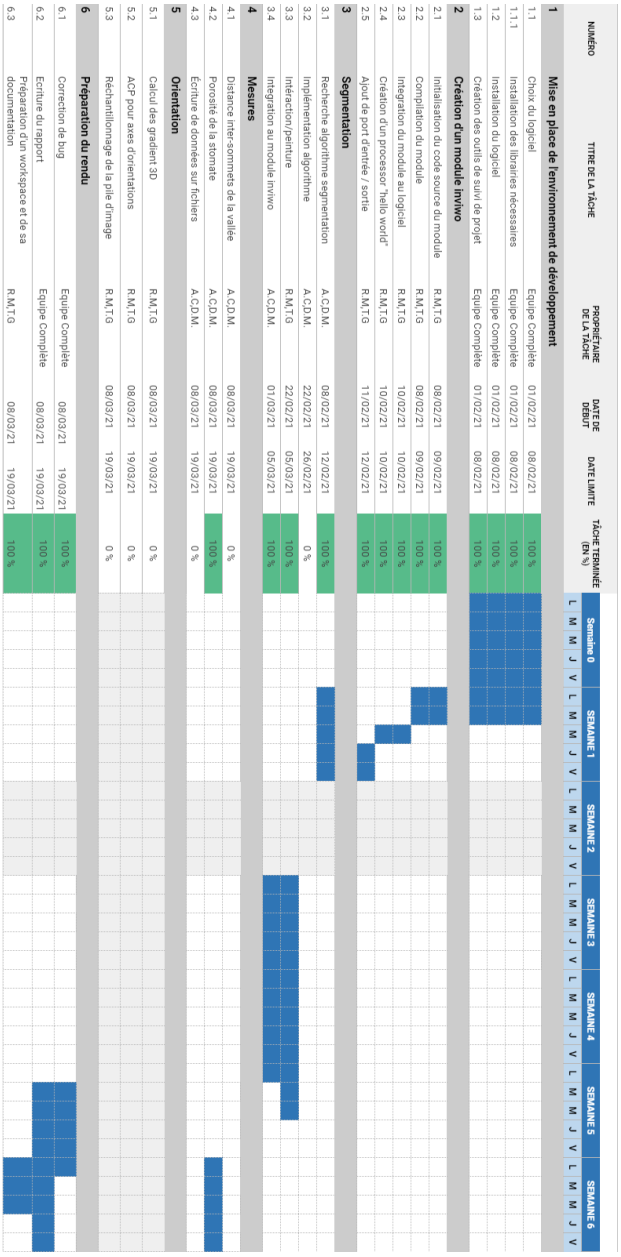


FIGURE 16: Gantt Effectif

15.3 Manuel d'utilisation

15.3.1 Environnement de travail

L'environnement de travail, appelé *workspace* au sein du logiciel Inviwo est fourni dans l'archive du projet. Celui-ci correspond au fichier *stomate-segmentation.inv*. Pour le charger dans Inviwo, il faut l'ouvrir en sélectionnant *File* → *Open Workspace*. Sauvegarder l'environnement de travail permet de conserver l'architecture des processeurs ainsi que leur paramétrage.

La figure 17 permet de voir l'environnement de travail dans son ensemble. Le panel de gauche permet d'ajouter des processeurs. Celui du centre permet de lier les ports et les propriétés des processeurs les uns aux autres. Le panel de droite permet de voir et de modifier les propriétés des processeurs.

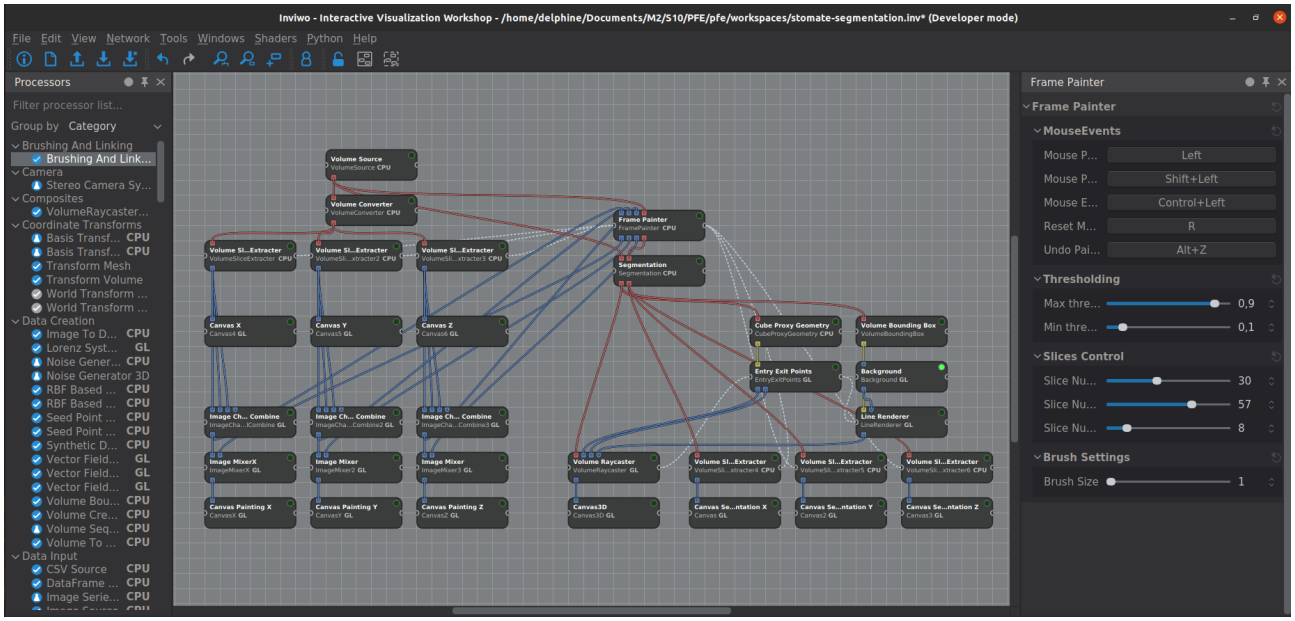


FIGURE 17: Vue de l'environnement de travail

15.3.2 Processeurs

On peut séparer les processeurs en trois groupes afin de faciliter la compréhension de l'environnement de travail : un groupe de lecture des données, un de calcul et un d'affichage.

La figure 18 montre les processeurs permettant de lire les données et de les envoyer aux processeurs d'affichage et de calcul. C'est le processeur *Volume Source* qui permet de sélectionner la pile d'image à charger. Les processeurs *Volume Slice Extracter* permettent de naviguer entre les différentes slices du volume d'origine. Ce sont ces processeurs qui envoient les slices actives aux processeurs d'affichage et de calcul.

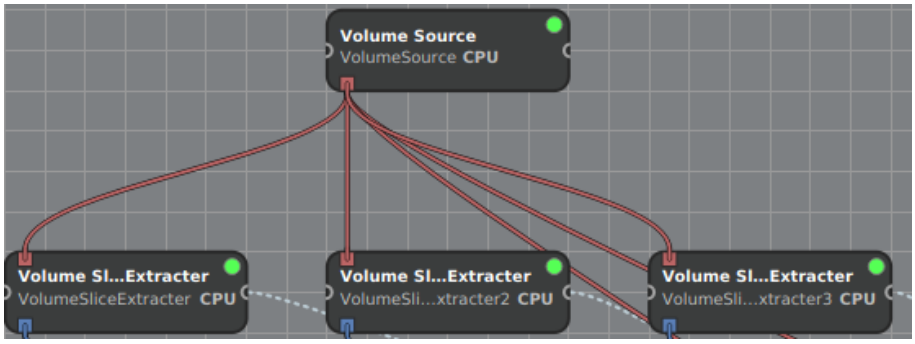


FIGURE 18: Processeurs de lecture des données

La figure 19 montre les processeurs implémentés pour ce projet permettant de réaliser la segmentation : *Frame Painter* et *Segmentation*.

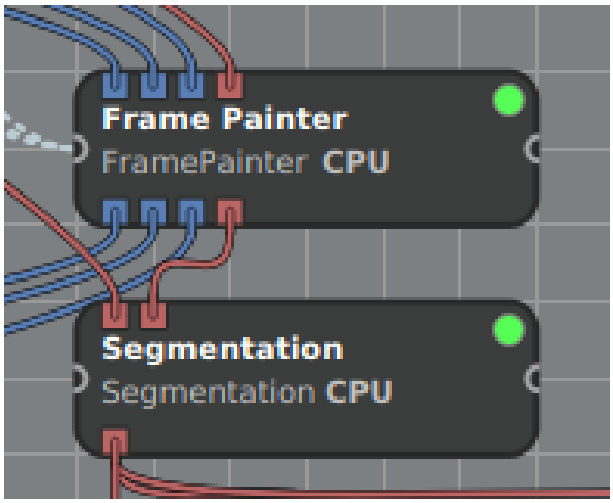


FIGURE 19: Processeurs permettant de réaliser la segmentation d'un volume

Le processeur *Frame Painter* permet d'annoter les slices en vue de la segmentation. Celui-ci offre différents paramètres dans l'onglet propriété (voir figure 20) dont deux seuils, les numéros de slices et le rayon de la brosse de peinture. Le seuil *Max treshold* permet d'annoter toutes les valeurs supérieures au seuil comme étant de la matière. Les pixels ainsi annotés seront alors dessinés en rouge sur le canvas 2D de peinture. Le seuil *Min treshold* permet d'annoter toutes les valeurs inférieures au seuil comme étant de l'air. Les pixels ainsi annotés seront alors dessinés en bleu sur le canvas 2D de peinture. De plus, pour annoter manuellement un pixel, il est possible de cliquer directement dans le canvas 2D de peinture. Un simple clic permet d'annoter les pixels correspondant à la matière et affichera le(s) pixel(s) sélectionné(s) en rouge (d'une teinte légèrement différente de celle utilisée pour les seuils). Si la touche *SHIFT* est utilisée en plus du clic, le(s) pixel(s) sélectionné(s) sera(ont) considéré(s) comme de l'air et affiché(s) en bleu sur le canvas 2D de peinture. Les numéros de slices sont reliés aux processeurs *Volume Slice Extracter*. Il est donc possible de naviguer entre les slices directement depuis le processeur *Frame Painter*. Le rayon de la brosse permet d'annoter un plus grand nombre de pixels avec un seul coup de pinceau. Celui-ci est limité à un rayon de 10 pixels.

FIGURE 20: Propriétés du processeur Frame Painter

Le processeur de segmentation réalise la segmentation du volume en entrée du *Volume Source* à l'aide de l'algorithme *random walker*. Pour cela, les données annotées grâce au processeur *Frame Painter* sont utilisées. Il est possible d'activer/désactiver la segmentation dans les propriétés, en cliquant sur le bouton *Apply Segmentation*. Cela peut être utile afin d'économiser des ressources pendant l'annotation des données. Ce processeur offre également la possibilité d'enregistrer différentes mesures dans un fichier texte (voir figure 21), en appuyant sur la touche *p*.

```
Dimensions : [       73,       81,       51]
Total volume : 301563 voxels
Outside air volume : 126381 voxels
Volume of the matter : 110253 voxels
Interior air volume : 64929 voxels
Total volume of the leaf : 175182 voxels
Proportion of air inside the leaf : 37.0637%
Proportion of matter inside the leaf : 62.9363%
```

FIGURE 21: Mesures des différentes parties segmentées

La figure 22 montre les processeurs permettant d'afficher les slices sur lesquelles se fait l'annotation (à l'aide des seuils ou de l'outil de peinture), ainsi que le résultat de la segmentation en 3D. Il est possible de paramétrer la fonction de transfert du processeur *Volume Raycaster* afin d'obtenir une frontière matière/air plus nette.

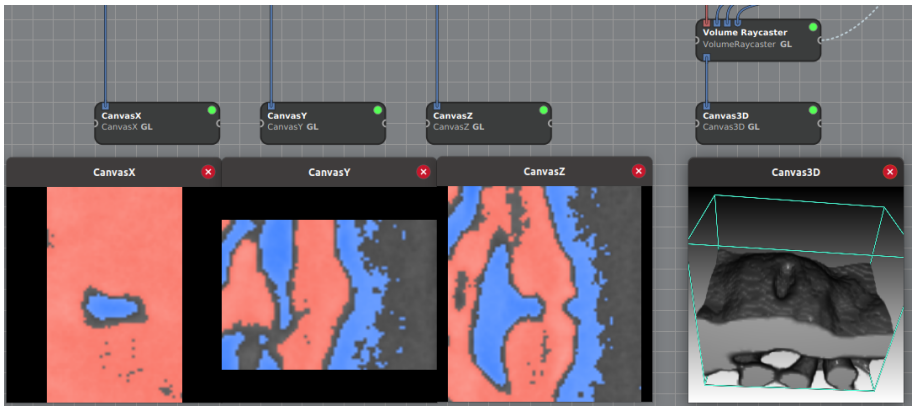


FIGURE 22: Affichage des canvas sur lesquelles sont annotées les données à segmenter et résultat de la segmentation en 3D

Références

[1] The GIMP Development Team. *GIMP*. Version 2.10.22. 4 oct. 2020. URL : https://www.gimp.org (visité le 20/03/2021).

[2] Wayne Rasband. *ImageJ*. Version 1.53h. 4 fév. 2021. URL : https://imagej.nih.gov/ij/ (visité le 20/03/2021).

[3] *GNU General Public License*. Version 3. Free Software Foundation, 29 juin 2007. URL : http://www.gnu.org/licenses/gpl.html (visité le 20/03/2021).

[4] Daniel Jönsson et al. "Inviwo - A Visualization System with Usage Abstraction Levels". In : *IEEE Transactions on Visualization and Computer Graphics* 26.11 (2019), p. 3241-3254. ISSN : 1077-2626. DOI : 10.1109/TVCG.2019.2920639.

[5] Kikinis R, Pieper SD, Vosburgh K. *3D Slicer*. Version 4.11.20210226. 1er mar. 2021. URL : https://www.slicer.org/ (visité le 20/03/2021).

[6] Daniel Jönsson et al. *Inviwo*. Version v0.9.11. 3 sept. 2019. URL : https://inviwo.org/ (visité le 20/03/2021).

[7] Jennis Meyer-Spradow, Timo Ropinski, Jörg Mensmann, Klaus H. Hinrich. *Voreen*. Version 5.2.0. 28 jan. 2021. URL : https://www.uni-muenster.de/Voreen/ (visité le 20/03/2021).

[8] Michael S. Noble. *Volview*. Version 3.4. 20 juil. 2011. URL : https://www.kitware.com/volview/ (visité le 20/03/2021).

[9] Johannes A. Koeppen. *Ogles2*. Version 2b. 20 juin 2016. URL : http://ogles.sourceforge.net/Ogles/ogles2-doc/index.html (visité le 20/03/2021).

[10] Paul Yushkevich, Guido Gerig. *ITK-Snap*. Version 3.8.0. 6 déc. 2019. URL : http://www.itksnap.org/pmwiki/pmwiki.php (visité le 20/03/2021).

[11] NIH/NIGMS Center for Integrative Biomedical Computing. *ImageVis3D*. Version 3.1.0. 8 avr. 2014. URL : https://www.sci.utah.edu/cibc-software/imagevis3d.html (visité le 20/03/2021).

[12] Gaël Guennebaud, Benoît Jacob et al. *Eigen*. Version 3.3.9. 12 avr. 2020. URL : http://eigen.tuxfamily.org (visité le 20/03/2021).

[13] Xiaofeng Xie et al. "An Iterative Boundary Random Walks Algorithm for Interactive Image Segmentation". In : *arXiv: Computer Vision and Pattern Recognition* (2018).