UCLouvain

epl

École polytechnique de Louvain

Construire une IUG universelle pour la gestion de l'hétérogénéité des formats d'imagerie cérébrale via conversion automatique et visualisation 3D intégrées

Auteur: **Michaël Halleux**
Promoteurs: **Benoît Macq, Laurence Dricot**
Lecteurs: **Sébastien Jodogne, Nicolas Delinte**
Année académique 2024–2025
Master [120] en sciences informatiques

Résumé

L'analyse des données d'imagerie cérébrale repose aujourd'hui sur une multitude d'outils logiciels spécialisés. Cette diversité s'accompagne d'une hétérogénéité importante des formats de fichiers utilisés, ce qui complique la visualisation, la conversion ou l'exploitation de ces données dans des flux de travail standardisés. Ce mémoire propose une réponse technique à une partie de cette problématique en développant un outil logiciel interactif combinant deux fonctionnalités essentielles : la visualisation tridimensionnelle de fichiers d'imagerie cérébrale (imagerie par résonance magnétique anatomique, fonctionnelle et de diffusion) et la conversion automatique entre plusieurs formats répandus (TRK, TCK, FBR, VOI, VMR, NIfTI).

Conçu en Python dans une démarche modulaire et extensible, cet outil repose sur une interface graphique intuitive et s'appuie sur des bibliothèques open source robustes comme `PyVista`, `DIPY`, `NumPy`, `PyQt6` ou `Nibabel`. Une attention particulière a été portée à la gestion des systèmes de coordonnées et des matrices de transformation afin de garantir la cohérence spatiale des données après conversion. L'ensemble du projet est distribué sous licence libre et documenté dans un dépôt public, avec pour ambition de servir de passerelle entre des outils hétérogènes sans imposer de rupture aux pratiques existantes. Ce travail s'inscrit ainsi dans une logique de science ouverte et vise à améliorer l'accessibilité, l'interopérabilité et la réutilisabilité des données en neuro-imagerie.

Remerciements

Je tiens tout d'abord à exprimer ma sincère gratitude à mes deux promoteurs, Benoît Macq et Laurence Dricot, pour leur disponibilité, leurs conseils avisés et le temps qu'ils m'ont consacré tout au long de ce travail. Leur encadrement et leur confiance ont constitué un cadre à la fois exigeant et bienveillant, qui m'a permis de progresser de manière structurée et stimulante. Merci également pour les réunions régulières et pour la clarté de leurs retours.

Je souhaite également remercier très chaleureusement Nicolas Delinte et Quentin Dessain, pour leur soutien constant, leur grande patience et la générosité avec laquelle ils ont partagé leur expertise. Leur accompagnement m'a été précieux, tant pour comprendre des notions complexes, que pour garder le cap dans les moments de doute ou de dispersion.

Enfin, je remercie l'équipe de support de *BrainVoyager* pour leur réactivité, leur disponibilité et la qualité des réponses apportées à mes nombreuses questions techniques. Leur collaboration m'a grandement aidé dans la compréhension des formats propriétaires et l'intégration de ces données dans le cadre du projet.

À toutes ces personnes, je témoigne ma reconnaissance pour leur contribution essentielle à la réalisation de ce travail, et je ne peux pas m'empêcher de faire un clin d'œil à mes parents pour leur soutien inconditionnel tout au long de mes cinq années de formation.

Table des matières

Introduction

Le cerveau humain est un organe fascinant, à la fois incroyablement complexe et encore largement mystérieux. Grâce à l'imagerie médicale et en particulier à la neuro-imagerie, il est aujourd'hui possible d'en explorer l'organisation, le fonctionnement et même la connectivité interne sans aucune intervention invasive. Ces techniques ont révolutionné les neurosciences, la neurologie et bien d'autres disciplines en donnant accès à des données d'une richesse exceptionnelle.

Mais cette richesse a un prix : celui de la complexité technique. En effet, chaque logiciel, chaque laboratoire, chaque pipeline de traitement semble avoir développé son propre format de fichier, ses propres standards, ses propres habitudes. Il en résulte une forme de désordre silencieux, souvent ignoré mais bien réel, qui complique le travail des chercheurs, ralentit les processus d'analyse et freine la collaboration entre équipes.

C'est dans ce contexte que s'inscrit le présent travail. L'objectif est de proposer une contribution utile, pragmatique et ouverte : un outil simple d'utilisation, capable de visualiser et de convertir plusieurs formats courants en neuro-imagerie. L'idée est de fournir une sorte de boîte noire dans laquelle un professionnel peut glisser son fichier et en ressortir une visualisation ou une conversion exploitable, sans devoir en maîtriser tous les détails techniques.

Ce mémoire est structuré en deux grandes parties. La première, de nature théorique, a pour but de poser le cadre scientifique, technique et conceptuel dans lequel s'inscrit ce travail. Elle débute par une présentation des différentes modalités de la neuro-imagerie, avec un accent particulier sur l'IRM structurelle, fonctionnelle et de diffusion. S'ensuivent une analyse détaillée des outils de visualisation existants, des formats de fichiers associés et des enjeux d'interopérabilité qui en découlent. Un chapitre est ensuite consacré aux systèmes de coordonnées en neuro-imagerie et aux matrices de transformation spatiale, éléments clés pour comprendre les conversions géométriques entre les formats. La seconde partie est centrée sur la contribution logicielle proprement dite. Elle présente le problème traité, la solution proposée ainsi que la méthodologie de développement adoptée. Sont ensuite détaillés les choix techniques, les formats pris en charge, les modules de conversion implémentés et l'interface graphique développée pour l'utilisateur. L'ensemble est enrichi d'une discussion critique portant sur les apports du projet, les difficultés rencontrées et les pistes d'amélioration identifiées.

Première partie

Contexte théorique et problématique

Chapitre 1

Variété des modalités en neuro-imagerie

L'imagerie médicale désigne l'ensemble des techniques utilisées pour visualiser le corps humain grâce à l'image. La neuro-imagerie, ou imagerie cérébrale, en est une branche spécifique dédiée à la visualisation et à l'étude du cerveau.
Deux grandes catégories de techniques sont distinguées en neuro-imagerie : l'imagerie structurelle et l'imagerie fonctionnelle. Selon la Fondation pour la Recherche Médicale (FRM), l'imagerie structurelle « permet d'étudier l'anatomie du cerveau et tout ce qui peut la perturber (tumeur, hémorragie, déformation pathologique, etc.). Elle se révèle très utile au diagnostic médical.» [1]. La méthode principale utilisée à cette fin est l'**IRM** (Imagerie par Résonance Magnétique) **structurelle**, aussi appelée **IRM anatomique**.
Concernant l'imagerie fonctionnelle, l'Institut du Cerveau de Paris la définit comme une « technique d'imagerie qui enregistre l'activité d'un organe (le cerveau par exemple), lors de l'exécution d'une tâche, comme un mouvement, la lecture, la réponse à une question... » [2]. Toujours selon la FRM, l'imagerie fonctionnelle (dans le cadre de la neuro-imagerie) « rend compte de l'activité des zones cérébrales durant certaines tâches (parole, mouvement, etc.). Elle est autant utilisée en recherche fondamentale qu'en clinique [...] » [1].
Cinq techniques majeures sont utilisées pour l'imagerie fonctionnelle : la **MEG** (Magnétoencéphalographie), l'**EEG** (Électroencéphalogramme), la **TEP** ou **PET** en anglais (Tomographie par émission de positons, PETscan), l'**IRMf** (Imagerie par Résonance Magnétique fonctionnelle) et l'**IRMd** (Imagerie par Résonance Magnétique de diffusion).

1.1 Principe de l'Imagerie par Résonance Magnétique

Cette section a pour objectif de présenter les principales modalités d'imagerie utilisées pour générer les fichiers exploités dans le cadre de ce travail. L'approche consiste à introduire les fondements des techniques concernées sans entrer dans les détails physiques approfondis, afin de rester en cohérence avec le cadre du mémoire.

L'IRM repose sur une technologie d'acquisition permettant d'obtenir différents types d'images du cerveau humain, notamment structurelles et fonctionnelles. Bien que cette technologie soit également utilisée pour l'exploration d'autres parties du corps, le présent travail se focalise sur son application à l'imagerie cérébrale même si les principes physiques sous-jacents restent identiques.

Un appareil d'IRM se présente comme une structure cylindrique imposante au sein de laquelle le patient est allongé sur une table mobile. Le sujet est positionné au centre du tunnel afin de bénéficier d'un champ magnétique homogène, orienté de manière stable et généré par l'aimant (fig 1.1). Ce champ principal, noté B_0, constitue la base sur laquelle l'ensemble du signal sera construit.

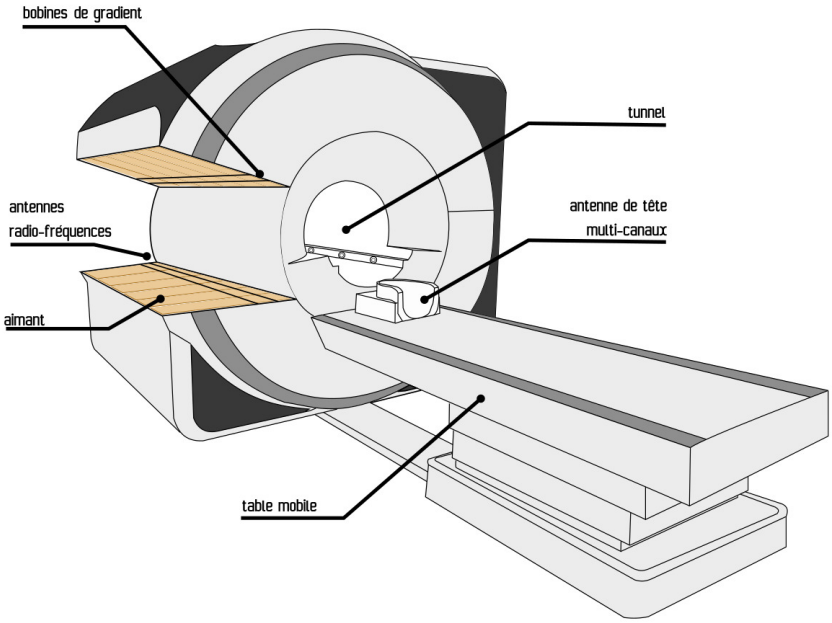


FIGURE 1.1 – *Illustration schématique des principales composantes d'un appareil IRM* [3]

Le principe fondamental de l'IRM repose sur les propriétés magnétiques des noyaux d'hydrogène présents dans les tissus biologiques.

Un proton (noyau d'hydrogène) se comporte comme un aimant qui tourne autour de son propre axe. Le moment angulaire qui caractérise cette rotation est appelé `spin` (fig 1.2). Et la fréquence à laquelle un proton effectue ses rotations autour de son axe est appelée `fréquence de Larmor`. En temps normal, les spins pointent dans des directions totalement aléatoires dans les tissus du corps humain. Cependant, l'aimant principal de l'IRM permet d'aligner les moments magnétiques des protons selon le même axe (de manière parallèle ou anti-parallèle) que le champ magnétique principal généré (B_0). Ce champ part en général des pieds du patient et se dirige vers sa tête.

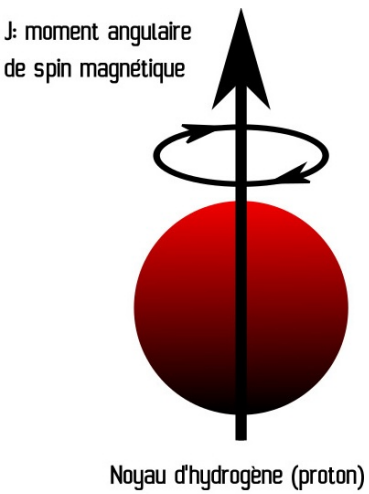


FIGURE 1.2 – *Oscillation du champ magnétique d'un proton* [3]

Les protons alignés au champ magnétique principal B_0 se trouvent dans un état d'équilibre. Pour obtenir une image, l'IRM repose sur la détection d'un signal mesurant les variations de comportement des protons selon le type de tissu traversé. Afin de générer ce signal, il est nécessaire de perturber cet équilibre en excitant les protons par l'application

de gradients de champ magnétique (des aimants secondaires plus petits) qui viennent perturber localement B_0. L'analyse de leur retour progressif à l'état d'équilibre permet alors de caractériser les propriétés des tissus et de reconstruire une image représentative de la structure interne du cerveau. L'IRM exploite le phénomène de résonance magnétique grâce à l'antenne radio-fréquence (fig 1.1) pour générer cette excitation. Cette antenne crée une série d'ondes dites "radio-fréquences" perpendiculaires au champ B_0 en direction du champ B_1. Cette série d'impulsion suit la fréquence de Larmor de l'hydrogène qui entre en résonance et bascule perpendiculairement à B_0. En arrêtant ces impulsions, le moment magnétique des atomes d'hydrogène va revenir à son état d'alignement avec le champ B_0. Cette étape s'appelle `la relaxation` des atomes. Ce phénomène joue un rôle central dans le processus d'imagerie car la vitesse à laquelle les protons effectuent cette relaxation (mesurée par les antennes de tête multi-canaux (fig 1.1)) dépend directement des propriétés du tissu environnant. Cette dépendance permet ainsi de différencier les types de tissus biologiques, offrant la possibilité de reconstruire une image des structures cérébrales.

Pour faciliter la compréhension, nous pouvons reprendre l'analogie faite par Pierre Bellec, Eddy Fortier et Samuel Guay dans leur cours sur les cartes cérébrales [4], le champ magnétique "*[...] est comme une toile vierge pour une peinture.*" et "*Comme différents tissus biologiques réagissent différemment à ces stimulations (celles des gradients), les gradients nous permettent de "peindre" une image du cerveau sur la "toile" B0.*".

En pratique, l'IRM fournit une série de coupes successives. L'ensemble de ces coupes constitue un volume complet permettant d'obtenir une représentation tridimensionnelle du cerveau.

1.2 Formation de l'image

Pour reconstruire une image en IRM, il faut identifier de manière précise de petits volumes de tissu appelés voxels (l'équivalent du pixel mais en 3D). Pour cela, trois étapes sont nécessaires afin de les localiser selon les trois axes de l'espace : sélectionner une coupe (dans la direction z), identifier une colonne de voxel (axe x, codage de phase) et repérer une ligne de voxel (axe y, codage de fréquence).

Ces opérations sont possibles grâce aux bobines de gradient (fig 1.3), utilisées pour faire varier l'amplitude du champ magnétique principal dans les trois directions.

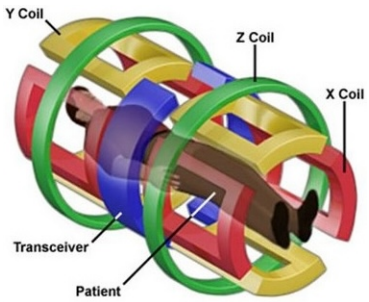


FIGURE 1.3 – *Bobine de gradients utilisées pour l'encodage en trois dimensions* [5]

Sélection de coupe : excitation sélective

La sélection du plan de coupe constitue la première étape du codage spatial [6]. Elle repose sur l'application d'un gradient de champ magnétique, le gradient de sélection de coupe (GSC) en mauve sur la figure 1.4a, perpendiculaire au plan visé (donc parallèle à B_0 et s'y additionne) qui modifie localement la fréquence de précession [1] des protons.

En plus, une impulsion radiofréquence sélective est émise à la fréquence correspondant uniquement aux protons du plan ciblé (fig 1.4b). Seuls ces protons sont alors excités et les autres, hors de la zone de résonance, restent inactifs et ne produisent aucun signal.

(a) (b)

FIGURE 1.4 – *(A) Application du GSC ; (B) Application de la RF* [6]

Ces protons, tous situés dans le plan de coupe sélectionné, vont ensuite être une nouvelle fois stimulés pour déterminer leur position horizontale et verticale (fig 1.5a et fig 1.5b).

(a) (b)

FIGURE 1.5 – *(A) Protons du plan de coupe sélectionné ; (B) Axes relatifs aux protons* [6]

1. Pour vulgariser et imager ce terme, le mouvement de précession désigne le "cercle" formé par l'axe de rotation du proton

Codage de phase

On applique ensuite un gradient de codage de phase (GCP). Ce gradient, actif brièvement, modifie temporairement la fréquence de précession des protons selon leur position ce qui va modifier la phase de chaque proton. Après l'arrêt du gradient, tous les protons ont une fréquence de précession identique mais avec des phases différentes selon leur position verticale (fig 1.6). Chaque ligne de protons (perpendiculaire à la direction du GCP) a donc la même phase mais celle-ci diffère d'une ligne à l'autre. Ce décalage de phase est conservé jusqu'à la mesure du signal, permettant ainsi d'identifier la position verticale des protons lors du traitement des données.

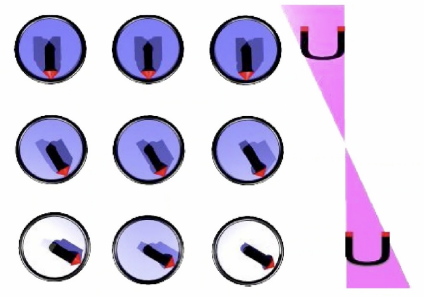


FIGURE 1.6 – *Illustration du décalage de phase entre les lignes* [6]

Codage fréquentiel

La dernière étape du codage spatial consiste à appliquer un gradient de codage par la fréquence. Ce gradient modifie la fréquence de précession des protons en fonction de leur position horizontale durant toute la mesure (fig 1.7). Ainsi, chaque colonne de protons possède une fréquence de précession distincte, ce qui permet d'identifier leur position horizontale lors du traitement du signal.

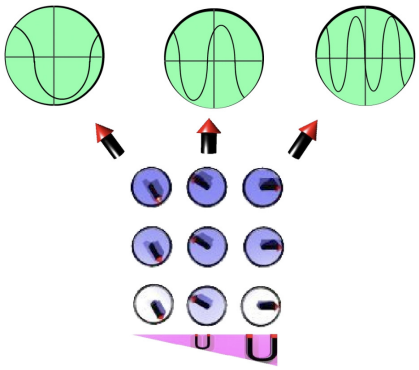


FIGURE 1.7 – *Illustration des différences de fréquence entre les colonnes* [6]

Construction de l'image (k-espace)

Les signaux mesurés sont stockés dans une matrice appelée k-espace, qui contient les informations sous forme de fréquences. Le k-espace est rempli ligne par ligne en variant le gradient de phase et en enregistrant les signaux sous différents gradients de fréquence.

Une fois toutes les lignes remplies (en répétant l'acquisition plusieurs fois), une transformée de Fourier inverse est appliquée pour convertir les données du k-espace en une image classique dans le domaine spatial, celle que l'on peut visualiser.

1.3 Contrastes T_1 et T_2

Comme expliqué ci-dessus, après l'excitation radiofréquence en IRM, les spins des protons d'hydrogène sont déviés de leur alignement initial avec le champ magnétique principal B_0 et tendent alors à revenir progressivement vers cette position d'équilibre, selon un processus appelé relaxation. Les temps de relaxation T_1 et T_2 sont des paramètres qui caractérisent cette relaxation des protons dans les tissus.

Le temps T_1 (*relaxation longitudinale*) caractérise le retour de la magnétisation selon l'axe du champ principal B_0 (axe z). Il reflète la vitesse à laquelle les spins se réalignent avec le champ magnétique.

La composante de la magnétisation selon l'axe z (M_z) augmente alors avec le temps, suivant une loi exponentielle croissante donnée par :

$$M_z(t) = M_0 \left(1 - e^{-t/T_1}\right) \tag{1.1}$$

où M_0 représente la magnétisation à l'équilibre, t le temps écoulé après l'impulsion radiofréquence et T_1 la constante de temps caractéristique de la relaxation longitudinale. Par définition, T_1 correspond au temps nécessaire pour que M_z atteigne 63 % de sa valeur finale M_0 (fig 1.8).

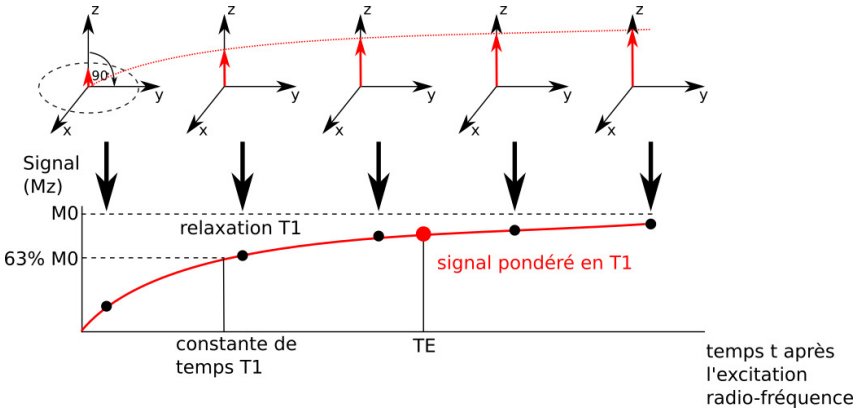


FIGURE 1.8 – *Processus de relaxation T1* [3]

En pratique, on n'observe généralement qu'un point de cette courbe, au temps d'écho TE, ce qui donne un *signal pondéré en* T_1. Si un tissu possède un T_1 long, sa magnétisation selon z augmente lentement, donc le signal mesuré à TE sera faible. À l'inverse, un T_1 court induit un signal plus élevé à ce même temps.

Chaque tissu biologique présente une valeur de T_1 propre, ce qui permet d'obtenir un contraste spécifique sur les images IRM pondérées en T_1 et ainsi de différencier les structures anatomiques.

Le temps T_2 (*relaxation transverse*) décrit la perte de cohérence de la magnétisation dans le plan transverse (xy), c'est-à-dire la diminution de la composante perpendiculaire à B_0 due au déphasage entre spins.

La composante de la magnétisation dans le plan transverse, notée M_{xy}, décroît progressivement avec le temps selon un processus appelé *relaxation transverse* ou *relaxation* T_2.

Cette décroissance est décrite par l'équation :

$$M_{xy}(t) = M_1 e^{-t/T_2} \tag{1.2}$$

où M_1 est la valeur initiale de la magnétisation transverse et T_2 la constante de temps caractéristique de cette décroissance. Par définition, T_2 correspond au temps nécessaire pour que M_{xy} atteigne 37 % de sa valeur initiale. Ce phénomène est également appelé *déphasage* car il résulte de la perte progressive de cohérence entre les spins dans le plan transverse (fig 1.9).

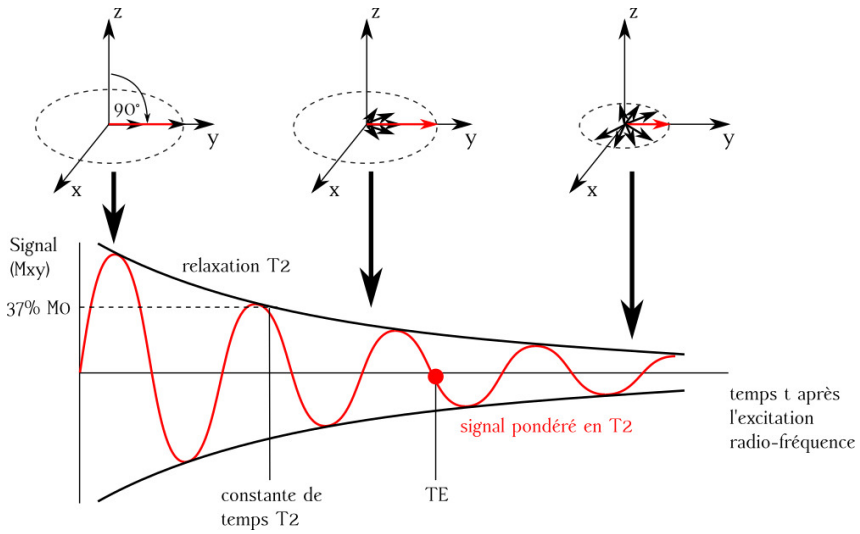


FIGURE 1.9 – *Processus de relaxation T2* [3]

Comme pour le contraste T_1, le signal mesuré est souvent enregistré à un temps précis (le temps d'écho TE), ce qui conduit à des images dites *pondérées en* T_2. Si un tissu présente une valeur de T_2 élevée, la décroissance de la magnétisation transverse est plus lente, et le signal mesuré à TE sera plus intense. À l'inverse, un T_2 court se traduit par une décroissance rapide et un signal faible.

Différents tissus (matière grise, matière blanche, liquide céphalo-rachidien) possèdent des constantes T_2 spécifiques, permettant d'obtenir des contrastes presque complémentaires à ceux des images pondérées en T_1.

En général, T_2 est toujours inférieur à T_1 pour un même tissu. Les images pondérées en T_1 et T_2 offrent des contrastes différents et complémentaires, ce qui permet de mieux distinguer les structures anatomiques en IRM.

1.4 IRM structurelle

L'IRM anatomique, ou structurelle, constitue une technique de référence pour l'exploration des structures internes du cerveau. Ce type de scan offre un contraste entre matière grise, matière blanche et liquide céphalo-rachidien qui permet de faire des analyses morphométriques [2] (fig 1.10).

2. La morphométrie est l'étude du cerveau et de ses structures

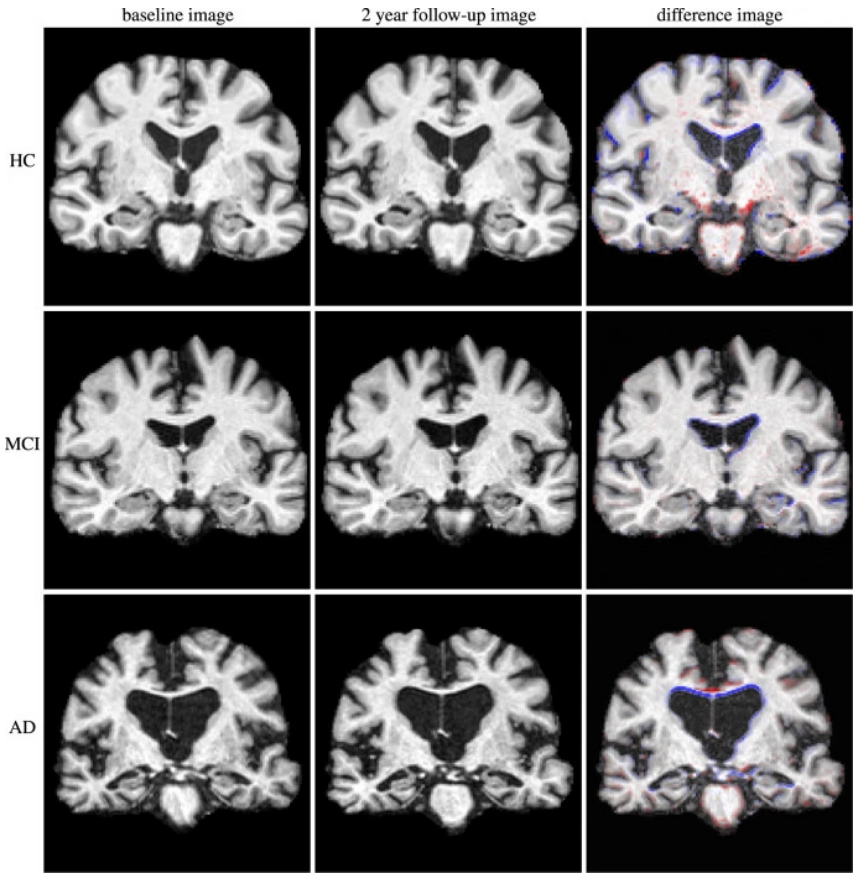


FIGURE 1.10 – *Différences morphologiques entre des individus* [7]

L'IRM anatomique permet ainsi de visualiser avec précision les différentes structures cérébrales, d'évaluer leur intégrité et de détecter d'éventuelles anomalies, telles que des tumeurs, des hémorragies, des malformations congénitales ou des lésions associées à des pathologies neurodégénératives. Elle fournit des images en trois dimensions avec une résolution spatiale qui est souvent de l'ordre du millimètre, ce qui la rend particulièrement adaptée aux analyses morphométriques fines. Les images d'une telle IRM sont constituées d'un seul volume en trois dimensions (x, y, z).
Contrairement aux techniques d'imagerie fonctionnelle, l'IRM anatomique fournit principalement des informations sur l'organisation statique du cerveau, sans indication directe sur son activité dynamique. Elle est toutefois essentielle pour la planification chirurgicale, le suivi de pathologies cérébrales et sert fréquemment de base de référence pour aligner d'autres types de données issues de techniques multimodales telles que l'IRM fonctionnelle ou l'IRM de diffusion.

1.5 IRM fonctionnelle

L'IRMf est une technique d'imagerie cérébrale permettant de localiser les zones du cerveau impliquées dans l'exécution de tâches spécifiques, telles que parler, compter, lire ou effectuer un mouvement volontaire. En enregistrant l'activité cérébrale, elle permet de délimiter avec précision les régions corticales responsables de fonctions cognitives ou motrices particulières. Les images IRMf présentent une série de volumes en trois dimensions. Nous parlons donc d'images 4D (ou 3D+t avec t qui correspond au temps) [8].
La capacité de l'IRMf à détecter l'activité cérébrale repose sur la mesure des variations

d'oxygénation du sang. L'activité neuronale s'accompagne d'une augmentation de la consommation en oxygène, mais surtout d'une élévation plus importante du débit sanguin local, phénomène qui est exploité pour générer le signal d'imagerie fonctionnelle.
Concrètement, l'activation neuronale entraîne une surcompensation hémodynamique : le flux sanguin croît davantage que la consommation réelle d'oxygène par les neurones. Ce déséquilibre est à l'origine du phénomène de contraste BOLD (*Blood Oxygenation Level Dependent*), qui constitue le fondement du signal exploité en IRMf. Le signal BOLD provient de la différence de comportement entre deux formes d'hémoglobine dans le sang :

— Oxyhémoglobine (avec oxygène)
— Désoxyhémoglobine (sans oxygène)

L'oxyhémoglobine n'influence pas le champ magnétique, mais la désoxyhémoglobine, elle, le perturbe. Cette perturbation rend le signal IRM plus faible et accélère le temps de relaxation T_2 dans les images IRM dites «pondérées T_2». Quand une région du cerveau devient plus active, elle reçoit plus de sang oxygéné : il y a alors moins de désoxyhémoglobine donc le signal IRM devient plus fort.

Le signal BOLD mesure donc les changements d'oxygénation du sang dans le cerveau, ce qui permet de repérer les zones cérébrales actives.

Grâce à l'IRM fonctionnelle, il devient ainsi possible de visualiser indirectement l'activité cérébrale et de cartographier avec précision les régions impliquées dans des fonctions spécifiques. Cette capacité à localiser dynamiquement les zones actives du cerveau constitue un outil essentiel en recherche fondamentale et en neurologie clinique.

1.6 IRM de diffusion

L'IRMd est une technique d'imagerie cérébrale permettant d'explorer l'architecture microstructurale du cerveau en mesurant le déplacement des molécules d'eau dans les tissus biologiques. Dans un environnement isotrope, comme le liquide céphalo-rachidien, les molécules d'eau diffusent librement dans toutes les directions. En revanche, dans les tissus cérébraux, la diffusion est souvent restreinte par des structures (les membranes cellulaires, les axones, etc), conduisant à une diffusion anisotrope.

Cette anisotropie est particulièrement prononcée dans la substance blanche, où les axones, orientés de manière parallèle, facilitent la diffusion de l'eau le long de leur axe, tout en la restreignant perpendiculairement (fig 1.11). Cette propriété est exploitée pour cartographier les trajectoires des fibres nerveuses et étudier la connectivité cérébrale [9].

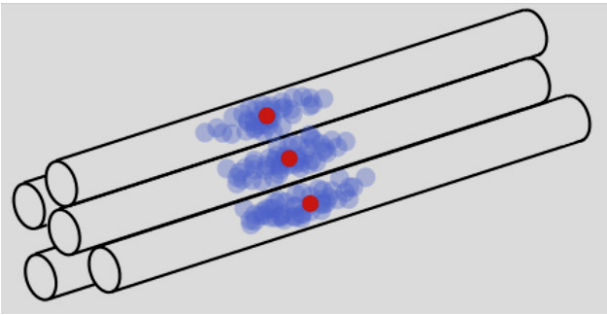


FIGURE 1.11 – *Schéma illustrant la diffusion de l'eau contrainte par les fibres (Pierre Bellec, sous licence CC-BY 4.0)*

Séquences pondérées

L'objectif d'une séquence pondérée en diffusion "est d'obtenir des images dont le contraste est influencé par les différences de mobilité de molécules d'eau." comme expliqué sur le site IMAIOS [10]. Cette séquence est appelée *Pulsed Gradient Spin Echo* (PGSE) et elle introduit deux gradients de diffusion de part et d'autre d'une impulsion radiofréquence à 180° (fig 1.12). Le premier sert à déphaser la magnétisation tandis que le second est utilisé pour la re-phaser. Les molécules d'eau immobiles retrouvent leur phase initiale après le second gradient, alors que les molécules mobiles restent déphasées. Plus la mobilité des molécules d'eau est importante dans un voxel, plus elles sont déphasées donc plus le signal IRM est atténué.

La sensibilité à la diffusion est caractérisée par le facteur b, qui dépend de l'intensité du gradient (g), de la durée (δ) et de l'intervalle (Δ) des gradients (entre le début du premier et le début du deuxième) [11, 12] :

$$b_{\text{PGSE}} = (\gamma g \delta)^2 \cdot (\Delta - \frac{1}{3}\delta) \tag{1.3}$$

où γ est le rapport gyromagnétique du proton (= rapport entre la fréquence de rotation et l'intensité du champ magnétique).

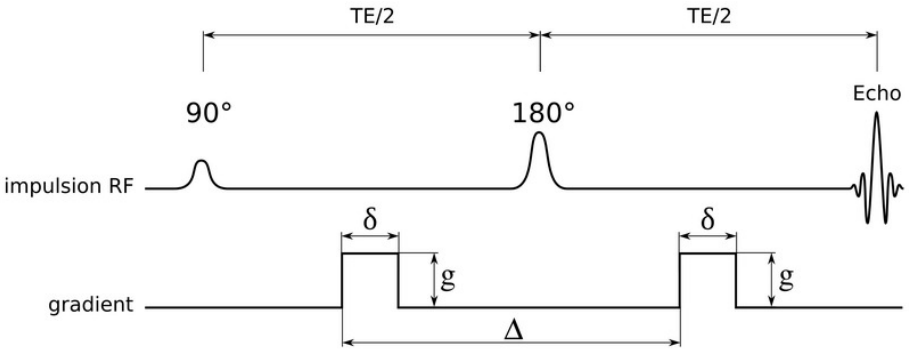


FIGURE 1.12 – *Graphique schématique pour représenter la séquence PGSE* [13]

Le signal de diffusion est modélisé par l'équation de Stejskal-Tanner :

$$I = I_0 \cdot e^{-b_{\text{PGSE}} \cdot ADC} \tag{1.4}$$

où I_0 est le signal de base (pondéré T_2) et ADC le coefficient apparent de diffusion, qui quantifie la mobilité moyenne des molécules d'eau dans le tissu [11, 10].

$$ADC = -\frac{\log\left(\frac{S(b)}{S_0}\right)}{b} \tag{1.5}$$

où $S(b)$ est le signal mesuré avec un facteur b donné, S_0 est le signal mesuré avec $b = 0$ (sans pondération en diffusion) et b est le facteur de sensibilité à la diffusion

Pour évaluer la diffusion dans toutes les directions, les gradients sont appliqués selon plusieurs axes. Cela permet de détecter une diffusion isotrope (égale dans toutes les directions) ou anisotrope (préférentielle, par exemple le long des axones de la substance blanche).

Généralement des images sont acquises avec un facteur $b = 0$ (pondération T_2 seule) et un facteur b élevé (avec une pondération en diffusion), ce qui permet de calculer l'ADC et de générer des cartographies quantitatives de la diffusion.

Tenseur de diffusion

Lorsque l'on réalise des acquisitions pondérées dans plusieurs directions, on peut extraire un tenseur de diffusion qui "synthétise" les informations. Le modèle de l'imagerie par tenseur de diffusion (DTI) permet de quantifier la directionnalité de la diffusion de l'eau en chaque point du cerveau. En modélisant la diffusion comme un ellipsoïde (fig 1.13), le tenseur fournit des informations sur l'orientation principale des fibres nerveuses, ce qui permet de calculer une cartographie des faisceaux de ces fibres.

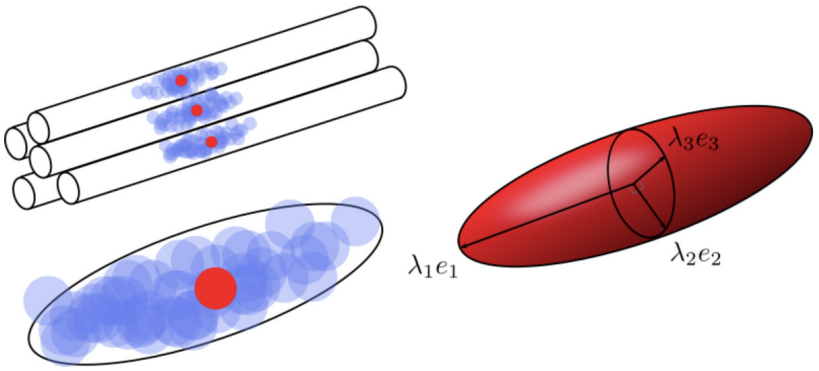


FIGURE 1.13 – *Illustration de la diffusion des molécules d'eau (Figure par Pierre Bellec sous licence CC-BY 4.0, inspirée par M. Descoteaux et C. Poupon)*

Les paramètres de ce modèle sont les directions principales de diffusions (e_1, e_2, e_3) et les valeurs de diffusions associées à ces directions $(\gamma_1 > \gamma_2 > \gamma_3)$. Il caractérise la diffusion : coefficient d'anisotropie, directions privilégiées et restreintes dans l'espace.

Cependant, DTI présente des limitations, notamment dans les régions où plusieurs faisceaux de fibres se croisent. Dans ces zones, le modèle tensoriel simple ne peut pas représenter adéquatement les orientations multiples. Pour surmonter ces limitations, des techniques avancées telles que le modèle multi-tensoriel ou l'imagerie à haute résolution angulaire (HARDI) ont été développés [14].

Tractographie

La tractographie est une technique d'imagerie qui permet d'estimer in vivo les trajectoires des fibres de matière blanche dans le cerveau. Elle repose sur les données issues de l'IRM de diffusion, en exploitant l'anisotropie des déplacements moléculaires de l'eau pour inférer la direction des connexions axonales.

Deux grandes approches peuvent être distinguées (fig 1.14). La tractographie dite déterministe consiste à suivre itérativement, à partir d'un point d'origine situé en substance blanche, la direction principale de diffusion jusqu'à ce que le trajet atteigne la matière grise ou ne puisse plus être poursuivi. Cette méthode génère une unique trajectoire par point de départ. En revanche, la tractographie probabiliste intègre l'incertitude inhérente à l'estimation de la direction des fibres : au lieu d'un unique tracé, elle génère un ensemble de trajectoires possibles, reflétant la variabilité potentielle autour de la direction principale. Cette approche permet ainsi une modélisation plus robuste dans les régions à architecture complexe ou à faible anisotropie.

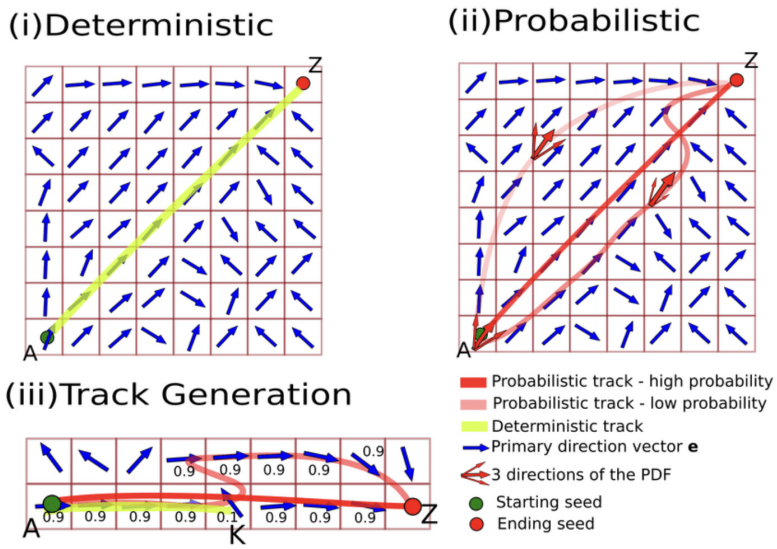


FIGURE 1.14 – *Exemple simplifié montrant les deux types d'approche de tractographie* [15]

Chapitre 2

Systèmes de coordonnées 3D en neuro-imagerie

La manipulation de données en neuro-imagerie nécessite une compréhension approfondie des systèmes de coordonnées tridimensionnels utilisés pour localiser et comparer les structures cérébrales. Ces systèmes fournissent un cadre standardisé permettant d'aligner et d'interpréter les données issues de différentes modalités d'imagerie.
Pour passer d'un espace de référence à un autre, on utilise des matrices de transformation affines. Ces matrices permettent d'exprimer les coordonnées d'une position spécifique initialement définie dans un référentiel, dans un autre système de coordonnées. Généralement, les matrices affines incluent des transformations géométriques élémentaires telles que les translations, les mises à l'échelle (scaling), les rotations et les déformations par cisaillement (shearing). L'application correcte de ces transformations garantit la cohérence spatiale nécessaire à la comparaison et à l'intégration multimodale des données.
Il convient de souligner que les termes "espace", "espace de référence", "référentiel" et "système de coordonnées" sont utilisés de manière interchangeable dans la littérature. Une compréhension claire de ces systèmes et de leur interrelation est fondamentale pour une analyse précise et reproductible des données de neuro-imagerie.

2.1 Conventions d'orientation des axes

Afin de clarifier la compréhension des différents espaces de référence, il convient de rappeler les trois axes anatomiques fondamentaux, dont la définition fait consensus dans la communauté scientifique :

- Gauche-droite (Left-Right)
- Supérieur-inférieur (Superior-Inferior) :
  - Supérieur = vers la tête
  - Inférieur = vers les pieds
- Antérieur-postérieur (Anterior-Posterior) :
  - Antérieur = vers l'avant du corps
  - Postérieur = vers l'arrière du corps

Les couleurs visibles sur la figure 2.1 sont utilisées à des fins purement illustratives et feront l'objet d'une explication détaillée dans la section 5.1.1. En pratique clinique, il existe une divergence dans la convention adoptée pour la représentation spatiale entre les radiologues et les neurologues [17, 18]. Les radiologues utilisent une convention selon laquelle les images sont visualisées comme si le praticien se tenait face au patient ; ainsi,

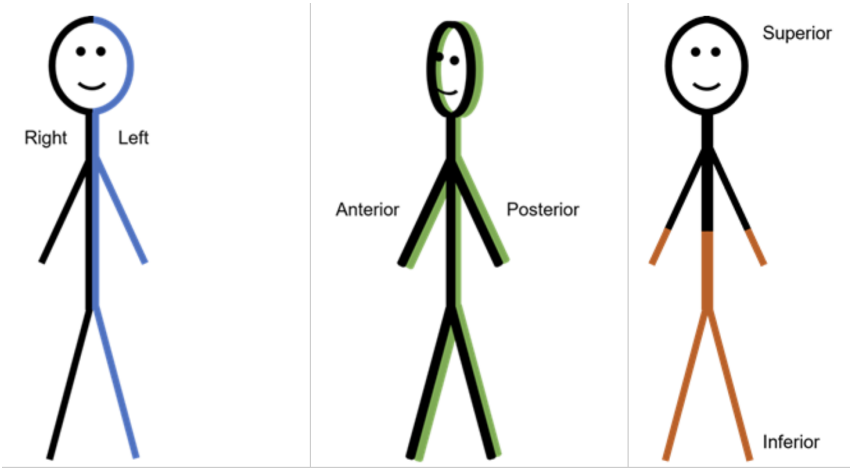


FIGURE 2.1 – *Illustration des plans de coupe anatomique (convention radiologique)* [16]

la gauche sur l'image correspond à la droite réelle du patient et inversement pour la droite. À l'inverse, les neurologues adoptent une perspective égocentrée, où la gauche et la droite sur l'image correspondent directement aux côtés gauche et droit du patient, respectivement. En revanche, les repères anatomiques verticaux et sagittaux (les directions inférieure/supérieure et antérieure/postérieure) sont communément admis par les deux disciplines et ne varient pas selon ces conventions. Nous utiliserons ici la convention neurologique lors de nos mentions de telles orientations.

2.1.1 RAS, RAS+

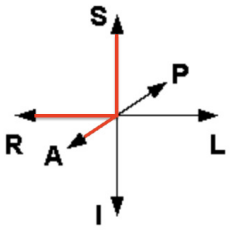


FIGURE 2.2 – *Orientation RAS+*

La notation RAS (*Right-Anterior-Superior*) est largement répandue dans le domaine de la neuro-imagerie, mais son interprétation peut varier selon les contextes logiciels ou les formats de données. Par convention, l'abréviation RAS désigne un système de coordonnées où l'axe x pointe vers la droite (*Right*), l'axe y vers l'avant (*Anterior*) et l'axe z vers le haut (*Superior*) (fig 2.2). Cette définition correspond à la convention neurologique. Cependant, certains formats définissent ces directions non pas en termes d'orientation des axes, mais en fonction du point d'origine à partir duquel les axes sont mesurés. Ainsi, dans certaines interprétations inversées, « R » pourrait désigner un axe orienté depuis la droite, auquel cas les coordonnées positives pointeraient vers la gauche et de même pour les axes y et z. Afin de lever toute ambiguïté, la notation RAS+ est utilisée pour indiquer explicitement que les directions positives des axes correspondent à la droite, à l'avant et au haut du patient, respectivement. Cette convention normalisée permet une interprétation univoque des coordonnées spatiales.

2.1.2 LPS, LPS+

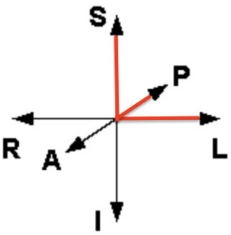


FIGURE 2.3 – *Orientation LPS+*

Le système de coordonnées LPS (Left-Posterior-Superior) est la convention standard adoptée par le format DICOM, largement utilisé pour le stockage et l'échange d'images médicales. Dans ce système, les axes sont définis comme suit : l'axe x va de la droite vers la gauche du patient, l'axe y de l'avant vers l'arrière et l'axe z du bas vers le haut (fig 2.3). La notation LPS+ indique que les directions positives

des axes correspondent respectivement à la gauche, l'arrière et le haut du patient [19].

2.1.3 LAS, LAS+

Le système de coordonnées LAS (Left-Anterior-Superior) correspond à la convention radiologique. Dans ce système, les axes sont orientés de la manière suivante : l'axe x va de la droite vers la gauche du patient, l'axe y de l'arrière vers l'avant et l'axe z du bas vers le haut (fig 2.4). Cette convention peut être utilisée dans certains logiciels ou formats de données et il est important de la reconnaître pour assurer une interprétation correcte des données. Comme pour les autres systèmes de coordonnées, une attention particulière doit être portée lors de l'intégration ou de la conversion de données entre différents systèmes pour maintenir la cohérence spatiale [20].

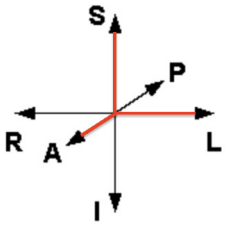


FIGURE 2.4 – *Orientation LAS+*

2.2 Les espaces de référence

Nous examinerons ici les espaces de référence et les standards les plus fréquemment employés en neuro-imagerie, sans en faire une liste exhaustive, afin d'en clarifier la terminologie et d'analyser leurs différences respectives. Certains logiciels utilisent parfois leur propre système de coordonnées (comme FreeSurfer ou BrainVoyager) mais nous n'en parlerons pas dans cette section. La table 2.1 propose un résumé concis de ce qui est discuté plus en détail ci-dessous. Un système de coordonnées tridimensionnel est défini par un point particulier appelé «origine», correspondant à l'intersection de trois axes orthogonaux (habituellement notés x, y et z). Pour comprendre et analyser un espace de référence, nous devons donc savoir :

1. Quelle est l'origine (0,0,0) ?
2. Quelle est l'orientation des 3 axes (+x, +y et +z) ?
3. Quelle est l'unité employée (millimètre, centimètre, voxels, ...) ?
4. La géométrie/forme est-elle adaptée/normalisée à un modèle/atlas ou bien est-ce qu'elle correspond toujours aux données brutes de l'individu ?

Ces axes, perpendiculaires les uns aux autres, permettent de décrire précisément toute position dans l'espace en termes de coordonnées numériques, facilitant ainsi la représentation spatiale, la visualisation et l'analyse des structures anatomiques ou fonctionnelles du cerveau.

2.2.1 L'espace monde

Le système de coordonnées monde est un repère cartésien global unique dans lequel tous les modèles (scanner, patient, instruments, etc.) sont positionnés et orientés (fig 2.5). Chaque modèle possède son propre système de coordonnées local pour décrire sa géométrie interne mais pour les faire interagir ou les superposer les uns par rapport aux autres, leurs coordonnées locales peuvent être transformées en coordonnées monde. Dans le domaine de la neuro-imagerie, les unités sont exprimées en millimètre et en général un point fixe de la salle d'examen (par exemple le centre géométrique de l'IRM ou un repère magnétique de la salle) est utilisé comme origine (0, 0, 0). Les axes sont orientés de façon cartésienne

standard (par exemple : X vers l'avant de la salle, Y vers la droite, Z vers le haut) mais peuvent varier d'un laboratoire à un autre (fig 2.5 propose un type d'orientation. Le « repère *monde* » fait référence au système de coordonnées global interne à une scène ou à un protocole donné et ne représente pas un repère universel entre les différents lieux médicaux.

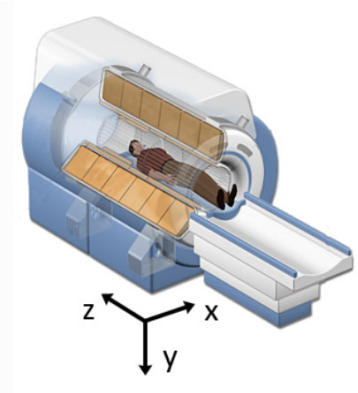


FIGURE 2.5 – *Illustration d'un système de coordonnées monde* [21]

2.2.2 L'espace natif ou l'espace patient

Les termes *espace natif* et *espace patient* sont utilisés de manière interchangeable pour désigner le référentiel anatomique propre à chaque individu, tel qu'il est défini au moment de l'acquisition des images médicales. Dans cet espace, les coordonnées sont exprimées en millimètres et font directement référence à la position réelle du patient dans le scanner, sans qu'aucune transformation géométrique n'ait encore été appliquée. L'origine est placée sur un point anatomique de référence (souvent la commissure antérieure (AC) du cerveau, ou une autre référence fixée lors de l'acquisition). Les trois axes anatomiques de l'espace patient sont définis comme suit (illustré sur la figure 2.6) :

- le plan **sagittal**, souvent noté **X** : perpendiculaire au sol et sépare la gauche de la droite.
- le plan **coronal**, souvent noté **Y** : perpendiculaire au sol et sépare l'avant (Antérieur) de l'arrière (Postérieur).
- le plan **axial**, souvent noté **Z** : parallèle au sol et sépare la tête (supérieure) des pieds (inférieurs).

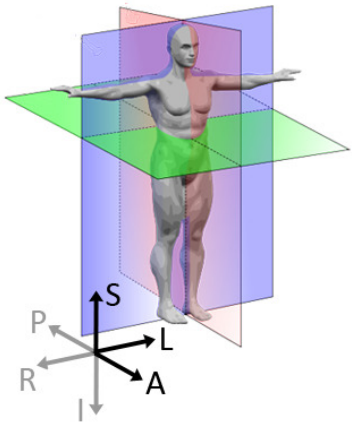


FIGURE 2.6 – *Représentation visuelle des trois axes anatomiques (axial, coronal et sagittal)* [21]

Cet espace, dont les axes suivent les notations vues au point 2.1, reflète l'état brut des données immédiatement après l'acquisition IRM : exprimées dans le référentiel propre au patient, sans enregistrement ni alignement avec un espace standardisé. Il constitue ainsi le point de départ de nombreuses étapes de prétraitement en neuro-imagerie.

2.2.3 L'espace voxel ou l'espace intrinsèque

Les termes *espace voxel* et *espace intrinsèque* sont fréquemment employés de manière interchangeable dans la littérature scientifique pour désigner le système de coordonnées associé à la grille de voxels d'une image volumique. L'origine de l'espace voxel est définie comme le centre du premier pixel (dans le cas d'une image 2D comme sur la figure 2.7) ou du premier voxel (dans une image 3D) représenté sur la figure 2.8, ce qui signifie que les axes i, j, k partent d'un coin de l'image, en suivant l'ordre de stockage des données. L'unité de ces axes est exprimée en voxels et la (taille) résolution d'une telle image est alors exprimée en nombre de voxels, qui eux-mêmes ont une taille (souvent en mm) physique. Ces indices i, j et k sont utilisés pour parcourir les dimensions de la matrice ((i, j, k) = position du voxel dans la matrice de l'image) image et peuvent être associés aux axes physiques x, y ou z selon l'orientation et le système de coordonnées adopté. Il est donc important de ne pas confondre ces indices de grille avec les coordonnées spatiales exprimées dans un espace anatomique réel [16].

FIGURE 2.7 – *Image 2D, type d'orientation selon Matlab* [16]

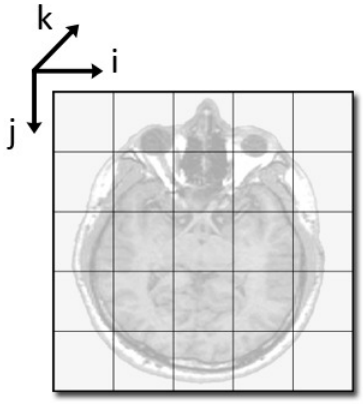


FIGURE 2.8 – *Représentation du référentiel intrinsèque d'une image 3D* ([21])

2.2.4 Le standard MNI

L'espace MNI (Montreal Neurological Institute) est un système de coordonnées standardisé utilisé en neuro-imagerie pour permettre la comparaison des données cérébrales. Il définit un cerveau standard moyen obtenu à partir de l'alignement d'images IRM de plusieurs sujets sains. Le modèle MNI152, par exemple, est dérivé de l'alignement de 152 cerveaux [22]. Son origine est située au niveau de la commissure antérieure [23]. Les axes de l'espace MNI sont orientés selon la convention RAS+ : l'axe X va de la gauche vers la droite, l'axe Y de l'arrière vers l'avant et l'axe Z du bas vers le haut [24].

2.2.5 Le standard Talairach

L'espace de Talairach (également appelé Talairach-Tournoux) est un système de coordonnées tridimensionnel introduit par Jean Talairach et Pierre Tournoux en 1988. Il est basé sur l'anatomie d'un cerveau humain unique et utilise la commissure antérieure (AC) comme origine. Les axes sont définis comme suit : l'axe X va de la gauche vers la droite, l'axe Y de l'arrière vers l'avant (le long de la ligne AC-PC [1]) et l'axe Z du bas vers le haut. Ce système permet de localiser les structures cérébrales de manière proportionnelle, indépendamment des variations individuelles de taille et de forme du cerveau. Bien que l'espace de Talairach ait été largement utilisé, il est progressivement remplacé par l'espace MNI dans les études de neuro-imagerie modernes [24].

1. AC-PC pour "Anterior Commissure - Posterior Commissure" [25]

TABLE 2.1 – *Schéma récapitulatif des repères et espaces de coordonnées*

Repère	**Origine typique**	**Axes orientés selon. . .**	**Unités**	**Usage principal**
Monde (X_m, Y_m, Z_m)	Repère global (centre de la salle d'examen ou de l'IRM)	Cartésien standard (ex. X : avant, Y : droite, Z : haut)	Millimètres	Fusion de modèles, calibration d'instruments, superposition multi-modalités
Patient/Anatomical (X_p, Y_p, Z_p)	Point anatomique de référence (souvent la commissure antérieure de l'individu)	Convention anatomique (X : gauche–droite, Y : postérieur–antérieur, Z : bas–haut)	Millimètres	Localisation anatomique, mesures sur volumes (morphométrie, ROI), coregistration intra-sujet
Image $(\mathrm{i}, \mathrm{j}, \mathrm{k})$	Coin d'une matrice de pixels/voxels $(0, 0, 0)$	Indices de lignes/-colonnes/coupes (sens des lignes d'image)	Indices	Stockage brut, accès aux matrices de voxels, visualisation slice par slice
MNI $(X_{\mathrm{MNI}}, Y_{\mathrm{MNI}}, Z_{\mathrm{MNI}})$	Commissure antérieure (AC) du cerveau standard MNI (p. ex. MNI152)	Convention RAS+	Millimètres	Normalisation dans l'espace template, comparaison inter-sujets, utilisation d'atlas
Talairach $(X_{\mathrm{Tal}}, Y_{\mathrm{Tal}}, Z_{\mathrm{Tal}})$	Commissure antérieure (AC) du cerveau de référence Talairach	Convention RAS+	Millimètres	Localisation proportionnelle dans le Talairach-Tournoux atlas, historique en neuro-imagerie, cartographie de structures cérébrales

2.3 Matrices de transformation spatiale

Les transformations affines sont essentielles en neuro-imagerie pour passer d'un espace de référence à un autre ou pour aligner des images entre elles. Elles relient en quelque sorte les indices d'un voxel (i, j, k) à des coordonnées physiques réelles (x, y, z) dans un espace tridimensionnel (souvent en millimètres). Elles permettent de modéliser des opérations géométriques telles que la translation, la mise à l'échelle, la rotation et le cisaillement. Ces transformations sont généralement représentées par des matrices 4×4 (fig 2.9 en coordonnées homogènes, facilitant leur combinaison et leur application successive.

$$\mathbf{M} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

FIGURE 2.9 – *Matrice affine* 4×4 [26]

La sous-matrice 3×3 représente la **rotation** ou le **scaling** (mise à l'échelle) ou le **shearing** (cisaillement). Et enfin, la matrice colonne 3×1 représente la **translation** [26]. Utilisé dans un système de coordonnées, la matrice 3×3 représente une orientation dans l'espace tandis que le matrice 3×1 représente une position dans l'espace. Pour mieux en appréhender le fonctionnement, nous allons décomposer cette matrice 4×4 étape par étape. La figure 2.10 en propose une illustration visuelle synthétique.

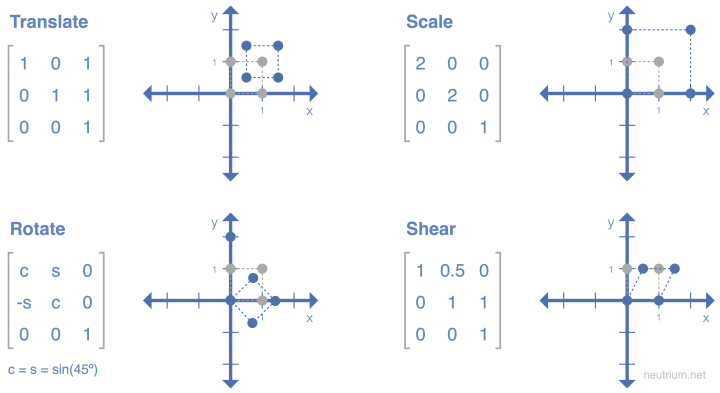


FIGURE 2.10 – *Illustrations des opérations géométriques* [27]

2.3.1 Translation

La translation déplace chaque point (ou voxel) d'une image d'un vecteur fixe (t_x, t_y, t_z) le long des axes du système. En coordonnées homogènes, la matrice de translation est :

$$T = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Cette opération est utilisée pour repositionner des images ou des structures anatomiques dans un espace de référence commun.

2.3.2 Mise à l'échelle (Scaling)

La mise à l'échelle modifie la taille des objets le long des axes x, y et z en les multipliant par des facteurs (s_x, s_y, s_z). La matrice correspondante est :

$$S = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Cette transformation est cruciale pour ajuster les dimensions des images à des échelles standardisées, notamment lors de la normalisation vers des atlas cérébraux.

2.3.3 Rotation

La rotation fait pivoter les points autour d'un axe donné. Cette matrice est légèrement plus complexe que les autres car elle nécessite l'utilisation de toute la matrice 3×3. Pour expliquer cela plus clairement, nous pouvons diviser la rotation en trois parties, une pour chaque axe autour duquel nous effectuons une rotation. Même si évidemment, c'est l'ensemble de ces 3 parties qui forment la rotation complète.

Une matrice de rotation autour de l'axe x par un angle θ est :

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Une matrice de rotation autour de l'axe y par un angle θ est :

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Une matrice de rotation autour de l'axe z par un angle θ est :

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2.3.4 Cisaillement (Shear)

Le cisaillement déforme l'image en déplaçant les points proportionnellement à leur position le long d'un axe formant une sorte d'inclinaison. Cette transformation est utilisée pour modéliser des déformations non uniformes. La matrice de cisaillement sur l'axe x est :

$$C_x(\theta) = \begin{bmatrix} 1 & \cot\theta & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Chapitre 3

Diversité et enjeux des formats de fichiers

L'analyse de l'imagerie par résonance magnétique anatomique, fonctionnelle (IRMf) et de diffusion (IRMd) repose sur un ensemble d'outils logiciels spécialisés, chacun apportant des fonctionnalités particulières pour le prétraitement, l'analyse statistique et la visualisation des données. Ce chapitre propose une sélection ciblée de logiciels de neuro-imagerie, choisis non pas dans une optique d'exhaustivité, mais dans le but de mettre en lumière quelques outils utilisant ou générant les mêmes formats de fichiers que ceux pris en charge par l'outil développé dans ce travail. Une présentation plus large des autres visualiseurs disponibles (la plus complète possible) est proposée en annexe (annexe B). Un tableau synthétique (table 3.1) est proposé afin d'offrir une vue d'ensemble claire et directe sur la diversité des outils existants dans le domaine. Enfin, la problématique de cette diversité dans les formats de fichiers est abordée.

3.1 Panorama des outils de visualisation pour la neuro-imagerie

3D Slicer

3D Slicer est une plateforme logicielle modulaire libre, utilisée en neuroimagerie pour la visualisation, le traitement et l'analyse de données médicales. Elle prend en charge une vaste gamme de formats (NIfTI, DICOM, NRRD) et permet l'affichage interactif en 2D, 3D, la segmentation manuelle ou automatique, l'analyse de surfaces, la fusion d'images multimodales et la planification chirurgicale. Le module SlicerDMRI ajoute le support de tractographies, de mesures de connectivité et de la reconstruction de fibres à partir de données DWI. Les résultats peuvent être exportés en formats standards (.nii, .vtk, .trk, .csv) [28].

BrainVoyager Viewer

BrainVoyager (BV) est un logiciel professionnel sous une licence payante qui propose un pipeline complet de visualisation et d'analyse incluant la création de documents FMR (volume fonctionnel), VMR (anatomie), SRF (surface), STC (temps) et ses autres formats propriétaires qu'on ne retrouve nulle part mais il supporte aussi le format Nifti. Son visualiseur offre des coupes 2D/3D, la reconstruction et le lissage de surfaces corticales, la

superposition de segmentations, ainsi qu'un rendu volumique avancé pour l'exploration des cartes fonctionnelles en profondeur [29].

MRIcroGL

MRIcroGL [30] est la version complète et améliorée de MRIcron [31]. Il fournit un rendu 2D et 3D, avec support du format NIfTI par défaut. Son interface drag-and-drop intègre un convertisseur DICOM→NIfTI, des fonctionnalités d'overlay de cartes statistiques (couleur, transparence) et la création de mosaïques personnalisées via un éditeur graphique ou des scripts. Les formats supportés pour la visualisation sont NIfTI (.nii, .nii.gz, .hdr/.img), .VOI, Bio-Rad Pic (.pic), NRRD, Philips (.par/.rec), ITK MetaImage (.mhd, .mha), AFNI (.head/.brik) et Freesurfer (.mgh, .mgz).

MRView

MRView est l'interface de visualisation de la suite MRtrix3, conçue pour l'affichage interactif de volumes anatomiques et de tractographies. Il supporte nativement les formats NIfTI (.nii, .nii.gz), DICOM, tiff, PNG, mgh, etc ainsi que les fichiers .tck générés par MRtrix pour les streamlines. MRView permet l'affichage multi-couches, le rendu 3D, la visualisation des tractographies, le contrôle du slicing, des colormaps, de la transparence, etc [32, 33].

TrackVis

TrackVis est un outil de visualisation 3D conçu pour afficher les tractographies générées par Diffusion Toolkit. Il supporte les fichiers .trk, un format propriétaire mais largement utilisé et offre une navigation fluide dans les streamlines avec colorisation selon la direction, le groupe ou les métriques. TrackVis inclut des outils de sélection de régions d'intérêt (ROI), d'analyse de faisceaux et d'export des visualisations en PNG ou TIFF. L'interface est optimisée pour le diagnostic rapide ou la recherche clinique [34, 35].

TABLE 3.1 – *Comparatif de visualiseurs existants en neuro-imagerie (36 outils)*

Visualiseur	Logiciel associé	Formats supportés	Licence
3D Slicer / SlicerDMRI	3D Slicer	.nii, .img/.hdr, .dcm, NRRD, .vtk, .trk	Open source
AFNI GUI	AFNI	.nii, .img/.hdr, .HEAD/.BRIK	Open source
Amira	Amira (Thermo Fisher)	.dcm, .nii, .img/.hdr, .obj	Commercial
Anatomist	BrainVISA	.nii, .img/.hdr, MINC, .ima, .arg, .tex	Open source
BioImage Suite	Yale / NIH	.dcm, .nii, .img/.hdr, .bval/.bvec	Open source
BrainBrowser	BrainBrowser	.nii, .img/.hdr, .gii	Open source
BrainNet Viewer	MATLAB	.nii, .img/.hdr, node/edge/surf	Open source
BrainSuite	BrainSuite	.nii, .img/.hdr, formats internes	Open source
BrainVoyager Viewer	BrainVoyager	.vmr, .fmr, .fbr, .nii, .img/.hdr, .srf, .stc, ...	Commercial
BrainanceMD Viewer	BrainanceMD	.dcm	Commercial
Brainstorm	MATLAB	.img/.hdr, .ima/.dim, .mri, .mnc, .fif	Open source
FSLeyes	FSL	.nii, .img/.hdr, .mgz, .mgh	Open source
FreeView	FreeSurfer	.nii, .img/.hdr, .mgz/.mgh	Open source
IMAIOS DICOM Viewer	IMAIOS	.dcm	Gratuit
ITK-SNAP	ITK-SNAP	.nii, .img/.hdr, .dcm	Open source
InVesalius	InVesalius	.dcm	Open source
Mango	Mango	.nii, .img/.hdr, .dcm, .mnc, .gii	Gratuit (non open source)
MicroDicom	MicroDicom	.dcm	Gratuit (non libre)
MNI Display	MNI BIC	.nii, .img/.hdr, .mnc, .ply, .dfs	Open source
MIPAV	NIH	.dcm, .nii, .img/.hdr, Analyze	Open source
MRIcroGL	MRIcroGL	.voi, .nii, .img/.hdr, .head/.brik, .mgh, .mgz, .mhd, .mha	Open source
MRView	MRtrix3	.nii, .img/.hdr, .dcm, .tck	Open source

À suivre...

Suite du tableau 3.1

Visualiseur	Logiciel associé	Principaux formats	Licence
NeuroElf GUI	NeuroElf (MATLAB)	.nii, .img/.hdr, .vmr, .sdm, .voi, .fmr, ...	Gratuit (nécessite MATLAB)
Olea Medical	Canon Medical	.dcm	Commercial
OHIF Viewer	OHIF / Cornerstone	.dcm	Open source
Orthanc	Orthanc Team	.dcm	Open source
OsiriX	Pixmeo	.dcm	Gratuit /Commercial
Papaya	Web (Javascript)	.nii, .img/.hdr, .dcm, .gii	Open source
Pycortex	Python/VTK	.nii, .gii	Open source
PySurfer	Python	.mgh/.mgz, .pial	Open source
RadiAnt DICOM Viewer	Medixant	.dcm	Gratuit (non libre)
SliceDrop	SliceDrop	.dcm, .nii, .img/.hdr, .mgh/.mgz, .nrrd, .vtk, .trk	Open source
TrackVis	TrackVis	.trk	Gratuit, non libre
Voreen	Voreen	.dcm, .nii, .raw, .tiff, .hdf5	Open source
VolView	Kitware	.dcm, .nii, .img/.hdr, .nrrd	Open source (Kitware)
Weasis	Weasis	.dcm	Open source

3.2 Diversité des formats de fichiers en neuro-imagerie

Les outils logiciels actuellement disponibles sur le marché de la recherche en neuro-imagerie offrent, dans l'ensemble, des fonctionnalités robustes, diversifiées et adaptées aux besoins des chercheurs et cliniciens. Ces solutions couvrent de nombreux aspects de l'analyse de données cérébrales, allant du prétraitement aux visualisations avancées en 3D. Cependant, un problème structurel persiste dans l'écosystème logiciel : une grande hétérogénéité de formats de fichiers, chacun développé pour répondre à des besoins spécifiques.

Parmi les formats les plus répandus, on distingue notamment le **format DICOM** (Digital Imaging and Communications in Medicine), qui constitue le standard de facto dans les environnements hospitaliers. C'est un format complet et complexe qui permet de stocker à la fois les images et les métadonnées associées au patient et à l'acquisition. Dans le milieu de la recherche, le **format NIfTI** (Neuroimaging Informatics Technology Initiative) est largement utilisé pour l'analyse d'images structurelles et fonctionnelles, notamment en IRMf, en raison de sa simplicité, de sa structure compacte et de sa compatibilité avec des pipelines analytiques populaires. Ce format s'est développé après le format DICOM qui est plus ancien et historiquement adopté dans les milieux cliniques. L'adoption du NIfTI dans le secteur médical reste progressive, notamment en raison des exigences strictes de certification réglementaire et de la nécessité de maintenir la compatibilité avec des équipements existants basés sur le standard DICOM. Le format NIfTI permet une visualisation rapide et complète des images cérébrales mais il stocke uniquement les éléments responsable de l'affichage de l'image tandis que le DICOM est un format bien plus complet en terme de données patient, acquisition, etc.

À côté de ces formats standards, de nombreux formats propriétaires ou spécifiques à certaines applications ont vu le jour. C'est notamment le cas des formats utilisés dans BrainVoyager (BV), tels que **VMR** (Voxel Map Representation) pour les volumes anatomiques et **VOI** (Volume of Interest) pour les régions d'intérêt ou encore **FBR** pour les tracking de fibres. A titre d'exemple, le logiciel BrainVoyager comptabilise à lui seul un peu plus de 50 formats propriétaires. Dans le domaine de la tractographie, les formats **TRK** (TrackVis), **TCK** (MRtrix), **TRX**, **TRACT**, **TRAKO**, **FIB** (DSI Studio), **VTK**, **VTP** ou **DPY** (DIPY) et beaucoup d'autres illustrent bien la fragmentation actuelle : chacun est lié à un outil ou une communauté logicielle, souvent avec des spécifications techniques non compatibles entre elles.

La plupart des outils disponibles ne sont compatibles qu'avec un sous-ensemble limité de formats, généralement restreint à ceux développés par la communauté à laquelle l'outil appartient, ou à ceux adoptés historiquement.

Cette limitation crée deux types de contraintes pour l'utilisateur :
- soit le format de ses données n'est pas reconnu par le logiciel choisi, l'empêchant de poursuivre son analyse sans possibilité de conversion ;
- soit il est contraint d'utiliser plusieurs outils différents pour traiter un même jeu de données, complexifiant ainsi son flux de travail.

Un autre facteur aggravant réside dans la tendance fréquente de certains logiciels à générer leur propre format propriétaire. Cette dynamique est motivée par la volonté d'optimiser les performances internes ou de structurer les données selon une logique spécifique, mais elle contribue également à accentuer la dispersion des standards. Il en

résulte une prolifération de formats parallèles, parfois incompatibles entre eux, rendant difficile la construction de pipelines de traitement homogènes et aggravant le problème initial reconnu de tous qu'est l'hétérogénéité du type de format.

Cette situation est d'autant plus paradoxale que, dans la littérature scientifique, les forums de discussion spécialisés et les échanges entre professionnels, un consensus semble exister sur la nécessité de rationaliser et de standardiser les formats. Des initiatives comme le format DICOM pour les données cliniques ou NIfTI pour les données de recherche ont tenté de répondre à cette problématique. Pourtant, malgré cet accord apparent sur le besoin de convergence, chaque communauté tend à proposer son propre format qu'elle estime optimal, souvent au détriment de l'interopérabilité.

En conséquence, plutôt que de s'orienter vers une solution consolidée, le domaine continue d'enrichir un paysage déjà fragmenté. Le nombre croissant de formats rend les processus de traitement et de visualisation plus complexes, ralentit l'adoption de solutions unifiées et alourdit considérablement la charge technique pour les utilisateurs finaux.

Chapitre 4

Problématique actuelle

La neuro-imagerie constitue un pilier essentiel pour la recherche neuroscientifique ainsi que pour le diagnostic et le suivi de nombreuses pathologies cérébrales.

Les différentes techniques d'imagerie cérébrale (structurelle, fonctionnelle et de diffusion) fournissent chacune une perspective spécifique sur le cerveau.

Bien que puissantes individuellement, ces approches prennent tout leur sens lorsqu'elles sont combinées. En effet, intégrer et traiter ces modalités de manière conjointe permet une compréhension plus globale et plus fine du fonctionnement cérébral. Une telle analyse multi-modale permet, par exemple, de corréler une activation fonctionnelle à une structure anatomique précise ou d'étudier comment la connectivité structurelle (tractographie) influence les dynamiques fonctionnelles mesurées.

Dans ce contexte, la capacité à visualiser et manipuler des données issues de plusieurs modalités s'avère un vrai plus. Elle facilite la validation croisée des résultats, améliore l'interprétabilité et répond aux exigences croissantes de la recherche neuroscientifique contemporaine qui tend vers une intégration systémique des données.

Les systèmes de coordonnées 3D utilisés en neuro-imagerie permettent d'aligner et interpréter les données issues des différentes modalités d'imagerie. Toutefois, il existe une divergence dans la convention adoptée pour la représentation spatiale selon les contextes logiciels ou les formats de données.

Le domaine de la neuro-imagerie est caractérisé par une grande **hétérogénéité de formats de fichier**. Ces formats émergent parfois dans le but d'offrir une alternative technique supposée plus performante ou plus adaptée aux besoins spécifiques que les formats précédents. Dans d'autres cas, ces nouveaux formats répondent à des stratégies commerciales visant à maintenir les utilisateurs captifs d'un logiciel particulier en proposant des formats propriétaires, compatibles uniquement avec leurs propres outils.

Cette multiplicité des formats pose des défis majeurs en matière de compatibilité logicielle et de reproductibilité scientifique. Elle engendre aussi plusieurs difficultés concrètes pour les utilisateurs :

- Interopérabilité réduite : Les données issues de différentes plateformes ou logiciels ne peuvent souvent pas être directement intégrées.
- Complexité des pipelines d'analyse : Lorsqu'il faut convertir manuellement les données entre différents formats, cela augmente la charge de travail, allonge les

délais de traitement et accroît le risque d'erreurs.
— Freins à la diffusion des connaissances : Une trop grande diversité de formats de fichiers peut restreindre le choix des logiciels exploitables, ce qui empêche les utilisateurs de tirer parti des avantages spécifiques offerts par chaque outil.

C'est dans cette perspective que s'inscrit la contribution de ce projet, en posant les bases d'un outil de conversion de différents formats de fichier et de visualisation intégrée.

Deuxième partie

Contribution du projet

Chapitre 1

Présentation du projet : Objectifs et impact

La neuro-imagerie s'appuie sur diverses modalités pour l'acquisition des images. Chacune de ces approches génère des types de données spécifiques, nécessitant l'utilisation de formats de fichiers adaptés. Les multiples logiciels destinés à l'analyse statistique, à la visualisation ou au traitement de données sont également à l'origine de nouveaux formats de fichier.

Face à cette situation problématique engendrée par l'hétérogénéité des formats en neuro-imagerie, deux approches principales peuvent être envisagées.

La première consisterait à établir un consensus général autour d'un format unique, afin de standardiser la gestion des données au sein de la communauté scientifique et clinique. Cependant, cette approche se heurte souvent à des difficultés majeures, en raison des intérêts divergents des différents acteurs impliqués (équipes de recherche, éditeurs de logiciels, industriels, etc), chacun privilégiant généralement les formats propres aux outils avec lesquels ils sont habitués ou leurs pratiques existantes [36]. Cette solution reste donc faisable en théorie et elle faciliterait grandement le travail effectué dans le domaine de la neuro-imagerie. Un format unique permettrait de concentrer l'ensemble des efforts d'optimisation, de précision et de recherche en général. Cependant cette solution dépend considérablement du facteur humain car bien que possible à réaliser techniquement, il est impératif d'avoir, en amont, un consensus global. Et actuellement nous observons aisément que ce consensus n'est pas encore atteint et que cela fait plusieurs années que c'est en cours de discussion.

La seconde approche, plus pragmatique et réalisable à court terme, sans nécessiter de consensus général compliqué à obtenir, est la mise en place d'une solution technique agissant comme une passerelle ou une « boîte noire » logicielle. Une telle solution permettrait de visualiser et de convertir automatiquement plusieurs formats de données cérébrales utilisés actuellement. Cette approche présente l'avantage majeur de respecter les pratiques et habitudes existantes, tout en assurant une certaine interopérabilité. L'idée derrière cette approche est également, à plus long terme, de favoriser une réduction progressive du nombre de formats utilisés pour agir comme un véritable « entonnoir » et amener une simplification du paysage des formats en neuro-imagerie.

Cette « boîte noire » serait également capable de fournir des fonctionnalités de visualisation tridimensionnelle des données, facilitant l'interprétation et l'analyse des résultats. En automatisant ces tâches souvent laborieuses et sources d'erreurs humaines, cette solution contribuerait significativement à la standardisation et à la simplification

des flux de travail en neuro-imagerie. Elle favoriserait ainsi l'accès aux données, leur réutilisation et une collaboration facilitée entre équipes et institutions.

Enfin, la mise en œuvre d'un tel outil pourrait s'inscrire pleinement dans une démarche plus large de science ouverte, visant à promouvoir non seulement l'interopérabilité technique des données neuroscientifiques, mais également leur accessibilité universelle. Elle permettrait à la communauté scientifique et académique de tirer pleinement profit des avantages offerts par chaque logiciel spécialisé, sans se heurter aux barrières techniques actuelles liées à la fragmentation des formats. Cette approche constituerait donc un pas vers une exploitation optimale des données neuroscientifiques, au bénéfice direct des étudiants du domaine, des patients, des chercheurs et des professionnels de santé.

C'est en poursuivant les objectifs de cette seconde approche que ce projet a été construit.

Concrètement, en partant des besoins du terrain qui étaient exprimés par *l'Institute of Neuroscience (IoNS)* de l'UCLouvain, la réflexion autour d'un outil "boite noire" a ainsi débuté afin de proposer une Interface Utilisateur Graphique (IUG) universelle pour la gestion de l'hétérogénéité des formats d'imagerie cérébrale via conversion automatique et visualisation intégrée. L'outil créé au terme de ce projet a été nommé "*VisuBrain*".

Chapitre 2

Méthodologie de développement

2.1 Cadre méthodologique

Dans le cadre de ce mémoire, la mise en place d'une méthodologie de développement rigoureuse et explicitement définie s'est imposée comme une condition essentielle pour assurer la cohérence et l'évolutivité de l'outil conçu. L'instauration d'un cadre structurant les modalités de travail et les interactions avec le client est essentielle pour garantir la cohérence, la qualité et l'évolutivité du projet. Dans ce contexte, la philosophie de développement *Agile* [37] a été adoptée en concertation avec l'équipe encadrante et plus spécifiquement, la méthode SCRUM a été retenue comme cadre organisationnel [38, 39].

La méthode SCRUM repose sur une planification itérative du travail à travers des cycles/phases court(e)s appelé(e)s *"sprints"*, permettant une évaluation régulière de l'avancement du projet et une validation progressive des fonctionnalités développées. Cette approche ne force pas l'usage de technologies particulières, mais privilégie le fait de recevoir des feedbacks (retours) constants et "rapprochés" dans le temps. Elle offre ainsi une grande flexibilité qui facilite l'adaptation aux demandes client et aux nouvelles idées.
La mise en œuvre de cette méthodologie a permis de structurer le développement autour de quatre points centraux : l'interaction humaine avec les utilisateurs finaux (ou en tout cas un échantillon représentatif de ceux-ci), la construction incrémentale de prototypes fonctionnels, la collaboration étroite avec le client et surtout la capacité à s'adapter au changement. La méthode repose sur le postulat que les connaissances initiales sont faibles et que la compréhension du produit, des besoins et du contexte évolue au fil du processus de développement. Cette dynamique d'apprentissage continu et d'amélioration incrémentale a fait de SCRUM un cadre particulièrement pertinent et efficace pour conduire à bien le présent travail.

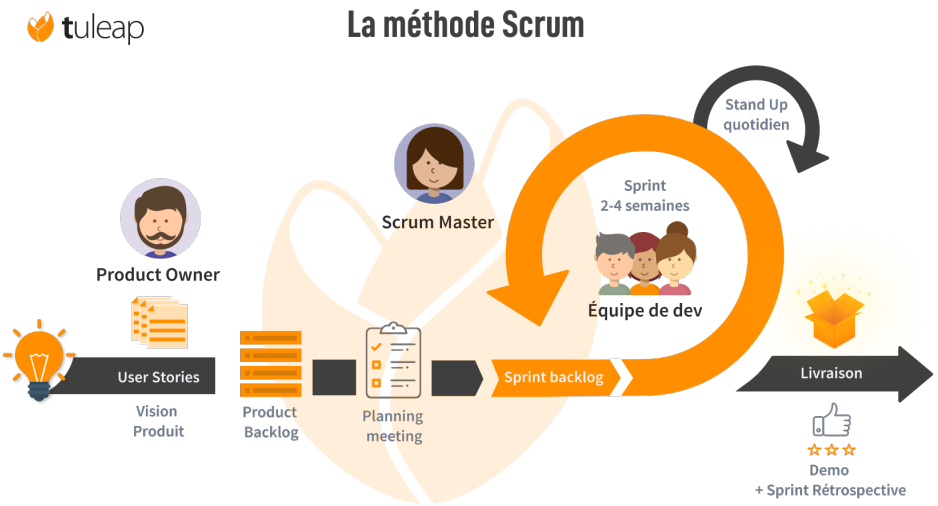


FIGURE 2.1 – *Illustration de la méthode SCRUM* [40]

L'illustration 2.1 synthétise le fonctionnement général de la méthode SCRUM. Il est toutefois essentiel de garder à l'esprit que cette méthode, bien qu'appuyée sur un cadre théorique structuré, reste fondamentalement adaptable. Chaque projet possède ses spécificités et la mise en œuvre de SCRUM doit être ajustée en fonction du contexte et des besoins particuliers.

Dans le cadre de ce travail, l'organisation adoptée reposait sur un rythme de réunions toutes les trois semaines avec l'équipe encadrante et le partenaire clinique IONS (le "client"). Ces points de contact réguliers permettaient de clarifier les objectifs du sprint en cours, de faire le point sur les éléments réalisés ou mal compris et d'ajuster collectivement la direction du projet. Des suggestions et axes d'amélioration étaient formulés à cette occasion, afin d'être mis en œuvre lors du sprint suivant. Par ailleurs, certaines avancées pouvaient également faire l'objet de discussions ponctuelles entre deux réunions, en fonction des besoins ou des urgences identifiées.

2.2 Conception logicielle et structuration modulaire du code

Le développement de l'outil s'est appuyé sur une architecture modulaire 100% en Python, visant à assurer la lisibilité, la maintenabilité et l'évolutivité du code . Le code est structuré en plusieurs modules spécialisés (fig 2.2), chacun répondant à une responsabilité fonctionnelle précise dans le cycle de traitement des fichiers d'imagerie cérébrale : conversion de formats, visualisation, gestion des sessions, interface utilisateur, etc.

L'un des aspects fondamentaux (si pas le plus déterminant) de ce travail réside dans sa volonté explicite d'évolutivité et d'ouverture à l'amélioration continue. Comme cela a été souligné dans les sections précédentes, le domaine de la neuro-imagerie informatisée est en perpétuelle évolution et possède une hétérogénéité croissante des formats de fichiers. L'outil développé dans le cadre de ce mémoire constitue ainsi une première brique dans un projet plus large, destiné à s'enrichir progressivement de nouvelles fonctionnalités, de formats supplémentaires et d'optimisations diverses.

Au-delà de l'ajout de conversion de fichiers, la conception de l'outil anticipe l'ajout de modules complémentaires, la correction des limitations actuelles ou encore l'adaptation à

de nouveaux cas d'usage. Pour permettre et encourager cette dynamique de développement collaboratif, le choix de publier ce projet sous une licence open source s'est imposé naturellement. Rendre le code accessible permet non seulement sa libre consultation mais également sa réutilisation, sa modification, sa redistribution et son amélioration.

Pour être tout à fait précis, le projet s'inscrit dans la philosophie du logiciel libre, qui va au-delà de l'open source en se distinguant par son attention particulière concernant les libertés de l'utilisateur. Comme le rappelle le projet GNU sur son site officiel [41] : « les utilisateurs ont la liberté d'exécuter, copier, distribuer, étudier, modifier et améliorer ces logiciels ». Souligner cet engagement est essentiel car il ne s'agit pas simplement d'un choix technique ou symbolique, mais d'un principe structurant le projet dans son ensemble. L'outil est pensé pour évoluer grâce à ses utilisateurs et par eux dans une logique de communauté, de pérennité et de transparence scientifique. Pour parvenir à cette fin, il est donc essentiel d'avoir un code clair et structuré afin qu'il soit facilement accessible pour assurer sa maintenabilité et son évolutivité.

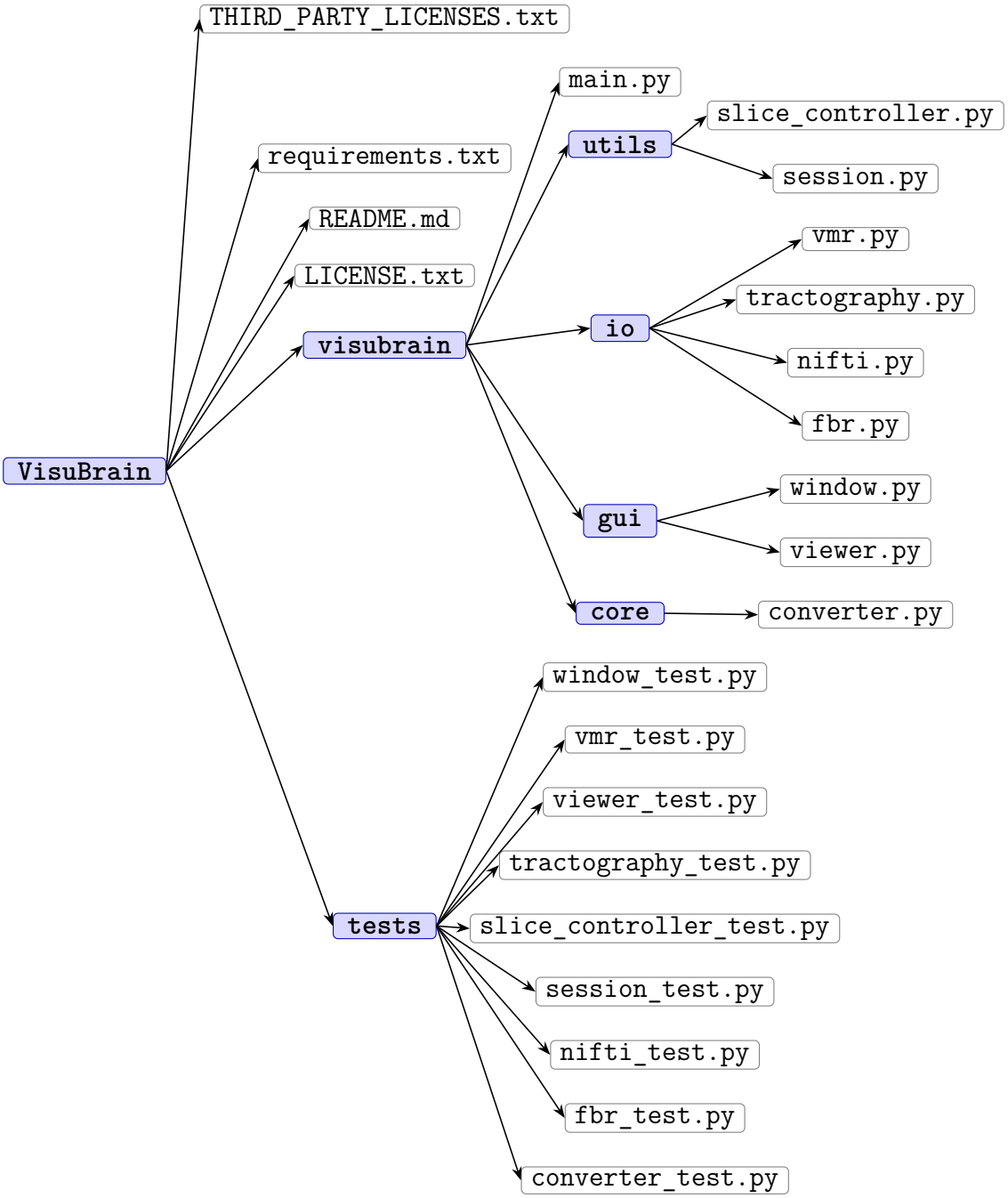


FIGURE 2.2 – *Architecture du code du projet* ***VisuBrain*** *(les dossiers sont représentés en bleu et les fichiers en blanc)*

2.2.1 Organisation principale

L'outil se compose des modules principaux suivants (schématisé sur la figure 2.2) :

— `main.py` : point d'entrée de l'application, responsable de l'initialisation de la fenêtre principale et du chargement des composants.

— `window.py` : définit la fenêtre graphique principale. Elle orchestre l'interface utilisateur avec le chargement des volumes/tractographies, les interactions avec les vues et l'accès au module du convertisseur.

— `viewer.py` : gère la visualisation 3D. Il permet de naviguer parmi et d'afficher les volumes anatomiques et les streamlines/fibres de tractographie superposées.

— `slice_controller.py` : permet la navigation dans les plans axiaux, coronaux et sagittaux à partir d'un volume chargé pour mettre à jour dynamiquement les coupes affichées.
— `session.py` : gère l'état de l'application et des objets chargés (fichiers, volumes, streamlines, paramètres) pour une division par "dossier"/session, assurant une centralisation de la logique et une séparation claire entre les données et la vue.

L'outil intègre des modules dédiés à la lecture, l'écriture et la conversion entre formats de fichiers d'imagerie cérébrale. Ces modules implémentent des classes et fonctions spécialisées selon le type de fichier :

— `converter.py` : centralise la logique de conversion en appelant dynamiquement les fonctions spécifiques aux formats d'entrée et de sortie. Il agit comme une couche intermédiaire abstraite entre l'interface et les modules/méthodes de bas niveau.
— `fbr.py`, `nifti.py`, `tractography.py`, `vmr.py` : modules spécialisés assurant respectivement le support du format FBR (BrainVoyager), du format NIfTI, des formats de tractographie (TRK, TCK) et du format VMR (BrainVoyager). Ces formats de fichiers ont nécessité une attention spécifique, que ce soit pour leur conversion, leur visualisation, ou dans certains cas pour les deux aspects à la fois.

Les fichiers présents dans le dossier tests sont, comme le nom du fichier l'indique, les fichiers qui permettent de tester séparément les fonctions de chaque module. Ces tests sont expliqués plus en détail dans une prochaine section (cf. 2.3.1)

2.2.2 Philosophie de conception

Le code élaboré suit une organisation selon le principe de séparation des responsabilités. Les traitements de données sont découplés de la logique d'interface graphique. Les conversions sont isolées dans des modules bas niveau, tandis que la gestion des interactions utilisateur et de la visualisation est centralisée dans les modules d'interface.

Ce découplage permet :

— une évolution indépendante de la logique d'interface et des fonctions de traitement
— une meilleure testabilité indépendante des composants
— la possibilité de réutiliser les modules de conversion dans d'autres contextes (ligne de commande, API, etc.).

La conception modulaire adoptée facilite l'ajout futur de nouveaux formats ou de nouvelles fonctionnalités utilisateurs. Par exemple, l'ajout d'un nouveau format de tractographie ne nécessite que la création d'un fichier dédié si nécessaire (par exemple `trx.py`), implémentant les méthodes attendues, puis son intégration dans le module `converter.py`.

2.3 Robustesse et maintenabilité

2.3.1 Tests unitaires et tests manuels

Afin d'assurer la fiabilité d'un logiciel, il est important d'y inscrire certains tests. Les tests unitaires s'occupent de vérifier des parties individuelles du code comme des fonctions/méthodes spécifiques. Ces tests sont primordiaux car ils permettent d'assurer le fonctionnement de base d'une méthode mais ils restent toutefois limités à tester les structures internes des fonctions indépendamment du reste. Ils permettent notamment de vérifier que le comportement de base de la fonction en question n'est pas altéré après une modification du code ou un ajout dans celui-ci.

`Pytest` (ainsi que son extension `pytest-qt`) [42] a été utilisée pour l'écriture des tests unitaires, pour sa simplicité, sa lisibilité et son adoption dans la communauté Python. La librairie `Unittest` [43] était également une possibilité pour effectuer de tels tests. Ces deux librairies présentent des avantages et des inconvénients. La bibliothèque `Unittest` présente un avantage important, à savoir qu'elle fait partie de la librairie standard de Python et son utilisation ne demande donc aucune installation supplémentaire comparé à `Pytest` qui, elle, nécessite une installation en plus. Cependant, le choix s'est porté vers `Pytest` pour sa simplicité d'implémentation et également car elle a l'avantage d'être largement utilisé dans la communauté Python. Le fait qu'elle dispose d'une communauté importante et active est en adéquation avec la philosophie de conception du présent travail.

Une large proportion des fonctions du code présentant une logique autonome (initialement découplées de l'interface graphique ou du système de fichiers, ou pouvant être adaptées en ce sens pour l'élaboration des tests) a été partiellement ou entièrement couverte par des tests unitaires automatisés (fig 2.3). Ces tests visent à vérifier la robustesse face aux erreurs, le comportement aux limites ainsi que la cohérence des transformations de données. Bien qu'ils ne couvrent pas l'ensemble des cas d'usage possibles, ils assurent d'une part la conformité du comportement attendu des fonctionnalités de base et d'autre part la stabilité de leur exécution lors de modifications ultérieures du code. Le projet est structuré selon les standards du développement python : le code source est organisé en modules et sous-modules et tous les tests unitaires sont regroupés dans un dossier `tests` séparé qui sont exécutés automatiquement via la commande `pytest` à la racine du projet (en utilisant des fichiers d'exemple).

De plus, des tests d'intégration ont également été mis en place pour vérifier certains aspects de la conversion de fichiers entre les formats, leur ouverture dans le visualisateur et la manipulation des principales fonctionnalités graphiques. Les scénarios automatiques faits par ces tests d'intégration permettent de garantir la robustesse de l'outil dans des conditions semblables à une utilisation réelle. L'argument *tmp_path* (ex. 2.1) présent dans certaines fonctions de test est une fixture fournie automatiquement par `pytest` qui permet de réaliser des tests d'écriture et de lecture de fichiers dans un dossier temporaire, isolé et proprement nettoyé après chaque test. L'argument *qtbot* (fig 2.2) dans les tests est également fourni de base dans le framework `pytest-qt` pour simplifier et fiabiliser les tests des interfaces graphiques `PyQt`. La fixture *monkeypatch* est également utilisée pour remplacer temporairement une fonction, une méthode ou une variable durant l'exécution d'un test. Elle permet d'isoler le code à tester, de simuler des dépendances externes (fichiers, bibliothèques, bases de données...) et de garantir des tests rapides, fiables et reproductibles. Ces méthodes ne permettent pas de vérifier le rendu graphique mais plutôt de s'assurer de la robustesse et de l'appelabilité des méthodes.

```python
def test_trk_to_tck_conversion(tmp_path):
    trk_file = "data/NT1_uf_right.trk"
    tck_file = tmp_path / "mini.tck"
    ref_file = "data/NT1_RD.nii.gz"
    conv = Converter(trk_file,
                     str(tck_file),
                     anatomical_ref=ref_file)
    conv.convert()
    assert tck_file.exists()
    try:
        img = nib.streamlines.load(str(tck_file))
        assert img.streamlines is not None
    except Exception as e:
        pytest.fail(f"Bad trk file: {e}")
```

Listing 2.1 – Exemple d'utilisation de la fixture *tmp_path*

```python
def test_windowapp_launch(qtbot):
    window = WindowApp()
    qtbot.addWidget(window)
    window.show()
    assert window.isVisible()
    assert hasattr(window, "_viewer")
    assert window.isVisible()
    assert hasattr(window, "_viewer")
    assert window.windowTitle() == "VisuBrain"
    assert window._main_layout is not None
```

Listing 2.2 – Exemple d'utilisation de la fixture *qtbot*

En raison de la complexité que peut représenter une interface utilisateur (IU) en terme d'interractions "homme-machine" ainsi que le caractère visuel des fichiers convertis, l'interface graphique ainsi que les résultats des diverses conversions ont été également validés par des tests manuels, des retours de professionnels du domaine ainsi que par les utilisateurs finaux. En plus des tests automatisés, pour le viewer et l'UI, certains tests manuels ont été effectué afin d'ouvrir de vrais fichiers, de manipuler l'interface, de vérifier les différents affichages et de pousser le système dans ses limites.

```
=================== test session starts ====================
platform darwin -- Python 3.13.2, pytest-8.3.5, pluggy-1.6.0
PyQt6 6.8.1 -- Qt runtime 6.8.2 -- Qt compiled 6.8.2
rootdir: VisuBrain/tests
plugins: qt-4.4.0, cov-6.1.1
collected 108 items

converter_test.py ..............................    [ 28%]
fbr_test.py ......                                  [ 34%]
nifti_test.py ..                                    [ 36%]
session_test.py .....                               [ 40%]
slice_controller_test.py ..........                 [ 50%]
tractography_test.py .........                      [ 58%]
viewer_test.py ..............                       [ 72%]
vmr_test.py ....                                    [ 75%]
window_test.py ........................             [100%]

================== tests coverage ======================
____coverage: platform darwin, python 3.13.2-final-0 ____

Name                                                 Cover
------------------------------------------------------------
VisuBrain/visubrain/core/converter.py                 98%
VisuBrain/visubrain/gui/viewer.py                     83%
VisuBrain/visubrain/gui/window.py                     81%
VisuBrain/visubrain/io/fbr.py                        100%
VisuBrain/visubrain/io/nifti.py                      100%
VisuBrain/visubrain/io/tractography.py               100%
VisuBrain/visubrain/io/vmr.py                        100%
VisuBrain/visubrain/utils/session.py                 100%
VisuBrain/visubrain/utils/slice_controller.py        100%
------------------------------------------------------------
TOTAL                                                 89%
================= 108 passed in 25.84s ==================
```

Listing 2.3 – Résultats de l'outil Pytest: pourcentage de recouvrement des tests effectués dans les différents modules du projet

2.3.2 Gestion explicite des erreurs

Une attention particulière a été portée à la gestion des erreurs face aux utilisations de l'IUG. Lors de l'utilisation d'un outil interactif, il est fréquent que l'utilisateur exécute une séquence d'actions inattendue, aboutissant à un comportement non souhaité ou non prévu par le programme. Il peut également tenter de charger un fichier dans un format non pris en charge, d'interagir avec une fonctionnalité sans avoir rempli les conditions préalables ou de provoquer involontairement une incohérence dans l'état de l'application.

Dans tous ces cas de figure, il est essentiel que l'application reste stable et ne se termine pas brutalement. Un crash systématique de l'outil en réponse à une mauvaise manipulation représenterait une barrière significative à son adoption, en particulier pour des utilisateurs non techniques et rendrait le diagnostic de l'erreur difficile. Afin de prévenir

ces comportements critiques, une stratégie défensive a été mise en place à l'aide de blocs `try` / `except`, permettant de capturer les exceptions, d'en gérer proprement les conséquences et dans certains cas d'en informer explicitement l'utilisateur (ex. 2.3).

L'utilisateur est averti d'une erreur ou d'un problème par un simple message prioritaire qui s'affiche à l'écran. Cela permet de prévenir qu'un problème est survenu, d'expliquer la raison de l'interruption et d'éviter un arrêt brutal de l'ensemble du logiciel.

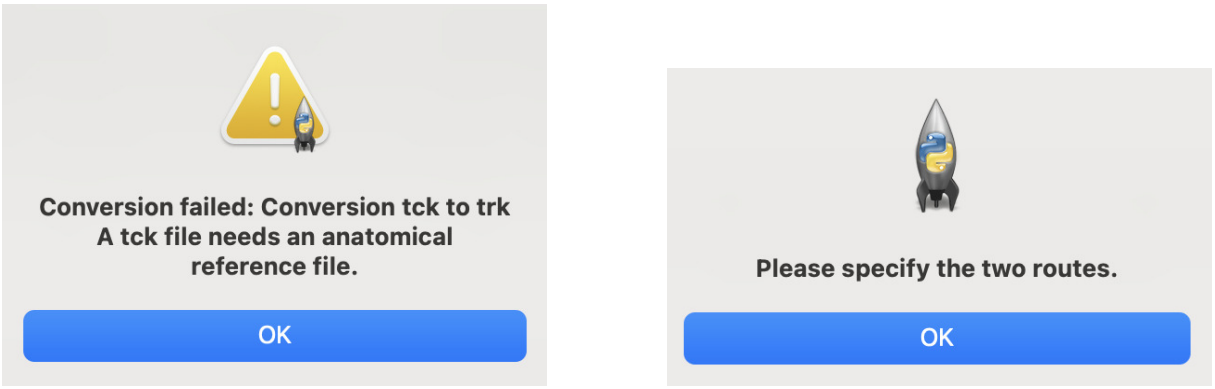


FIGURE 2.3 – *Exemples de message d'erreur pour l'utilisateur*

Cette approche contribue non seulement à la robustesse du logiciel, mais aussi à l'expérience utilisateur globale en assurant une utilisation fluide même en présence d'actions erronées ou non anticipées.

2.3.3 Processus de refactoring

Le *code refactoring* (refactorisation du code) correspond à une transformation logicielle qui préserve le comportement externe de celui-ci tout en améliorant sa structure interne. C'est une façon disciplinée d'améliorer la lisibilité et les fondations du code afin de minimiser les chances d'introduire des bugs. Cette technique permet de comprendre plus facilement le code mais surtout de le rendre moins "coûteux" à modifier. Lors des phases de développement, afin de garder une structure propre et de ne pas la complexifier inutilement le code a subi plusieurs étapes de refactorisation. Les avantages d'un tel processus sont multiples :

- Améliorer le design du logiciel (structure interne)
- Contrer la dégradation du code (vieillissement du logiciel)
- Augmenter la compréhension du code
- Trouver des bugs aisément et écrire du code plus robuste
- Améliorer la productivité sur le long terme
- Réduire les coûts de maintenance
- Réduire les besoins de tests automatiques
- Préparer et faciliter les personnalisations futures

Dans l'optique de garantir une certaine qualité minimum du code et de respecter les standards Python (PEP8 [1]), l'outil `Pylint` [2] a été utilisé. Il permet de détecter des erreurs potentielles, des mauvaises pratiques ou des écarts de style dans le code source. Pylint procède à une analyse statique du code, c'est-à-dire qu'il examine le code source sans

1. `https://peps.python.org/pep-0008/`
2. `https://pypi.org/project/pylint/`

l'exécuter. Il attribue une note globale au fichier analysé (sur 10), signale précisément la ligne et la nature de chaque problème détecté (erreur, avertissement, convention de nommage, etc.) et propose des recommandations de correction. Il contribue ainsi à améliorer la robustesse et la maintenabilité du code. Dans le cadre du présent projet, des analyses régulières avec Pylint ont été effectuées tout au long du développement. Les retours fournis ont permis d'identifier rapidement des axes d'amélioration et d'optimiser progressivement la qualité du code. Grâce à ces démarches il a été possible de garantir un code plus propre, cohérent et conforme aux standards professionnels, ce qui facilite à la fois la compréhension par un tiers et donc la pérennité du projet. Le tableau suivant (2.1) présente les différents scores obtenus suite à l'utilisation de cet outil sur l'ensemble des fichiers Python du projet.

Fichier	**Note Pylint (/10)**	# lignes testées
converter.py	10.00	353
viewer.py	9.72	404
window.py	9.59	755
fbr.py	9.88	196
nifti.py	10.00	115
tractography.py	10.00	130
vmr.py	9.55	173
session.py	9.77	125
slice_controller.py	10.00	108
Moyenne	9.83	2359

TABLE 2.1 – *Notes Pylint obtenues pour chaque module du projet le 31/05/2025*

Chapitre 3

Description des choix techniques

3.1 Langage et outils de développement

Dans le cadre de la conception de l'interface utilisateur ainsi que des modules de conversion, l'un des objectifs fondamentaux était de proposer une solution innovante et utile, tout en évitant de « réinventer la roue ». Le projet s'inscrit pleinement dans une logique d'ouverture et de partage avec la communauté informatique et neuroscientifique : il a vocation à être réutilisé, modifié et enrichi par d'autres utilisateurs. Il était donc cohérent, d'un point de vue méthodologique, de s'appuyer sur des composants existants, éprouvés et bien documentés afin de concentrer les efforts de développement sur les aspects véritablement spécifiques à la problématique étudiée.

Le choix du langage de programmation s'est posé naturellement dans ce contexte. Plusieurs alternatives étaient envisageables, mais le langage Python s'est imposé comme le plus adapté aux exigences du projet. Ce choix repose sur plusieurs considérations complémentaires : sa très large adoption dans la communauté scientifique et informatique, sa syntaxe claire et accessible, sa robustesse, sa gratuité, sa libre utilisation ainsi que la richesse de son écosystème de bibliothèques.

L'intégralité de l'outil présenté dans ce travail a ainsi été développée en Python. Ce langage bénéficie d'une documentation abondante, d'un support communautaire actif et d'une interopérabilité facilitée avec d'autres technologies. De plus, certaines bibliothèques essentielles à ce projet (telles que `nibabel` notamment pour la lecture des fichiers NIfTI, `dipy` pour le traitement des données de diffusion, `pyvista` pour la visualisation 3D, `bvbabel` pour certains fichiers BrainVoyager ou encore `PyQt6` pour la construction de l'interface graphique) sont spécifiquement conçues pour répondre aux besoins du domaine. Leur disponibilité et leur stabilité ont été des facteurs déterminants dans le choix du langage de développement.

3.1.1 Bibliothèques standards utilisées

PyVista

PyVista [44] est une bibliothèque open source Python de visualisation 3D reposant sur VTK (Visualization Toolkit) [45], conçue pour simplifier la manipulation de maillages et de données volumétriques. Elle permet une intégration fluide avec des interfaces interactives et propose une API intuitive pour le rendu scientifique. Dans ce travail, PyVista est utilisée pour l'affichage des structures cérébrales et des fibres de tractographie en trois dimensions, facilitant ainsi l'exploration visuelle des données.,

PyVistaQt

PyVistaQt [46] est un module complémentaire de PyVista qui permet l'intégration fluide de visualisations 3D dans des interfaces graphiques basées sur Qt (comme PyQt6). Il fournit une fenêtre de rendu interactive capable de s'insérer comme un widget dans une application PyQt, ce qui facilite la création d'interfaces mêlant contrôle utilisateur et visualisation scientifique. PyVistaQt joue un rôle clé dans l'interface, en rendant possible l'affichage dynamique de volumes et de fibres dans la fenêtre principale de l'application tout en conservant une interactivité complète avec le reste des composants de l'IUG.

NumPy

NumPy [47] constitue la base du calcul scientifique en Python. Elle fournit des structures de données performantes et des fonctions vectorisées pour les opérations numériques. NumPy est donc au cœur des opérations numériques de bas niveau dans les modules de conversion et de visualisation.

NiBabel

NiBabel [48] est une bibliothèque open source conçue pour la lecture, l'écriture et la manipulation de formats de fichiers d'imagerie neuro [1], notamment NIfTI et Analyze. Elle permet de charger des volumes, d'accéder aux métadonnées (dimensions, orientations, affines, etc) et de naviguer entre les différents systèmes de coordonnées. NiBabel est donc utilisée pour l'importation et la conversion des images anatomiques et fonctionnelles dans un format standardisé compatible avec les autres modules.

DIPY

DIPY (Diffusion Imaging in Python) [49] est aussi une bibliothèque open source qui se concentre sur le traitement, l'analyse et la visualisation des données issues de l'IRM de diffusion. Elle fournit des outils avancés pour le calcul des tenseurs, la reconstruction de fibres (tractographie) et la gestion des fichiers de streamline (TRK, TCK, etc.). DIPY est ici utilisée pour lire, transformer et visualiser les trajectoires des fibres de matière blanche mais est également utilisée dans le module de conversion.

PyQt6

PyQt (utilisé ici dans sa version PyQt6) [50] sert de lien Python depuis le framework Qt. Il offre plus de 35 modules d'extension et permet d'utiliser Python comme langage de développement d'applications alternatif à C++. Il offre un ensemble complet de composants (fenêtres, boutons, menus, barres de navigation, etc) permettant de construire des interfaces modernes et interactives. Nous l'utilisons dans ce travail pour concevoir l'interface principale de l'application, intégrer les contrôles de visualisation, gérer les événements utilisateur et piloter les modules sous-jacents. Sa flexibilité a permis d'intégrer des éléments de visualisation 3D de manière fluide au sein de l'interface.

1. Voici la liste complète des formats pris en charge par le package NiBabel : ANALYZE (simple, SPM99, SPM2 et ultérieurs), GIFTI, NIfTI1, NIfTI2, CIFTI-2, MINC1, MINC2, AFNI BRIK/HEAD, ECAT, MGH, Philips PAR/REC ainsi qu'un support limité pour DICOM

BvBabel

BvBabel [51] est une bibliothèque Python open source maintenue par Omer Faruk Gulban. Elle vise à faciliter la lecture, l'écriture et l'analyse des formats de fichiers propriétaires utilisés par BrainVoyager (`.vmr`, `.fmr`, `.srf`, `.voi`, `.map`, `.fbr`, ...). Cette bibliothèque constitue une passerelle essentielle entre l'écosystème propriétaire de BrainVoyager et les outils open source de la communauté en neuro-imagerie. Dans le cadre de ce travail, BvBabel est utilisée principalement pour extraire et manipuler les données contenues dans les fichiers `.vmr`.

3.2 Licence logicielle et dépôt GitHub

Puisque l'outil développé dans ce travail est mis à disposition en open source, le choix d'une licence appropriée est non seulement recommandé, mais indispensable. En effet, la mise en ligne publique du code (avec possibilité de lecture, de modification et de réutilisation par toute personne) exige un cadre juridique clair pour encadrer les conditions d'usage, de contribution et de redistribution. La licence n'est pas une formalité accessoire : elle définit les droits et obligations des utilisateurs tout en protégeant le ou les auteurs initial-aux du projet.

Le fait d'apposer une licence libre (ou open source) à un code ne retire en aucun cas les droits d'auteur au(x) créateur(s) de ce dernier. En adoptant une licence open source, l'auteur conserve ses droits tout en fixant les modalités selon lesquelles le projet peut être utilisé, modifié et partagé. Elle permet ainsi de garantir que les valeurs d'ouverture et de transparence soient effectivement respectées. Le choix de la licence est donc un acte fondateur dans la vie publique du projet et un levier essentiel pour encourager les contributions communautaires, tout en assurant la sécurité juridique de toutes les parties impliquées.

3.2.1 Choix et justification de la licence

Lorsqu'on développe un logiciel, choisir la bonne licence open source n'est jamais anodin. Dans le cas présent, l'application s'appuie sur plusieurs bibliothèques comme PyQt6, DIPY, NiBabel, NumPy, PyVista, PyVistaQt et bvbabel. Toutes ces librairies sont open source, mais chacune a sa propre philosophie en matière de licence, ce qui a un impact direct sur la manière dont nous pouvons partager notre travail.

La plupart des dépendances utilisées (NiBabel, PyVista, PyVistaQt, bvbabel) utilisent la licence MIT. Il s'agit d'une licence à la fois simple et particulièrement permissive, autorisant la modification, la réutilisation et la redistribution du code, y compris dans un cadre commercial. La seule condition requise est la préservation de la mention des auteurs originaux ainsi que de la licence au sein du projet distribué. NumPy et DIPY, elles, sont sous licence BSD. Celle-ci ressemble beaucoup à la MIT, elle est très permissive, ne pose pratiquement aucune contrainte et encourage la réutilisation et le partage du code.

Mais il y a une exception : PyQt6. Cette bibliothèque fonctionne sous la licence GPL v3. La GPL v3 indique que si vous utilisez du code sous GPL dans votre projet, alors tout votre projet doit aussi être sous GPL. C'est ce qu'on appelle une licence «copyleft». Elle est là pour s'assurer que tous les logiciels dérivés restent libres et ouverts au bénéfice de tous sans pouvoir restreindre la liberté des autres utilisateurs.

Donc même si la grande majorité de nos librairies sont sous des licences très permissives, la présence de PyQt6 sous GPL v3 nous oblige à adopter cette licence pour notre propre code. Cela veut dire que si nous partageons ou distribuons notre application, nous devons fournir le code source et permettre à chacun de le modifier et de le redistribuer à son tour toujours sous licence GPL v3.

En résumé (fig 3.1), l'utilisation de PyQt6 impose la licence GPL v3 à notre projet, même si toutes les autres librairies sont plus souples. C'est un choix qui respecte la loi mais qui s'inscrit aussi dans une démarche de partage et d'ouverture. Et finalement, cela permet à d'autres de s'appuyer sur le présent travail pour aller encore plus loin, ce qui fait partie de nos objectifs.

Caractéristique	**MIT**	**BSD**	**GPL v3**
Type de licence	Permissive	Permissive	Copyleft fort
Réutilisation commerciale	Oui	Oui	Oui
Redistribution	Oui	Oui	Oui (même licence)
Modification	Oui	Oui	Oui
Obligation de publier les sources dérivées	Non	Non	Oui
Mention de l'auteur/droits d'auteur	Oui	Oui	Oui
Texte de licence à inclure	Oui	Oui	Oui

TABLE 3.1 – *Comparaison des licences open source principales utilisées*

3.2.2 Sémantique de version

La numérotation de version du projet a débuté à `v0.1.0`, en accord avec les recommandations formulées dans le document de référence Semantic Versioning 2.0.0 [52] de Tom Preston-Werner, cofondateur de GitHub (plateforme d'hébergement et de partage du code source concerné par le présent travail). Ce système de versionnement sémantique repose sur une convention simple et rigoureuse qui permet de communiquer de manière explicite l'état de stabilité et les changements apportés au logiciel.

Dans cette approche, une version se compose de trois entiers `MAJEUR.MINEUR.CORRECTIF` :

- Le numéro `MAJEUR` est incrémenté en cas de changements non rétrocompatibles.
- Le numéro `MINEUR` est augmenté lorsqu'une nouvelle fonctionnalité rétrocompatible est ajoutée.
- Le numéro `CORRECTIF` est réservé aux corrections de bogues ou améliorations mineures, sans changement de comportement visible.

La version initiale `v0.1.0` marque donc le point de départ d'un logiciel encore en phase de développement mais disposant déjà d'un socle fonctionnel et exploitable. Les incréments suivants ont été appliqués en fonction de l'évolution du code, notamment en lien avec l'ajout progressif de formats supportés, l'intégration de la visualisation 3D ou encore la stabilisation des modules de conversion. Ce choix de versionnement vise à garantir une lisibilité optimale du cycle de développement pour les utilisateurs et les éventuels contributeurs.

Le dépôt contenant le code source présent sur la plateforme GitHub est accessible dans l'annexe A.1

Chapitre 4

Implémentation des modules de conversion

Après avoir exposé les choix techniques qui ont servi de base pour le développement de l'outil, ce chapitre décrit en détail les modules de conversion mis en place. L'objectif est d'illustrer concrètement comment ces choix se traduisent dans l'architecture du projet.

4.1 Sélection des formats prioritaires

La grande diversité des formats de fichiers d'imagerie médicale a nécessité une phase initiale de sélection ciblée. L'objectif de cette sélection était d'identifier les formats les plus pertinents dans le cadre du présent travail, tout en laissant ouverte la possibilité d'une extension progressive de cette liste dans le futur. Deux critères principaux ont guidé ce choix, dans un ordre hiérarchisé :

1. Les besoins exprimés par le partenaire clinique (le « client »),
2. La fréquence d'utilisation des formats dans la littérature scientifique.

Le premier critère a conduit à privilégier en priorité le format `.fbr`, un format propriétaire utilisé par le logiciel BrainVoyager. Ce format est au cœur des problématiques soulevées dans ce mémoire, notamment en ce qui concerne l'hétérogénéité des formats de fichiers dans le domaine de la neuro-imagerie. Ce besoin a été spécifiquement formulé par *l'Institute of Neuroscience (IoNS)* de l'UCLouvain. Le format `.fbr` étant associé à la tractographie issue de l'IRM de diffusion, il a semblé pertinent d'étendre l'analyse aux autres formats majeurs de ce domaine.

Sur la base du second critère (la prévalence dans la littérature) et après validation auprès de l'équipe encadrante, les formats `.trk` et `.tck` ont été retenus en complément du format `.fbr`. Ces formats sont largement utilisés pour la visualisation et l'échange de données de tractographie.

En parallèle, il a été jugé pertinent d'explorer également les formats associés à l'imagerie anatomique et fonctionnelle. Dans une logique de continuité avec le format `.fbr`, le format `.vmr`, également propriétaire de BrainVoyager, a été retenu. D'autres formats choisis sur la base de leur popularité et de leur complémentarité avec les objectifs de ce mémoire, ont été inclus : `.nii`, `.nii.gz` (NIfTI) et `.voi`. Le format NIfTI a été privilégié dans le cadre de ce travail par rapport au format DICOM, largement reconnu comme le standard universel en imagerie médicale. Ce choix ne repose pas sur une supériorité quelconque du NIfTI mais sur l'adéquation entre ses caractéristiques et les besoins spécifiques du projet.

En effet, le format DICOM offre de nombreux avantages supplémentaires par rapport au NIfTI, notamment en intégrant des métadonnées cliniques détaillées : informations patient, paramètres d'acquisition (date, heure, localisation, ...), identifiants du dispositif médical, etc.

Toutefois, ces informations, bien qu'indispensables dans un contexte clinique, n'étaient pas nécessaires ici. L'objectif principal de ce travail portant sur la manipulation, la visualisation et la conversion de volumes d'imagerie cérébrale, le format NIfTI (plus simple et centré sur les données volumétriques telles que les dimensions, orientation spatiale de l'image, système de coordonnées, transformations affines, etc.) s'est avéré plus adapté, notamment pour le développement, le prototypage rapide et l'intégration avec les bibliothèques utilisées. De plus, ce format est largement utilisé en recherche, ce qui correspond au cas d'application du présent travail.

Il est donc important de souligner que ce choix ne remet en aucun cas en question la pertinence ni la fiabilité du format DICOM qui demeure le socle des échanges d'imagerie médicale dans les environnements hospitaliers et réglementés à l'échelle mondiale. Le NIfTI s'inscrit davantage dans un cadre de recherche et d'analyse simplifiée.

Cette sélection constitue ainsi une base représentative des formats couramment utilisés en neuro-imagerie, tout en étant structurée autour d'un besoin concret issu du terrain clinique.

4.2 Description des formats concernés

Ce projet s'est alors concentré sur un ensemble fixe de formats en neuro-imagerie. Chaque format présente des spécificités en termes de structure, de métadonnées et de cas d'usage justifiant leur inclusion dans cette première version de l'outil. Un tableau synthétique présentant ces formats est proposé en fin de section (table 4.9).

4.2.1 FBR

Le format `.fbr` [53] est un format propriétaire utilisé initialement uniquement par le logiciel BrainVoyager pour stocker les coordonnées des fibres d'une tractographie. C'est un fichier encodé en binaire donc ses données ne sont pas directement lisibles. La table 4.2 décrit en détail le contenu et l'organisation d'un tel fichier.

Paramètre	Description
Magic number (int32)	Valeur définissant un fichier fibre binaire valide
FileVersion (int32)	Version du fichier `.fbr`
CoordsType (int32)	Système de coordonnées BrainVoyager : 2 = BVI (défaut), 1 = SYS, 0 = TAL
FibersOriginX (float32) FibersOriginY (float32) FibersOriginZ (float32)	Origine des fibres (coordonnée X, Y, Z)
NrOfGroups (int32)	Nombre de groupes de fibres
Boucle sur les groupes :	
Name (char*, 0-terminated)	Nom du groupe
Visible (int32)	Indique si le groupe est affiché (booléen)
Animate (int32)	Animation du groupe (usage spécifique)
Thickness (float32)	Épaisseur des fibres (mm)
Color (3*8-bit)	Valeurs RGB (0-255) pour le groupe
NrOfFibers (int32)	Nombre de fibres dans le groupe
Boucle sur les fibres du groupe :	
NrOfPoints (int32)	Nombre de points constituant la fibre
X positions (float)	Coordonnées X des points
Y positions (float)	Coordonnées Y des points
Z positions (float)	Coordonnées Z des points
R color (char)	Rouge (0-255) pour chaque point
G color (char)	Vert (0-255) pour chaque point
B color (char)	Bleu (0-255) pour chaque point

TABLE 4.2 – *Structure des paramètres d'un fichier FBR.*

4.2.2 TRK

Le format `.trk` [54] est le format natif de TrackVis. Il permet de stocker des streamlines, c'est-à-dire des représentations tridimensionnelles des fibres de matière blanche estimées par des modèles de diffusion. Il est simple, efficace et largement utilisé dans la recherche. Il comprend une en-tête structurée sur les 1000 premiers bytes (table 4.3) ainsi qu'un ensemble de *tracts* où chaque streamline est définie comme une suite de points dans l'espace (table 4.5).

Nom	Type	Octets	Commentaire
id_string[6]	char	6	Chaîne d'identification du fichier track. [1]
dim[3]	short int	6	Dimensions du volume image.

voxel_size[3]	float	12	Taille des voxels du volume image.
origin[3]	float	12	Origine du volume image (non utilisée, toujours (0,0,0)).
n_scalars	short int	2	Nombre de scalaires enregistrés (en plus des coordonnées x, y, z).
scalar_name[10][20]	char	200	Nom de chaque scalaire
n_properties	short int	2	Nombre de propriétés enregistrées pour chaque tract.
property_name[10][20]	char	200	Nom de chaque propriété
vox_to_ras[4][4]	float	64	Matrice 4x4 de transformation voxel vers RAS. [2]
reserved[444]	char	444	Espace réservé pour de futures versions.
voxel_order[4]	char	4	Orientation des données image originales.
pad2[4]	char	4	Remplissage (padding).
image_orientation_patient[6]	float	24	Orientation de l'image
pad1[2]	char	2	Remplissage (padding).
invert_x	unsigned char	1	Indicateur d'inversion/rotation (usage interne).
invert_y	unsigned char	1	Comme ci-dessus.
invert_z	unsigned char	1	Comme ci-dessus.
swap_xy	unsigned char	1	Comme ci-dessus.
swap_yz	unsigned char	1	Comme ci-dessus.
swap_zx	unsigned char	1	Comme ci-dessus.
n_count	int	4	Nombre de tracts contenus dans ce fichier (0 = non enregistré).
version	int	4	Numéro de version (actuellement 2).
hdr_size	int	4	Taille de l'en-tête (devrait être 1000)

TABLE 4.3 – *Structure de l'en-tête d'un fichier TrackVis (`.trk`).*

Après l'en-tête, ce sont les données (table 4.5) sur les tracts eux-mêmes qui sont stockées. Dans cette description, n correspond au nombre total de tracts, n_s est le nombre de scalaire(s) (chaque point d'un tract a son propre scalaire) et n_p est le nombre de propriété(s) (chaque tract a sa propre "propriété") mais ces deux dernières données valent souvent 0.

1. Les 5 premiers caractères doivent être “TRACK”.
2. Si vox_to_ras[3][3] vaut 0, la matrice n'est pas enregistrée.

Tract	Type de donnée	Octets	Commentaire
Tract #1	int	4	Nombre de points dans ce tract, noté m.
	float	$(3 + n_s) \times 4$	Point du tract #1. Contient 3 plus n_s nombres flottants. Les 3 premiers sont les coordonnées $x/y/z$, suivis des n_s scalaires de ce point.
	float	$(3 + n_s) \times 4$	Point du tract #2. Identique au précédent.
	...	...	...
	float	$(3 + n_s) \times 4$	Point du tract #m. Identique au précédent.
	float	$n_p \times 4$	n_p nombres flottants représentant les propriétés de ce tract.
Tract #2	Identique au précédent.		
⋮	Identique au précédent.		
Tract #n	Identique au précédent.		

TABLE 4.5 – *Structure du "corps" dans un fichier TrackVis (`.trk`).*

4.2.3 TCK

Le format `.tck` [55] est utilisé et créé par MRtrix. Il permet de stocker des tracts de façon comparable au format `.trk`. Il dispose d'une en-tête au format texte (table 4.6) et ses données stockées en binaire. La structure de son en-tête peut varier selon l'utilisation qu'on en fait car elle est délibérément extensible. Celle qui est présentée ci-dessous correspond à un cas de base déterminé d'un fichier exemple mais ne représente pas une généralité.

Champ	Type	Commentaire
mrtrix tracks	string	Marqueur de début de fichier MRtrix (.tck)
command_history	string	Trace complète de l'historique de génération.
downsample_factor	entier	Facteur de sous-échantillonnage
fod_power	flottant	Puissance appliquée au FOD (fonction de distribution d'orientation)
init_threshold	flottant	Seuil initial utilisé pour la génération des tracts.
lmax	entier	Ordre maximal des harmoniques sphériques (SH) utilisés.
max_angle	entier	Angle maximal autorisé entre deux points consécutifs d'une streamline (en degrés).
max_dist	entier	Longueur maximale d'une streamline (en mm ou voxels selon le contexte).
max_num_seeds	entier	Nombre maximal de seeds (points de départ) utilisés pour la tractographie.
max_num_tracks	entier	Nombre maximal de tracts conservés dans le fichier.
max_seed_attempts	entier	Nombre maximal de tentatives de seed par streamline.

max_trials	entier	Nombre maximal d'essais pour générer une streamline.
method	string	Algorithme de tractographie utilisé (ex : iFOD2).
min_dist	entier	Longueur minimale d'une streamline.
mrtrix_version	string	Version de MRtrix utilisée pour générer le fichier.
rk4	booléen (0/1)	Indique si la méthode d'intégration Runge-Kutta 4 a été utilisée.
samples_per_step	entier	Nombre d'échantillons par pas de tractographie.
sh_precomputed	booléen (0/1)	Indique si les harmoniques sphériques étaient pré-calculés.
source	string	Chemin du fichier source utilisé pour la tractographie (ex : FOD).
step_size	flottant	Taille d'un pas de tractographie (en mm).
stop_on_all_include	booléen (0/1)	Arrêter le tract si toutes les inclusions sont atteintes.
threshold	flottant	Seuil pour l'arrêt de la propagation d'un tract.
timestamp	flottant	Date et heure de la génération (format timestamp Unix).
unidirectional	booléen (0/1)	Indique si la propagation se fait dans une seule direction.
roi	string	Région d'intérêt (seed ou include). Peut apparaître plusieurs fois selon les ROI utilisées.
datatype	string	Format des données binaires stockées (ex : Float32LE).
file	string	Emplacement et offset des données binaire contenant les streamlines
count	entier	Nombre de streamlines écrites dans le fichier
total_count	entier	Nombre total de streamlines générées (avant filtrage)
END	string	Marqueur de fin d'entête

TABLE 4.6 – *Structure de l'en-tête d'un fichier MRtrix (`.tck`).*

Les données binaires des tracts sont elles-mêmes stockées dans ce même fichier à la suite du marqueur de fin sous forme de triplets de valeurs à virgule flottante (un triplet par point le long d'un tract). Les tracts sont séparés par un triplet de valeurs NaN (Not A Number) et un triplet de valeurs Inf (infini) est utilisé pour indiquer la fin du fichier.

4.2.4 NIfTI

Le format NIfTI (`.nii` ou `.nii.gz`) est un standard de la recherche en neuro-imagerie. Sa version la plus récente NIfTI-2 [56] est une extension de la version NIfTI-1 [57] qui dérive elle-même du format ANALYZE™ 7.5. Il est principalement utilisé pour stocker des volumes (ex : IRM structurelle, IRMf) et est pris en charge par la quasi-totalité des outils modernes (SPM, FSL, AFNI, BrainVisa, BrainVoyager, FreeSurfer, FSL, ITK, Mango, MRIcron, NiBabel, etc). La table 4.7 décrit en détail l'organisation de ce fichier.

Type	Nom	Offset	Taille	Description
int	sizeof_hdr	0B	4B	Taille de l'en-tête. Doit être 348 (octets).
char	data_type[10]	4B	10B	Non utilisé ; compatibilité avec Analyze.
char	db_name[18]	14B	18B	Non utilisé ; compatibilité avec Analyze.
int	extents	32B	4B	Non utilisé ; compatibilité avec Analyze.
short	session_error	36B	2B	Non utilisé ; compatibilité avec Analyze.
char	regular	38B	1B	Non utilisé ; compatibilité avec Analyze.
char	dim_info	39B	1B	Encodage des directions (phase, fréquence, coupe).
short	dim[8]	40B	16B	Dimensions du tableau de données.
float	intent_p1	56B	4B	1er paramètre d'intention.
float	intent_p2	60B	4B	2e paramètre d'intention.
float	intent_p3	64B	4B	3e paramètre d'intention.
short	intent_code	68B	2B	Code d'intention NIfTI.
short	datatype	70B	2B	Type de donnée.
short	bitpix	72B	2B	Nombre de bits par voxel
short	slice_start	74B	2B	Indice de la première coupe
float	pixdim[8]	76B	32B	Dimension pour chaque voxel, respectivement dans dim[8]
float	vox_offset	108B	4B	Décalage (en byte) vers les données dans un fichier .nii
float	scl_slope	112B	4B	Facteur d'échelle des données.
float	scl_inter	116B	4B	Offset d'échelle des données.
short	slice_end	120B	2B	Indice de la dernière coupe.
char	slice_code	122B	1B	Ordre du timing des coupes.
char	xyzt_units	123B	1B	Unités de pixdim[1..4].
float	cal_max	124B	4B	Intensité d'affichage maximale.
float	cal_min	128B	4B	Intensité d'affichage minimale.
float	slice_duration	132B	4B	Temps pour une coupe.
float	toffset	136B	4B	Décalage temporel (axe temps).
int	glmax	140B	4B	Non utilisé ; compatibilité avec Analyze.
int	glmin	144B	4B	Non utilisé ; compatibilité avec Analyze.
char	descrip[80]	148B	80B	Texte descriptif.
char	aux_file[24]	228B	24B	Nom de fichier auxiliaire.
short	qform_code	252B	2B	Utilisation des champs quaternion.
short	sform_code	254B	2B	Utilisation des champs affine.
float	quatern_b	256B	4B	Paramètre quaternion b.
float	quatern_c	260B	4B	Paramètre quaternion c.
float	quatern_d	264B	4B	Paramètre quaternion d.
float	qoffset_x	268B	4B	Décalage quaternion x.

float	qoffset_y	272B	4B	Décalage quaternion y.
float	qoffset_z	276B	4B	Décalage quaternion z.
float	srow_x[4]	280B	16B	1ere ligne de la matrice affine.
float	srow_y[4]	296B	16B	2e ligne de la matrice affine.
float	srow_z[4]	312B	16B	3e ligne de la matrice affine.
char	intent_name[16]	328B	16B	Nom ou signification de l'intention.
char	magic[4]	344B	4B	Chaîne d'identification ("magic string").

TABLE 4.7 – *Structure de l'en-tête d'un fichier NIfTI-1 (`.nii`).*

4.2.5 VOI

Le format `.voi` est souvent utilisé en recherche. Il correspond au format propriétaire de MRIcron, logiciel créé par Chris Rorden [58]. Comme le mentionne à plusieurs reprises son créateur [59, 60], le format `.voi` est identique au format `.nii` (NIfTI-1). La seule et unique différence, en plus de la compression par défaut du format .voi, est l'extension du fichier.

4.2.6 VMR

Le format `.vmr` (Voxel Map Representation) [61] est un format propriétaire utilisé et développé par BrainVoyager (BV) pour stocker l'ensemble des données des volumes anatomiques 3D. Les fichiers VMR codent les données en entiers non signés (8 ou 16 bits) avec un système de coordonnées propre qui est souvent celui de "BV Internal" (PIL+). Etant donné que c'est un format propriétaire de BV, il n'est pas directement compatible avec les outils utilisant des standards ouverts comme NIfTI. La structure de ce format est un peu différente des autres car elle contient d'abord une en-tête "pre-data" (avant-donnée) suivie des données et enfin une en-tête "post-data" (après-donnée) (table 4.8).

Nom du champ	**Type**	**Octets**	**Description**
File version	unsigned int	2	Version du fichier
DimX	unsigned int	2	Taille axe X (voxels)
DimY	unsigned int	2	Taille axe Y (voxels)
DimZ	unsigned int	2	Taille axe Z (voxels)
Bloc données : boucle dans l'ordre Z-Y-X (conventions BV internal)			
DimZ	unsigned char	1	Coordonnées Z
DimY	unsigned char	1	Coordonnées Y
DimX	unsigned char	1	Coordonnées X
Post-data header (conventions DICOM standard), seulement pour version $\geq$ 2 :			
OffsetX	short int	2	Décalage (offset) sur l'axe X.
OffsetY	short int	2	Décalage (offset) sur l'axe Y.
OffsetZ	short int	2	Décalage (offset) sur l'axe Z.
FramingCubeDim	short int	2	Taille du cube d'encadrement (iso-dimensions).
PosInfosVerified	int	4	Statut de vérification des informations de position.

CoordinateSystem	int	4	Type de système de coordonnées utilisé.
Slice1CenterX	float	4	Coordonnée X du centre du premier slice.
Slice1CenterY	float	4	Coordonnée Y du centre du premier slice.
Slice1CenterZ	float	4	Coordonnée Z du centre du premier slice.
SliceNCenterX	float	4	Coordonnée X du centre du dernier slice.
SliceNCenterY	float	4	Coordonnée Y du centre du dernier slice.
SliceNCenterZ	float	4	Coordonnée Z du centre du dernier slice.
RowDirX	float	4	Composante X du vecteur direction ligne.
RowDirY	float	4	Composante Y du vecteur direction ligne.
RowDirZ	float	4	Composante Z du vecteur direction ligne.
ColDirX	float	4	Composante X du vecteur direction colonne.
ColDirY	float	4	Composante Y du vecteur direction colonne.
ColDirZ	float	4	Composante Z du vecteur direction colonne.
NRows	int	4	Nombre de lignes de la matrice d'image de slice.
NCols	int	4	Nombre de colonnes de la matrice d'image de slice.
FoVRows	float	4	Champ de vue (FoV) dans la direction ligne (en mm).
FoVCols	float	4	Champ de vue (FoV) dans la direction colonne (en mm).
SliceThickness	float	4	Épaisseur du slice (en mm).
GapThickness	float	4	Épaisseur de l'intervalle entre slices (en mm).
NrOfPastSpatialTransformations	int	4	Nombre de transformations spatiales appliquées.
PastTransformation	struct/tableau	variable	Pour chaque transformation passée : nom (string), type (int), fichier source (string), nombre de valeurs (int), valeurs (liste de float).
LeftRightConvention	unsigned char	1	(modifié en v4) 1=radiologique, 0=neurologique
ReferenceSpaceVMR	unsigned char	1 (si v4)	Espace de référence (nouveau en v4)

VoxelSizeX	float	4	Résolution du voxel sur X (en mm)
VoxelSizeY	float	4	Résolution du voxel sur Y (en mm)
VoxelSizeZ	float	4	Résolution du voxel sur Z (en mm)
VoxelResolutionVerified	unsigned char	1	Indicateur de vérification de la résolution voxel
VoxelResolutionInTALmm	unsigned char	1	Indicateur résolution en mm Talairach
VMROrigV16MinValue	int	4	Intensité minimale du volume d'origine V16
VMROrigV16MeanValue	int	4	Intensité moyenne du volume d'origine V16
VMROrigV16MaxValue	int	4	Intensité maximale du volume d'origine V16

TABLE 4.8 – *Structure interne d'un fichier VMR (`.vmr`)*

Tableau récapitulatif des formats étudiés

Format	**Logiciel**	**Principal usage**
`.fbr`	BrainVoyager	Tractographie (streamlines)
`.trk`	TrackVis	Tractographie (streamlines)
`.tck`	MRtrix	Tractographie (streamlines)
`.nii`	Standard (multitude de logiciels)	Volumes anatomiques 3D
`.voi`	MRIcron	Volumes anatomiques 3D
`.vmr`	BrainVoyager	Volumes anatomiques 3D

TABLE 4.9 – *Résumé des formats de fichiers concernés par le projet.*

4.3 Scripts de conversion

4.3.1 $TRK \leftrightarrow TCK$

Les fichiers `.trk` (TrackVis) et `.tck` (MRtrix3) sont deux formats couramment utilisés pour représenter des tractographies. Ces fichiers encodent une collection de streamlines (fibres), chaque streamline étant représentée comme une séquence de coordonnées de points en trois dimensions. Bien que leur structure interne diffère, ces formats partagent une sémantique commune, ce qui permet leur conversion relativement directe à l'aide de bibliothèques existantes.

Dans ce travail, les conversions entre ces deux formats ont été implémentées à l'aide des fonctions `load_tractogram` et `save_tractogram` fournies par la bibliothèque `DIPY`

qui supporte nativement les deux formats. Deux fonctions ont été développées :

- `_trk_to_tck` : cette fonction charge un fichier `.trk` à l'aide de `load_tractogram`, en spécifiant le mot-clé `'same'` pour maintenir le même espace de référence que celui défini dans le fichier source. Le tractogramme est ensuite sauvegardé au format `.tck` via `save_tractogram`. Aucune modification des streamlines n'est nécessaire : la conversion est effectuée en mémoire sans transformation géométrique.
- `_tck_to_trk` : cette fonction convertit un fichier `.tck` vers le format `.trk`. À la différence de la conversion inverse, cette opération nécessite obligatoirement la présence d'une image anatomique de référence (habituellement au format NIfTI comme c'est le cas dans le présent travail). Cette exigence vient du fait que le format `.trk` encode explicitement l'espace image de référence (dimensions, voxel size, orientation, position), alors que le format `.tck` permet d'être plus souple. Si aucun fichier de référence n'est spécifié, une exception est levée et le conversion ne s'effectue donc pas. Après chargement du tractogramme et de la référence adéquate, le fichier est enregistré au format `.trk`.

La spécificité technique de ces fonctions réside donc dans l'asymétrie de la conversion : convertir un fichier `.trk` vers `.tck` est direct et autonome, tandis que la conversion inverse (`.tck` vers `.trk`) nécessite un contexte spatial externe (l'image de référence) pour assurer une définition cohérente de l'espace image. Ce comportement est conforme aux spécifications internes des formats `.trk` et `.tck`.

Cette approche garantit l'intégrité des données spatiales lors des conversions, tout en assurant une compatibilité maximale avec les outils logiciels standards du domaine (TrackVis, MRtrix, DIPY, etc.). Elle respecte également le principe général du projet : tirer parti des bibliothèques robustes de l'écosystème Python pour gérer la complexité des formats tout en assurant une traçabilité complète des opérations.

4.3.2 $TRK \leftrightarrow FBR$

$TRK \rightarrow FBR$

Nous présentons ici la procédure de conversion d'un fichier de tractographie au format `.trk` vers le format propriétaire `.fbr` utilisé par BrainVoyager. En raison de l'absence de documentation claire sur la spécification exacte du format `.fbr` dans la littérature, une hypothèse méthodologique a été adoptée concernant le système d'unités utilisé pour les coordonnées spatiales. Il est supposé que les coordonnées des points de streamline sont exprimées en millimètres et non en indices voxel. Cette hypothèse repose sur l'analyse empirique d'un ensemble de fichiers `.fbr` valides. Le fichier TRK contient des *streamlines* exprimées dans le repère RAS+ en millimètres. Le format FBR quant à lui utilise un espace dit *BV Internal* basé sur la convention PIL+ (Posterior-Inferior-Left), avec un système de coordonnées centré au milieu du volume.

1. Chargement initial Le fichier `.trk` est d'abord chargé via un objet `Tractography` du module `visubrain.tractography.py` (fig 2.2, qui extrait le contenu sous forme d'un objet `StatefulTractogram`. Celui-ci contient entre autres :

- Les **dimensions** du volume de référence : `dim` $= (d_x, d_y, d_z)$
- Les **tailles de voxel** : `voxel_sizes` $= (s_x, s_y, s_z)$

Ces valeurs permettent de calculer les dimensions physiques du volume en millimètres :

$$\vec{L} = \texttt{dim} \cdot \texttt{voxel\_sizes} = (d_x s_x, d_y s_y, d_z s_z)$$

Le **centre du volume**, utilisé comme origine pour le format FBR, est ensuite défini par :

$$\vec{o}_{\text{PIL}} = \left(\frac{L_y}{2}, \frac{L_z}{2}, \frac{L_x}{2} \right)$$

où l'on note que l'axe $x_{\text{BVI}} = y_{\text{RAS}}$, $y_{\text{BVI}} = z_{\text{RAS}}$, $z_{\text{BVI}} = x_{\text{RAS}}$, conformément à la conversion PIL+.

2. Transformation spatiale Chaque streamline est une liste ordonnée de points $\vec{p}_i = (x_i, y_i, z_i)$ exprimés en coordonnées RAS+ (Right-Anterior-Superior). Ces coordonnées sont transformées en PIL+ (4.1) par la relation suivante :

$$\vec{p}_{\text{PIL}} = \begin{bmatrix} -y_i \\ -z_i \\ -x_i \end{bmatrix} + \vec{o}_{\text{PIL}}$$

Ce changement d'orientation est implémenté par une inversion des axes suivie d'un recentrage relatif au milieu du volume.

(a) (b)

FIGURE 4.1 – *(A) Orientation des axes du fichier* `.trk` *en RAS+ ; (B) Orientation des axes du fichier* `.fbr` *en PIL+*

3. Attribution des couleurs Les couleurs sont déterminées pour chaque point à partir des vecteurs tangents des streamlines. Pour un point i, on note :

$$\vec{t}_i = \frac{\vec{p}_{i+1} - \vec{p}_i}{\|\vec{p}_{i+1} - \vec{p}_i\|}$$

Les composantes (t_x, t_y, t_z) sont ensuite mappées en valeurs de couleur RGB par :

$$\texttt{RGB}_i = (|t_x|, |t_y|, |t_z|) \cdot 255$$

Le choix de la valeur absolue garantit des couleurs positives et reflète uniquement la direction dans l'espace. La section 5.1.1 discute plus en détails des étapes de cette attribution.

4. Construction des métadonnées FBR Chaque fibre est encodée comme un dictionnaire contenant :

— `NrOfPoints` : le nombre de points dans la streamline
— `Points` : une liste de tuples `(x, y, z, r, g, b)` pour chaque point

L'en-tête FBR (`header`) est également construit pour inclure :

— La version du fichier : `FileVersion` = 5
— Le type de coordonnées : `CoordsType` = 2 (coordonnées BVI par défaut)
— Le point d'origine `FibersOrigin` = $\vec{o}_{\text{PIL}}$
— Les métadonnées de groupe : nom, couleur, visibilité

5. Écriture du fichier FBR L'écriture binaire du fichier `.fbr` est assurée par une fonction dédiée, `write_fbr` du module `visubrain.fbr.py` (fig 2.2), qui prend en entrée le chemin de sortie du fichier, un dictionnaire `header` contenant les métadonnées globales et les propriétés des groupes de fibres, ainsi qu'une liste `fibers` décrivant chaque fibre individuellement.

La structure binaire suit une séquence strictement définie :

1. **Écriture du marqueur magique** : Les 4 premiers octets du fichier sont des *magic bytes* fixes, codés en hexadécimal `0xA4D3C2B1`, permettant d'identifier le fichier comme un fichier `.fbr` valide.
2. **Écriture de l'en-tête principal** :
   — `FileVersion` : version du format (`uint32`).
   — `CoordsType` : identifiant du système de coordonnées utilisé (e.g. `2` pour PIL+ mm) (`uint32`).
   — `FibersOrigin` : vecteur $\vec{o} = (x_o, y_o, z_o) \in R^3$ indiquant l'origine des coordonnées spatiales, codé en `float32`.
   — `NrOfGroups` : nombre de groupes de fibres (typiquement 1) (`uint32`).
3. **Écriture des groupes de fibres** : Pour chaque groupe défini dans le champ `header["Groups"]`, les informations suivantes sont encodées :
   — `Name` : nom du groupe, encodé en `latin-1` et terminé par un octet nul.
   — `Visible` : indicateur de visibilité (`uint32`).
   — `Animate` : paramètre d'animation (`int32`).
   — `Thickness` : épaisseur d'affichage des fibres (`float32`).
   — `Color` : couleur RGB par défaut du groupe, codée en trois octets (`uint8` $\times 3$).
   — `NrOfFibers` : nombre total de fibres contenues dans le groupe (`uint32`).
4. **Écriture des fibres** : Chaque fibre est représentée par une structure contenant :
   — `NrOfPoints` : nombre de points dans la fibre (`uint32`).
   — **Coordonnées spatiales** :

$$\texttt{X} = [x_1, x_2, \ldots, x_n], \quad \texttt{Y} = [y_1, y_2, \ldots, y_n], \quad \texttt{Z} = [z_1, z_2, \ldots, z_n]$$

   Ces vecteurs sont écrits consécutivement sous forme de tableaux `float32`.
   — **Couleurs par point** :

$$\texttt{R} = [r_1, \ldots, r_n], \quad \texttt{G} = [g_1, \ldots, g_n], \quad \texttt{B} = [b_1, \ldots, b_n]$$

   Chaque canal couleur est encodé séparément en `uint8`, suivant la convention RGB.

Ce format binaire est structuré pour une lecture optimisée par BrainVoyager, qui attend cette séquence précise pour interpréter et visualiser les données de tractographie.

$FBR \rightarrow TRK$

La suite logique est de proposer la conversion d'un fichier `.fbr` vers un fichier de tractographie standard `.trk`. Cette opération permet d'intégrer les données de tractographie générées ou visualisées dans BrainVoyager à des outils open source tels que ceux proposés par `DIPY` ou `MRtrix`.

Hypothèses de départ. Pour la même raison que celle expliquer au début de la conversion $TRK \rightarrow FBR$ (début de la section 4.3.2), la routine de conversion suppose que les coordonnées des points dans le fichier `.fbr` sont exprimées en millimètres dans le système de coordonnées interne de BrainVoyager. Seuls les fichiers pour lesquels `CoordsType` $= 2$ (code correspondant au référentiel interne de BV), méta-donnée par défaut, sont acceptés pour cette conversion.

Traitement des fibres. Chaque fibre est décrite par une liste ordonnée de points tridimensionnels $\vec{p_i} = (x_i, y_i, z_i) \in R^3$ pour $i \in \{1, \ldots, N\}$. Une transformation affine simple est alors appliquée à chaque point, afin de convertir les coordonnées PIL+ vers RAS+, le repère attendu par le format `.trk` :

$$\begin{pmatrix} x_{RAS} \\ y_{RAS} \\ z_{RAS} \end{pmatrix} = \begin{pmatrix} -z_{PIL} \\ -x_{PIL} \\ -y_{PIL} \end{pmatrix}$$

Cette transformation peut être interprétée comme une permutation suivie d'un retournement d'axes (flip) :

- L'axe x_{RAS} (gauche-droite) est obtenu en inversant l'axe z_{PIL} (gauche-droite).
- L'axe y_{RAS} (antérieur-postérieur) est obtenu en inversant l'axe x_{PIL} (postérieur-antérieur).
- L'axe z_{RAS} (inférieur-supérieur) est obtenu en inversant l'axe y_{PIL} (inférieur-supérieur).

Création de l'objet `StatefulTractogram`. Une fois les streamlines correctement transformées dans l'espace RAS+ (en millimètres), elles sont encapsulées dans un objet `StatefulTractogram`, une structure définie par la bibliothèque `DIPY`. Cet objet permet de lier les coordonnées des streamlines à un volume de référence anatomique (généralement un fichier NIfTI, comme demandé à l'utilisateur lors de la conversion fig 5.10) et d'assurer leur positionnement correct dans l'espace 3D.

Encodage du fichier `.trk`. Le fichier final au format `.trk` est généré à l'aide de la fonction `save_tractogram` provenant également de la libraire `DIPY`. Cette méthode tire pleinement parti de l'objet `StatefulTractogram`, en utilisant automatiquement les métadonnées (dimensions, voxel size, affine) du fichier de référence pour générer un entête complet et valide au format `.trk`, sans que l'utilisateur n'ait besoin de spécifier manuellement ces informations.

Il a été pertinent de réutiliser ces fonctions, dont la robustesse et la fiabilité ont été éprouvées, afin de garantir la solidité du code tout en s'inscrivant dans une logique d'efficacité. Cette démarche vise, une fois de plus, à éviter de « réinventer la roue » en s'appuyant sur des outils open source existants et fonctionnels.

4.3.3 $VOI \leftrightarrow NII$

La conversion entre les fichiers VOI et les fichiers NIfTI repose ici sur un principe de compatibilité structurelle entre les fichiers VOI compressés (`.voi.gz`) et les fichiers NIfTI [59, 60]. Contrairement à d'autres formats plus complexes cette conversion ne nécessite pas de transformation du contenu des données elles-mêmes mais uniquement une manipulation d'enveloppe de compression.

Les quatre fonctions suivantes ont été mises en œuvre pour couvrir les cas possibles :

- `_voi_to_nii` : cette fonction permet de convertir un fichier `.voi` (compressé) vers un fichier `.nii` non compressé. La méthode ouvre le fichier VOI en mode binaire compressé via `gzip.open()` puis le copie en sortie sous forme décompressée via `shutil.copyfileobj()`.
- `_voi_to_nii_gz` : cette fonction réalise une simple copie binaire entre deux fichiers compressés. Elle permet de renommer un fichier `.voi` vers un fichier `.nii.gz` tout en conservant son état compressé en utilisant la fonction `shutil.copy()`.
- `_nii_to_voi` : cette fonction effectue l'opération inverse de `_voi_to_nii`. Elle prend un fichier NIfTI non compressé (`.nii`) et le compresse au format GZIP en sortie (`.voi`) en utilisant les mêmes fonctions standards de lecture et d'écriture binaire.
- `_nii_gz_to_voi` : de la même manière que la conversion `_voi_to_nii_gz`, cette fonction effectue une simple copie d'un fichier compressé `.nii.gz` vers un fichier `.voi.gz`, sans altération du contenu.

Ces conversions supposent que le contenu binaire des fichiers VOI compressés est compatible avec une structure NIfTI comme le mentionne le créateur du format VOI.

Cette approche minimaliste mais fonctionnelle permet une interopérabilité rapide entre formats sans altérer les données. Elle illustre de nouveau le principe important de réutiliser les structures existantes et standardisées dès que possible afin d'éviter une complexité inutile et de favoriser la fluidité des conversions entre formats.

4.3.4 $VMR \leftrightarrow NII$

Comme décrit précédemment, l'en-tête du fichier VMR contient un ensemble riche de métadonnées nécessaires à une interprétation correcte du volume, notamment en ce qui concerne l'orientation et la position du volume dans l'espace. Il est nécessaire de préciser certains points concernant ces informations afin de mieux comprendre les explications de la conversion à proprement parler.

Parmi ces métadonnées figurent les champs `RowDirX, RowDirY, RowDirZ` et `ColDirX, ColDirY, ColDirZ`, qui décrivent respectivement deux des trois vecteurs d'orientation du volume, aussi appelés vecteurs de direction. Le vecteur `RowDir`, défini par ses composantes en x, y et z, décrit l'orientation de l'axe X, tandis que le vecteur `ColDir` définit l'orientation de l'axe Y de l'espace. Le troisième vecteur, nécessaire pour décrire complètement l'orientation tridimensionnelle du volume (axe Z), n'est pas explicitement stocké dans l'en-tête.

Ce choix est justifié par une optimisation de l'espace de stockage, dans la mesure où les trois vecteurs formant une base orthonormée, le troisième vecteur peut être aisément déduit par le produit vectoriel des deux premiers. En d'autres termes, la direction du troisième axe (axe Z) est reconstruite dynamiquement à partir des vecteurs `RowDir` et `ColDir` via un calcul de produit croisé.

En complément des informations d'orientation, l'en-tête VMR contient également des données relatives à la position absolue du volume. Ces informations sont stockées dans les champs `Slice1CenterX, Slice1CenterY, Slice1CenterZ` et `SliceNCenterX, SliceNCenterY, SliceNCenterZ`, qui décrivent les coordonnées spatiales du centre de la première et de la dernière coupe du volume. Ensemble, ces champs permettent de déduire la position du volume par rapport à l'origine du repère tridimensionnel $(0, 0, 0)$, c'est-à-dire la position du premier voxel dans l'espace (fig 4.2).

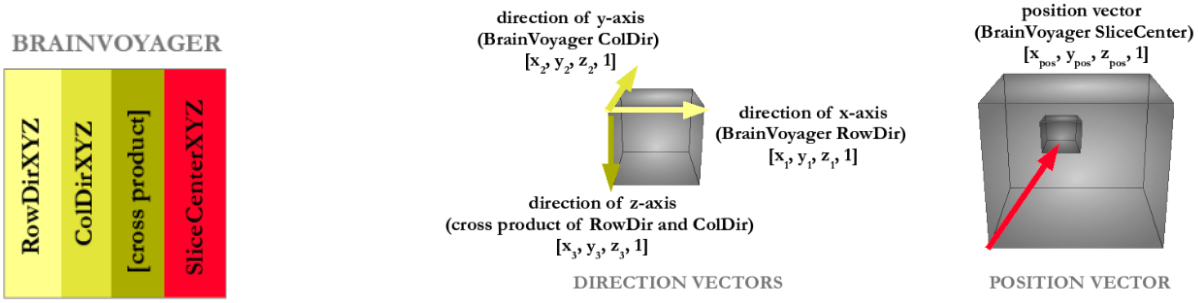


FIGURE 4.2 – *Terminologie du positionnement et représentation des vecteurs d'orientation et de position [62]*

L'ensemble de ces informations d'orientation et de position peuvent ensuite être rassemblées en une matrice (fig 4.3) qui permet de convertir les positions des voxels dans le système de coordonnées DICOM/World (x,y,z) en indices voxel (i,j,k) de BrainVoyager [62].

$$\text{world2voxel} = \begin{pmatrix} RowDir_x & ColDir_x & normal_x & SliceCenter_x \\ RowDir_y & ColDir_y & normal_y & SliceCenter_y \\ RowDir_z & ColDir_z & normal_z & SliceCenter_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

FIGURE 4.3 – *Transformation BrainVoyager "world-to-voxel"*

$NII \rightarrow VMR$

La première étape du processus de conversion d'un fichier NIfTI vers le format VMR consiste à charger le volume à l'aide de la bibliothèque `Nibabel`, via la fonction `nib.load()` [63]. Cette bibliothèque est reconnue dans la communauté pour sa robustesse et sa compatibilité avec les formats d'imagerie neuro comme le NIfTI et constitue donc un choix naturel pour cette tâche.

Par défaut, les fichiers NIfTI utilisent un système de coordonnées de type RAS+ (Right-Anterior-Superior), dans lequel l'origine se situe dans le coin inférieur gauche à l'arrière du volume. Toutefois, cette convention peut varier selon la provenance du fichier, en fonction du logiciel d'acquisition ou des étapes de prétraitement déjà appliquées. Certains fichiers peuvent ainsi ne pas être orientés en RAS+, ce qui introduit un risque d'interprétation erronée lors de la visualisation ou de la conversion.

Afin de garantir une base commune et d'assurer un comportement homogène du convertisseur, toutes les données NIfTI sont préalablement converties en orientation RAS+

(fig 4.4) si ce n'est pas déjà le cas. Cette standardisation est réalisée à l'aide de la fonction `nib.as_closest_canonical()` [64], qui réorganise les axes du volume pour les aligner avec la convention RAS+. Dans la documentation de NiBabel, l'orientation "canonical" fait précisément référence à cette configuration RAS+ [65], ce qui correspond à notre besoin de standardisation des axes du fichier NIfTI entrant.

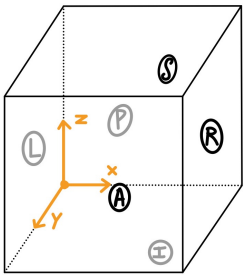


FIGURE 4.4 – *Orientation de l'espace de coordonnées NIfTI*

Une fois les données NIfTI converties en orientation RAS+, c'est-à-dire dans un format commun et standardisé, l'attention peut être portée sur les exigences spécifiques du format VMR. Une bonne gestion des éléments à indiquer dans la matrice de positionnement et d'orientation est essentielle pour garantir une correspondance correcte entre les coordonnées du fichier source (NIfTI) et celles attendues par le format de sortie (VMR).

Calcul des informations d'orientation et de position

Il faut donc trouver et/ou calculer ces informations nécessaires à la création du fichier VMR. Nous récupérons les dimensions des voxels ainsi que les dimensions de l'image nifti dans son en-tête, respectivement pixdim et dim. Ensuite nous créons la matrice affine qui permet de passer de l'espace image du fichier NIfTI vers l'espace image d'un fichier DICOM standard (fig 4.5).

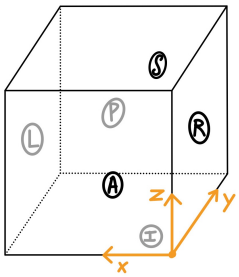


FIGURE 4.5 – *Orientation de l'espace de coordonnées DICOM*

Afin de convertir les coordonnées issues d'un volume NIfTI vers un repère conforme à la convention spatiale standard du format DICOM, il est nécessaire d'appliquer une transformation affine inversant les axes x et y (car dans la convention NIfTI, ces axes croissent respectivement vers la droite et vers l'avant du patient, tandis que dans celle de DICOM, ils croissent vers la gauche et vers l'arrière).
De plus, cette transformation doit recentrer l'origine du volume dans le coin "inférieur-avant-droit", en décalant les coordonnées de translation selon les dimensions du volume. Si l'on note dim_x et dim_y le nombre de voxels le long des axes x et y, la matrice affine de transformation à appliquer est donnée par :

$$\text{nii2dcm} = \begin{bmatrix} -1 & 0 & 0 & \dim_x - 1 \\ 0 & -1 & 0 & \dim_y - 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

En multipliant cette matrice par la matrice affine initiale du fichier NIfTI, notée $\text{nii}_{\text{affine}}$, on obtient une nouvelle matrice de position exprimée dans le repère standard DICOM :

$$\text{dcmposmatrix} = \text{nii}_{\text{affine}} \cdot \text{nii2dcm}$$

Cette opération calcule les coordonnées spatiales de manière cohérente et permet d'avoir les positions des voxels relatifs à l'espace image standard de DICOM tel que géré dans le logiciel BrainVoyager.
Ensuite on définit deux matrices auxiliaires : `dcm2bv1` et `dcm2bvn`. Ces matrices expriment la position du centre d'une coupe dans le repère voxel, en prenant en compte l'échelle des voxels.
La matrice `dcm2bv1` cible la première coupe ($z = 0$), centrée au milieu de l'image ($x = \dim_x/2$, $y = \dim_y/2$) :

$$\texttt{dcm2bv1} = \begin{pmatrix} \frac{1}{v_x} & 0 & 0 & \frac{\dim_x}{2} \\ 0 & \frac{1}{v_y} & 0 & \frac{\dim_y}{2} \\ 0 & 0 & \frac{1}{v_z} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

La matrice `dcm2bvn` cible la dernière coupe ($z = \dim_z - 1$), également centrée en x et y :

$$\texttt{dcm2bvn} = \begin{pmatrix} \frac{1}{v_x} & 0 & 0 & \frac{\dim_x}{2} \\ 0 & \frac{1}{v_y} & 0 & \frac{\dim_y}{2} \\ 0 & 0 & \frac{1}{v_z} & \dim_z - 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

On applique ensuite la transformation spatiale, on multiplie la matrice de positionnement globale `dcmposmatrix` par chacune de ces matrices de coupe pour obtenir leur position et orientation dans le repère :

- `first` = `dcmposmatrix`×`dcm2bv1` donne la position et l'orientation de la première coupe ;
- `last` = `dcmposmatrix` × `dcm2bvn` fait de même pour la dernière coupe.

De la matrice résultante pour chaque coupe, on extrait les valeurs suivantes qui vont nous permettre de fournir les champs adéquats à l'en-tête du fichier VMR :

- le centre de la coupe : coordonnées $[x, y, z]$ du centre de la première (`first[0,3]`, `first[1,3]`, `first[2,3]`) et de la dernière coupe (`last[0,3]`, `last[1,3]`, `last[2,3]`) ;
- les vecteurs d'orientation : le vecteur ligne `rowDir` (direction physique de l'axe X, i.e. `first[0:3, 0]`) et le vecteur colonne `colDir` (direction physique de l'axe Y, i.e. `first[0:3, 1]`).

Une fois ces différentes données calculées, l'étape suivante consiste les intégrer, ainsi que les coordonnées des voxels issues du volume NIfTI, dans la construction de l'en-tête du fichier VMR. Pour ce faire, une nouvelle en-tête VMR est initialisée et nous discutons ici des champs particulièrement sensibles et non triviaux.

- Les vecteurs d'orientation calculés précédemment sont insérés dans les champs : `RowDirX`, `RowDirY`, `RowDirZ`, `ColDirX`, `ColDirY`, `ColDirZ`, ainsi que les coordonnées du centre des coupes extrêmes dans : `Slice1CenterX`, `Slice1CenterY`, `Slice1CenterZ` et `SliceNCenterX`, `SliceNCenterY`, `SliceNCenterZ`. Ces données se situent dans l'en-tête "d'après données" (Post-Data Header) et celles-ci ont la particularité d'être inscrite selon la terminologie du système de référence DICOM standard.
- En ce qui concerne les dimensions du volume, la terminologie utilisée suit les conventions de l'espace de coordonnées de *BV (BrainVoyager) Internal*. Une correspondance attentive doit donc être appliquée entre les axes du fichier NIfTI et ceux attendus par le format VMR. En effet, pour rappel, un fichier NIfTI en convention RAS+ adopte l'orientation suivante :

— x_{nii} = gauche/droite (sagittal),
— y_{nii} = avant/arrière (coronal),
— z_{nii} = bas/haut (axial)

Tandis que le format interne de BrainVoyager (BV internal), en convention PIL+, suit l'ordre suivant :

— x_{bvi} = avant/arrière (coronal),
— y_{bvi} = bas/haut (axial),
— z_{bvi} = gauche/droite (sagittal)

Par conséquent, les dimensions sont réassignées comme suit :

$$\dim_x^{\text{VMR}} = \dim_y^{\text{NIfTI}}, \quad \dim_y^{\text{VMR}} = \dim_z^{\text{NIfTI}}, \quad \dim_z^{\text{VMR}} = \dim_x^{\text{NIfTI}}$$

Cette permutation garantit une concordance correcte entre les repères spatiaux des deux formats. En effet, contrairement aux informations contenues dans l'en-tête d'après données, les données à proprement parler sont quant à elle stockées selon la terminologie du référentiel BrainVoyager Internal (PIL+) (fig 4.6).

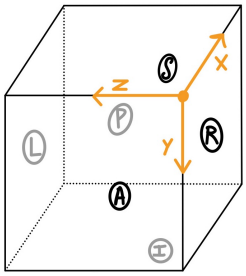


FIGURE 4.6 – *Orientation de l'espace de coordonnées BVI*

— Le champ `LeftRightConvention`, situé dans l'en-tête d'après données, est fixé à 1 ce qui correspond à la convention radiologique, cohérente avec le référentiel DICOM standard.

L'écriture effective du fichier VMR, une fois toutes les informations nécessaires préparées, est réalisée à l'aide de la fonction `write_vmr` fournie dans le module `vmr.py` de `bvbabel` [66]. Ce module, développé par Omer Faruk Gulban (chercheur et ingénieur logiciel chez Brain Innovation, l'éditeur de la suite BrainVoyager), permet d'encoder à la fois les métadonnées de l'en-tête et les données volumétriques sous forme binaire, conformément aux spécifications du format VMR.

L'intégration de cette fonction s'inscrit dans une logique de réutilisation d'outils existants et fiables : plutôt que de réimplémenter une logique d'écriture binaire, l'utilisation de `write_vmr`, déjà éprouvée et optimisée, permet de gagner en robustesse tout en s'alignant avec l'esprit collaboratif et réutilisable des logiciels open source.

Toutefois, même si nous pouvons compter sur cette méthode pour l'écriture, il est important de comprendre et analyser son fonctionnement. L'encodage de l'en-tête dans cette fonction ne nécessite pas d'explication technique supplémentaire à ce qui a été expliqué jusqu'ici, cependant une attention particulière doit être portée concernant l'écriture des données issues du fichier NIfTI. En effet, pour que l'écriture soit conforme à l'espace de référence attendu par BrainVoyager pour les données d'un fichier VMR (BV Internal, en convention PIL+), les données doivent subir une transformation préalable non-triviale.

Cette nécessité provient des différences d'orientation entre l'espace RAS+ du NIfTI (dans lequel : x = gauche-droite, y = avant-arrière, z = bas-haut) et l'espace PIL+ du VMR (où : x = avant-arrière, y = bas-haut, z = gauche-droite).

Pour garantir une correspondance correcte, la structure des données doit être modifiée avant écriture. Cette transformation ne concerne que les données car BrainVoyager stocke celles-ci avec la terminologie du référentiel BV Internal PIL+ contrairement au stockage du *"Post-Data header"* (en-tête située après les données dans le fichier VMR) qui lui se fait sur base de la terminologie du référentiel DICOM standard.

Dans la fonction d'écriture, les données sont transformées par les instructions suivantes :

```
1 data_img = data_img[::-1, ::-1, ::-1]
2 data_img = np.transpose(data_img, (0, 2, 1))
```

Listing 4.1 – Lignes 276–277 de la fonction *write_vmr* de *bvbabel*

Voici l'explication de cette transformation :

- **Inversion des axes** : la commande `[::-1, ::-1, ::-1]` applique un *flip* sur les trois axes du volume. Cela permet de réaligner les directions des axes du volume NIfTI avec celles de l'espace BV Internal (le sens de chaque axe est opposé entre les deux systèmes).
- **Permutation des axes** : la commande `np.transpose(data_img, (0, 2, 1))` réordonne les dimensions pour correspondre au schéma de parcours du volume attendu par le format VMR, qui encode les données selon l'ordre (Z, Y, X) [61]. Concrètement, nous pouvons effectuer la correspondance suivante : les coordonnées NIfTI $(\texttt{X}, \texttt{Y}, \texttt{Z})$ correspondent aux coordonnées $(\texttt{Z}, \texttt{X}, \texttt{Y})$ dans l'espace standard de BrainVoyager ("BV Internal"). En pratique, cette transposée revient à inverser les deux derniers axes, soit : $(Z, X, Y) \rightarrow (Z, Y, X)$.

Cette transformation assure que les données volumétriques sont cohérentes avec les attentes du référentiel standard de BrainVoyager (PIL+) et celles du système de coordonnées du NIfTI (RAS+).

$VMR \rightarrow NII$

Dans un premier temps, l'en-tête et les données volumétriques du fichier VMR sont extraites à l'aide de la fonction `read_vmr` issue de la bibliothèque `bvbabel` [66]. Cette fonction permet d'accéder à l'ensemble des informations encodées dans le fichier en entrée, à la fois les données brutes et les métadonnées contenues dans l'en-tête.

Il est essentiel de souligner que la fonction `read_vmr` applique automatiquement une transformation vers l'espace RAS+, en réalignant les données brutes issues du VMR selon un repère anatomique standardisé. Cette transformation est exactement celle appliquée lors de l'écriture d'un fichier VMR à partir d'un volume NIfTI (fig 4.1), seulement l'ordre des deux étapes effectuées est inversés. Ainsi, les données obtenues à la lecture grâce à la méthode de `bvbabel` sont directement prêtes à l'emploi pour générer un fichier NIfTI cohérent avec les conventions attendues sans qu'il soit nécessaire d'appliquer une transformation supplémentaire.

Comme expliqué précédemment, l'en-tête d'un fichier VMR contient notamment des méta-données qui permettent, en les assemblant judicieusement, de former une matrice

de transformation des coordonnées DICOM/monde (x,y,z) vers les coordonnées d'indices voxel BrainVoyager(i, j, k) (fig 4.3). On extrait des champs de l'en-tête ces méta-données qui représentent les vecteurs de direction de l'image (`rowDir` pour l'axe X et `colDir` pour l'axe Y). Ensuite on calcule le vecteur normal (qui détermine la direction de l'axe Z) à l'aide du produit vectoriel. Ces trois vecteurs sont ensuite assemblés sous forme de colonnes dans une matrice d'orientation appelée `ImageOrientationDCM`, de dimension 3×3 :

$$\texttt{ImageOrientationDCM} = \begin{bmatrix} \texttt{RowDir}_x & \texttt{ColDir}_x & \texttt{Normal}_x \\ \texttt{RowDir}_y & \texttt{ColDir}_y & \texttt{Normal}_y \\ \texttt{RowDir}_z & \texttt{ColDir}_z & \texttt{Normal}_z \end{bmatrix}$$

Elle définit le repère orthonormé propre au volume selon le référentiel DICOM standard (fig 4.5).

On détermine ensuite la position du centre du volume, calculée comme la moyenne des centres de la première et de la dernière coupe (`Slice1Center` et `SliceNCenter`). Cette matrice 1×3 appelée `ImagePositionCentreDCM` représente le point d'ancrage spatial du volume dans l'espace monde.

$$\texttt{ImagePositionCentreDCM} = \frac{1}{2}\left(\texttt{Slice1Center} + \texttt{SliceNCenter}\right)$$

La matrice d'orientation (`ImageOrientationDCM`), l'échelle des voxels et la position du centre du volume sont combinées pour former une matrice affine (`dcm_to_patient`).

La sous-matrice 3×3 située en haut à gauche de cette matrice affine est obtenue en multipliant chaque vecteur directionnel (axes colonnes, lignes, slices) de `ImageOrientationDCM` par la taille de voxel correspondante. Cette opération a pour effet de convertir les déplacements exprimés en nombre de voxels en déplacements physiques de sorte que chaque déplacement d'un voxel le long d'un axe se traduise par un déplacement de la taille du voxel dans la direction appropriée dans l'espace patient.

La colonne de translation ($[:3, 3]$) est ensuite renseignée avec la position du centre du volume, déjà exprimée en millimètres dans le référentiel patient, ce qui permet de localiser précisément l'image dans l'espace physique sans nécessiter d'autre mise à l'échelle.

$$\texttt{dicom\_to\_patient} = \begin{bmatrix} s_x \cdot \texttt{RowDir}_x & s_y \cdot \texttt{ColDir}_x & s_z \cdot \texttt{SliceDir}_x & \texttt{ImagePositionCentre}_x \\ s_x \cdot \texttt{RowDir}_y & s_y \cdot \texttt{ColDir}_y & s_z \cdot \texttt{SliceDir}_y & \texttt{ImagePositionCentre}_y \\ s_x \cdot \texttt{RowDir}_z & s_y \cdot \texttt{ColDir}_z & s_z \cdot \texttt{SliceDir}_z & \texttt{ImagePositionCentre}_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

où s_x, s_y et s_z sont les résolutions voxel selon les axes X, Y et Z.

Il est ensuite nécessaire de construire la partie *translation* de la matrice de transformation affine permettant de passer du centre du volume (origine du système de référence de BrainVoyager Internal) au coin du volume (utilisé comme origine du système NIfTI). Cette opération s'effectue via la matrice `ShiftCenter` qui ajuste les coordonnées pour placer correctement le volume dans le nouveau référentiel.

$$\texttt{ShiftCenter} = \begin{bmatrix} 1 & 0 & 0 & -\frac{\texttt{dim}_x}{2} \\ 0 & 1 & 0 & -\frac{\texttt{dim}_y}{2} \\ 0 & 0 & 1 & -\frac{\texttt{dim}_z}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

La multiplication de la matrice de translation `ShiftCenter` à la matrice `dicom_to_patient` garantit l'alignement correct de l'origine voxel.

Enfin, un passage au repère RAS+ (convention NIfTI) est réalisé via un flip sur les axes X et Y (`patient_to_ras`), ce qui permet d'obtenir la matrice finale `Voxel2World` utilisée comme affine lors de la création du fichier NIfTI.

$$\texttt{patient\_to\_ras} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

La matrice affine finale utilisée pour écrire le fichier NIfTI est donc :

$$\texttt{voxel2world} = \texttt{patient\_to\_ras} \cdot \texttt{dicom\_to\_patient} \cdot \texttt{ShiftCentre}$$

Cette construction affine encode à la fois l'orientation, l'échelle et la localisation du volume dans l'espace patient, garantissant ainsi une conversion fidèle et cohérente des images VMR vers le format NIfTI dans le référentiel RAS+.

On renseigne ensuite explicitement les résolutions de voxel et les dimensions dans l'en-tête NIfTI avant de sauvegarder le fichier.

Il faut cependant être prudent quant à l'encodage des dimensions de l'image. Dans le format VMR, les dimensions du volume sont indiquées sous les champs `DimX`, `DimY` et `DimZ`, qui se réfèrent respectivement aux axes X_{BVI}, Y_{BVI} et Z_{BVI} [61]. Selon la convention PIL+ utilisée en interne par BrainVoyager, ces axes sont orientés comme suit :

— X_{BVI} : antéro-postérieur (avant → arrière)
— Y_{BVI} : inféro-supérieur (bas → haut)
— Z_{BVI} : gauche → droite (latéral)

En revanche, dans le format NIfTI, les axes sont exprimés dans la convention RAS+ :

— X_{NIfTI} : droite → gauche (sagittal)
— Y_{NIfTI} : postérieur → antérieur (coronal)
— Z_{NIfTI} : inférieur → supérieur (axial)

Par conséquent, il est impératif de tenir compte de la correspondance suivante lors de l'écriture du fichier NIfTI :

$$\begin{aligned} X_{\text{NIfTI}} &\longleftarrow Z_{\text{BVI}} \\ Y_{\text{NIfTI}} &\longleftarrow X_{\text{BVI}} \\ Z_{\text{NIfTI}} &\longleftarrow Y_{\text{BVI}} \end{aligned}$$

L'architecture modulaire de conversion a été pensée pour évoluer facilement, de nouveaux formats peuvent être intégrés sans remettre en question les fondations existantes. Cette base fonctionnelle ouvre ainsi naturellement la voie à l'intégration des modules dans une interface graphique dédiée, objet du chapitre suivant, qui vise à rendre ces fonctionnalités accessibles de manière interactive et intuitive à l'utilisateur final tout en offrant la possibilité de visualiser certains de ces formats.

Chapitre 5

Développement et fonctionnement de l'interface graphique utilisateur (IUG)

5.1 Fonctionnalités intégrées de visualisation 3D interactive

L'interface principale de l'application (fig 5.1) a été conçue pour permettre la visualisation interactive de fichiers anatomiques (au format NIfTI) ainsi que de fichiers de tractographie (aux formats TRK ou TCK). Bien que ces formats aient été ciblés en priorité dans le cadre de ce travail, cette sélection ne constitue en aucun cas une limite fixe. L'objectif à long terme est d'étendre progressivement la compatibilité à un maximum de formats couramment utilisés en neuro-imagerie. Néanmoins, pour des raisons de cohérence et de portée, le choix s'est ici porté sur ces formats spécifiques car ils figurent parmi les plus fréquemment rencontrés dans la littérature scientifique dédiée à la visualisation de données cérébrales. En tant que visualiseur d'images minimal, il s'agit d'un outil pratique pour les utilisateurs qui ont besoin d'un outil rapide et facile pour visualiser des images médicales.

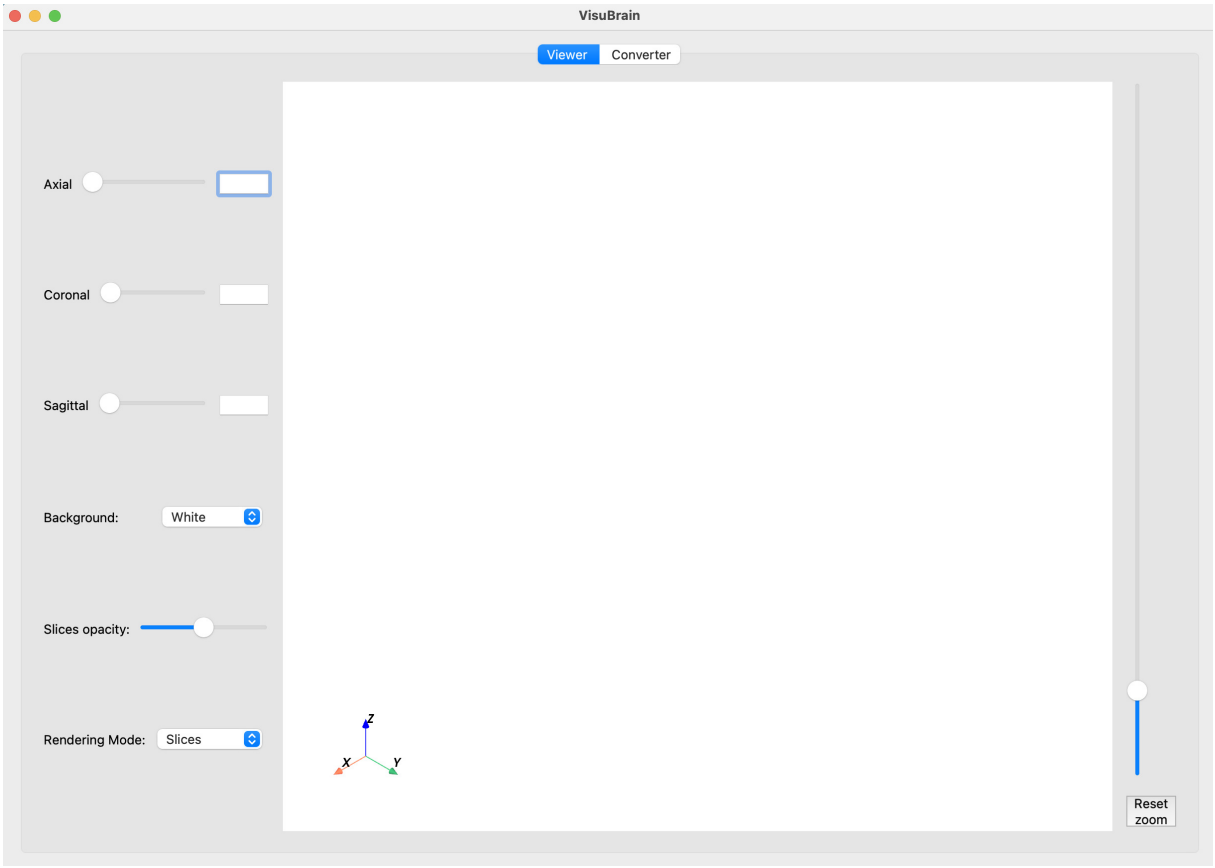


FIGURE 5.1 – *Interface principale de l'outil de visualisation*

Chargement des images

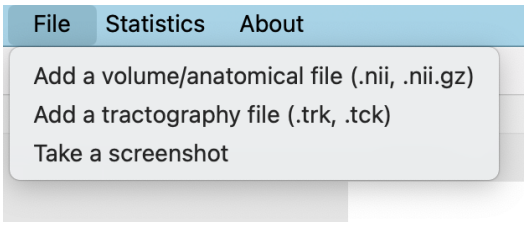


FIGURE 5.2 – *Charger un fichier*

L'onglet *File* (fig 5.2), situé en haut à gauche de l'interface, offre une méthode intuitive et classique pour charger les trois types de fichiers pris en charge par l'application. Lorsque l'utilisateur commence par charger un fichier anatomique, les fichiers de tractographie ajoutés par la suite sont automatiquement superposés à ce volume de référence en cours de visualisation. Inversement, si le premier fichier chargé est un fichier de tractographie, l'image anatomique importée ultérieurement est utilisée comme base pour repositionner correctement les streamlines, assurant ainsi leur alignement spatial avec le volume de fond.

Transformation affine des streamlines

Lors du chargement d'un fichier de tractographie (`.trk` ou `.tck`), l'outil utilise la fonction `load_tractogram` fournie par la bibliothèque `DIPY`, afin de charger les streamlines dans leur espace de coordonnées d'origine. Si le fichier d'entrée est au format `.tck`, une image anatomique de référence au format NIfTI est requise pour définir le système de coordonnées spatiales dans lequel les streamlines doivent être interprétées.

Rappel sur les streamlines

Les streamlines (ou fibres) sont représentées comme des listes de points tridimensionnels dans l'espace, soit :

$$\mathbf{s} = [\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n], \quad \text{avec } \mathbf{p}_i \in \mathbf{R^3}. \tag{5.1}$$

Chaque point $\mathbf{p}_i$ est une coordonnée exprimée dans le système de référence du fichier d'origine.

Transformation vers l'espace monde

Lorsque l'image anatomique de référence est fournie, une transformation affine est appliquée aux streamlines pour exprimer leurs coordonnées dans l'espace physique (aussi appelé espace patient). Cette transformation est définie par la matrice affine $\mathbf{A} \in \mathbf{R}^{4\times4}$ fournie par le fichier NIfTI :

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & t_x \\ a_{21} & a_{22} & a_{23} & t_y \\ a_{31} & a_{32} & a_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.2}$$

où les a_{ij} représentent l'orientation et l'échelle des voxels dans chaque direction et (t_x, t_y, t_z) représente la translation du volume dans l'espace monde.

Application de l'inverse de l'affine

Dans le code, la transformation appliquée est l'inverse de la matrice affine :

$$\mathbf{A}^{-1} = \texttt{np.linalg.inv(affine)} \tag{5.3}$$

Cette opération permet de convertir les coordonnées des streamlines exprimées dans l'espace monde RAS+ [a] (défini par le fichier `.tck` ou `.trk`) vers l'espace voxel correspondant à l'image NIfTI [67].
Pour chaque point $\mathbf{p}_i = (x_i, y_i, z_i)$ d'une streamline, on construit le vecteur homogène :

$$\tilde{\mathbf{p}}_i = \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix} \tag{5.4}$$

et on applique la transformation :

$$\tilde{\mathbf{p}}_i^{\text{voxel}} = \mathbf{A}^{-1} \cdot \tilde{\mathbf{p}}_i \tag{5.5}$$

Cela aligne les streamlines avec l'espace voxel de l'image anatomique.

Résultat

Le tableau final de streamlines transformées est ensuite retourné par la fonction `transform_streamlines`, garantissant que les coordonnées des fibres sont cohérentes avec l'image de fond.
Cette étape est cruciale pour assurer une visualisation précise et spatialement alignée des données de tractographie avec leur support anatomique.

a. Les coordonnées stockées dans les fichiers `.trk` et `.tck` sont exprimées en voxel, cependant, lorsque `DIPY` charge les tractographies, par défaut, ils transforment les coordonnées en espace monde RAS+.

Exemple d'utilisation et options disponibles

Nous illustrons ici le fonctionnement de l'outil à l'aide d'une image anatomique pondérée en T_1 accompagnée de deux fichiers de tractographie. L'application repose sur un système de « session » (ex. 5.3) dans lequel chaque chargement d'une nouvelle image anatomique initialise automatiquement une nouvelle session de travail. Chaque session conserve l'ensemble des paramètres définis par l'utilisateur ainsi que toutes les tractographies précédemment chargées et superposées à l'image anatomique correspondante. Ce mécanisme garantit une organisation claire et indépendante des données manipulées au sein de l'interface.

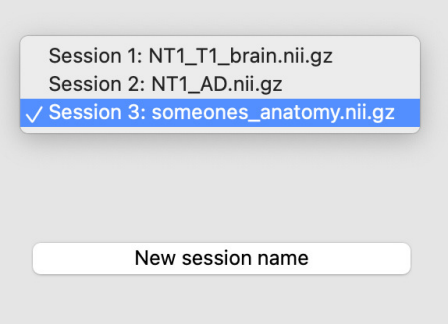


FIGURE 5.3 – *Plusieurs sessions de travail*

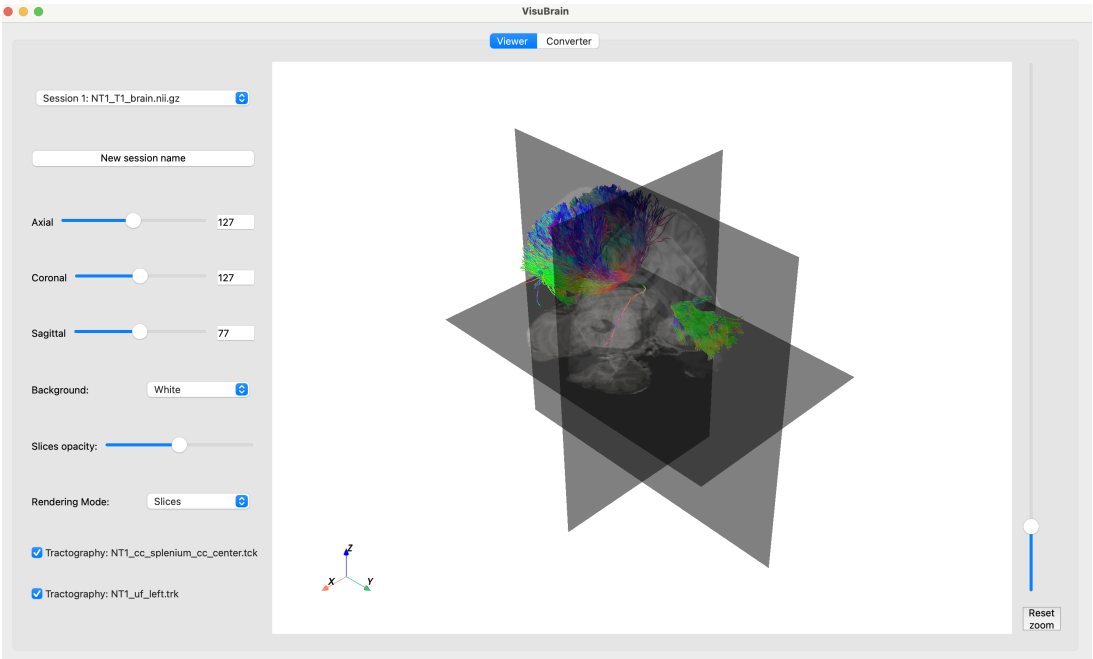


FIGURE 5.4 – *Exemple 1 : visualisation des fichiers chargés, vue de deux tracts simultanés*

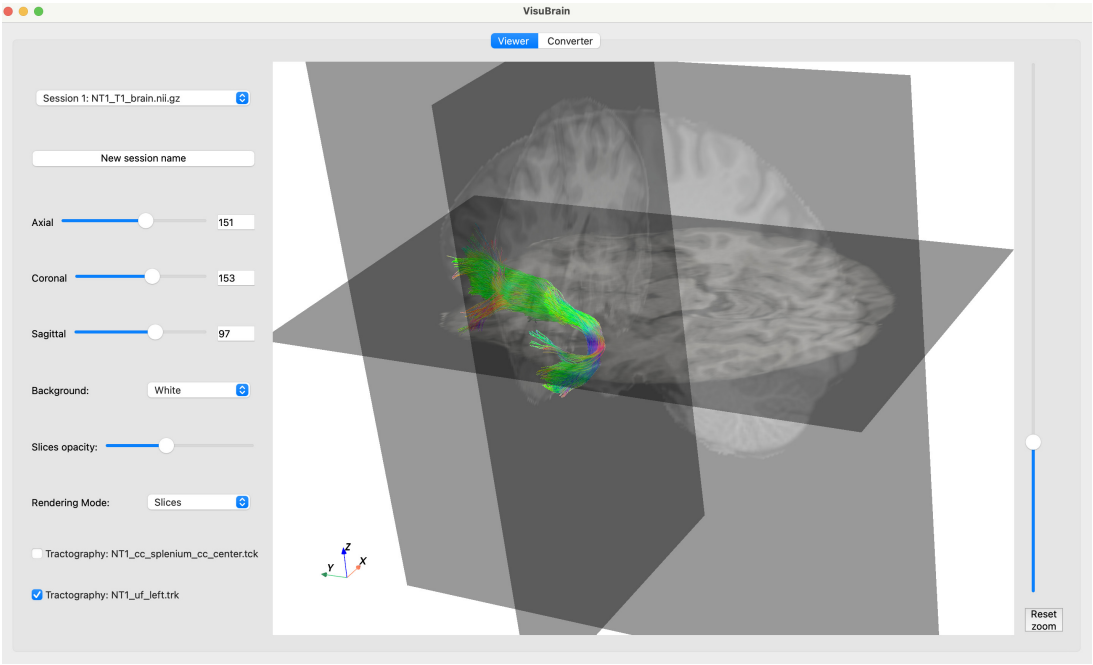


FIGURE 5.5 – *Exemple 2 : visualisation des fichiers chargés, vue de zoom variable et sélection spécifique des tracts*

L'outil propose un ensemble d'options d'affichage et de manipulation (fig 5.4, 5.5) :

— Les trois premiers curseurs situés à gauche permettent de naviguer à travers les coupes axiale, coronale et sagittale du volume anatomique actuellement affiché.

— Le paramètre *Background* permet de modifier la couleur de l'arrière-plan, offrant ainsi un meilleur contraste pour faire ressortir certaines structures de l'image.

— Le curseur *Slices Opacity* ajuste le niveau de transparence des coupes orthogonales du volume NIfTI, permettant une superposition plus ou moins marquée avec d'autres éléments visuels.

— Le paramètre *Rendering Mode* permet de basculer entre deux modes d'affichage : soit les trois plans orthogonaux classiques (axial, coronal, sagittal), soit un rendu volumique 3D de l'image anatomique (ex. 5.6).

— Le bouton *New session name* offre à l'utilisateur la possibilité de renommer la session en cours, afin de mieux organiser son espace de travail ou d'y apporter une identification plus explicite selon ses besoins.

— Enfin, le curseur vertical situé à droite facilite le zoom avant/arrière sur l'image, notamment pour les utilisateurs d'ordinateurs portables équipés d'un pavé tactile, offrant une alternative ergonomique au geste de pincement.

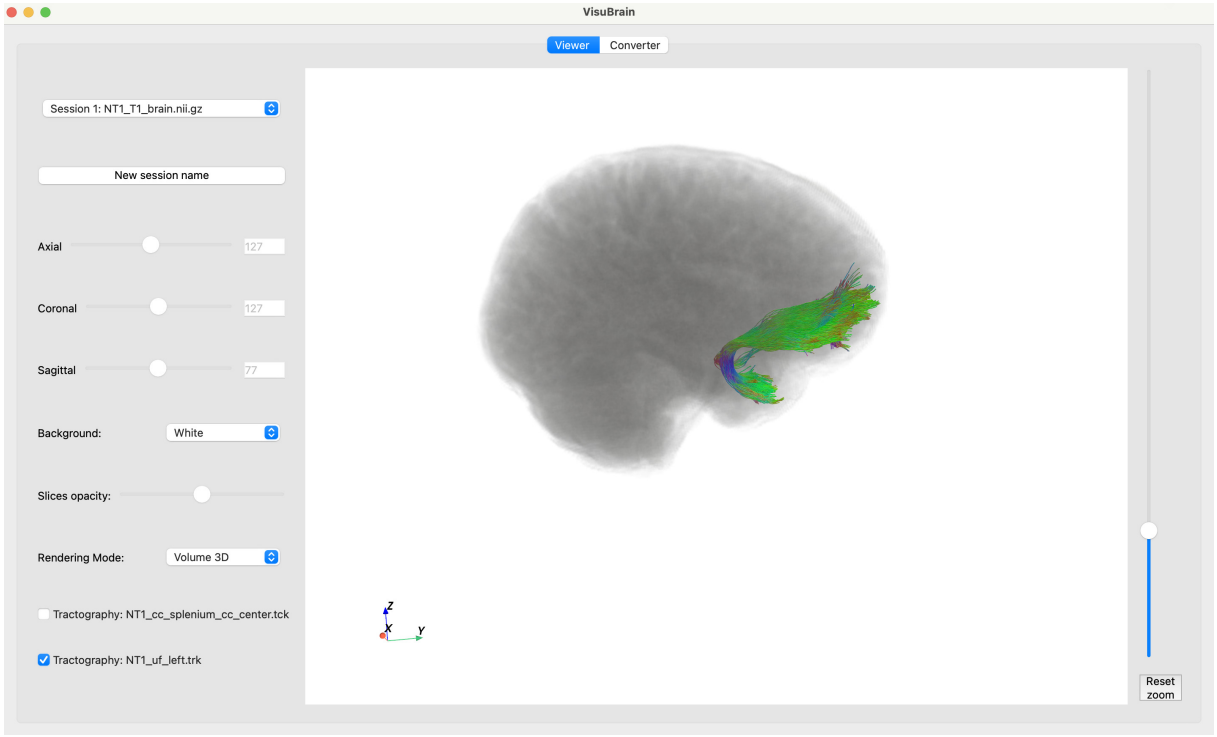
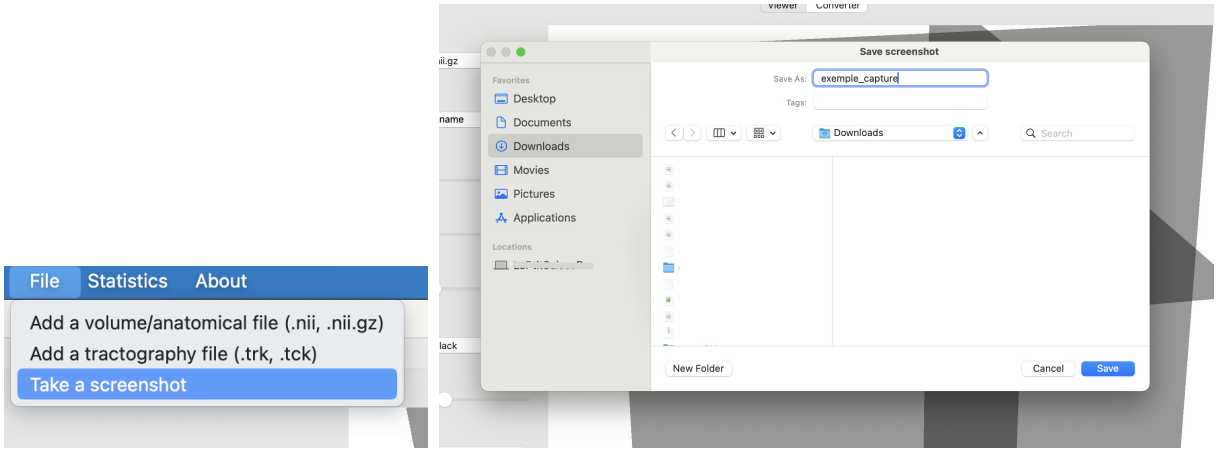


FIGURE 5.6 – *Exemple du rendu 3D d'un volume de cerveau pondéré en T1 avec les tracts spécifiquement sélectionnés*

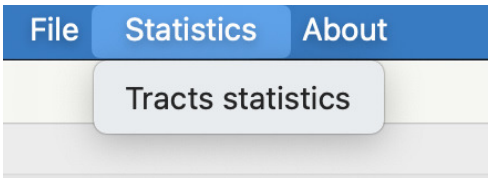
L'onglet *File* offre également la possibilité de capturer rapidement une image de la vue utilisateur actuellement affichée. L'utilisateur peut ensuite choisir d'enregistrer cette capture d'écran directement sur sa machine dans un répertoire local de son choix (fig 5.7).



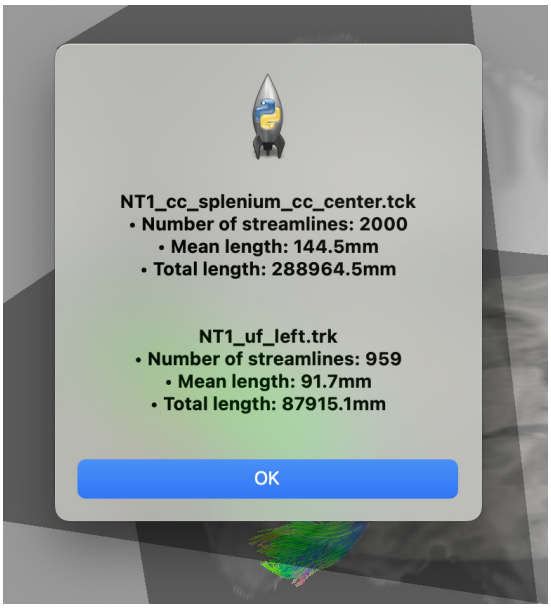
(a) (b)

FIGURE 5.7 – *(A) Onglet « Take a screenshot » ; (B) Sauvegarde d'une capture d'écran localement sur la machine de l'utilisateur*

L'onglet *Statistics* fournit un ensemble d'informations relatives aux tractographies chargées. Pour chaque fichier importé, sont affichés : le nom du fichier, la longueur moyenne des streamlines, le nombre total de streamlines contenues et la longueur cumulée de l'ensemble des fibres (fig 5.8).



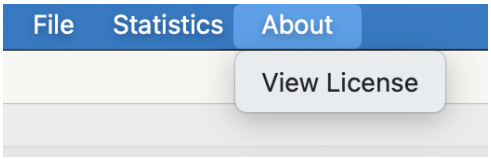
(a)



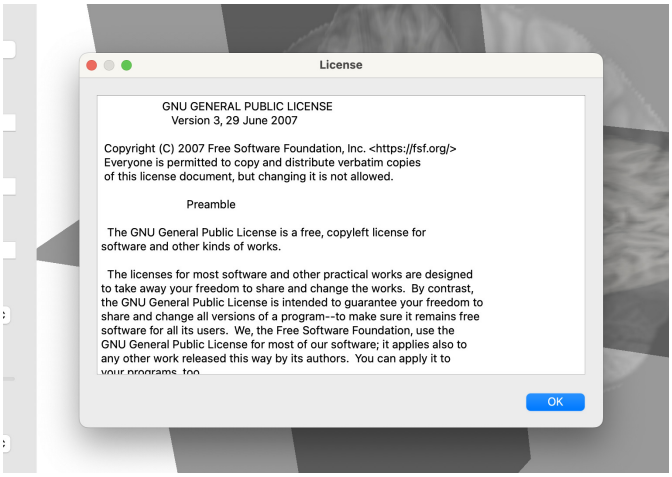
(b)

FIGURE 5.8 – *(A) Onglet « Statistics » ; (B) Exemple de statistiques de deux tractographies chargées*

Enfin, l'onglet *About* permet de consulter le texte complet de la licence logicielle sous laquelle l'application est distribuée (fig 5.9).



(a)



(b)

FIGURE 5.9 – *(A) Onglet « About » ; (B) Affichage de la licence logicielle depuis l'onglet « About »*

5.1.1 Visualisation des directions de diffusion

Dans les images de DTI, les directions principales de diffusion sont souvent codées en couleur pour faciliter l'interprétation [68, 69] :

— **Rouge** : gauche-droite (axe X)
— **Vert** : antéro-postérieur (axe Y)
— **Bleu** : supérieur-inférieur (axe Z)

Cette convention permet une visualisation intuitive des orientations des fibres nerveuses dans l'espace tridimensionnel.

L'affichage visuel des streamlines dans le visualisateur repose sur une coloration codée selon la direction spatiale locale du mouvement le long de chaque fibre. L'objectif est de fournir une information intuitive sur l'orientation tridimensionnelle des segments de tractographie en associant chaque direction principale à une couleur de base.

Principe de détermination de la couleur

Pour chaque streamline, représentée comme une liste ordonnée de points dans $\mathbf{R}^3$, soit :

$$\mathbf{s} = [\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n], \quad \text{avec } \mathbf{p}_i = (x_i, y_i, z_i) \in \mathbf{R}^3 \tag{5.6}$$

on définit un vecteur tangent local $\mathbf{t}_i$ pour chaque point comme la différence entre deux points consécutifs :

$$\mathbf{t}_i = \mathbf{p}_{i+1} - \mathbf{p}_i \quad \text{pour } i = 1, \ldots, n-1 \tag{5.7}$$

Pour garantir un vecteur couleur de même taille que la streamline, le dernier vecteur $\mathbf{t}_n$ est défini comme $\mathbf{t}_n = \mathbf{t}_{n-1}$.
Ces vecteurs tangents sont ensuite normalisés pour obtenir des vecteurs unitaires :

$$\hat{\mathbf{t}}_i = \frac{\mathbf{t}_i}{\|\mathbf{t}_i\|} \quad \text{avec } \|\mathbf{t}_i\| = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 + (z_{i+1} - z_i)^2} \tag{5.8}$$

Projection en couleur RGB

Les composantes du vecteur tangent normalisé sont ensuite projetées dans l'espace de couleurs RGB, selon la convention suivante :

- composante $X \rightarrow$ Rouge (Red)
- composante $Y \rightarrow$ Vert (Green)
- composante $Z \rightarrow$ Bleu (Blue)

Pour chaque point on applique :

$$\text{RGB}_i = \left(|\hat{t}_{i,x}|, |\hat{t}_{i,y}|, |\hat{t}_{i,z}|\right) \cdot 255 \tag{5.9}$$

On prend la valeur absolue pour éliminer l'effet de l'orientation du vecteur (un vecteur vers la gauche ou vers la droite produit la même couleur). Le résultat est converti en entiers non signés (`uint8`) pour la compatibilité avec l'encodage des couleurs des moteurs de rendu graphique.

Ce codage permet d'identifier visuellement les orientations dominantes :

- Les fibres majoritairement orientées selon l'axe gauche-droite (X) apparaissent en rouge.
- Celles orientées selon l'axe antéro-postérieur (Y) apparaissent en vert.
- Celles suivant l'axe inféro-supérieur (Z) apparaissent en bleu.
- Les orientations mixtes produisent des couleurs composites (par exemple, rouge + bleu donne une sorte de mauve).

Ce type de visualisation est un standard dans les logiciels de tractographie car il permet une lecture rapide de l'organisation spatiale des faisceaux de matière blanche dans le cerveau.

5.2 Intégration des scripts de conversion dans la IUG

L'interface principale de l'onglet `Conversion` (fig 5.10) permet à l'utilisateur d'ajouter facilement un fichier en entrée qu'il souhaite convertir (1). Pour certains formats (`.fbr` et `.tck`) la fourniture d'une image de référence anatomique est requise (2). Cette exigence répond à la fois à des contraintes techniques (liées à la façon dont ces fichiers sont définis) et à un souci de précision afin de garantir une conversion aussi fiable et cohérente que possible.

Une fois le fichier source chargé, l'outil analyse automatiquement son format et propose à l'utilisateur une liste des conversions disponibles (3). L'utilisateur peut alors choisir le format cible souhaité. Enfin, l'outil permet d'enregistrer le fichier converti à l'emplacement de son choix sur la machine (4). Il suffit ensuite de presser le bouton "Convert" pour effectuer la conversion voulue (5).

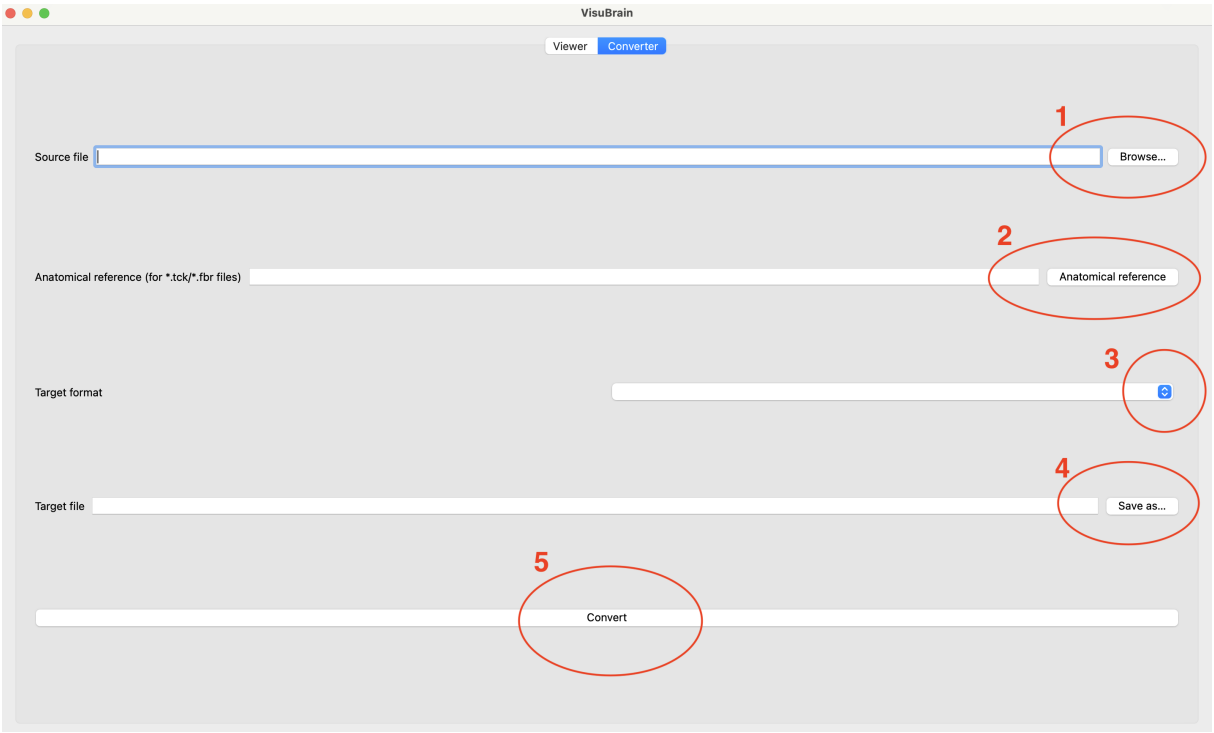


FIGURE 5.10 – *Interface principale de l'outil de conversion. (1) Charger un fichier (2) Charger une référence anatomique (3) Sélectionner le format cible (4) Emplacement de sauvegarde (5) Confirmer la conversion*

Exemple d'utilisation

Nous illustrons ici un cas d'exemple (ex. 5.11) avec un fichier au format `.trk`. L'objectif est de le transformer au format `.tck`. Il faut d'abord charger le fichier souhaité.

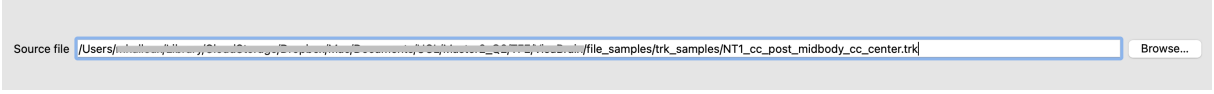


FIGURE 5.11 – *Exemple de chargement d'un fichier* `.trk`

Ensuite il faut sélectionner le format souhaité parmi ceux proposé (ex. 5.12), dans le cas d'exemple nous choisissons le format `.tck`.

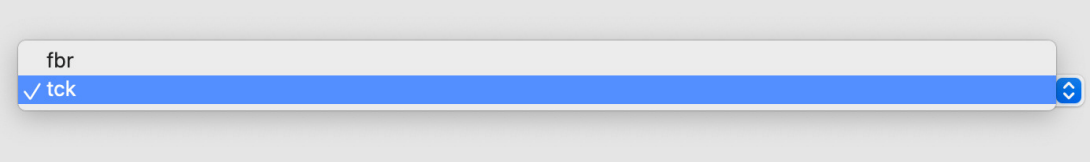


FIGURE 5.12 – *Sélection du format cible*

Il suffit ensuite à l'utilisateur de sélectionner un nom de fichier ainsi qu'un emplacement de sauvegarde approprié (**1**, **2**), puis de confirmer son choix (**3**) pour finaliser l'enregistrement du fichier converti (fig 5.13). L'étape finale consiste à cliquer sur le bouton `Convert`. Si le processus se déroule correctement, une fenêtre contextuelle (popup) informe l'utilisateur que la conversion a été effectuée avec succès (fig 5.14).

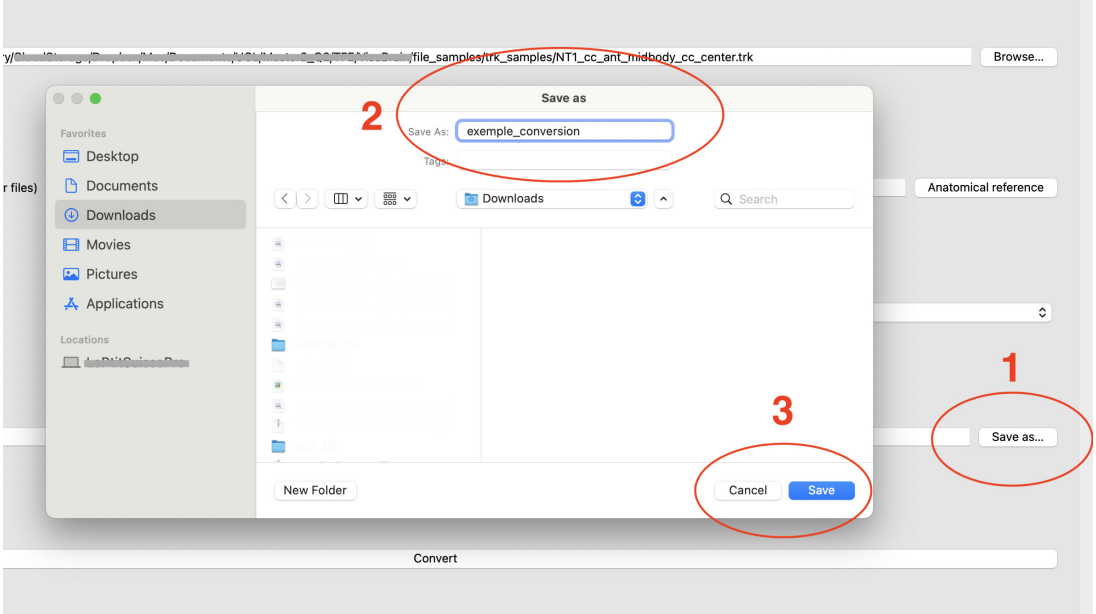


FIGURE 5.13 – *Sauvegarde locale, sur la machine de l'utilisateur, du fichier converti. (1) Emplacement de sauvegarde (2) Nom du fichier (3) Confirmer la sauvegarde*

FIGURE 5.14 – *Message de confirmation de finalisation de la conversion. (1) Confirmer la conversion (2) Pop-up affirmant l'état de la conversion*

5.3 Anonymisation des données

Toutes les données, fichiers et informations à caractère personnel utilisées dans le cadre de ce travail ont été traitées dans le strict respect des règles d'éthique et de confidentialité. En plus d'être rendues totalement anonymes, leur utilisation a systématiquement fait l'objet d'un consentement explicite, libre et éclairé, fourni par écrit par la (ou les) personne(s) concernée(s).

Les images présentées dans ce rapport à des fins d'illustration ne contiennent aucune information identifiable et leur qualité visuelle ne permet pas une reconnaissance faciale ou corporelle des sujets scannés. Par souci d'intégrité personnelle et conformément aux principes éthiques que ce travail entend promouvoir, aucune image ou donnée d'origine humaine n'a été incluse sans un accord formel et documenté.

Les images utilisées pour les tests durant les phases de développement du logiciel ont été soumises au même processus rigoureux d'anonymisation que les autres données exploitées dans ce travail. Par ailleurs, afin de préserver pleinement la confidentialité des sujets, ces fichiers ne sont en aucun cas inclus ou accessibles dans le dépôt public en ligne associé au projet.

De plus, aucune donnée n'est enregistrée ni stockée de manière persistante : l'ensemble des opérations est effectué localement sur la machine de l'utilisateur et toutes les données chargées sont automatiquement supprimées à la fermeture du logiciel garantissant ainsi une confidentialité totale.

Cette rigueur garantit que l'ensemble du travail s'inscrive dans une démarche respectueuse de la vie privée, de la dignité et des droits des individus, en accord avec les exigences légales et les bonnes pratiques en recherche scientifique et médicale.

Chapitre 6

Discussion et perspectives

6.1 Apports du projet au domaine de la neuro-imagerie

Le travail présenté dans ce mémoire vise à proposer une solution concrète, fonctionnelle et extensible à une problématique bien connue dans le domaine de la neuro-imagerie : la difficulté d'interopérabilité entre formats de fichiers d'imagerie médicale issus d'écosystèmes logiciels distincts. Cette fragmentation limite les synergies entre outils, freine la réutilisation des données et rend les flux de travail rigides.

L'un des apports de ce projet est de permettre une meilleure communication entre deux univers qui jusqu'à présent restaient relativement hermétiques : d'une part, l'écosystème ouvert des formats `NIfTI`, `VOI`, `TCK` et `TRK`, largement utilisés dans les pipelines modernes d'analyse et dans les outils open source (tels que MRtrix3 ou DIPY), et d'autre part, le système plus fermé de *BrainVoyager*, qui repose sur ses propres formats propriétaires utilisés dans un outil "closed source". Dans la pratique, BrainVoyager reste aujourd'hui un outil central et performant dans certaines structures cliniques et de recherche, notamment grâce à ses capacités avancées de visualisation, à son intégration des modalités fonctionnelles et à sa prise en charge directe des analyses de tractographie couplées à des données IRMf. Ces fonctionnalités sont encore rarement égalées par les alternatives open source.

C'est précisément à cette interface que se positionne la contribution de ce travail : offrir un pont bidirectionnel entre ces mondes afin de faciliter l'intégration des outils open source dans des environnements tels que BrainVoyager. En cela, l'outil répond à une demande concrète exprimée par l'équipe de l'Institute of Neuroscience à l'UCLouvain, confrontée à des contraintes d'incompatibilité de formats qui rendaient jusque-là difficile l'adoption de certains outils modernes de tractographie.

En pratique, le logiciel développé permet une visualisation interactive de fichiers `NIfTI`, `TRK` et `TCK`, ainsi que la conversion automatique entre ces formats et ceux utilisés par BrainVoyager. L'approche modulaire du code, sa documentation complète et sa licence libre garantissent une maintenabilité à long terme et une ouverture à la collaboration. Le projet est ainsi conçu non pas comme une solution figée mais comme une plateforme évolutive que d'autres membres de la communauté pourront adapter, enrichir et intégrer à leur propre pipeline.

En somme, ce travail propose plus qu'un outil technique, il souhaite une contribution structurelle à la neuro-imagerie en abaissant les barrières d'entrée à l'usage des formats ouverts et en favorisant une adoption progressive d'approches plus interopérables et transparentes pour s'intégrer dans la démarche des sciences ouvertes.

Contribution à la communauté logicielle BrainVoyager

Dans une démarche d'ouverture et de partage des connaissances, ce projet a également donné lieu à la soumission d'une contribution [1] au dépôt public `bvbabel`, une bibliothèque maintenue par Omer Faruk Gulban [2], visant à proposer des outils en Python pour lire et écrire les formats propriétaires de BrainVoyager. Plus spécifiquement, une routine de lecture binaire du format `.fbr` (fichier de tractographie utilisé par BrainVoyager), développée dans le cadre de ce mémoire, a été proposée à l'intégration dans ce dépôt.

Cette routine a été conçue pour offrir une lecture fiable et fidèle des fichiers `.fbr`, dans une optique d'interopérabilité avec des outils open source. Elle permet ainsi d'utiliser ces fichiers au sein de pipelines de traitement alternatifs, favorisant l'ouverture des formats à la communauté scientifique. La société Brain Innovation a accueilli favorablement cette initiative et a invité à soumettre formellement la contribution, en vue d'un examen pour une intégration future. Cette dernière est actuellement en attente d'acceptation.

Cette reconnaissance manifeste l'intérêt pratique et la pertinence de l'approche adoptée dans ce travail et ouvre la voie à des collaborations plus étroites entre l'écosystème BrainVoyager et les environnements open source de la neuro-imagerie.

6.2 Défis techniques rencontrés

L'un des principaux défis rencontrés au cours de ce projet réside dans la disponibilité limitée de la documentation scientifique et technique concernant les formats de fichiers traités. Bien que certains formats standards tels que `NIfTI` ou `DICOM` soient relativement bien documentés et accompagnés de spécifications précises, d'autres, notamment les formats utilisés pour la tractographie ou certains formats propriétaires, ont nécessité une approche plus exploratoire combinant analyse manuelle de fichiers, vérifications empiriques et échanges répétés avec les développeurs ou le support technique.

Les difficultés ont été particulièrement marquées avec les formats propriétaires utilisés par `BrainVoyager`, tels que `.vmr` ou `.fbr`. La documentation publique sur ces formats est soit partielle, soit obsolète, reposant parfois sur des spécifications qui ne sont plus d'actualité. À titre d'illustration, les fichiers `VMR` générés actuellement reposent sur la version 4 du format, alors que la documentation accessible en ligne détaille principalement la version 2. De plus, des évolutions de terminologie ou même des divergences dans les définitions de certains champs entre versions, compliquent davantage l'interprétation de ces fichiers.

Ces limitations rencontrées soulignent une problématique plus large inhérente à l'utilisation d'outils propriétaires dans le cadre d'une recherche scientifique. L'absence de documentation exhaustive, combinée à un manque de transparence sur les structures internes de manipulation de fichiers, constitue un frein important à la reproductibilité des analyses. Ces obstacles peuvent ralentir la diffusion des méthodes et limiter l'interopérabilité avec d'autres environnements de recherche. L'expérience vécue avec BrainVoyager dans ce travail illustre de manière concrète les conséquences de ces verrous techniques : bien que puissants et largement utilisés, certains outils propriétaires restent difficiles à intégrer pleinement dans des écosystèmes scientifiques ouverts tant que leurs formats internes et processus ne sont pas documentés de manière formelle et accessible.

1. `https://github.com/ofgulban/bvbabel/pull/68`
2. `https://github.com/ofgulban/bvbabel`

6.3 Limites actuelles et voies d'amélioration

Bien que l'outil développé constitue une base fonctionnelle et prometteuse pour la conversion et la visualisation de données en neuro-imagerie, plusieurs pistes d'amélioration peuvent être envisagées pour renforcer ses performances, sa flexibilité et son utilité à long terme.

Tout d'abord, concernant la conversion de fichiers de tractographie vers le format propriétaire `.fbr` de BrainVoyager, les phases finales de test ont confirmé que les coordonnées des streamlines étaient correctement encodées : la forme globale des faisceaux correspond fidèlement au fichier source et les couleurs représentant les directions de diffusion sont également correctement restituées. Cependant, un dysfonctionnement a été observé au niveau de l'alignement spatial des fibres une fois visualisées dans BrainVoyager : celles-ci ne sont pas correctement superposées au volume anatomique. Ce décalage suggère une incompatibilité avec le système de coordonnées attendu par BrainVoyager lors de l'importation du fichier `fbr`. Malgré les efforts pour respecter les spécifications connues de ce format, le comportement observé indique un traitement interne du logiciel qui n'est pas documenté de manière suffisamment transparente que pour déduire la source du problème. La difficulté est renforcée par le caractère propriétaire et fermé du logiciel qui ne permet pas de vérifier précisément les transformations appliquées. Des échanges ont été menés avec l'équipe de support de BrainVoyager à ce sujet, sans aboutir à une solution claire. Deux pistes d'amélioration restent à explorer : la première consisterait à valider que les coordonnées attendues dans le fichier `fbr` sont bien exprimées en millimètres et non en indices voxel (point sur lequel BrainVoyager n'a pas encore fourni de confirmation définitive). La seconde piste serait d'envisager l'association d'un fichier complémentaire (par exemple un fichier de transformation ou de géométrie propre à BrainVoyager) susceptible d'assurer une correspondance spatiale correcte entre les fibres et le volume anatomique. Ce cas illustre une limite importante dans l'interopérabilité avec des formats propriétaires peu documentés et souligne la nécessité de poursuivre les efforts vers une meilleure standardisation et ouverture des formats en neuro-imagerie.

Ensuite, l'élargissement du nombre de formats pris en charge reste une priorité. Le domaine de la neuro-imagerie utilise une grande variété de formats, certains très spécifiques à un outil ou à une communauté de pratique. Ajouter de nouveaux modules de conversion permettrait de rendre l'outil plus universel et de répondre à des cas d'usage encore non couverts.

De la même manière, étendre la compatibilité de la visualisation à davantage de formats de fichiers permettrait d'utiliser l'outil dans des contextes plus variés.

Sur le plan fonctionnel, l'intégration d'un système de superposition de régions d'intérêt serait un ajout pertinent pour visualiser et interpréter les résultats d'analyses directement sur l'image anatomique.

Enfin, la transformation du projet en package Python distribuable via `pip` constitue une évolution naturelle qui permettrait de simplifier son installation, sa diffusion et son intégration dans d'autres environnements. Le versionnage sémantique déjà en place dans le code facilite cette démarche et ouvre la voie à un cycle de publication structuré, incluant une gestion claire des mises à jour. Dans cette perspective, il serait intéressant de se consacrer à la rédaction d'une documentation accessible. Ce besoin a été mis en évidence

par les difficultés rencontrées au cours de ce travail, notamment en lien avec l'insuffisance ou l'obsolescence de la documentation de certains formats propriétaires comme ceux de BrainVoyager (cf. 6.2). Bien que la rédaction de cette documentation n'ait pas été prioritaire dans les phases initiales du développement, elle figure clairement parmi les améliorations futures prévues. Elle participerait à rendre le projet plus transparent pour être encore plus facilement modifiable par d'autres utilisateurs ou développeurs.

Conclusion

Ce travail de fin d'études s'inscrit dans un contexte technique et scientifique où la nécessité d'harmonisation des formats et d'interopérabilité entre outils est bien présente. Le domaine de la neuro-imagerie, bien qu'extrêmement riche et innovant, souffre encore d'une forte fragmentation en matière de formats de données, de chaînes de traitement et de visualisation. Cette réalité complique le travail des chercheurs et freine les efforts de reproductibilité.

L'objectif de ce projet était de proposer une solution simple, unifiée et accessible qui fasse un premier pas vers la réponse à ce besoin d'intégration. En construisant un outil capable de convertir automatiquement plusieurs formats de fichiers d'imagerie cérébrale, tout en permettant la visualisation de certains d'entre eux dans une interface claire et interactive, ce travail apporte une réponse concrète à une difficulté récurrente du domaine. Il a également permis d'explorer en profondeur la logique des différents formats manipulés, les conventions spatiales associées ainsi que les défis de conversion d'un espace de référence à un autre.

D'un point de vue technique, le projet a mobilisé un ensemble d'outils robustes et bien établis dans la communauté Python en mettant un accent particulier lors du développement logiciel sur la structuration modulaire du code, la maintenabilité, l'évolutivité et la documentation. L'approche adoptée a été résolument orientée vers la reproductibilité et la communauté, comme en témoigne la publication open source du code accompagnée d'une licence libre. Ce projet propose également une démarche éthique : aucune donnée personnelle n'est stockée, tous les traitements s'effectuent localement et les visuels utilisés dans ce travail sont anonymisés et accompagnés d'un accord écrit. L'engagement éthique renforce la légitimité de l'outil dans un contexte où la gestion responsable des données est une exigence croissante.

Sur le plan personnel, ce travail a représenté un véritable terrain d'apprentissage, autant sur les plans technique et scientifique que sur les plans organisationnel et collaboratif. Il m'a permis de mettre en œuvre un ensemble de techniques de développement logiciel et de méthodologies organisationnelles abordées au cours du cursus universitaire, en les appliquant de manière concrète dans un contexte de projet complet. Étant initialement novice dans le domaine médical et plus particulièrement en neuro-imagerie, les défis techniques liés au développement se sont accompagnés d'une courbe d'apprentissage importante concernant les normes, la terminologie, les conventions spatiales, les structures de fichiers et autres spécificités propres à ce domaine. Cette double difficulté a exigé une adaptation constante mais a également renforcé la dimension formatrice du travail. Il m'a permis d'approfondir la compréhension des enjeux propres à la neuro-imagerie, d'appliquer de manière concrète les principes de développement logiciel enseignés durant le cursus et de contribuer de manière utile à une problématique bien réelle du monde scientifique.

Ce mémoire a été rempli de découvertes, de remises en question, d'essais, d'erreurs et de réussites.

Si ce projet ne constitue qu'une première étape, j'ai l'espoir qu'il pose les bases d'une solution élargie, appelée à évoluer avec les besoins dans ce domaine. Il est en effet souhaitable que d'autres formats, outils et fonctionnalités viennent enrichir cette base. Le travail réalisé ici pourrait ainsi servir de socle fiable pour d'autres développeurs, chercheurs et institutions souhaitant améliorer l'interopérabilité et la qualité des outils en neuro-imagerie. Ce travail apporte ainsi un début de réponse que je pense prometteuse, claire et simple à une problématique fragmentaire qui agite la communauté des chercheurs en neuro-imagerie depuis des années.

Bibliographie

[1] Fondation pour la Recherche Médicale (FRM). Imagerie cérébrale : percer les mystères du cerveau. https://www.frm.org/fr/actualites/imagerie-cerebrale-percer-les-mysteres-du-cerveau, 16/02/2016. Dernièrement consulté le 28 avril 2025.

[2] Institut du Cerveau. Imagerie fonctionnelle : Technique d'imagerie. https://institutducerveau.org/lexique/imagerie-fonctionnelle. Dernièrement consulté le 28 avril 2025.

[3] By Pierre Bellec et l'équipe PSY3018. Imagerie par résonance magnétique. https://methodes-cogneuro.github.io/irm.html. illustration faite par P. Bellec (2021) sous licence CC-BY. Dernièrement consulté le 28/05/2025.

[4] By Pierre Bellec et l'équipe PSY3018. Cartes cérébrales. https://methodes-cogneuro.github.io/cartes_cerebrales.html. 2023. Dernièrement consulté le 29/05/2025.

[5] Dmitry Kurzhunov. Novel reconstruction and quantification methods for oxygen-17 magnetic resonance imaging at clinical field strengths. https://www.researchgate.net/publication/318658798_Novel_Reconstruction_and_Quantification_Methods_for_Oxygen-17_Magnetic_Resonance_Imaging_at_Clinical_Field_Strengths?channel=doi&linkId=5975fc11aca2728d0267b0a3&showFulltext=true. September 2017. Dernièrement consulté le 28/05/2025.

[6] IMAIOS. Sélection du plan de coupe : Excitation sélective. https://www.imaios.com/fr/e-mri/codage-spatial-du-signal/selection-du-plan-de-coupe. 2008-2025. Dernièrement consulté le 29/05/2025.

[7] Christian Ledig Andreas Schuh Ricardo Guerrero Rolf A Heckemann and Daniel Rueckert. Structural brain imaging in alzheimer's disease and mild cognitive impairment : biomarker analysis and shared morphometry database. Sci. Rep., 8(1) :1–16, July 2018.

[8] ELSTER LLC AD Elster. Invention of functional mri : Who invented functional mr imaging (fmri) ? https://mriquestions.com/who-invented-fmri.html. © 2024. Dernièrement consulté le 29/05/2025.

[9] D. le Bihan, médecin radiologue français, membre de l'Académie nationale de Médecine (2003). imagerie de diffusion en irm l.f. https://www.academie-medecine.fr/le-dictionnaire/index.php?q=imagerie%20de%20diffusion%20%28en%20IRM%29, 2018. Dernièrement consulté le 29 avril 2025.

[10] IMAIOS. Séquences pondérées en diffusion. https://www.imaios.com/fr/e-mri/irm-tenseur-de-diffusion/sequences-ponderees-en-diffusion. © 2024. Dernièrement consulté le 29/05/2025.

[11] Nicolas Delinte. Quantifying the microstructural changes in the brain of patients with alcohol use disorder after abstinence via diffusion mri. https://hdl.handle.net/2078.2/16359. 2019-2020. Mémoire de fin d'étude de Master [120] : ingénieur civil biomédical.

[12] Michele Cerra. Assessing microstructural brain differences in epileptic patients with vagus nerve stimulation via diffusion mri, tractography and machine learning. https://hdl.handle.net/2078.2/35379. 2022-2023. End of study thesis of Master [120] in Computer Science and Engineering.

[13] Franck Mauconduit. Diffusion tensor magnetic resonance imaging of the brain : from modeling to high resolved 3d imaging. applications and fibre tracking in a schizophrenia mouse model. quantitative diffusion tensor imaging by magnetic resonance : methodological developments, modeling and fibre tracking imaging. brain applications in three models : trauma, schizophrenia and tumors. https://www.researchgate.net/publication/278640102_Diffusion_tensor_magnetic_resonance_imaging_of_the_brain_from_modeling_to_high_resolved_3D_imaging_Applications_and_fibre_tracking_in_a_schizophrenia_mouse_model_Quantitative_Diffusion_Tensor_Imaging_. December 2011.

[14] JI Berman, MR Lanza, L Blaskey, JC Edgar, TPL Roberts. High angular resolution diffusion imaging probabilistic tractography of the auditory radiation. https://pmc.ncbi.nlm.nih.gov/articles/PMC3987782/, 19/11/2012. Dernièrement consulté le 29 avril 2025.

[15] Eleftherios Garyfallidis. "towards an accurate brain tractography", phd thesis, university of cambridge. https://www.researchgate.net/publication/233960603_Eleftherios_Garyfallidis_Towards_an_accurate_brain_tractography_PhD_thesis_University_of_Cambridge_2012. 2012. Lien vers l'image : https://www.researchgate.net/figure/A-simplified-example-showing-in-i-and-ii-the-same-data-set-i-The-yellow-line-sh
fig12_233960603.

[16] Matlab Help Center. Medical image coordinate systems. https://nl.mathworks.com/help/medical-imaging/ug/medical-image-coordinate-systems.html, 2024. Dernièrement consulté le 02/05/2025.

[17] Nibabel. Radiological vs neurological convention, 2006. Dernièrement consulté le 02/05/2025.

[18] Graham Wideman. Orientation and voxel-order terminology : Ras, las, lpi, rpi, xyz and all that, 29/08/2005. Dernièrement consulté le 02/05/2025.

[19] BrainVoyager. Coordinate systems. https://www.brainvoyager.com/bv/doc/UsersGuide/CoordsAndTransforms/CoordinateSystems.html, 2025. Dernièrement consulté le 2025-05-02.

[20] NiBabel developers. Radiological vs neurological conventions. https://nipy.org/nibabel/neuro_radio_conventions.html, 2025. Dernièrement consulté le 02/05/2025.

[21] 3D Slicer. Coordinate systems. https://slicer.readthedocs.io/en/latest/user_guide/coordinate_systems.html#coordinate-systems. © Copyright 2025, Slicer Community.

[22] BrainMap. The mni brain and the talairach atlas. https://brainmap.org/training/BrettTransform.html, 2025. Dernièrement consulté le 06/05/2025.

[23] Radiopaedia : Craig Hacking. Basal ganglia (gray's illustrations). https://prod-images-static.radiopaedia.org/images/56735167/0._big_gallery.jpeg. Dernièrement consulté le 06/05/2025.

[24] BrainMap. How are the different head and mri coordinate systems defined ? https://www.fieldtriptoolbox.org/faq/source/coordsys/, 05/02/2025. Dernièrement consulté le 06/05/2025.

[25] Radiopaedia : Franck Gaillard. Ac-pc line (diagram). https://prod-images-static.radiopaedia.org/images/27832865/a37d34594d61e30738a4b488349b73_big_gallery.jpeg. Dernièrement consulté le 06/05/2025.

[26] BrainVoyager v23.0. Spatial transformation matrices. https://www.brainvoyager.com/bv/doc/UsersGuide/CoordsAndTransforms/SpatialTransformationMatrices.html. Dernièrement consulté le 07/05/2025.

[27] Neutrium. Basics of affine transformation. https://neutrium.net/articles/mathematics/basics-of-affine-transformation/, 11/09/2012. Dernièrement consulté le 07/05/2025.

[28] 3D Slicer Community. 3d slicer user documentation. https://www.slicer.org, 2004. Slicer Foundation.

[29] Rainer Goebel. Brainvoyager. https://www.brainvoyager.com/products/brainvoyager.html, 2024.

[30] NITRC : NeuroImaging Tools and Resources Collaboratory. mricrogl :main-page. https://www.nitrc.org/plugins/mwiki/index.php/mricrogl:MainPage, 12/12/2022.

[31] mricron introduction. https://www.nitrc.org/projects/mricron/.

[32] MRtrix3 Development Team. Mrview user manual. https://www.mrtrix.org, 2019.

[33] Jean-Donald Tournier, Rudiger Smith, Derek Raffelt, et al. Mrtrix3 : A fast, flexible and open software framework for medical image processing and visualisation. *NeuroImage*, 202 :116137, 2019.

[34] Athinoula A. Martinos Ruopeng Wang, Van J. Wedeen. Trackvis user manual. http://www.trackvis.org, 2015. Center for Biomedical Imaging, Department of Radiology, Massachusetts General Hospital.

[35] R. Wang T. Benner A. G. Sorensen and V. J. Wedeen. Diffusion toolkit : A software package for diffusion imaging data processing and tractography. *Martinos Center for Biomedical Imaging, MGH, Charlestown, MA, United States*, 2007.

[36] francopestilli et autres (forum de discussion). Tractography data format 942. https://github.com/nipy/nibabel/issues/942, 30/07/2020. Dernièrement consulté le 10/05/2025.

[37] Atlassian. Qu'est-ce que la méthodologie agile ? https://www.atlassian.com/fr/agile. Dernièrement consulté le 15/05/2025.

[38] Alain Collignon et Joachim Schöpfel. Méthodologie de gestion agile d'un projet. scrum – les principes de base. *CAIRN Info*, pages 12–15, 05/07/2016. https://shs.cairn.info/revue-i2d-information-donnees-et-documents-2016-2-page-12?lang=fr&tab=texte-integral.

[39] Atlassian. Qu'est-ce que scrum ? https://www.atlassian.com/fr/agile/scrum. Dernièrement consulté le 18/05/2025.

[40] Tuleap. Comprendre la méthode agile scrum en 10 min. https://www.tuleap.org/fr/agile/comprendre-methode-agile-scrum-10-minutes. Dernièrement consulté le 19/05/2025.

[41] Copyright © 1996-2002 2004-2019 2021 2022 Free Software Foundation Inc. [Traduit de l'anglais par Frédéric Couchet et Karl Pradène]. Qu'est-ce que le logiciel libre ? https://www.gnu.org/philosophy/free-sw.html. Dernièrement consulté le 19/05/2025.

[42] holger krekel and pytest-dev team. pytest : helps you write better programs. https://docs.pytest.org/en/stable/#. Copyright © 2015,.

[43] PyCharm : Maha Taqi. Pytest vs. unittest : Which is better ? https://blog.jetbrains.com/pycharm/2024/03/pytest-vs-unittest/#. March 15, 2024, Dernièrement consulté le 10/05/2025.

[44] Documentation pyvista. https://docs.pyvista.org/. Dernièrement consulté le 12/05/2025.

[45] Documentation vtk. https://vtk.org/. Dernièrement consulté le 12/05/2025.

[46] Documentation pyvistaqt. https://pypi.org/project/pyvistaqt/. Dernièrement consulté le 12/05/2025.

[47] Documentation numpy. https://numpy.org/. Dernièrement consulté le 12/05/2025.

[48] Documentation nibabel. https://nipy.org/nibabel/gettingstarted.html. Dernièrement consulté le 12/05/2025.

[49] Documentation dipy. https://dipy.org/index.html. Dernièrement consulté le 12/05/2025.

[50] Documentation pyqt6. https://pypi.org/project/PyQt6/. Dernièrement consulté le 12/05/2025.

[51] Omer Faruk Gulban. bvbabel : Python tools for reading and writing brainvoyager files. https://github.com/ofgulban/bvbabel. Dernièrement consulté le 12/05/2025.

[52] Tom Preston-Werner : inventeur de Gravatars et cofondateur de GitHub. Gestion sémantique de version 2.0.0. https://semver.org/lang/fr/. Licence Creative Commons — CC BY 3.0. Dernièrement consulté le 10/05/2025.

[53] Brain Innovation B.V. The format documentation of the fbr file. https://helpdesk.brainvoyager.com/603922-The-Format-of-FBR-Files. 19/05/2025. Dernièrement consulté le 22/05/2025.

[54] TrackVis. The format documentation of the trk file. https://trackvis.org/docs/?subsect=fileformat. Copyright © 2015 Ruopeng Wang, Van J. Wedeen, Athinoula A. Martinos Center for Biomedical Imaging, Department of Radiology, Massachusetts General Hospital. Dernièrement consulté le 20/05/2025.

[55] MRtrix3. The format documentation of the tck file. https://mrtrix.readthedocs.io/en/dev/getting_started/image_data.html. Dernièrement consulté le 18/05/2025.

[56] MRtrix3. The format documentation of the nifti-2 file. https://brainder.org/2015/04/03/the-nifti-2-file-format/. 03/04/2015. Dernièrement consulté le 10/05/2025.

[57] MRtrix3. The format documentation of the nifti-1 file. https://brainder.org/2012/09/23/the-nifti-file-format/. 23/09/2012. Dernièrement consulté le 10/05/2025.

[58] Chris Rorden : développeur de MRIcron and Columbia SC USA Professor University of South Carolina Columbia SC USA Co-Director McCausland Center for Brain Imaging Columbia SC USA Principal Investigator, Neuropsychology Lab. Chris rorden's home. https://people.cas.sc.edu/rorden/. Dernièrement consulté le 18/05/2025.

[59] Chris Rorden : développeur de MRIcron. Forum de discussion mricron. https://www.nitrc.org/forum/message.php?msg_id=21363. 01/06/2017. Dernièrement consulté le 09/05/2025.

[60] Chris Rorden : développeur de MRIcron. Forum de discussion mricron. https://www.nitrc.org/forum/message.php?msg_id=24541. 26/05/2018. Dernièrement consulté le 05/05/2025.

[61] Brain Innovation B.V. The format documentation of the vmr file. https://helpdesk.brainvoyager.com/952829-The-Format-of-VMR-Files. 09/04/2025. Dernièrement consulté le 15/05/2025.

[62] Hester Breman. Positioning and transformations in brainvoyager, July 22, 2008.

[63] Nibabel. Api nibabel : Documentation load_save. https://nipy.org/nibabel/reference/nibabel.loadsave.html#module-nibabel.loadsave. 2006. Dernièrement consulté le 17/05/2025.

[64] Nibabel. Api nibabel : Documentation funcs. https://nipy.org/nibabel/reference/nibabel.funcs.html#module-nibabel.funcs. 2006. Dernièrement consulté le 17/05/2025.

[65] Nibabel. Image voxel orientation. https://nipy.org/nibabel/image_orientation.html. 2006. Dernièrement consulté le 14/05/2025.

[66] Omer Faruk Gulban. Read, write, create brainvoyager vmr file format. https://github.com/ofgulban/bvbabel/blob/main/bvbabel/vmr.py, 2023. Dernièrement consulté le 10/05/2025.

[67] Nibabel. The inverse of the affine gives the mapping from scanner to voxel. https://nipy.org/nibabel/coordinate_systems.html#the-inverse-of-the-affine-gives-the-mapping-from-scanner-to-voxel. Dernièrement consulté le 19/05/2025.

[68] esanum. Comment lire une tractographie ? https://www.esanum.fr/today/posts/tractographie#:~:text=Comment%20lire%20une%20tractographie%3F. 09/06/2015. Dernièrement consulté le 03/05/2025.

[69] Pierre Bellec et l'équipe PSY3018. Irm de diffusion. https://methods-cogneuro.github.io/irm_diffusion.html. 2023. Dernièrement consulté le 04/05/2025. Paragraphe "Caractéristiques des tenseurs".

[70] Nicolas Delinte. Github "pilab medical imaging". https://github.com/PiLAB-Medical-Imaging. A collection of repositories made by/for PiLAB students.

[71] Afni. 5.1.1. afni gui : Startup. https://afni.nimh.nih.gov/pub/dist/doc/htmldoc/afniandafni/gui_guide/afni_gui_start.html.

[72] Afni. Afni file : Readme.afnigui. https://afni.nimh.nih.gov/pub/dist/doc/program_help/README.afnigui.html, 2025.

[73] Thermo Fisher Scientific. Amira user's guide. https://www.thermofisher.com/be/en/home/electron-microscopy/products/software-em-3d-vis/amira-software.html?SID=srch-srp-AMIRA. Thermo Fisher Scientific.

[74] Bioimage suite : An integrated medical image analysis suite. Papademetris X. and Scheinost D. PIs, Dept. of Radiology and Biomedical Imaging, Yale School of Medicine.

[75] MedicalExpoConnect. Image analysis software brainance md. https://www.medicalexpo.com/prod/advantis-medical-imaging/product-122346-900633.html.

[76] Advantis Medical Imaging. Advantis brain (brainance md). https://advantis.io/brain/.

[77] Brainbrowser. https://brainbrowser.cbrain.mcgill.ca, 2018. McGill University.

[78] Mingrui Xia. Brainnet viewer manual. https://www.nitrc.org/docman/view.php/504/1280/brainnet, 2013. National Key Laboratory of Cognitive Neuroscience and Learning, Beijing Normal University.

[79] BrainStorm Team. Brainstorm user's manual. https://neuroimage.usc.edu/brainstorm, Dernière édition le 2025-04-23 16 :15 :56 par TakfarinasMedani.

[80] Supported file format for brainstorm software. https://neuroimage.usc.edu/brainstorm/Introduction#Supported_file_formats, Dernière édition le 2025-04-23 16 :15 :56 par TakfarinasMedani.

[81] BrainSuite Developers. Brainsuite documentation. http://brainsuite.org, 2002.

[82] Brainvisa. https://brainvisa.info/web/. © 2024, BrainVisa group.

[83] FMRIB Analysis Group. Fsleyes : Fsl image viewer. https://open.win.ox.ac.uk/pages/fsl/fsleyes/fsleyes/userdoc/, 2016.

[84] FreeSurfer Community. Freeview user's guide. https://surfer.nmr.mgh.harvard.edu/fswiki/Freeview, 2010.

[85] IMAIOS. Imaios dicom viewer. https://www.imaios.com/fr/imaios-dicom-viewer.

[86] Invesalius. https://invesalius.github.io. CTI Renato Archer.

[87] Itk-snap tutorial. http://www.itksnap.org, 2006. Paul Yushkevich, Jilei Hao, Alison Pouch, Sadhana Ravikumar and colleagues at the Penn Image Computing and Science Laboratory (PICSL) at the University of Pennsylvania.

[88] Research Imaging Institute. Mango user manual. http://ric.uthscsa.edu/mango, 2020. University of Texas Health Science Center at San Antonio.

[89] Microdicom - free dicom viewer for windows. https://www.microdicom.com. MicroDicom Ltd.

[90] Mni display - software for visualization and segmentation of surfaces and volumes. https://www.bic.mni.mcgill.ca/software/Display/Display.html, 2016. Institut et hôpital neurologiques de Montréal.

[91] Neuroelf. https://neuroelf.net/. 2017.

[92] Open health imaging foundation viewer : Extensible web imaging platform. https://ohif.org/. Open Health Imaging Foundation.

[93] Dicomweb™. https://www.dicomstandard.org/using/dicomweb/. The Medical Imaging Technology Association (MITA), a division of NEMA, serves as the DICOM Secretariat.

[94] Olea Medical. Mri software sphere® 3.0. https://www.medicalexpo.com/prod/olea-medical/product-130687-1028918.html.

[95] © 2012-2016 Sébastien Jodogne University Hospital of Liège. © 2017-2023 Osimis S.A. © 2024-2024 Orthanc Team SRL. © 2021-2024 Sébastien Jodogne ICTEAM UCLouvain. Orthanc-server. https://www.orthanc-server.com/.

[96] Copyright © 2025 Pixmeo. Osirix dicom viewer. https://www.osirix-viewer.com/, 2025.

[97] Bayard Roth. Papaya : A native html5 medical research image viewer. https://github.com/rii-mango/Papaya, 2004. Stanford University.

[98] Inria CEA Neurospin Scott Burns Vanderbilt University Martin Luessi Harvard Medical School MGH Martinos Center Eric Larson University of Washington ILABS Michael Waskom, New-York University (NYU) Alexandre Gramfort. Pysurfer documentation. https://pysurfer.github.io. 2012-2020.

[99] Pycortex documentation. https://gallantlab.org/pycortex/index.html. Dernièrement consulté le 09/05/2025.

[100] J. S. Gao, A. G. Huth, M. D. Lescroart, and J. L. Gallant. Pycortex : an interactive surface visualizer for fmri. *Frontiers in Neuroinformatics*, 2015.

[101] Copyright © 2009-2025 Medixant. Radiant dicom viewer. https://www.radiantviewer.com/.

[102] Github de slice drop. https://github.com/slicedrop/slicedrop.github.com. Dernièrement consulté le 09/05/2025.

[103] Voreen documentation. https://voreen.uni-muenster.de. University of Münster.

[104] Volview. https://volview.kitware.com/. © 2024 Kitware, Inc.

[105] Supported file formats. https://kitware.github.io/VolView/loading_data.html#file-formats. Copyright © 2020-PRESENT Kitware, Inc.

[106] Weasis dicom viewer. https://weasis.org/en/index.html. Dernièrement consulté le 30/04/2025.

Annexe A

Annexes techniques

A.1 Code source du travail

Le code source de l'outil présenté dans ce travail est accessible via le dépôt suivant sur GitHub : `https://github.com/MichaHllx/VisuBrain`. Le choix de la plateforme s'est fait naturellement car celle-ci a été utilisée tout au long de la formation à l'UCLouvain pour la gestion de projets, le versionnage de code et la collaboration. GitHub est une plateforme largement adoptée dans le monde académique et professionnel en raison de sa fiabilité, de sa compatibilité avec de nombreux environnements de développement et de son intégration fluide avec les outils utilisés lors de ce projet.

Par ailleurs, le dépôt officiel du *PiLAB Medical Imaging* [70] présenté comme « une collection de référentiels créés par/pour les étudiants du PiLAB » est également hébergé sur GitHub, ce qui renforce la cohérence du choix de la plateforme.

A.2 Usage d'assistance numérique

Dans une démarche d'optimisation textuelle, l'assistant linguistique ChatGPT (modèle GPT-4, développé par OpenAI) a été utilisé de manière ponctuelle pour aider à la reformulation de certaines phrases et à la clarification stylistique. Cet outil n'a en aucun cas généré de contenu scientifique ou technique. L'ensemble du contenu technique, conceptuel et scientifique a été conçu et rédigé par l'auteur du présent mémoire (Michaël Halleux).

Annexe B

Panorama des outils en neuro-imagerie

AFNI GUI

AFNI (Analysis of Functional NeuroImages) est un environnement open-source riche pour le traitement et la visualisation de données fMRI, combinant coupe 2D/3D, etc. L'interface GUI permet le réglage interactif des overlays, la navigation dans les slices et la synchronisation pour afficher des contours de surface sur les volumes [71]. Ce visualisateur gère les formats propre à *afni* (les pairs de fichiers .HEAD and .BRIK) et NIfTI-1 (.nii, .nii.gz) [72].

Amira

Amira [73] est un logiciel commercial de visualisation 3D hautement interactif développé par Thermo Fisher. Il est utilisé pour l'analyse avancée de structures neuronales, incluant la reconstruction de réseaux, l'électrophysiologie, la visualisation multimodale et bien d'autres fonctions. Il lit DICOM, NIfTI, TIFF, STL, OBJ et d'autres formats. Il génère des fichiers de projet .hx ainsi que des exports volumétriques et de surfaces.

BioImage Suite

BioImage Suite [74] est une suite logicielle développée à l'Université de Yale, conçue pour l'analyse et la visualisation d'images médicales. Elle intègre des outils pour le traitement d'images, l'enregistrement, la segmentation et l'analyse fonctionnelle, notamment pour les données IRMf. BioImage Suite prend en charge une variété de formats d'images, y compris NIfTI et DICOM et offre des fonctionnalités avancées pour la visualisation 3D, la tractographie et l'analyse de surfaces corticales. La suite est disponible en version de bureau et en version web (BioImage Suite Web), cette dernière permettant une utilisation sans installation préalable via un navigateur web. BioImage Suite est largement utilisée dans la communauté de la neuroimagerie pour des applications de recherche. Cependant ce logiciel n'est pas approuvé pour un usage en clinique professionnel.

BrainanceMD Viewer

BrainanceMD Viewer est un visualiseur DICOM web-based [75] conçu spécifiquement pour l'imagerie avancée en neurodiagnostic. Il prend en charge l'affichage des volumes

de diffusion (DTI), perfusion et cartes fonctionnelles issues d'IRM multimodale. L'outil permet la superposition de différentes modalités (T1, T2, BOLD, DWI) ainsi que l'export de visualisations en images statiques (PNG) via son interface web sécurisée. Il ne génère pas de format propriétaire, mais peut produire des rapports PDF pour les cliniciens [76].

BrainBrowser

BrainBrowser est une suite d'outils JavaScript open source pour la visualisation WebGL de données en neurosciences. Il permet l'affichage en ligne de surfaces corticales (au format GIFTI) et de volumes 3D (NIfTI), sans nécessiter d'installation locale. Il fonctionne via un navigateur et n'écrit aucun format propriétaire [77].

BrainNet Viewer

BrainNet Viewer [78] est une application MATLAB qui permet la visualisation de graphes de connectivité cérébrale sur des surfaces 3D. Il accepte des fichiers de surface *.nv ainsi que les fichiers de surface FreeSurfer (.pial), des fichiers de coordonnées de nœuds (.node) qui représentent des ROIs (régions d'intérêt), des fichiers "d'arrêtes" (.edge) qui représentent les connexions entre les noeuds et des fichiers volumiques (uniquement NifTi, .nii ou la paire .img/.hdr). L'interface graphique permet de choisir le style de rendu, la transparence, la taille des nœuds et la pondération des liens. Les visualisations sont exportables en image haute qualité, sans création de format propriétaire.

Brainstorm

Brainstorm [79] est une application open source complète développé avec le soutien du NIH (National Institutes of Health) pour l'analyse et la visualisation de données MEG, EEG, fNIRS, ECoG, depth electrodes et multiunit electrophysiology. Elle offre des visualisations en coupe, en rendu 3D de surfaces corticales, en topographie 2D ainsi qu'une multitude de fonctionnalités d'analyse. Ce logiciel accepte beaucoup de formats de différents [80], par exemple pour les volumes IRM, ceux qui sont supportés sont les suivants : Analyze (.img/.hdr), BrainVISA GIS (.ima/.dim), CTF (.mri), DICOM (using SPM converter), MINC (.mnc), MGH (.mgh, .mgz), Neuromag (.fif) et Nifti-1 (.nii, .nii.gz).

BrainSuite

BrainSuite est un environnement de traitement et visualisation pour la neuroimagerie structurelle et diffusionnelle. Il lit les formats NIfTI et ses propres formats intermédiaires et permet le rendu de surfaces corticales, la segmentation de la matière grise/blanche, la génération de tractographies à partir de données DTI, la visualisation des résultats en 3D, ... [81].

BrainVISA

BrainVISA [82] est une plateforme logicielle développée pour l'analyse morphologique du cerveau à partir d'images IRM. Elle s'appuie sur un ensemble d'outils modulaires (notamment Morphologist) permettant la segmentation des tissus cérébraux et la reconstruction de surfaces corticales. BrainVISA intègre un visualiseur graphique, Anatomist, conçu pour la visualisation avancée de données volumétriques, de surfaces et de graphes

de structures corticales. Il supporte les formats NIfTI, MINC et ses propres formats propriétaires spécifiques (.ima, .arg, .tex).

FSLeyes

FSLeyes [83], le viewer de FSL, permet la visualisation multi-couches de volumes anatomiques (T1, T2) et fonctionnels (séries temporelles fMRI) en coupes orthogonales et en rendu 3D interactif. Il remplace petit à petit FSLView en offrant les mêmes fonctionnalités mais en en améliorant certaines. Il prend en charge les formats NIfTI (.nii, .nii.gz), Analyze (.hdr/.img), Gifti, VTK et FreeSurfer (.mgz, .mgh) sans pour autant écrire un format propriétaire.

FreeView

FreeView [84], inclus dans la suite FreeSurfer, affiche des volumes anatomiques et leurs segmentations ainsi que des surfaces corticales reconstruites, permettant des mesures de distance, des tracés de surface et la superposition de cartes statistiques en 3D. Il peut charger des images NIFTI et des formats spécifiques à FreeSurfer, tels que les données avec les extensions .mgz et .inflated.

IMAIOS DICOM Viewer

IMAIOS Viewer est un visualiseur en ligne accessible depuis n'importe quel navigateur, conçu à but éducatif pour l'enseignement de l'anatomie humaine. Il supporte le format DICOM, permet la navigation 2D (axiale, coronale, sagittale) et intègre des annotations superposables interactives issues de leur base de données anatomique. Il est principalement utilisé en mode lecture, sans édition et ne permet pas d'export de données modifiées, uniquement des captures d'écran [85].

InVesalius

InVesalius est un logiciel libre développé pour la reconstruction 3D à partir de tomodensitométrie (CT) et IRM, principalement au format DICOM. Il permet le seuillage interactif, la génération de modèles de surface en temps réel et l'export vers les formats STL, OBJ et autres. Il n'utilise pas de format propriétaire et il convient à la neuroanatomie clinique et à l'éducation [86].

ITK-SNAP

ITK-SNAP est un outil dédié à la segmentation semi-automatique d'images cérébrales 3D. Il permet le chargement de volumes NIfTI et DICOM, l'application de régions d'intérêt (ROI) manuelles ou assistées par région et leur affichage en coupes 2D synchronisées avec un rendu 3D. Il exporte les masques sous forme de fichiers NIfTI binaires, sans format propriétaire. Son interface simple est conçue pour faciliter le travail des cliniciens et chercheurs en segmentation structurelle [87].

Mango

Mango (Multi-image Analysis GUI) [88] est un visualiseur de neuroimagerie extensible développé à l'Université du Texas. Il offre des vues multiplan, des outils d'annotation et de ROI, l'affichage de surfaces 3D et la possibilité d'appliquer des scripts personnalisés via un environnement de plugin. Il lit de nombreux formats (Analyze, DICOM, NEMA-DES, MINC, NIFTI, NIFTI2, VTK, GIFTI (.surf.gii) et BrainVisa) et permet l'export d'images, de masques et de rapports au format PNG ou CSV. Mango ne génère pas de format propriétaire mais crée des fichiers de projet .mnc ou .roi.

MicroDicom

MicroDicom [89] est un visualiseur DICOM léger, orienté vers la visualisation et l'export de séries médicales. Il permet d'extraire des images, de naviguer par tranches, d'ajuster les fenêtres de contraste/luminosité et de mesurer des structures anatomiques. Il ne prend en charge que le format DICOM en entrée et permet l'export en BMP, JPEG, TIFF, PNG, GIF et autres. Aucun format propriétaire n'est donc généré avec ce logiciel.

MNI Display

MNI Display est un outil de visualisation développé au McConnell Brain Imaging Centre, conçu pour afficher des images anatomiques (IRM, TEP, etc.) en coupes orthogonales, en surfaces et en rendu 3D. Il supporte les formats FreeSurfer (.mgz/.mgh), MINC (Medical Imaging NetCDF), NIfTI et Analyze, Wavefront OBJ, Stanford (.ply), GIFTI (.gii), BrainSuite surfaces (.dfs), BrainSuite tractography (.dft) avec la possibilité de superposer plusieurs volumes (par exemple, atlas ou segmentations). Il permet aussi la visualisation de surfaces corticales reconstruites [90].

NeuroElf

NeuroElf [91] est un environnement de visualisation et d'analyse de données d'imagerie cérébrale entièrement intégré dans MATLAB. Il propose une interface graphique riche pour le traitement et l'inspection de données IRMf, la définition de modèles statistiques et l'exploration des résultats. Il prend en charge les formats NIfTI, Analyze, certains formats de BrainVoyager (VTC, AMR, VMR, ...), ainsi que ses propres objets internes (.sdm, .glm). NeuroElf met l'accent sur l'accessibilité des pipelines d'analyse et la traçabilité des étapes de traitement.

OHIF Viewer

OHIF Viewer (Open Health Imaging Foundation) [92] est une plateforme d'imagerie médicale open source, extensible et basée sur le web, conçue pour la visualisation d'images médicales. Elle offre une interface modulaire permettant l'intégration dans divers systèmes cliniques et de recherche. OHIF Viewer prend en charge les images médicales multidimensionnelles (2D, 3D et 4D), notamment les données DICOM. Le visualiseur fonctionne avec les archives d'images prenant en charge DICOMWeb [93] et offre une API de source de données pour communiquer avec des archives via des formats d'API propriétaires.

Olea Sphere

Olea Medical propose une suite logicielle avancée pour le traitement des images médicales. Il proposent un logiciel pour la neuroimagerie, Olea Sphere [94], qui offre des outils pour l'analyse de données, analyse d'images, amélioration d'images, visualisation, diagnostic, post-traitement, importation d'images médicales, dépistage, capture d'images, évaluation, interprétation, etc.

Orthanc

Orthanc [95] est un serveur DICOM open-source, simple d'utilisation mais complet, conçu pour la gestion, le stockage et le partage d'images médicales dans les hôpitaux et en recherche. Développé à l'UCLouvain (ICTEAM) à l'initiative de Sébastien Jodogne, Orthanc est reconnu pour sa facilité de déploiement, son interface web intuitive, sa compatibilité avec les standards DICOMweb et son système de plugins qui en fait un outil modulaire et évolutif. Il ne génère pas de formats propriétaires et propose de puissants outils d'anonymisation, tout en restant multiplateforme. Orthanc bénéficie d'une large communauté et s'inscrit dans une démarche d'innovation ouverte, fidèle à la volonté de son créateur de rendre la gestion des images médicales accessible et interopérable à tous.

OsiriX

OsiriX est un visualiseur d'images médicales DICOM avancé, largement utilisé dans le monde cependant il est uniquement conçu pour les systèmes macOS. Il prend en charge une large gamme de modalités d'imagerie, notamment l'IRM, la TDM, la TEP et l'échographie, en utilisant le format DICOM standard. Il offre des techniques avancées de post-traitement en 2D et 3D, une technique innovante exclusive pour la navigation 3D et 4D et une intégration complète avec n'importe quel PACS. La version commerciale, OsiriX MD, est certifiée pour une utilisation clinique (FDA et CE) [96].

Papaya

Papaya est un visualiseur d'images médicales écrit en JavaScript pur, conçu pour l'affichage interactif dans un navigateur. Il supporte le chargement local ou distant de fichiers NIfTI, GIFTI et DICOM, avec rendu des volumes en coupes orthogonales et outils de zoom, scroll et réglage des contrastes. Il permet l'incrustation de surfaces et l'ajout de calques overlays. Bien qu'il ne sauvegarde pas de données, il peut exporter les vues sous forme d'images [97].

PySurfer

PySurfer est une bibliothèque Python basée sur VTK, conçue pour visualiser des surfaces corticales (prévu initialement uniquement pour les données issues de FreeSurfer) et y superposer des résultats d'analyse (activation IRMf, courbes, statistiques). Il lit les formats GIFTI, NIfTI et les surfaces .pial/.inflated de FreeSurfer. Il permet le scripting complet d'animations, la génération de rendus pour publication et l'interaction directe avec les scènes via Python. Aucun format propriétaire n'est généré [98].

Pycortex

Pycortex [99] est un logiciel Python permettant de générer des visualisations 3D interactives de données IRMf projetées sur des modèles de surface corticale. Il peut également générer des visualisations corticales 2D aplaties de haute qualité. [100].

RadiAnt DICOM Viewer

RadiAnt [101] est un visualiseur DICOM léger et performant pour images médicales, conçu pour les systèmes Windows. Il prend en charge une variété de modalités d'imagerie, notamment la radiographie numérique (CR, DX), la mammographie (MG), la tomodensitométrie (CT), l'IRM, la TEP, l'échographie (US), l'angiographie numérique (XA) et la médecine nucléaire (NM), en utilisant le format DICOM standard. RadiAnt offre des fonctionnalités telles que la lecture asynchrone des images, la gestion avancée de la mémoire pour l'ouverture simultanée de grandes études et des outils de mesure et d'annotation. Il ne génère pas de formats propriétaires, facilitant ainsi l'intégration dans divers flux de travail cliniques.

Slice Drop

SliceDrop [102] est logiciel basique web-based développé pour afficher des images médicales. Il permet l'affichage d'images individuellement ou de façon superposée mais non allignée. Il supporte plusieurs formats pour les volumes (DICOM, NIfTI, MGH/MGZ, NRRD), pour les formes (VTK PolyData, Freesurfer, STL) et pour les fibres (trk). SliceDrop ne produit pas de format propriétaire.

Voreen

Voreen (Volume Rendering Engine) [103] est une plateforme de visualisation scientifique conçue pour la recherche biomédicale. Elle permet un rendu 3D temps réel de volumes NIfTI, DICOM, RAW et autres, avec prise en charge de l'éclairage interactif, des gradients et de l'opacité. Voreen propose un environnement de développement graphique pour construire des workflows personnalisés incluant segmentation, fusion multimodale et analyse visuelle. Il est compatible avec les formats suivants : DICOM, TIFF stacks, HDF5, RAW, NetCDF, VTI, NIfTI-1.

VolView

VolView [104] est un système de rendu volumique développé par Kitware, permettant aux chercheurs et aux professionnels de santé de visualiser de manière profonde les données DICOM en 3D grâce à un rendu de volume interactif et cinématique. Ce logiciel tourne sur tous les navigateurs et ne nécessite donc pas d'installation. Il supporte plusieurs formats différents en plus du DICOM, tel que Nifti, NRRD et MHA [105].

Weasis

Weasis est un visualiseur DICOM multiplateforme, autonome et basé sur le web, conçu pour une intégration flexible dans les systèmes d'information cliniques. Il est conçu pour répondre aux besoins évolutifs des systèmes d'information clinique en imagerie médicale.

Ses fonctionnalités se concentrent sur l'accès web aux images radiologiques, le support d'une large gamme de types DICOM et l'offre de capacités multimédias. Weasis fournit des outils pour visualiser et analyser les images obtenues à partir d'équipements d'imagerie médicale conformément à la norme DICOM [106].

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**