

UNIVERSITÉ DE SHERBROOKE
Faculté de génie
Département de génie électrique et de génie informatique

Recalage pour imagerie multimodale tomographique sur petit animal

Mémoire de maîtrise
Spécialité : génie électrique

Massinissa BANDOU

Jury : Yves BÉRUBÉ-LAUZIÈRE (directeur)
Ruben GONZALEZ-RUBIO
Eric PLOURDE

Sherbrooke (Québec) Canada

Avril 2014

RÉSUMÉ

L'imagerie médicale joue désormais un rôle central tant en recherche fondamentale sur le développement des maladies que dans l'aide au diagnostic des pathologies. Il existe plusieurs techniques ou modalités d'imagerie qui permettent de visualiser de façon non invasive des corps biologiques et d'en fournir leurs propriétés structurelles et fonctionnelles. Un des problèmes majeurs rencontré est de pouvoir analyser et traiter ces techniques d'imagerie dans un référentiel commun. Ce problème, connu sous le nom de recalage ou registration, consiste en une estimation d'une transformation géométrique permettant la superposition spatiale des caractéristiques correspondantes entre les images. Deux contextes d'application peuvent être distingués : Le recalage monomodal qui traite les séquences temporelles d'images provenant de la même modalité et le recalage multimodal qui traite la mise en correspondance d'images venant de différentes modalités.

Le laboratoire TomOptUs de l'Université de Sherbrooke a développé un lit multimodal, avec marqueurs fiduciaux, destiné à accueillir un petit animal pour des prises de mesures avec différentes modalités. Il est donc intéressant de recaler ces modalités en utilisant l'information basée sur les attributs géométriques ou sur l'intensité des pixels ou voxels. Le laboratoire a aussi développé un système laser-scanner permettant de mesurer le relief d'un objet ou d'un animal afin d'obtenir des images profilométriques. Avec ces images, il devient possible de faire un recalage de surface avec d'autres modalités d'imagerie dans la mesure où on veut obtenir le profil extérieur de l'animal afin de faire une reconstruction interne 3D.

Le cadre de cette recherche consiste à développer un programme informatique permettant de répondre au besoin du laboratoire en matière de recalage d'image à l'aide de marqueurs fiduciaux et de recalage d'images profilométriques. Le présent document couvre la théorie et les algorithmes utilisés ainsi que la conception informatique détaillée permettant d'obtenir des résultats concluants.

Mots-clés : recalage monomodal, recalage multimodal, marqueurs fiduciaux, recalage de surface, image profilométrique.

TABLE DES MATIÈRES

1 INTRODUCTION	1
1.1 Mise en contexte et problématique	1
1.2 Laboratoire TomOptUS	2
1.3 Motivation	3
1.4 Objectifs du projet de recherche	3
1.5 Contributions originales	4
1.6 Plan du document	4
2 ÉTAT DE L'ART	7
2.1 L'imagerie par résonance magnétique (IRM)	7
2.2 La tomodensitométrie (TDM)	8
2.3 La tomographie d'émission par positrons (TEP)	10
2.4 La tomographie optique diffuse (TOD)	11
2.5 Principe du recalage d'image	12
2.6 Transformation géométrique	13
2.6.1 Transformation rigide	13
2.6.2 Transformation non-rigide	14
2.7 La métrique	16
2.7.1 Métrique quadratique	16
2.7.2 Corrélation croisée normalisée	17
2.7.3 Maximisation de l'information mutuelle	18
2.8 L'optimisation	21
2.9 L'interpolation	22
2.9.1 L'interpolation du plus proche voisin	22
2.9.2 L'interpolation linéaire	22
2.9.3 L'interpolation B-spline	23
2.10 Recalage par intensité des images	23
2.11 Recalage par traits caractéristiques	23
2.12 Recalage par points de référence	24
2.12.1 Marqueurs fiduciaux	24
2.13 Recalage de surface	26
2.14 Conclusion	26
3 CONCEPTION INFORMATIQUE	29
3.1 Installation informatique	29
3.1.1 CMake	30
3.1.2 Qt Designer	30
3.1.3 Conception de l'interface	32
3.2 Librairie VTK	32
3.2.1 Architecture de VTK	33
3.3 Librairie ITK	35

3.4 Recalage à l'aide des marqueurs fiduciaux	36
3.4.1 Calcul des centroïdes	41
3.4.2 Calcul de transformation rigide	42
3.5 Recalage de surface	45
3.5.1 Structure de l'algorithme	46
3.5.2 Extraction de surface à partir d'une image 3D	46
3.5.3 Positionnement des points de référence	49
3.6 Algorithme pour le raffinage des résultats	49
3.6.1 Algorithme ICP (iterative closest points)	50
3.6.2 Algorithme de l'information mutuelle	55
3.7 Conclusion	56
4 RÉSULTATS ET DISCUSSION	57
4.1 Résultat du recalage de surface	57
4.1.1 Recalage surface à surface	57
4.1.2 Recalage de surface avec volume tomographique	61
4.2 Résultat du recalage à l'aide des marqueurs fiduciaux	63
4.2.1 Recalage TEP avec TDM	63
4.2.2 Recalage TEP avec IRM	65
4.3 Discussion	67
5 CONCLUSION	69
5.1 Sommaire	69
5.2 Perspectives futures	70
ANNEXE A Exemple de code C++ pour afficher un maillage	71
ANNEXE B Programmation de la transformation rigide à l'aide de VTK	73
ANNEXE C Programmation de l'algorithme de l'information mutuelle à l'aide d'ITK	75
ANNEXE D La métrique du logiciel	79
LISTE DES RÉFÉRENCES	81

LISTE DES FIGURES

2.1	Gradient d'un champ magnétique dans la direction de y	8
2.2	Acquisition et reconstruction par rétroprojection	9
2.3	Série d'images TDM (gauche) ainsi que le volume obtenu à partir de ces séries (droite)	10
2.4	Image TEP de la vessie (surface rouge) d'une souris	11
2.5	Les composantes de base du recalage d'image : une métrique, une optimisation, une transformation et une interpolation	12
2.6	Corrélation croisée	17
2.7	Fonction de Parzen	20
2.8	Histogramme pour différent alignement d'image.	21
2.9	Différence entre interpolation du plus proche voisin à gauche et l'interpolation linéaire à droite	22
2.10	Image obtenue par la TDM d'un volume de 3 marqueurs fiduciaux autour d'un support pour petit animal	25
2.11	Image obtenue par la TEP des mêmes marqueurs fiduciaux présentés à la figure 2.10	25
3.1	Fenêtre exécutable CMake	30
3.2	Exemple d'un fichier CMakeList.txt	31
3.3	Configuration CMake, VTK et ITK.	31
3.4	<i>QVTKWidget</i> dans l'environnement de Qt Designer	31
3.5	Interface principale du programme	32
3.6	Structure pipeline composée de séries de filtres pour la lecture d'image 2D et 3D	33
3.7	Architecture de VTK de la source jusqu'à l'affichage	34
3.8	Contrôle (<i>Trackball</i>) de la scène dans un environnement 3D	34
3.9	Architecture programmée pour l'affichage d'image.	35
3.10	Exemple d'affichage d'un volume TEP d'une souris dans le logiciel	36
3.11	Structure pipeline d'intégration de VTK et ITK utilisée pour le recalage par information mutuelle	36
3.12	Structure pipeline VTK pour afficher un volume d'image TEP	37
3.13	Marqueurs fiduciaux vue par l'image TEP à gauche, l'image TDM au centre et l'image IRM à droite	38
3.14	Structure du <i>BoxWidget</i>	38
3.15	Diagramme UML servant à la création du <i>BoxWidget</i> via la classe <i>myBoxCallback</i> . La programmation du <i>BoxWidget</i> a été réalisée dans la fonction <i>Execute</i>	39
3.16	Structure d'implémentation d'un <i>widget</i> dans VTK	40
3.17	Image TEP d'une souris avec 3 marqueurs fiduciaux	40
3.18	Positionnement des <i>BoxWidgets</i> sur les 3 marqueurs fiduciaux	40
3.19	Diagramme UML pour la programmation de l'extraction du volume d'intérêt	41

3.20	Image TDM d'une souris avec 3 marqueurs fiduciaux	41
3.21	Positionnement des <i>BoxWidgets</i> sur les 3 marqueurs fiduciaux	42
3.22	Extraction des volumes des marqueurs fiduciaux de la TEP (droite), TDM (milieu) et l'IRM (gauche)	43
3.23	Triade formée en fonction des positions des marqueurs fiduciaux	43
3.24	Changement de repère de l'image source à l'image cible	44
3.25	Interface graphique de <i>Data Processing</i>	46
3.26	Algorithme de recalage de surface	47
3.27	Couches d'isodensité d'une image PET en fonction d'une certaine valeur <i>Contour value</i> pour l'algorithme Marching Cubes	48
3.28	Extraction de plusieurs couches d'isodensité constituant un organe d'une souris	48
3.29	Différentes topologies de polygones servant à la reconstruction de surface	49
3.30	Structure des points de référence pour le recalage de surface	49
3.31	Positionnement des points de référence correspondent pour l'image source et cible	50
3.32	Algorithme Iterative Closest Points	51
3.33	Profil d'une souris du système QOS à gauche et extraction du profil d'intérêt à droite	52
3.34	Interface d'option ICP	53
3.35	Structure de programmation suivi pour un recalage à l'aide des marqueurs fiduciaux et rafinage par l'algorithme de l'information mutuelle	55
4.1	Structure de programmation suivi pour un recalage d'un profil à un autre profil à l'aide des points de référence	57
4.2	Résultat d'un recalage surface avec une autre surface d'un diamant avec 3 points de référence	58
4.3	Structure de programmation suivi pour un recalage de surface d'une souris à l'aide de l'algorithme d'ICP	59
4.4	Résultat d'un recalage de surface d'une souris avec la même surface mais ayant subi une transformation rigide	60
4.5	Graphique de la distance moyenne entre les points correspondants en fonction du nombre d'itération	60
4.6	Structure de programmation suivi pour un recalage d'image profilométrique et d'un volume d'image TEP	61
4.7	Extraction de surface d'un volume TEP dont la valeur d'isodensité est de 0.0438	61
4.8	Résultat d'un recalage de surface d'une souris avec un volume TEP de la même souris	62
4.9	Graphique de la distance moyenne entre les points correspondants en fonction du nombre d'itération	63
4.10	Structure de programmation suivi pour un reclage d'image à l'aide des marqueurs fiduciaux	63

4.11 Résultat d'un recalage d'un volume de souris TEP (supérieur gauche) et d'un volume TDM (supérieur droit) à l'aide des marqueurs fiduciaux seulement	64
4.12 Recalage TEP-TDM avec marqueurs fiduciaux et information mutuelle	65
4.13 Vue du dessus (droite) et inclinée (gauche) du recalage TEP-TDM	65
4.14 Vue de trois coupes 2D du résultat du recalage TEP-TDM avec seulement marqueurs fiduciaux (droite) et rafinage par information mutuelle (gauche)	66
4.15 Recalage TEP-IRM avec marqueurs fiduciaux seulement	66
4.16 Vue de deux coupes 2D du résultat du recalage TEP-IRM avec seulement marqueurs fiduciaux (droite) et rafinage par information mutuelle (gauche)	67
D.1 Kiviat Chart	79

LISTE DES TABLEAUX

3.1	Tableau des unités de Hounsfield.	37
4.1	Mesure expérimentale du temps d'exécution du code informatique pour un recalage de surface à géométries simples.	58
4.2	Mesure expérimentale du temps d'exécution du code informatique pour un recalage avec ICP de surface à géométries complexes.	61
4.3	Mesure expérimentale du temps d'exécution du code informatique pour un recalage avec ICP de surface avec volume tomographique.	62
4.4	Mesure expérimentale du temps d'exécution du code informatique pour un recalage TEP-TDM avec information mutuelle.	65
4.5	Mesure expérimentale du temps d'exécution du code informatique pour un recalage TEP-IRM avec information mutuelle.	67
D.1	La métrique du logiciel.	79

LISTE DES ACRONYMES

Acronyme	Définition
IRM	Imagerie par résonnance magnétique
TDM	Tomodensitométrie
TEP	Tomographie d'émission par positrons
TOD	Tomographie optique diffuse
FDG	Fluorodésoxyglucose
PIR	Proche infrarouge
QOS	Quidd Optical System
ITK	Insight Toolkit
VTK	Visualization Toolkit
ICP	Iterative Closest Points

CHAPITRE 1

INTRODUCTION

1.1 Mise en contexte et problématique

Plusieurs modalités d'imagerie sont utilisées au quotidien dans la recherche biomédicale ou pharmaceutique permettant de visualiser indirectement de façon non invasive l'anatomie ou un processus physiologique ou métabolique d'un tissu biologique. Ces modalités sont obtenues sur la base de plusieurs phénomènes physiques tels que la résonance magnétique, la radioactivité, la réflexion d'ondes d'ultrason, etc. Ils interagissent avec le tissu sous forme d'absorption, d'atténuation et de diffusion. À partir de ces formes, il est possible de faire une reconstruction multi planaire bidirectionnelle, qui permet à travers d'un volume de générer des coupes frontales, sagittales ou axiales et de faire une reconstruction tridimensionnelle du tissu biologique. Cette reconstruction est très utile. Par exemple, elle permet d'élaborer des techniques chirurgicales plus sûres ou de simuler des opérations afin d'optimiser les chances d'un bon traitement ou d'un bon diagnostic.

Parmi les modalités d'imagerie les plus connues, on trouve la tomodensitométrie (TDM ou tomographie par rayons X), basée sur la mesure de l'atténuation des rayons X par les tissus biologiques. On trouve aussi l'imagerie par résonance magnétique (IRM) qui est basée sur l'excitation du spin des protons plongés dans un champ magnétique statique et irradiés par des ondes radio-fréquences, la tomographie d'émission par positrons (TEP), basée sur l'utilisation d'un traceur radioactif fonctionnalisé, qui via l'émission de positrons, permet de localiser et de quantifier l'activité métabolique et moléculaire dans un organisme vivant.

Ces modalités présentent des avantages et des inconvénients, par exemple l'IRM fournit des images de très haute résolution et elle est totalement inoffensive car elle n'utilise pas de radiation ionisante, mais elle se caractérise par la difficulté de synthétiser des traceurs fonctionnalisés ayant la capacité de modifier les temps de relaxation afin d'améliorer le contraste permettant de cibler des pathologies spécifiques. L'IRM est également excellente pour imager des tissus mous, par contre elle est fortement limitée pour des tissus durs comme les os. Elle nécessite en outre des aimants très puissants et l'appareillage est très cher. Également, une séance d'IRM dure typiquement 30 minutes, ce qui est relativement long. Comparativement à l'IRM, la TDM permet également d'obtenir de très hautes résolutions spatiales. Par contre, le contraste obtenu pour les tissus mous est intrinsèquement

moins bon que pour l'IRM ; la TDM étant en fait plus appropriée pour visualiser les tissus durs. On peut visualiser des tissus mous par TDM, mais dans ce cas on doit habituellement utiliser un agent de contraste injecté dans le sujet. Il est par contre difficile, voire impossible à ce jour, de synthétiser des agents de contraste fonctionnalisés pour la TDM. La TDM est extrêmement rapide (un corps entier peut être imaginé en moins de 5 minutes, ce qui est un grand avantage en milieu hospitalier. La TDM nécessite toutefois l'utilisation de fortes doses de rayons X qui peuvent induire des cancers lorsque répétées trop souvent. La TEP permet d'obtenir des images de processus fonctionnels et moléculaire par marquage. Cependant, comparée aux autres modalités, la TEP fournit des images à faible résolution spatiale.

Fabriquer un scanner combinant différentes modalités d'imagerie et leurs avantages respectifs s'avère complexe et très dispendieux. Une alternative est d'utiliser différents appareils et d'ensuite combiner les images par une technique de traitement d'images appelée recalage d'image (*registration* en anglais). Le recalage permet de mettre en correspondance les informations de deux images de même ou de différentes modalités afin d'obtenir une image combinée. Le recalage est en fait une méthode de recherche de transformation géométrique permettant d'aligner les points d'une image source aux points correspondants d'une image cible. Si l'objectif du recalage est de compenser uniquement des différences de repères d'acquisition, alors on utilisera une transformation rigide. Il est possible d'exécuter cette transformation à l'aide des marqueurs fiduciaux facilement repérables dans les champs de vue par les différentes modalités d'imagerie. Si l'objectif est de recaler deux modalités afin de compenser des différences structurelles ou anatomiques, alors on fait appel à des algorithmes impliquant les intensités des pixels.

Il est donc intéressant de développer un programme informatique capable d'appréhender des images multidimensionnelles et capable d'exploiter un algorithme de recalage pour les modalités TDM, IRM et TEP.

1.2 Laboratoire TomOptUS

Le laboratoire TomOptUS développe un scanner par tomographie optique diffuse (TOD) pour imagerie optique sur petit animal. Le laboratoire développe tous les aspects matériels (conception et réalisation de l'optique, l'électronique et la mécanique), la fabrication de milieux diffusants et absorbants synthétiques mimant les propriétés optiques de tissus biologiques appelés fantôme, les aspects informatiques (contrôle du scanner par ordinateur, visualisation des mesures, vision numérique 3D pour mesure de la forme extérieure des

objets) et mathématique (développement d’algorithmes de reconstruction tomographique, simulations Monte-Carlo de la propagation de la lumière en milieu diffusant à géométrie complexe). Le laboratoire se spécialise en imagerie intrinsèque permettant d’acquérir des données sur les structures internes d’un petit animal, ainsi qu’en imagerie par fluorescence afin de cartographier en 3D la concentration d’un traceur fonctionnalisé injecté dans celui-ci. Le laboratoire TomOptUS travaille également sur le développement de l’appareil d’imagerie pour petit animal, QOS (Quidd Optical System) de la compagnie Quidd opérationnel au Centre hospitalier universitaire de Sherbrooke (CHUS), afin de le doter de capacités d’imagerie tomographique (cet appareil ne permet pour le moment d’obtenir que des images planaires). Le groupe de recherche a également développé un lit multimodal pour petit animal avec marqueurs fiduciaux amovibles. Ces marqueurs contiennent des pastilles qui peuvent être identifiées dans une modalité d’imagerie comme la TEP, l’IRM ou la TDM. Il a aussi développé un profilomètre, un système de métrologie stréréoscopique utilisant deux caméras CCD, permettant de localiser dans l’espace le point de projection d’un faisceau laser sur une surface et ainsi mesurer cette surface.

1.3 Motivation

Afin d’obtenir le maximum d’information possible des modalités d’imagerie, on désire les combiner par une transformation rigide ou non rigide à l’aide des marqueurs fiduciaux. Puisque le laboratoire possède un système de balayage laser permettant d’obtenir une image profilométrique 3D, il est aussi intéressant de faire un recalage surfacique, c’est-à-dire combiner un profile 3D avec son homologue issu des autres modalités.

1.4 Objectifs du projet de recherche

Le projet de recherche vise à développer un programme informatique facile à utiliser et capable de faire des traitements de haut niveau tels que la segmentation ou l’extraction de surface. Le programme doit répondre aux besoins suivants :

- Donner la possibilité à l’utilisateur de recaler des images de différentes modalités comme la TEP, la TDM et l’IRM à l’aide des marqueurs fiduciaux.
- Permettre à l’utilisateur d’avoir une interaction avec les images et d’identifier les marqueurs fiduciaux dans un environnement 3D.
- Donner la possibilité de recaler une image profilométrique avec une image tomographique.

- Donner la possibilité de segmenter et d'extraire la surface d'intérêt d'une image profilométrique.
- Afficher le résultat visuel d'un recalage afin qu'il soit aisément appréciable.
- Donner à l'utilisateur la possibilité de choisir l'algorithme qui convient le mieux pour un meilleur résultat visuel.
- Donner le choix à l'utilisateur de faire un recalage rigide ou non rigide.
- Identifier la matrice de transformation finale permettant de superposer l'image source à l'image cible.
- Fournir la documentation nécessaire pour l'utilisation du programme et la compréhension de ce qu'il fait.
- Donner la possibilité de faire un développement ultérieur.

1.5 Contributions originales

À l'aide des librairies de visualisation et de segmentation existantes, ce projet de recherche apporte un soutien informatique au laboratoire TomOptUS en matière de recalage d'image 3D à l'aide des marqueurs fiduciaux. Il fournit un moyen d'exploiter un algorithme basé sur l'intensités des voxels pour un recalage multimodal ainsi que l'erreur quadratique moyenne pour un recalage monomodal. Ce projet donne aussi au laboratoire un outil informatique permettant de superposer, par une transformation géométrique, l'image profilométrique 3D avec son homologue issue d'une autre modalité d'imagerie telle la TEP, la TDM ou l'IRM, auquel nous avons extrait le profil 3D à l'aide d'un algorithme d'extraction de contour[Lorensen et Cline, 1987].

1.6 Plan du document

Ce document comprend cinq chapitres, le premier (présent chapitre) a permis de fournir une introduction au projet et d'énoncer les objectifs visés. Le second chapitre présente l'état de l'art qui comprend une description et du fonctionnement de toutes les modalités utilisées dans le projet ainsi que la théorie et les différentes méthodes de recalage proposées dans la littérature. Le troisième chapitre traite de la conception informatique qui a mené à la réalisation du projet. Le quatrième chapitre présente les résultats obtenus du projet

et finalement le dernier chapitre conclue les travaux et présente les points à améliorer au projet.

CHAPITRE 2

ÉTAT DE L'ART

Ce chapitre a pour objectif de donner une description des principes physiques des images médicales sur lesquelles s'est basé ce projet, des quatre critères pour caractériser une méthode de recalage, ainsi que les méthodes existantes de recalage d'image afin de pouvoir mieux situer les contributions et la conception de ce projet.

2.1 L'imagerie par résonance magnétique (IRM)

L'IRM repose sur la résonance magnétique nucléaire (RMN) des noyaux (protons) des atomes d'hydrogène présents dans les molécules d'eau qui sont très nombreuses dans les organismes vivants [Noblet, 2006]. En l'absence de champ magnétique externe, l'orientation des moments magnétiques des protons (moments qui sont déterminés par le spin du proton) est aléatoire et cela engendre une magnétisation nulle. Lorsque ces protons sont soumis à un champ magnétique extérieur constant \vec{B}_0 , leurs moments magnétiques entrent en mouvement rotatoire autour de la direction de ce champ appelé mouvement de précession qui a lieu à une certaine fréquence angulaire appelée fréquence de Larmor. Alors les protons s'alignent de façon parallèle ou anti-parallèle par rapport au champ magnétique engendrant une aimantation résultante non nulle dû à un léger excès des protons alignés dans le même sens que le champ. L'application d'une excitation électromagnétique radiofréquence modulée à la même fréquence de précession que les protons orientée perpendiculairement au champ \vec{B}_0 durant un bref moment, a pour conséquence de perturber l'équilibre et permet de basculer l'aimantation des protons à un certain angle par rapport au champ \vec{B}_0 . À l'arrêt de cette excitation, le retour de l'aimantation des protons à l'équilibre induit un signal de résonance qui est le signal de RMN. Ce retour à l'équilibre permet de mesurer les temps de relaxation longitudinale et transversale afin d'obtenir des images pondérées par ces temps de relaxation. En outre, la mesure de l'intensité du signal permet d'obtenir des images pondérées par la densité protonique dans les tissus [Desgrez *et al.*, 1994].

La fréquence de résonance dépend de la valeur du champ magnétique, la figure 2.1 montre le champ magnétique \vec{B}_0 auquel a été ajouté un gradient le long de la direction y . En raison des différences des intensités du champ magnétique engendré par le gradient, la mesure du signal RMN sur le long de l'axe y aura des fréquences différentes. En appliquant un

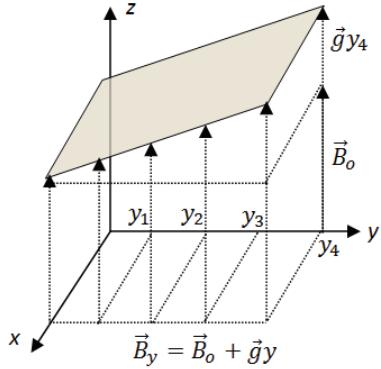


Figure 2.1 Gradient d'un champ magnétique dans la direction de y . Image provenant de [Université de Rennes, s. d.].

deuxième gradient sur le long de l'axe x et en induisant un pulse radiofréquence à l'aide d'une antenne, alors il sera possible de stimuler les protons d'une coupe le long de l'axe z et de pouvoir générer l'image IRM.

2.2 La tomodensitométrie (TDM)

La tomodensitométrie aussi appelée *CT scan (computed tomography scan)* est une méthode non invasive (mis à part l'utilisation de rayons X qui posent un risque pour la santé car il s'agit de radiation ionisante). La TDM est la plus ancienne modalité d'imagerie médicale tomographique. Elle est basée sur la mesure de l'atténuation des rayons-X lorsqu'ils traversent une section plane de tissu biologique [Goldman, 2007]. La tomographie (à savoir l'imagerie en coupes) est utilisée dans de nombreux domaines tels que la radioastronomie, la médecine, le milieu industriel, la sismique, la microscopie électronique, etc.

La TDM comporte deux étapes majeures, la première consiste à acquérir des projections en soumettant un objet (p.ex. un tissu biologique) à un balayage par des faisceaux de rayons-X sous différents angles de projection comme le montre la figure 2.2 [Romans, 2010]. C'est-à-dire qu'on applique une translation de la source de rayons-X le long de l'objet et on enregistre dans quelle proportion les rayons sont atténués à la sortie de l'objet et on répète la procédure à chaque incrémentation d'angle de projection. Les projections peuvent être représentées par la transformée de Radon décrite à l'équation donnée par

$$P_\theta(t) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) \delta(x \cos(\theta) + y \sin(\theta) - t) dx dy, \quad (2.1)$$

où $P_\theta(t)$ est l'intégrale de ligne de l'intensité de l'image $f(x, y)$ le long de la droite donnée par

$$x \cos(\theta) + y \sin(\theta) = t, \quad (2.2)$$

où t est la distance perpendiculaire de l'origine du plan xy à la droite elle-même. L'intégrale de ligne est calculée pour tous les angles θ de projection. À partir des projections acquises,

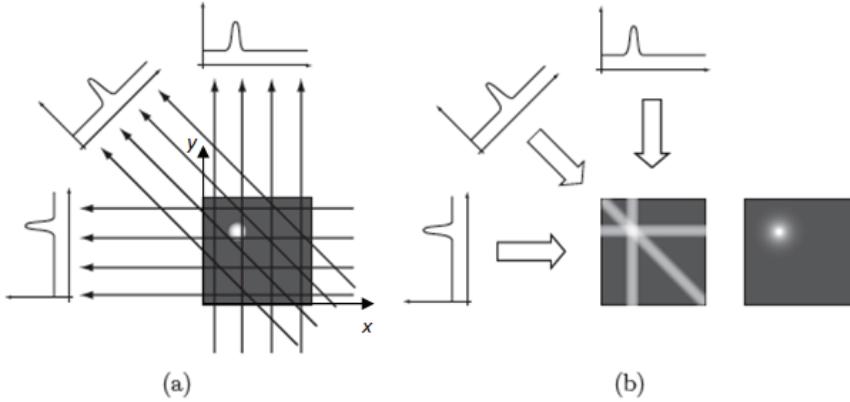


Figure 2.2 (a) Première étape de la TDM qui consiste en l'acquisition des projections à différents angles autour du tissu. (b) Seconde étape de la TDM qui consiste à la reconstruction par rétroposition filtrée. Image provenant de [Groller *et al.*, 2009].

la seconde étape consiste à une appliquer de l'algorithme de rétroposition filtrée (*filtered back-projection*) aux projections mesurées, algorithme qui est basé sur le théorème de la coupe centrale dans le plan de Fourier (*Fourier slice theorem*), afin de reconstituer l'image 2D [Brooks et Chiro, 1975]. Si on applique une transformée de Fourier par rapport à la variable t de l'équation de projection $P_\theta(t)$, cela donne la transformée de Fourier 2D de l'image évaluée sur la ligne de projection correspondant à la valeur t comme le montre l'équation suivante :

$$\begin{aligned} \int_{-\infty}^{\infty} P_\theta(t) e^{-jw t} dt &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) e^{-jw t} \delta(x \cos(\theta) + y \sin(\theta) - t) dt dx dy \\ &= \int_{-\infty}^{\infty} e^{-jw t} \delta(x \cos(\theta) + y \sin(\theta) - t) dt \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) dx dy \\ &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) e^{-jw(x \cos(\theta) + y \sin(\theta))} dx dy. \end{aligned} \quad (2.3)$$

Plus le nombre de projection est élevé plus il est possible d'évaluer ces transformés et donc obtenir le plus d'informations possibles de l'image [Kak et Slaney, 2001]. Au final, il suffirait en principe de faire de l'interpolation et de prendre la transformée de Fourier inverse afin d'obtenir l'image originale. Toutefois, il y a avantage à filtrer les projections tel que c'est fait dans l'algorithme de rétroposition filtrée, et dont nous n'entrerons pas dans les détails ici, pour compenser l'échantillonnage plus épars de l'espace de Fourier plus on s'éloigne du centre. Après avoir obtenu des images 2D de tranches successives du tissu biologique, il est possible de faire une reconstruction 3D complète du tissu en empilant ces tranches et en appliquant au besoin une interpolation linéaire entre les pixels correspondants des tranches. La figure 2.3 montre un simple exemple d'une série des tranches d'images 2D de la tête d'une personne avec lesquelles une image 3D a pu être générée.

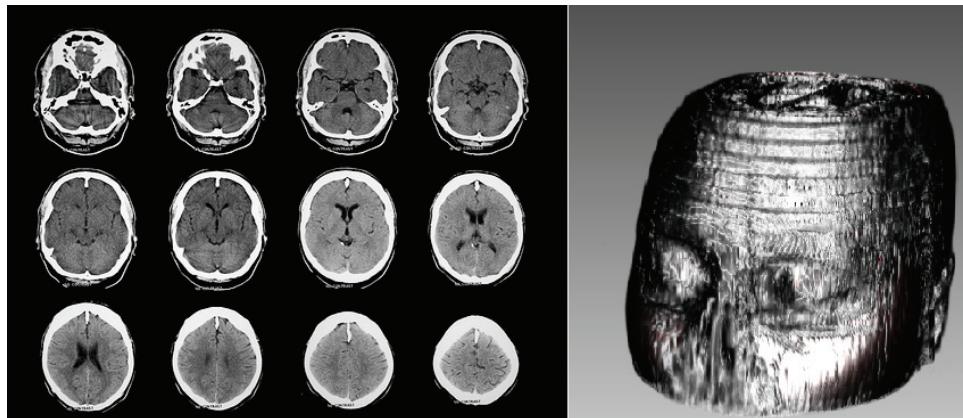


Figure 2.3 Série d'images TDM (gauche) ainsi que le volume obtenu à partir de ces séries (droite). Image provenant du logiciel réalisé dans le cadre de la présente maîtrise (Note : Dorénavant dans cet ouvrage, les images provenant de ce logiciel ne porteront plus cette mention afin d'alléger le texte ; il sera donc entendu qu'une image sans référence provient de ce logiciel.).

2.3 La tomographie d'émission par positrons (TEP)

La TEP, connue aussi sous le nom de *PET-Scan*, est une technique d'imagerie non invasive (en faisant abstraction de l'utilisation de traceurs radioactifs injectés chez le patient). La TEP est très utilisée en cardiologie afin d'analyser le flux sanguin dans les artères coronaires ou dans les cavités cardiaques, en neurologie pour évaluer les fonctions cérébrales et l'intégrité du cerveau et en diagnostic du cancer ainsi que pour déceler certaines anomalies inaccessibles par les autres techniques d'imagerie. La TEP est aussi utilisée dans le milieu

pharmaceutique pour l'analyse et le suivi de l'évolution d'un médicament dans le corps ou l'activité métabolique dans les tissus.

Le principe de détection de la TEP est basé sur l'émission des photons gamma d'une énergie de 511 keV suite à une annihilation entre les positrons libérés par les noyaux d'un traceur radioactif, dont on connaît les propriétés biologiques, et injecté dans le corps, et les électrons présents dans le milieu d'intérêt [Townsend, 2004]. Des détecteurs, chacun constitué d'un cristal scintillateur et d'un tube photomultiplicateur ou d'une photodiode à avalanche, permettent de déterminer la droite le long de laquelle s'est produite une annihilation (appelée ligne de réponse). Suite à la localisation du lieu de l'émission des photons sur l'ensemble des lignes de réponse, il est possible de déterminer de façon quantitative la concentration du traceur en chaque point du corps [Saha, 2010] [Wernick et Aarsvold, 2004]. C'est cette information quantitative que l'on représente sous la forme d'une image faisant apparaître en couleurs les zones de forte concentration du traceur comme le montre la figure 2.4, obtenue du laboratoire LabTEP du Centre d'imagerie moléculaire de Sherbrooke, dont le traceur est un mélange d'une substance radioactive avec du glucose(FDG) dans la vessie d'une petite souris.

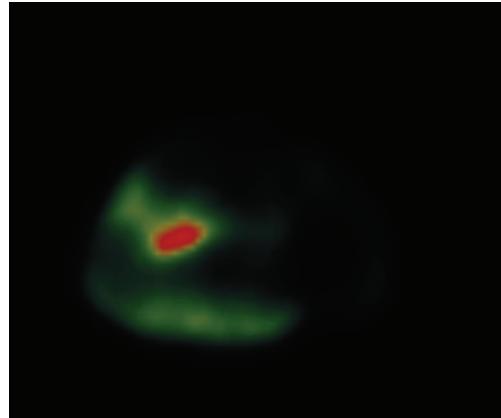


Figure 2.4 Image TEP de la vessie (surface rouge) d'une souris.

2.4 La tomographie optique diffuse (TOD)

La tomographie optique diffuse est une modalité d'imagerie non invasive permettant d'établir des cartes des propriétés optiques intrinsèques, comme les coefficients d'absorption μ_a et de diffusion μ_s , des tissus déterminant comment la lumière s'y propage dans le proche infrarouge (PIR, 650-1000nm)[Hervé *et al.*, 2007][Unlu *et al.*, 2008][Boffety, 2010]. Pour ce faire, on mesure d'abord la lumière diffuse lorsqu'on illumine un sujet avec une source lumineuse. Ensuite, on mesure tout autour la lumière qui ressort du sujet. Ces mesures acquises

sont éventuellement utilisées par un algorithme mathématique permettant d'extraire des cartes 3D des coefficients μ_a et μ_s qui nous informent sur la composition interne du tissu (structure, densité cellulaire, taille des noyaux, présence de fibres de collagène,...)[Boas *et al.*, 2001].

2.5 Principe du recalage d'image

Le recalage d'image se résume dans sa plus simple expression à trouver une transformation spatiale permettant de mettre en correspondance un ensemble de points connus d'une image source avec un autre ensemble de points d'une image cible. Il faut tout d'abord identifier sur quelles critères le recalage sera effectué. En général, Il y a quatre critères permettant de caractériser le recalage d'image [Noblet, 2006][Maintz et Viergever, 1998][Gonzalez et Woods, 2007] :

- **les attributs** : caractéristiques extraites des images qui permettent de guider le recalage, par exemple les niveaux de gris ou des marqueurs fiduciaux.
- **le critère de similarité** : il définit une certaine distance entre les attributs afin de quantifier la notion de ressemblance.
- **le modèle de déformation** : la géométrie de transformation (p.ex. transformation rigide ou non-rigide).
- **la stratégie d'optimisation** : méthode permettant d'identifier la meilleure transformation en fonction du critère de similarité et du modèle de transformation.

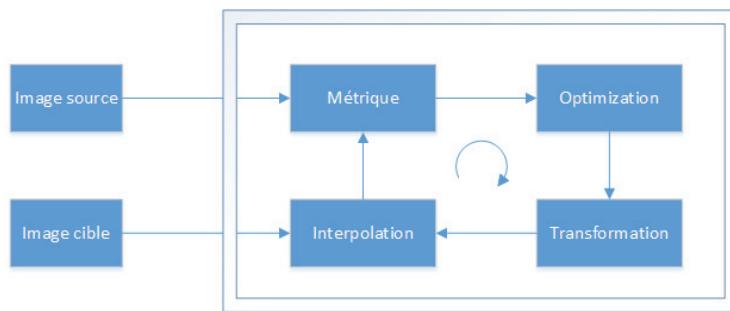


Figure 2.5 Les composantes de base du recalage d'image : une métrique, une optimisation, une transformation et une interpolation.

À partir de ces critères, il est possible de construire une structure, comme le montre la figure 2.5, permettant de représenter la problématique du recalage. La structure comprend : une métrique, permettant de quantifier la ressemblance entre l'image cible et l'image source, une optimisation qui consiste à déterminer la transformation optimale qui

minimise la métrique, un type de transformation géométrique qui permet d'associer les coordonnées d'une image source à celles de l'image cible et une interpolation car en pratique, la représentation utilisée des images est discrète et les points de l'image déformée ne correspondent pas nécessairement à l'image de référence [Ibanez *et al.*, 2005].

2.6 Transformation géométrique

La transformation géométrique est basée sur l'extraction dans les images source et cible de sous ensembles de primitives géométriques, points, courbes et surfaces, qui sont homologues, représentés par x et x' , et qu'il s'agit ensuite de mettre en correspondance en fonction des paramètres p du type de transformations T (comme p.ex. l'ensemble des transformations rigides).

$$x' = T(x|p). \quad (2.4)$$

2.6.1 Transformation rigide

Une transformation rigide consiste en l'estimation d'une translation et d'une rotation afin de repositionner un objet supposé rigide. Ce type de transformation conserve les distances, les angles et le parallélisme. Cette transformation possède 3 degrés de liberté pour des transformations 2D et 6 pour des transformations 3D comme le montre l'équation suivante :

$$x' = T(x|\theta, t_x, t_y). \quad (2.5)$$

où θ est l'angle de rotation, t_x et t_y représentent les composantes du vecteur de translation.

$$x' = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} x \quad (2.6)$$

La matrice de rotation est représentée par

$$R = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \quad (2.7)$$

où $R^{-1} = R^T$ (matrice transposée). Pour une image 2D, la transformation géométrique est appliquée dans le domaine spatial avant de passer dans le domaine discret (pixels) comme le montre l'équation 2.8. Les centres des pixels sont représentés par leurs positions

discrètes spécifiées par les indices i et j .

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \cdot \left(\begin{bmatrix} D_x & 0 \\ 0 & D_y \end{bmatrix} \cdot \begin{bmatrix} i \\ j \end{bmatrix} + \begin{bmatrix} O_x \\ O_y \end{bmatrix} \right) + \begin{bmatrix} t_x \\ t_y \end{bmatrix} \quad (2.8)$$

où O_x et O_y représente les coordonnées de l'origine de l'image et D_x et D_y sont les valeurs d'espacement en horizontal et en vertical des pixels.

Pour le cas 3D, la matrice ci-dessous est un exemple d'une transformation qui comporte une rotation d'un angle θ autour d'un axe z et une translation de (t_x, t_y, t_z) appliquées à des points (x_1, x_2, x_3) .

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 & T_x \\ -\sin(\theta) & \cos(\theta) & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{bmatrix} \quad (2.9)$$

Ce modèle est principalement utilisé dans le contexte de l'imagerie médicale pour recaler des images d'un même individu acquises à des instants différents ou pour différentes modalités d'imagerie [Noblet, 2006].

2.6.2 Transformation non-rigide

Une transformation non rigide implique des déformations lors de prises de mesures d'une séance d'imagerie à une autre. Représenter une transformation de déformation est très complexe surtout lorsque les sujets en question sont différents. En effet, les transformations non-rigides sont souvent irrégulières et il est impossible de trouver une transformation parfaite capable de corriger la déformation dans de tels cas. Les déformations sont généralement constituées de contractions, dilatations, rotations, translations, cisaillements et réflexions. Les catégories de transformation de déformation les plus connues sont la transformation de similarité et la transformation affine qui seront utilisées pour le projet.

Transformation de similarité

La transformation de similarité est utilisée dans le contexte de l'imagerie médicale pour le recalage d'images d'un même individu provenant de modalités différentes pour lesquelles la résolution n'est pas la même. Cette transformation inclue une rotation, une translation et un changement d'échelle. Elle préserve la linéarité et les angles entre les points mais elle ne peut garantir la préservation de la longueur entre les points après transformation

[Fitzpatrick et Sonka, 2009]. Un exemple de transformation de similarité pour un point X d'une image peut être écrit comme suit :

$$X' = RSX + t, \quad (2.10)$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} \quad (2.11)$$

où R est la matrice de rotation orthogonale, S est la matrice diagonale de changement d'échelle, et $t = [t_x, t_y]^T$ est le vecteur de translation le long de chaque axe pour une image 2D [Ibanez *et al.*, 2005].

Transformation affine

La transformation affine comporte les mêmes transformations que la transformation de similarité et elle peut avoir des transformations d'étirement et de cisaillement. Cependant, elle ne préserve pas les angles entre les images, elle préserve seulement les points constituant des droites parallèles [Fitzpatrick et Sonka, 2009]. L'équation générale de la transformation affine est la même que la transformation de similarité, mais la matrice R de l'équation précédente est remplacée par la matrice A qui n'est pas nécessairement orthogonale

$$X' = ASX + t \quad (2.12)$$

La transformation affine pour une image 3D peut être représentée en une seule matrice comme suit :

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & t_x \\ a_{21} & a_{22} & a_{23} & t_y \\ a_{31} & a_{32} & a_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \quad (2.13)$$

où a décrit les paramètres de changement d'échelle, de rotation et de cisaillement.

Les transformations rigide, affine et de similarité sont dites linéaires et elles peuvent être formulées en considérant les coordonnées homogènes, grâce à un produit matriciel comme suit [Noblet, 2006] :

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} a_{00} & a_{01} & a_{02} & b_0 \\ a_{10} & a_{11} & a_{12} & b_1 \\ a_{20} & a_{21} & a_{22} & b_2 \\ a_{30} & a_{31} & a_{32} & b_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}, \quad (2.14)$$

où (x, y, z) et (x', y', z') sont respectivement les coordonnées dans l'image source et les coordonnées dans l'image cible (transformées).

2.7 La métrique

La métrique définit une mesure de similarité entre deux images à recaler. Elle permet d'obtenir une certaine distance entre l'image source et l'image cible qui se caractérise par une valeur maximale ou minimale dès que la ressemblance entre les deux images est la plus forte. Dans cette section nous évoquerons la métrique quadratique, la corrélation croisée et l'information mutuelle.

2.7.1 Métrique quadratique

Pour une image source S , une image cible R et pour toute valeur d'intensité de pixel x , le but de la métrique quadratique est de mesurer (éventuellement minimiser) l'erreur $E(p)$ entre ces deux images

$$E(p) = \frac{1}{N} \sum_{i=1}^N [S(x_i) - R(T(x_i|p))]^2. \quad (2.15)$$

où N est le nombre total de pixels ou de voxels et T est la transformation spatiale en fonction des paramètres p comme la rotation ainsi que la translation pour une transformation rigide. En développant l'équation 2.15, on obtient

$$E(p) = \frac{1}{N} \sum_{i=1}^N (S^2(x_i) - 2S(x_i)R(T(x_i|p)) + R^2(T(x_i|p))). \quad (2.16)$$

Les termes $S^2(x_i)$ et $R^2(T(x_i|p))$ sont constants et donc indépendants de la minimisation de $E(p)$. Donc l'erreur peut s'crire comme suit

$$E(p) = \frac{-2}{N} \sum_{i=1}^N S(x_i)R(T(x_i|p)). \quad (2.17)$$

L'équation 2.17 prend la forme d'une fonction de corrélation générale entre l'image S et R . Cette corrélation varie en fonction de la surface commune des deux images. Plus la surface sera grande plus la corrélation sera importante. Dans le cas où les deux images se superposent sur une petite surface alors la corrélation sera faible et on cela peut engendrer

une erreur sur la recherche de la transformation minimisant l'équation 2.17. Afin d'éviter ce problème, il est préférable d'adopter la corrélation croisée normalisée.

2.7.2 Corrélation croisée normalisée

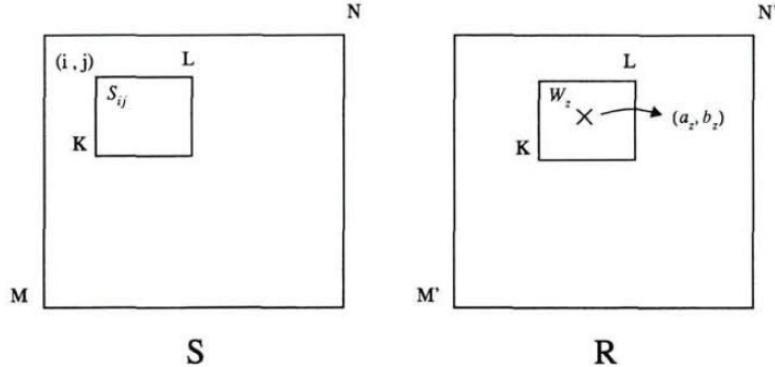


Figure 2.6 Méthode de corrélation croisée utilisant des sous-images. Image provenant de [Fonseca et Manjunath, 1996].

La corrélation croisée normalisée est très efficace pour comparer des images d'une même modalité ayant subi une transformation rigide. On sélectionne d'abord des sous images, n fenêtres représentées par W_z où $z=1, \dots, n$, et c'est à l'intérieur de celles-ci que la corrélation est calculée (il est aussi valable de considérer les images entières). Dans ce qui suit $K \times L$ représente la taille des sous-images, alors que $M \times N$ et $M' \times N'$ sont les tailles respectives des images S et R à l'intérieur desquelles on fait le calcul comme le montre la figure 2.6 et on itère le calcul de la corrélation sur toute l'image. Ce calcul peut être exprimé comme

$$C_{ij} = \frac{\sum_{l=0}^{K-1} \sum_{m=0}^{L-1} W_z(l, m) S_{ij}(l, m)}{\sqrt{\sum_{l=0}^{K-1} \sum_{m=0}^{L-1} W_z^2(l, m) \sum_{l=0}^{K-1} \sum_{m=0}^{L-1} S_{ij}^2(l, m)}}, \quad (2.18)$$

où

$$S_{ij}(l, m) = S(i + l, j + m). \quad (2.19)$$

La meilleure corrélation entre les deux sous images est obtenue lorsque la valeur C_{ij} , de l'équation 2.18, est à son maximum. Ainsi la valeur maximale calculée permet d'identifier la meilleure transformation reliant les deux sous-images [Fonseca et Manjunath, 1996]. En pratique, il est possible d'utiliser des méthodes numériques comme par exemple la méthode de Powell servant à calculer et optimiser la corrélation croisée normalisée [Barillot, 1999].

2.7.3 Maximisation de l'information mutuelle

La maximisation de l'information mutuelle, cette dernière servant de métrique, est la plus utilisée dans le traitement de signal, le transport de l'information et surtout pour un recalage d'images de différentes modalités. Cette méthode formule le recalage comme un problème statistique où la transformation géométrique est atteinte par maximisation de l'information mutuelle entre deux images. Ce critère est basé sur l'entropie des variables aléatoires. La forme la plus générique de l'entropie d'une variable aléatoire X , notée par $H(X)$, est donnée par l'espérance mathématique de la quantité d'information par événement donnée par [Ibanez *et al.*, 2005] [Viola, 1995] [Maes *et al.*, 1997] [Cover et Thomas, 1991].

$$H(X) = - \int_{-\infty}^{\infty} p(x) \log(p(x)) dx = -E_x[\log(p(x))], \quad (2.20)$$

où $p(x)$ est la densité de probabilité de la variable aléatoire X . Dans le cas du recalage d'image 2D, la densité de probabilité est exprimée par une fonction continue régissant la distribution des niveaux de gris des pixels de l'image.

L'objectif primaire de l'information mutuelle est de minimiser la différence d'intensité des pixels de l'image source par rapport à l'image cible. Pour ce faire, il faut procéder à une étude de distribution de probabilité conjointe sur les pixels des deux images. Cette distribution peut être représentée par un histogramme conjoint des valeurs des pixels. Par cet histogramme, il est possible ensuite de représenter l'information mutuelle des deux images en terme d'entropie conjointe, c'est-à-dire de combien la probabilité conjointe est-elle dispersée.

L'information mutuelle existante entre l'image source et l'image cible est donnée par

$$I(V, U) = H(V) + H(U) - H(V, U), \quad (2.21)$$

où V et U correspondent respectivement aux distributions d'intensité de l'image source et de l'image cible. $I(V, U)$ est la mesure de la réduction de l'entropie de U pour une valeur donnée de V . Si les variables aléatoires V et U sont indépendantes, alors

$$P(V, U) = P(V)P(U). \quad (2.22)$$

Dans ce cas, il n'y aura aucune information mutuelle car on aura

$$H(V, U) = H(V) + H(U) \quad (2.23)$$

et donc $I(U, V) = 0$. Dans le cas où V et U ont une certaine dépendance alors

$$H(V, U) < H(V) + H(U). \quad (2.24)$$

Dans ce cas, il y aura une certaine quantité d'information mutuelle. Dans le cas des variables aléatoires continues, on peut obtenir $H(V, U)$ par l'équation

$$H(V, U) = - \iint p_{V,U}(v, u) \log(p_{V,U}(v, u)) \, dV \, dU. \quad (2.25)$$

Dans le cas discret, il est possible d'exprimer l'information mutuelle entre deux variables indépendantes par [Ibanez *et al.*, 2005].

$$I(V, U) = \sum_{v \in V, u \in U} p(v, u) \log_2 \left(\frac{p(v, u)}{p(v)p(u)} \right). \quad (2.26)$$

Souvent, on ne peut pas avoir accès directement aux densités de probabilité. Par contre, il est possible d'estimer la densité de probabilité de l'image par la méthode de fenêtrage de Parzen. Cet estimé est construit par une superposition de fonctions noyau ("kernel functions" ; par exemple des fonctions gaussiennes), centrées sur les éléments d'un échantillonnage $A = x_1, x_2, \dots, x_{N_A}$ de taille N_A de la variable aléatoire X [Viola, 1995] [Viola *et al.*, 1996].

$$p(x) = \frac{1}{N_A} \sum_{x_i \in A} K_h(x - x_i). \quad (2.27)$$

Ici, K_h est la fonction noyau et l'indice h spécifie la largeur de cette fonction (h est aussi appelé paramètre de lissage). Plus ce paramètre est grand, plus l'approximation sera lisse ; bien qu'il puisse sembler avantageux d'utiliser de grandes valeurs de h , cela n'est pas toujours le cas, car il est possible que l'approximation omette des détails sur la distribution à reproduire). Dans la majorité des cas, on utilise des fonctions noyaux sous forme gaussiennes comme le montre la figure 2.7.

Il devient alors possible de calculer l'intégrale d'entropie par une simple moyenne d'un second échantillonnage B de taille N_B comme suit [Ibanez *et al.*, 2005] :

$$H(X) \approx \frac{-1}{N_B} \sum_{x_j \in B} \log(p(x_j)). \quad (2.28)$$

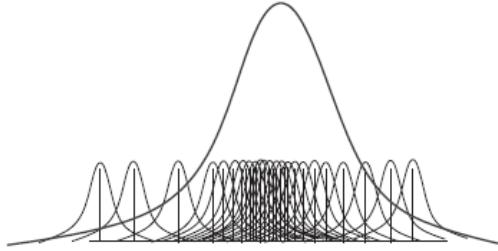


Figure 2.7 La méthode de Parzen utilise une superposition de fonctions noyau centrée aux échantillons des intensités de l'image. Graphique provenant de [Ibanez *et al.*, 2005].

Pour calculer l'entropie, il faut donc deux échantillonnages A et B . Le premier est utilisé pour estimer la densité de probabilité et le second pour estimer l'entropie. Donc on peut approximer l'entropie d'une variable aléatoire X par

$$H(X) \equiv \frac{-1}{N_B} \sum_{x \in B} \log \left(\frac{1}{N_A} \sum_{x_i \in A} K_h(x - x_i) \right). \quad (2.29)$$

Dans le but de maximiser l'information mutuelle, il faut ensuite calculer le gradient de cette dernière par rapport aux paramètres de la transformation que subit l'image à recaler $H(z) \rightarrow H(v(T(z)))$, où T est la transformation géométrique, et de trouver le maximum local permettant de maximiser l'information mutuelle.

L'information mutuelle normalisée

L'information mutuelle peut échouer pour certaines images ayant une grande présence de bruit autour de l'objet d'étude. Des études ont montré que l'information mutuelle normalisée fonctionne aussi bien et même mieux que l'information mutuelle simple [Studholme *et al.*, 1999]. Elle peut être calculée comme suit :

$$IN(V, U) = \frac{H(V) + H(U)}{H(V, U)}. \quad (2.30)$$

L'histogramme conjoint

Un moyen de développer son intuition sur comment fonctionne l'information mutuelle est de considérer l'histogramme conjoint de deux images. Un tel histogramme a pour abscisse les niveaux de gris de la première image et comme ordonnées les niveaux de gris de la seconde image. En supposant les deux images de mêmes dimensions, l'histogramme est formé en balayant simultanément les pixels des deux images et en ajoutant pour chaque

paire de pixels la valeur 1 à la cellule correspondant au couple d'intensités de ces pixels. Un exemple est illustré à la figure 2.8 où l'image au centre correspond à l'histogramme conjoint de l'image gauche et une copie identique de celle-ci. Dans ce cas, les images sont parfaitement alignées et l'histogramme est entièrement concentré sur la diagonale [Kostelec et Periaswamy, 2003] [Hill *et al.*, 1993]. L'image de droite correspond à l'histogramme conjoint de la même image et une version décalée de 5 pixels de celle-ci. On voit alors que l'histogramme est étalé comparativement au cas précédent, ce qui correspond à une plus grande entropie conjointe [Ibanez *et al.*, 2005] [Viola, 1995].



Figure 2.8 Histogramme pour différent alignement d'image. Image provenant de [Ibanez *et al.*, 2005].

Le recalage d'images tomographiques pourrait donc être formulé comme la recherche d'une transformation qui permet d'aligner à 45° l'histogramme conjoint, le plus parfaitement possible. L'expérience montre toutefois que la minimisation de l'entropie jointe a tendance à favoriser des transformations qui sont loin l'une de l'autre. L'information mutuelle, à laquelle contribue l'entropie conjointe permet de pallier à ce problème en faisant intervenir l'information individuelle de chacune des images.

2.8 L'optimisation

L'optimisation consiste à trouver le maximum de la mesure de similarité en fonction des degrés de liberté de la transformation géométrique recherchée. Prenons le cas d'une transformation rigide où seule la rotation et la translation sont impliquées. Des algorithmes d'optimisation sophistiqués comme Gauss-Newton permettent de minimiser la différence au carré ou l'algorithme d'optimisation de Levenberg Marquardt permet de minimiser la variance entre les intensités des pixels pour un recalage de même modalité [Ibanez *et al.*, 2005]. L'algorithme le plus utilisé en recalage d'image est la descente de gradient (*gradient descent*). Il suffit de dériver la métrique S (figure 2.5) en fonction des paramètres de transformation. Pour une transformation rigide, une optimisation de la métrique produit

le vecteur suivant

$$\left[\frac{\partial S}{\partial \theta}, \frac{\partial S}{\partial t_x}, \frac{\partial S}{\partial t_y}, \frac{\partial S}{\partial t_z} \right] \quad (2.31)$$

et ensuite, le gradient calcule la nouvelle valeur des paramètres par la longueur de pas λ

$$\theta + \frac{\partial S}{\partial \theta} \lambda, t_x + \frac{\partial S}{\partial t_x} \lambda, t_y + \frac{\partial S}{\partial t_y} \lambda, t_z + \frac{\partial S}{\partial t_z} \lambda. \quad (2.32)$$

2.9 L'interpolation

Dans le recalage d'image, l'interpolation a deux fonctions : interpoler l'intensité des points et détecter si ces points se trouvent dans le domaine de l'image à recaler. On y trouve trois types d'interpolation les plus utilisées : l'interpolation du plus proche voisin, l'interpolation linéaire et l'interpolation B-spline [Ibanez *et al.*, 2005].

2.9.1 L'interpolation du plus proche voisin

Cette interpolation est la plus facile à réaliser, elle ne nécessite pas de grand calcul. Elle consiste à utiliser l'intensité, valeur arrondie, du plus proche point du quadrillage de l'image. Le degré polynomial de cette interpolation est nul.

2.9.2 L'interpolation linéaire

Comme on peut le voir sur la figure 2.9, l'interpolation linéaire donne un résultat supérieur par rapport à l'interpolation du plus proche voisin. Dans le cas d'une d'image 2D, l'interpolation est estimée à partir de ses deux plus proches voisins dans chaque direction du plan xy . Elle est basée sur la technique quadratique par la méthode des trapèzes [Ibanez *et al.*, 2005].

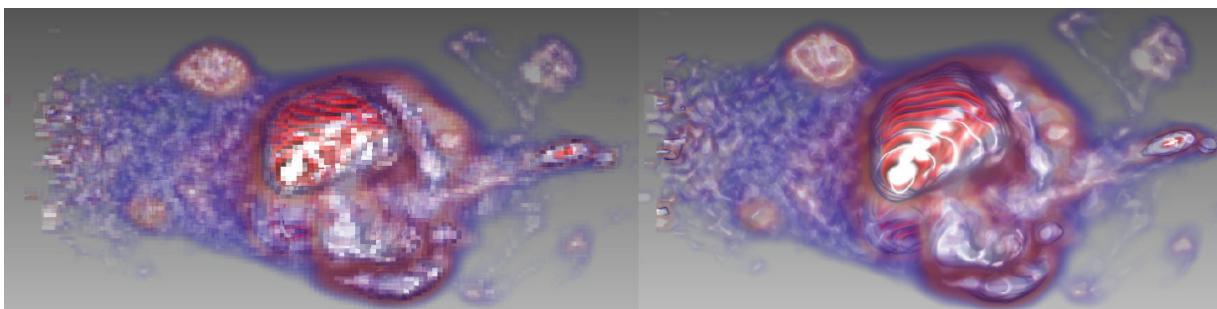


Figure 2.9 Différence entre interpolation du plus proche voisin à gauche et l'interpolation linéaire à droite.

2.9.3 L'interpolation B-spline

L'interpolation B-spline consiste en une combinaison linéaire de sous-fonctions ou coefficients, appelée spline, dans un intervalle bien défini. Avec ces coefficients, l'objectif est de résoudre un système linéaire avec les points de l'image suite à la transformation [Ibanez *et al.*, 2005].

2.10 Recalage par intensité des images

Le recalage par intensité consiste à trouver une transformation entre les deux images à recaler en fonction des valeurs des pixels ou voxels. L'image source et l'image cible sont représentées en niveau de gris et si elles sont de même modalité, la métrique de la corrélation croisée est la plus efficace pour calculer la différence en fonction de l'intensité des pixels ou voxels. Plus la similarité entre les images est importante plus la corrélation croisée sera petite. Dans un cas d'images de différentes modalités, il est difficile de trouver une relation linéaire entre les pixels des deux images. Il faut donc appliquer la métrique de l'information mutuelle vue dans la section précédente.

2.11 Recalage par traits caractéristiques

Comparé au recalage par intensité, le recalage par traits caractéristiques ne tient pas compte de l'intensité des pixels ou voxels, mais il est plutôt basé sur la recherche de un ou plusieurs critères de ressemblance qu'on peut facilement identifier sur les deux images à recaler.

Ces critères de ressemblance devraient être stables et fixes dans le temps et dans le processus de recalage. Ils peuvent être identifiables par des régions spécifiques, comme les organes ou les structures solides comme les os. Généralement, ces régions spécifiques sont détectées soit par des algorithmes de segmentation [Fitzpatrick et Sonka, 2009] ou soit de façon manuelle par l'utilisateur.

Les traits caractéristiques peuvent aussi être identifiés par des lignes, comme les contours de structure ou le périmètre d'une surface. Des détecteurs comme le filtre de Canny est souvent utilisé en traitement d'image pour la détection de contours [Ibanez *et al.*, 2005]. Ce détecteur utilise un filtre gaussien pour la réduction du bruit afin d'éliminer l'intensité des pixels aberrants et ensuite il calcule le gradient d'intensité de l'image. Le maximum

local du gradient indiquera une forte intensité donc une forte probabilité de présence d'un contour.

On peut aussi se baser sur des points saillants, des jonctions ou des centroïdes comme traits caractéristiques [Fitzpatrick et Sonka, 2009]. Ce genre de trait est très utilisé dans le domaine de la robotique pour la reconnaissance d'objets ou la navigation de robots mobiles, ou lorsqu'on veut faire de la stereo-vision pour la reconstruction 3D. Ils sont également très utiles lorsqu'on veut faire un recalage d'image médicale de même ou de différentes modalités avec points de référence (marqueurs fiduciaux) [Nahrendorf *et al.*, 2010]. Dans ce dernier cas, les points devraient être facilement visibles dans les deux types de modalité. L'extraction de ces points peut se faire de façon automatique par des algorithmes de segmentation et de détection de points saillants [Wang *et al.*, 1995] [Wang *et al.*, 1996] [Xiao, 2012] ou par l'utilisateur de façon manuelle comme le cas de ce projet.

2.12 Recalage par points de référence

La méthode de recalage par points de référence permet de trouver la transformation géométrique rigide à l'aide des positions spatiales d'un ensemble de points utilisés comme points de repère à l'intérieur ou à l'extérieur d'un sujet. Ces points sont obtenus soit par une identification manuelle par l'utilisateur ou soit par un processus de segmentation et de traitement d'image avancé [Nahrendorf *et al.*, 2010].

2.12.1 Marqueurs fiduciaux

Les marqueurs fiduciaux sont des objets détectables par plusieurs modalités d'imagerie et sont utilisés comme points de mesure permettant de calibrer les zones imagées par les deux images qu'on veut recaler [Wang *et al.*, 1996]. Ils peuvent prendre la forme d'un ensemble de points ou d'un ensemble de pièces mécaniques comme le cas de ce projet de recherche. Il est nécessaire de faire en sorte que les marqueurs soient visibles dans les deux modalités d'imagerie [Xiao, 2012] afin de faciliter l'identification par l'utilisateur. Le laboratoire TomOptUS a conçu des marqueurs fiduciaux munis de cavités et pastilles pouvant contenir des substances visibles par la TEP, la TDM et l'IRM. La figure 2.10 montre un exemple de la structure mécanique de 3 marqueurs fiduciaux fournis par l'imagerie TDM dont les cavités sont vides. Tandis que la figure 2.11 montre les mêmes marqueurs fiduciaux lors d'une séance d'imagerie TEP dont les cavités contiennent du fluorodésoxyglucose (FDG).

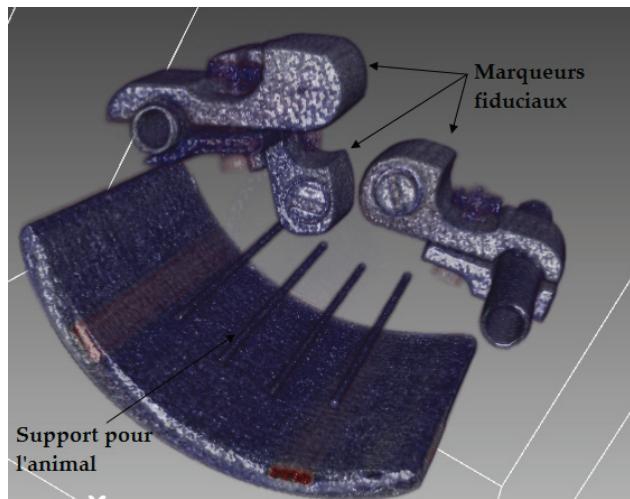


Figure 2.10 Image obtenue par la TDM d'un volume de 3 marqueurs fiduciaux autour d'un support pour petit animal.

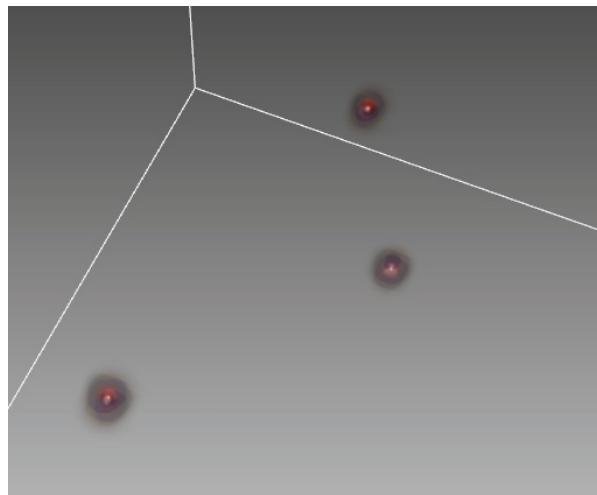


Figure 2.11 Image obtenue par la TEP des mêmes marqueurs fiduciaux présentés à la figure 2.10.

Le recalage par points de référence peut produire des erreurs aléatoires lors de la localisation des marqueurs correspondants entre les deux images. Ces erreurs peuvent être causées par le bruit dans l'image source et l'image cible ou par l'insertitude de la position exacte de chaque marqueur fiducial par rapport à la grille de voxels que constituent les deux images [Fitzpatrick et Sonka, 2009]. Cela peut fortement engendrer un mauvais recalage. Afin de corriger ce problème, il est préférable de raffiner le résultat de la transformation par un algorithme basé sur l'intensité des pixels ou voxels comme l'information mutuelle. La résolution de l'image peut jouer un rôle important en ce qui concerne la visibilité des marqueurs fiduciaux. Plus la résolution est grande plus il sera facile d'identifier les marqueurs fiduciaux dans l'image source à leurs homologues dans l'image cible.

2.13 Recalage de surface

Le recalage de surfaces est utilisé dans le domaine de la géophysique pour une étude de déformation structurale entre deux surfaces terrestres. Il peut être aussi utilisé dans le domaine de la recherche informatique comme le développement d'algorithmes de détection faciale ou de reconstruction d'environnements 3D. Ce type de recalage est également utilisé dans le domaine médical pour identifier deux surfaces correspondantes de même ou de différente modalité d'imagerie comme un recalage d'image profilométrique avec une modalité d'imagerie comme la TEP, l'IRM ou la TDM.

Le principal objectif du recalage de surface est de minimiser une distance particulière entre deux surfaces ayant un même ou différent nombre de points. Plusieurs algorithmes ont été développés et donnent de bons résultats. Parmi ces algorithmes, l'algorithme ICP (*Iterative Closest Point*) est le plus utilisé [Umeyama, 1991][Arun *et al.*, 1987][Besl et McKay, 1992][Yao *et al.*, 2011][Johnson et Hebert, 1997]. C'est un processus itératif qui minimise le carré des distances des points correspondants entre deux surfaces. Par cette minimisation, l'algorithme permet de trouver une transformation rigide, de similarité ou affine appliquées aux points de la surface source afin de mieux correspondre à ceux de la surface cible.

L'algorithme de recalage de surface par ICP se résume en cinq grandes étapes comme suite

1. La surface cible est extraite sous forme de points et on en obtient un maillage triangulé.
2. On représente la surface source par un ensemble de points.
3. Pour chaque point (vertex) de la surface source, on calcule le point le plus proche sur la triangulation de la surface cible.
4. Une transformation rigide, de similarité ou affine est appliquée à l'image source et on calcule l'erreur résiduelle entre points correspondants.
5. On itère à partir de la nouvelle position des points de l'image source jusqu'à ce que l'erreur résiduelle tombe sous un seuil de référence.

2.14 Conclusion

Le présent chapitre a présenté les concepts et méthodes liés au recalage d'images. On a parlé des différentes modalités d'imagerie qui génèrent des images d'intérêt pour les

présents travaux (TDM, TEP, IRM, TOD). On a discuté du principe du recalage d'images et de ses différentes étapes impliquant des transformations géométriques, des mesures de similitude entre les images, l'optimisation de telles mesures par rapport aux paramètres du type de transformations géométriques considérées et de l'interpolation. Au chapitre qui suit, on discutera de la conception d'un logiciel de recalage faisant appel aux différents outils du recalage d'images.

CHAPITRE 3

CONCEPTION INFORMATIQUE

Les entités informatiques traités dans le projet sont des objets : points dans l'espace 3D, maillages, champs scalaires, pixels, voxels etc. Le choix d'un langage orienté objet nous a paru judicieux. Le programme informatique à été donc réalisé en langage C++ sous Visual Studio 2010 en intégrant Qt Designer pour l'interface usager et pour sa capacité à être intégré par les outils de traitement d'image, de segmentation et de visualisation open source, Visualization ToolKit (VTK) et Insight ToolKit (ITK), de la compagnie *Kitware.inc.*

Ce projet a pour but de donner la possibilité à l'utilisateur d'effectuer deux types de recalages : le recalage de surface d'images profilométriques avec des images tomographiques et le recalage d'images tomographiques 3D à l'aide des marqueurs fiduciaux.

3.1 Installation informatique

Le projet de recherche a été réalisé en utilisant les librairies open source VTK5.10.1 et ITK4.2.2 dont on peut télécharger les fichiers sources du site web de *Kitware* [Kitware, s. d.a] [Kitware, s. d.b]. Pour intégrer VTK et ITK avec une interface graphique Qt, il faut les éléments suivants :

- Qt4.8.5.
- CMake-2.8.10.2-win32-x86.exe.
- Un fichier CMakeList.txt.
- Un compilateur C++ (Visual Studio 2010 à été choisi pour le projet).

Certaines images tomographiques 3D obtenues du laboratoire sont de grande taille comme le cas des images TDM dont les tailles sont de 512x512x512. Une compilation à 32 bits (x86) dont la mémoire est limitée à 2GB et généralement fragmentée, ne permet pas en pratique l'utilisation d'un espace de plus de 250MB, ce qui est limitant. Il est préférable de compiler tout les modules du projet à 64 bits (x64) afin d'exploiter le plus de mémoire possible jusqu'à 4GB. Pour cela, il faut tout d'abord télécharger la version 4.8.5 de Qt

pour Windows disponible sur le site web de Qt [Qt, s. d.] et suivre les étapes fournies sur le lien Qt pour la compilation à 64 bits sous Visual Studio 2010.

3.1.1 CMake

Après avoir compilé et intégré Qt Designer avec Visual Studio 2010, il faut télécharger la dernière version exécutable de CMake disponible sur le site de *Kitware* [Kitware, s. d.c]. CMake est une application séparée et de plus haut niveau que l'outil make. C'est un système multiplateforme qui permet de générer un environnement de construction afin de compiler le code source, créer des bibliothèques et construire des exécutables en fonction du compilateur choisi par l'utilisateur. La figure 3.1 montre l'interface principale de CMake.

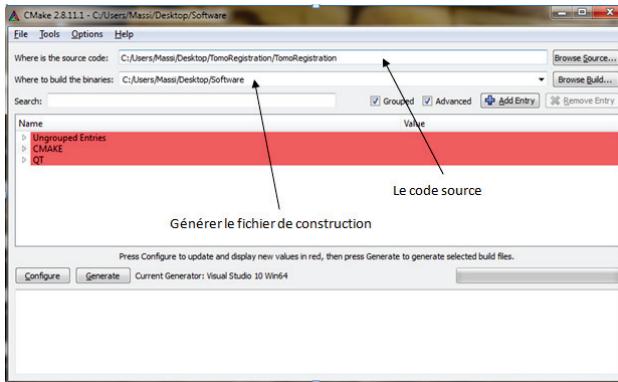


Figure 3.1 Fenêtre exécutable CMake. Image provenant de [Kitware, s. d.c].

Cette interface exige qu'on fournisse deux informations : là où le code source est localisé et là où le code binaire doit être produit. Sous Windows, la configuration de VTK et d'ITK par CMake permet de générer une multitude de fichiers de projet de Visual Studio. Afin que ces fichiers soient configurables, chaque fichier source doit contenir un fichier de configuration appelé CMakeLists.txt qui permet de contrôler le processus de construction logiciel afin d'obtenir des fichiers de construction standard. La figure 3.2 montre un exemple simple de la programmation du fichier CMakeList.txt d'un projet appelé *myProject* qui utilise les bibliothèques de Qt et VTK.

3.1.2 Qt Designer

Qt Designer est un environnement de développement multiplateforme pour concevoir des interfaces graphiques. Afin d'intégrer VTK et ITK avec Qt, il faut activer les cases *vtk-use-Qt* et *vtk-use-guisupport* lors de leur configuration avec CMake comme le montre la figure 3.3. Cela créera des fichiers *QVTKWidgetPlugin.dll* et *QVTKWidgetPlugin.lib* qui

PROJECT(myProject)

```

FIND_PACKAGE(Qt4)
INCLUDE(${QT_USE_FILE})
FIND_PACKAGE(VTK)
IF( VTK_FOUND )
    INCLUDE( ${USE_VTK_FILE} )
ENDIF( VTK_FOUND )
INCLUDE_DIRECTORIES(${myProject_SOURCE_DIR})
ADD_EXECUTABLE(myProject myProject.cxx)
TARGET_LINK_LIBRARIES(myProject ${VTK_LIBRARIES} QVTK)

```

Figure 3.2 Exemple d'un fichier CMakeList.txt. Code provenant du logiciel réalisé.

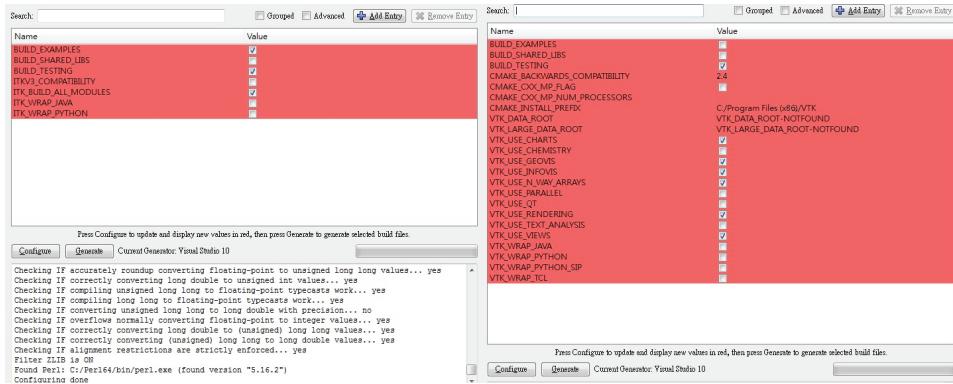


Figure 3.3 Configuration CMake, VTK (droite) et ITK (gauche). Image provenant de [Kitware, s. d.c].

doivent être déplacés dans le répertoire design de Qt. À l'ouverture de Qt Designer, le widget de VTK, *QVTKWidget*, apparaîtra comme le montre la figure 3.4.

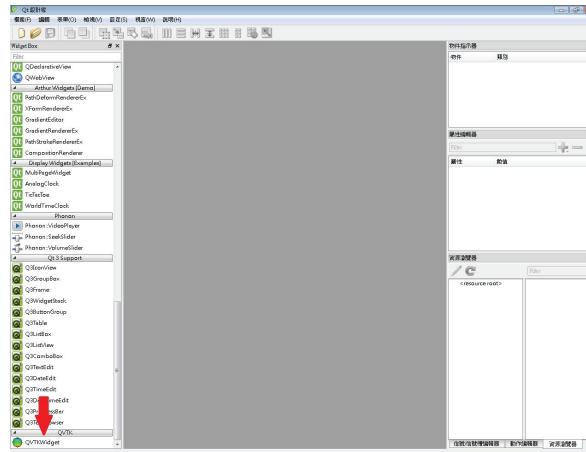


Figure 3.4 *QVTKWidget* dans l'environement de Qt Designer. Image provenant de [Qt, s. d.].

3.1.3 Conception de l'interface

L'interface est conçue pour qu'elle soit cohérente et simple à utiliser. La figure 3.5 présente l'interface principale (qui a été conçue dans le présent projet) lors de l'ouverture du fichier exécutable. Elle comprend trois fenêtres *QVTKwidgets* qui permettent d'afficher l'image source, l'image cible ainsi que le résultat du recalage. Pour le choix du type de recalage, ce dernier se fait sur une liste déroulante dans la barre d'état.



Figure 3.5 Interface principale du programme.

3.2 Librairie VTK

La librairie de VTK est d'une grande polyvalence. Elle est utilisée dans plusieurs domaines notamment en géophysique pour visualiser la nature et les effets séismiques et dans le domaine de la mécanique, pour l'étude d'élément finis ou maillage de volume et extraction de surface. Elle est aussi utilisée dans le domaine médical pour la visualisation et le traitement d'images tomographiques. Le choix de l'utilisation de la librairie VTK est non seulement pour la visualisation des données 2D/3D mais aussi en grande partie par sa facilité à être interfacée avec des langages de programmation tels que Tcl-Tk, Java ou Python et que le noyau de l'architecture de VTK est composé principalement de classes C++ compilées. Un des grands avantages du choix de VTK est qu'elle est utilisée par une large communauté scientifique. Ses caractéristiques sont :

- Librairie indépendante de la plateforme d'utilisation.
- Se base sur le standard de OpenGL.
- Intégrable avec Qt Designer.

- Possibilité de fournir des objets 3D (*widgets*).
- Possibilité d'interaction 3D avec les données.
- Possibilité d'afficher des données 2D ou 3D dans la même scène.
- Des exemples, des formations ainsi qu'un forum sont disponibles aux développeurs et aux utilisateurs [Kitware, s. d.a].

3.2.1 Architecture de VTK

L'architecture de VTK est basée en une structure modulaire composée de filtres en série. Ces filtres sont composés de plusieurs classes C++ et suivent une structure de pipeline comme on peut voir un exemple sur la figure 3.6 [Schroeder *et al.*, 1996].



Figure 3.6 Structure pipeline composée de séries de filtres pour la lecture d'image 2D et 3D. Schéma adapté de la documentation de VTK.

Cette architecture se divise en deux parties comme le montre la figure 3.7. La première partie, appelée *Data processing pipeline* en anglais, comprend la lecture, l'interprétation et le traitement des données qu'on utilisera (*vtkPolydata*, *vtkImageData*, *vtkSphere*,...), ainsi que des algorithmes permettant la manipulation des données et le traitement d'image (*vtkExtractSurface*, *vtkMarchingCubes*, *vtkTransform*,...).

La seconde partie comprend la structure de visualisation, appelée *Rendering pipeline* en anglais. Cette partie comprend la classe *ViewProp* utilisée par une sous-classe *vtkProp* pour afficher un type d'acteur (*actor*) ou d'objet comme (*vtkActor*, *vtkActor2D*, *vtkImageActor*, *vtkAnnotation*, *vtkVolume*,...). Cette classe donne aussi la possibilité à l'utilisateur de modifier l'orientation ou appliquer un changement d'échelle sur l'objet à visualiser. Afin d'afficher l'acteur, il faut un *Mapper*, un mappeur, qui permet de lier les données des sorties des filtres de la première partie afin de procéder au processus de visualisation. Par exemple si les données sont de type géométrique composées de vertex et d'arrêt, comme le cas d'une image profilométrique, le *Mapper vtkPolyDataMapper* sera donc utilisé pour afficher ce type d'image.

Le *Renderer* permet de spécifier comment afficher l'objet et permet aussi d'afficher plusieurs acteurs de différentes natures comme un champ scalaire, des maillages ou des points. Le *Renderer*, via *L'interactor*, permet à l'utilisateur d'avoir le contrôle de la position de

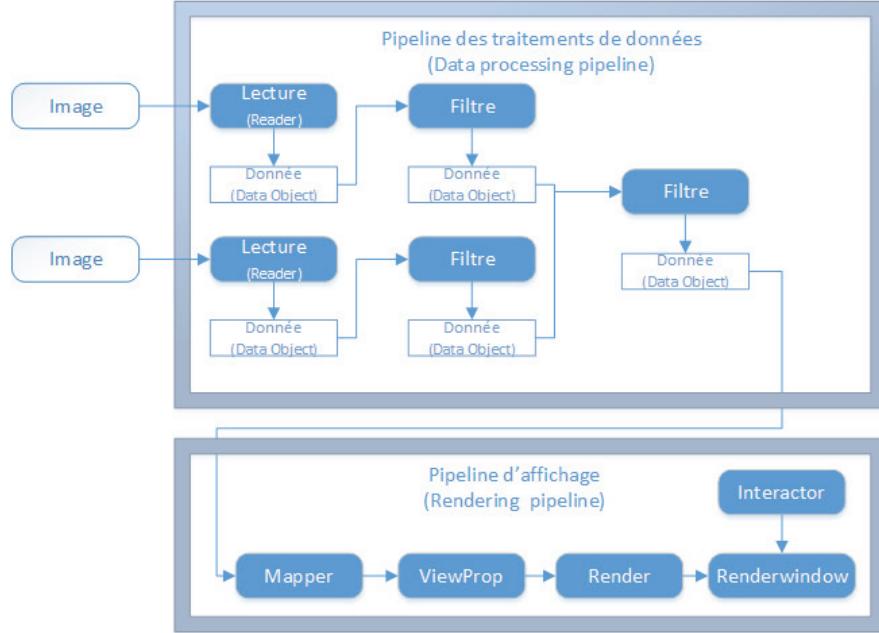


Figure 3.7 Architecture de VTK de la source jusqu'à l'affichage. Schéma développé pour le logiciel réalisé dans les présents travaux (note : dans ce qui suit, aucune mention ne sera dorénavant faite pour des schémas réalisés pour ce logiciel).

la caméra, ou point focal de la scène, par rapport à l'acteur en mode *Trackball* comme le montre la figure 3.8. Il permet aussi de fixer la position initiale de l'acteur et de fixer les propriétés intrinsèques de la scène.

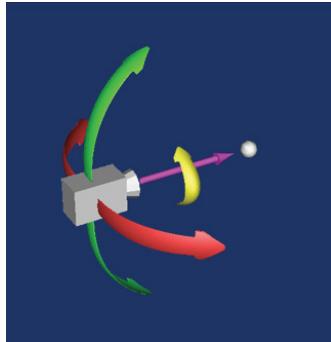


Figure 3.8 Contrôle (*Trackball*) de la scène dans un environnement 3D. Image provenant de [Kitware, s. d.a]. Cette figure est utile pour l'utilisateur du logiciel réalisé dans le cadre de la présente maîtrise.

Le *RenderWindow* permet de lier le système d'exploitation avec le système d'affichage de VTK. L'*interactor* contrôle les événements issus de la souris, du clavier et des signaux externes ou internes. Il permet aussi de donner à l'utilisateur la possibilité d'interagir à tout moment avec la caméra par des mouvements de rotation et de zoom avant et arrière.

L'*interactor* donne aussi la possibilité à l'utilisateur de créer sa propre interaction avec l'objet en spécifiant les événements qu'il veut avoir. Il permet aussi l'ajout d'objets par le biais de la classe *widget*. Celle-ci est très importante au point de vue de la programmation informatique lors du recalage par marqueurs fiduciaux. La figure 3.9 présente la structure

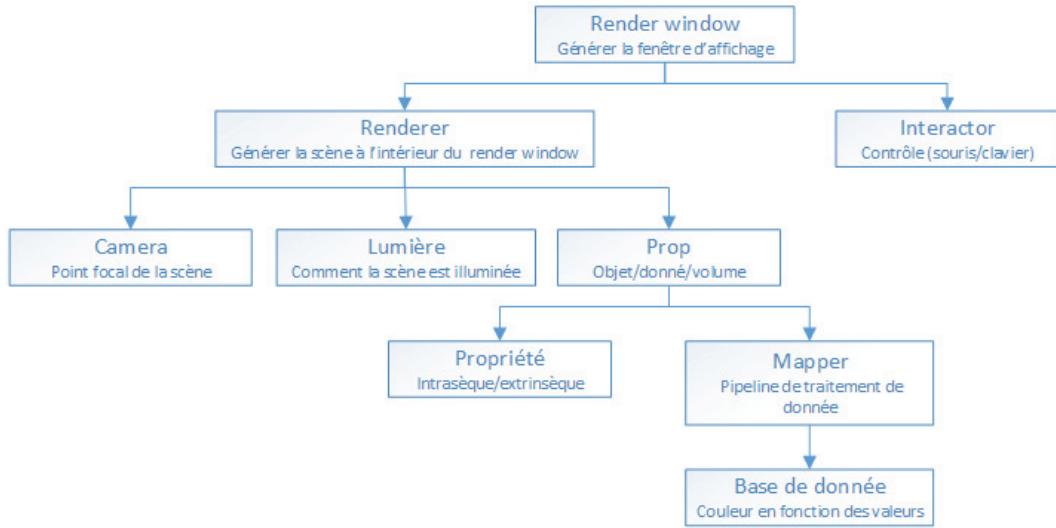


Figure 3.9 Architecture programmée pour l'affichage d'image.

générale utilisée pour l'affichage de toutes les modalités d'imagerie obtenues du laboratoire. La figure 3.10 est un exemple de construction d'un volume à partir de 136 séries d'images 2D de TEP auxquelles nous avons appliqué un filtre de lissage pour réduire les irrégularités à la surface volumique. La fenêtre *RenderWindow* comprend les acteurs suivants : une annotation, une représentation axiale du plan *xyz* ainsi que le volume d'image.

L'annexe A présente un simple exemple de programmation de l'affichage d'un maillage en utilisant la structure de pipeline.

3.3 Librairie ITK

La librairie ITK est utilisée en analyse d'image pour la segmentation et le recalage, mais elle ne prend pas en charge les aspects de visualisation et d'interface graphique d'où l'importance de l'intégrer avec la librairie de VTK. Sans entrer dans les détails, la librairie ITK suit une programmation générique (template en C++) [Kitware, s. d.b]. Pour le présent projet, la librairie ITK est utilisée que pour le recalage basé sur l'intensité des pixels c'est-à-dire l'application d'algorithmes de raffinement suite au recalage à l'aide des marqueurs fiduciaux. La figure 3.11 présente une structure pipeline d'intégration de la librairie d'ITK

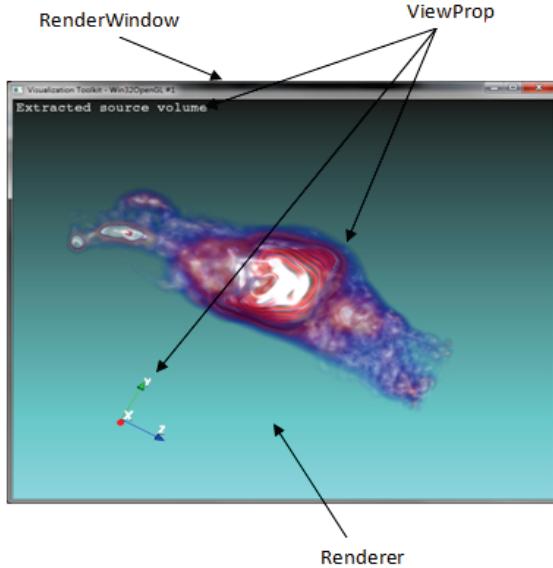


Figure 3.10 Exemple d'affichage d'un volume TEP d'une souris dans le logiciel.

avec celle de VTK programmée pour un recalage 3D de différente modalités d'imagerie impliquant la métrique de l'information mutuelle.

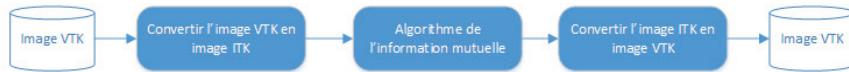


Figure 3.11 Structure pipeline d'intégration de VTK et ITK utilisée pour le recalage par information mutuelle.

3.4 Recalage à l'aide des marqueurs fiduciaux

Le recalage avec marqueurs fiduciaux se fait dans un environnement 3D. Il faut donc tout d'abord générer un volume à partir des images fournies par le laboratoire. La figure 3.12 présente la structure pipeline implémentée dans le programme afin d'afficher et modifier les propriétés intrinsèques d'un volume. Par exemple, les images TEP sont généralement dans un dossier contenant une série de fichiers de format Dicom. La lecture de ces fichiers est effectuée par le filtre *vtkDICOMImageReader*. En ce qui concerne les images IRM et TDM, elles sont obtenues en fichier de format raw qui est lu par le filtre *vtkImageReader*. La sortie de ces filtres permet la connexion au filtre *vtkGPUVolumeRayCastMapper*. Ce filtre utilise le raycasting, une technique de calcul d'images de synthèse 3D, pour rendre un volume. Cette technique consiste à envoyer un rayon à partir d'un point et ensuite une couleur est attribuée à chaque voxel du volume en fonction de la contribution de l'émission et l'absorption de la lumière le long du rayon traversant le volume [Groller *et al.*, 2009].

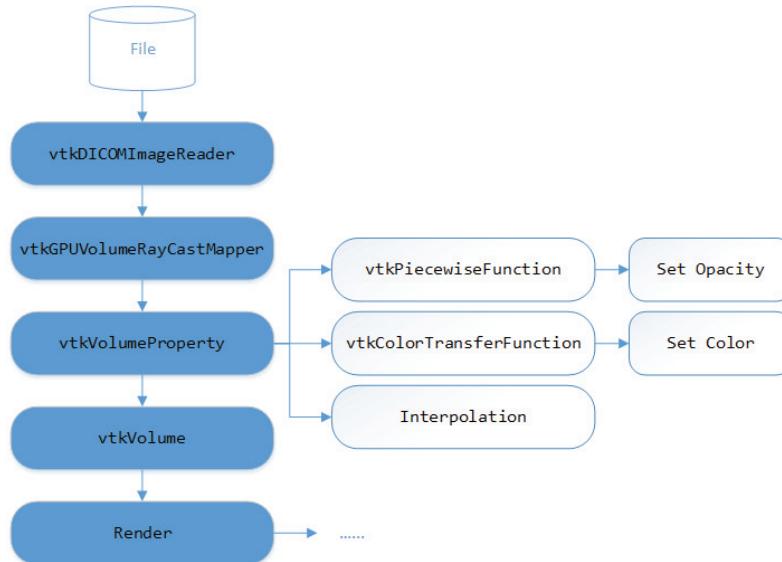


Figure 3.12 Structure pipeline VTK pour afficher un volume d'image TEP.

Les informations portées par les voxels indiquent des niveaux de densité de matière liés à la densité réelle exprimée en unités Hounsfield comme indiqué dans le tableau 3.1. *vtkVolumeProperty* permet de contrôler l'apparence de ces densités en utilisant la classe *vtkPiecewiseFunction* qui se base sur une fonction de transfert du gradient d'opacité, plus le voxel est dense plus il aura une grande opacité. La classe *vtkColorTransferFunction* permet de convertir les voxels en niveau de gris ou en RGB pour l'affichage du volume.

Tableau 3.1 Tableau des unités de Hounsfield.

Élément	Unités de Hounsfield
Os	1000
Foie	40-60
Matière blanche	46
Matière grise	43
Sang	40
Muscle	10-40
Rein	30
Liquide céphalorachidien	15
Eau	0
Gras	-50 - -100
L'air	-1000

Il faut spécifier les propriétés d'opacité ainsi que la couleur des voxels à la classe *vtkVolume* qui permet d'accéder au pipeline d'affichage. La figure 3.10 est un résultat d'un volume issu d'une série d'image de type *vtkImageData* dont le volume a été rendu avec la classe

vtkGPUVolumeRayCastMapper. La figure 3.13 présente le volume des marqueurs fiduciaux obtenus par les modalités d'imagerie TEP, TDM et IRM en suivant la structure d'affichage de volume.

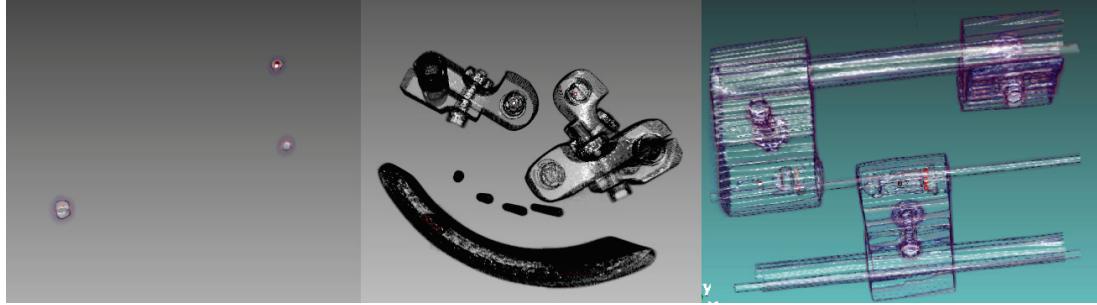


Figure 3.13 Marqueurs fiduciaux vue par l'image TEP à gauche, l'image TDM au centre et l'image IRM à droite. Ces images proviennent d'une même fenêtre (source) du logiciel réalisé.

Pour pouvoir localiser les marqueurs fiduciaux, il faut établir un moyen d'interaction permettant à l'utilisateur de les isoler de façon manuelle. Pour cela, il a été décidé d'établir une structure de programmation de *l'interactor* permettant d'exploiter la classe *widget* de VTK qui offre la possibilité d'utiliser des objets manipulables qui permettent de modifier les dimensions d'un objet. Parmi les objets les plus connus de la classe *widget*, on y trouve le *BoxWidget* montré à la droite de la figure 3.14. Le *BoxWidget* est un polygone d'arrêt qu'on peut manipuler et modifier ses dimensions par la présence de "poignées" (*handles*) sur chaque surface du polygone.

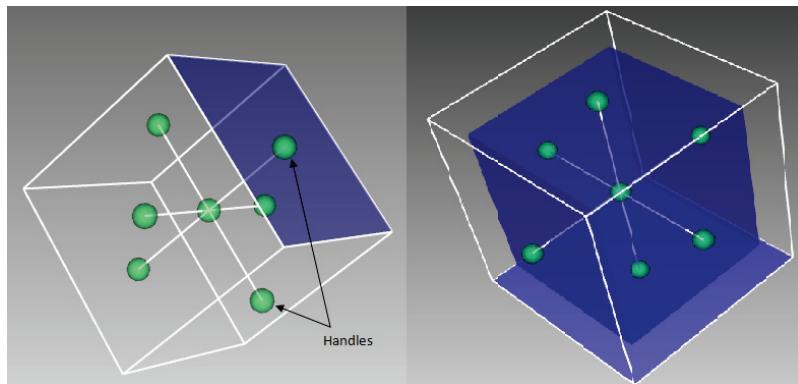


Figure 3.14 Structure du *BoxWidget*. Ces images proviennent d'une même fenêtre (source) du logiciel réalisé.

Nous avons décidé d'implanter un cube de la classe *vtkCubeSource*, d'une faible opacité, à l'intérieur du *BoxWidget* afin qu'il puisse agir comme un acteur manipulable par le *BoxWidget* comme montré à la gauche de la figure 3.14. Cela permet à l'utilisateur de

visualiser le volume de couverture des marqueurs fiduciaux et ainsi pouvoir évaluer la position spatiale afin de procéder au recalage. La figure 3.15 présente un diagramme UML utilisé pour programmer l'environnement d'interaction avec le *BoxWidget*.

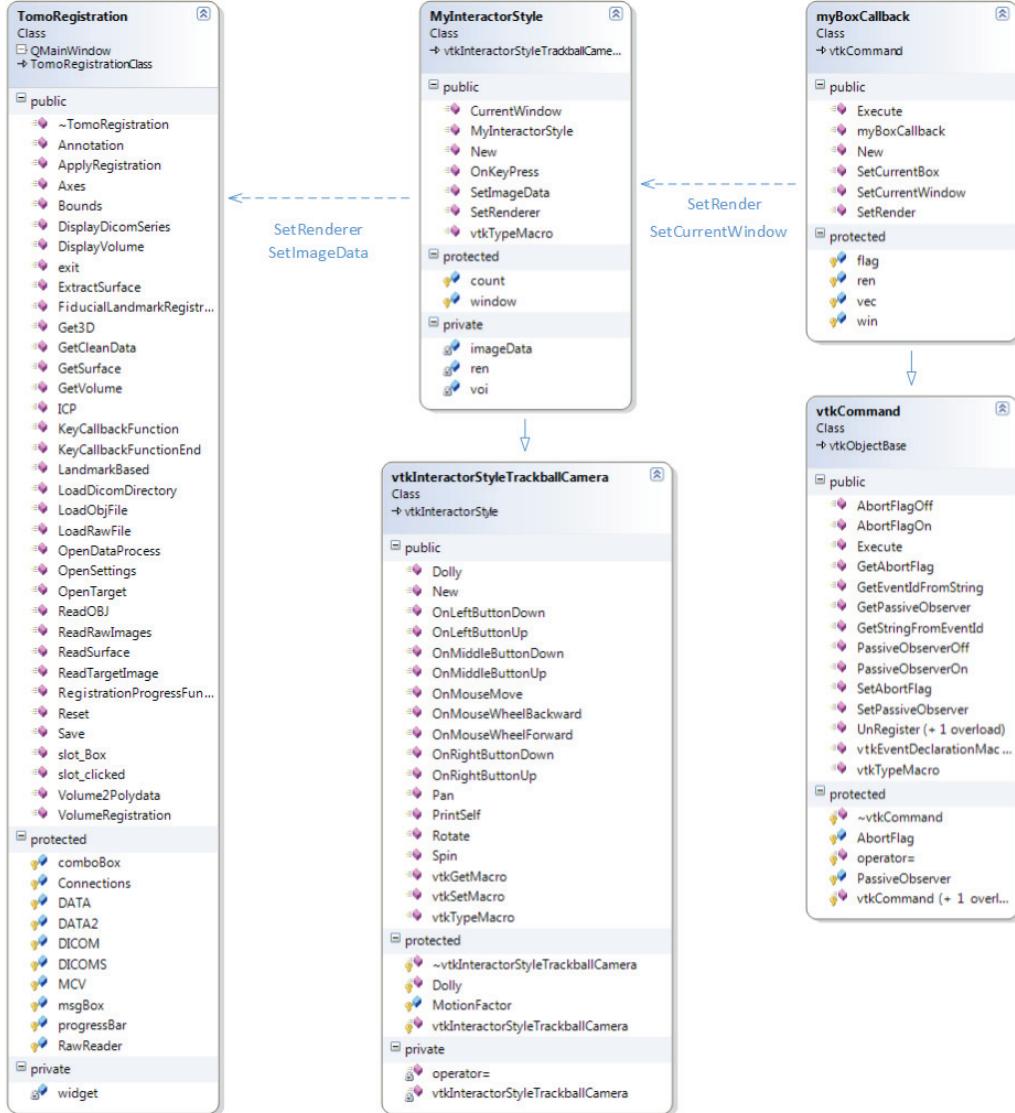


Figure 3.15 Diagramme UML servant à la création du *BoxWidget* via la classe *myBoxCallback*. La programmation du *BoxWidget* a été réalisée dans la fonction *Execute*.

La classe *TomoRegistration* de la figure 3.15 est la classe principale du projet dont dépendent la majorité des autres classes. Elle comprend toutes les fonctions nécessaires permettant à l'utilisateur d'interagir avec l'interface principale. La classe *MyInteractorStyle* est programmée de façon à spécifier une interaction à l'aide de la fonction virtuelle *OnKeyPress* qui permet de détecter un signal du clavier. Cette classe hérite de la classe *vtkInteractorStyleTrackBallCamera* qui permet le contrôle de la scène en mode *TrackBall*.

comme le montre la figure 3.8. La classe *myBoxCallback* permet à l'utilisateur de déplacer ou de modifier les dimensions du *BoxWidget*. Cette classe est dépendante de la classe *MyInteractorStyle* qui permet de gérer l'interaction et la détection du signal menant à la création du *BoxWidget*. La classe *myBoxCallback* hérite de la classe de *vtkCommand* qui par la fonction *Execute* permet de gérer le mouvement de déplacement que l'utilisateur exécute avec le *BoxWidget* afin de cibler les volumes des marqueurs fiduciaux. La structure de pipeline du fonctionnement du *BoxWidget* est présentée à la figure 3.16 dans lequel on génère le *BoxWidget* via l'interaction par un simple signal du clavier.

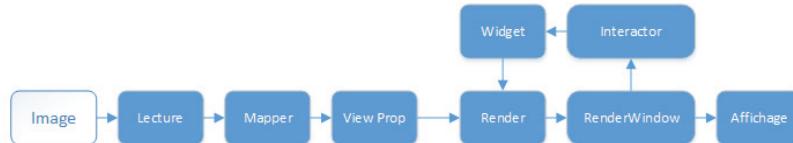


Figure 3.16 Structure d'implémentation d'un *widget* dans VTK.

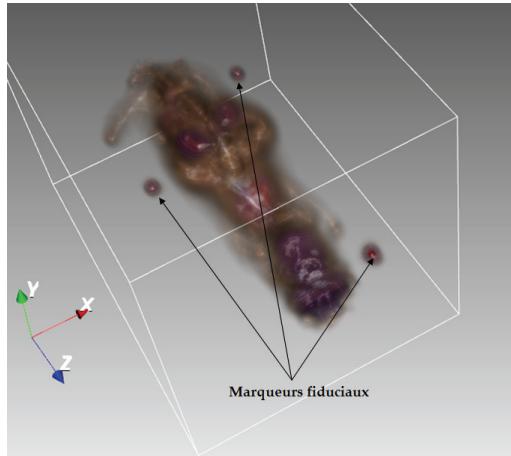


Figure 3.17 Image TEP d'une souris avec 3 marqueurs fiduciaux.

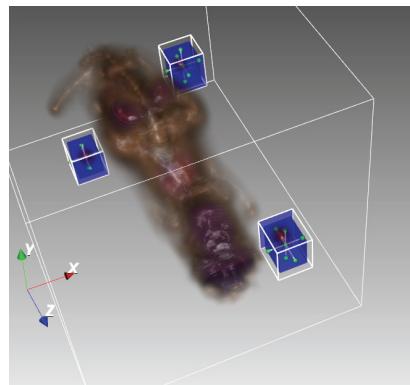


Figure 3.18 Positionnement des *BoxWidgets* sur les 3 marqueurs fiduciaux.

La figure 3.17 présente un volume d'une image PET d'une souris entourée de trois marqueurs fiduciaux et la figure 3.18 présente la même souris dont les marqueurs fiduciaux sont entourés de *BoxWidgets*. La figure 3.19 présente un diagramme UML utilisée pour programmer un moyen d'extraire un volume d'intérêt. La structure mécanique des marqueurs fiduciaux est très visible en IRM et en TDM comme le montre à la figure 3.20 d'une image TDM. De la même manière que l'image TEP, il est possible d'entourer sans difficulté de visualisation ces marqueurs fiduciaux par les *BoxWidgets* comme le montre la figure 3.21.

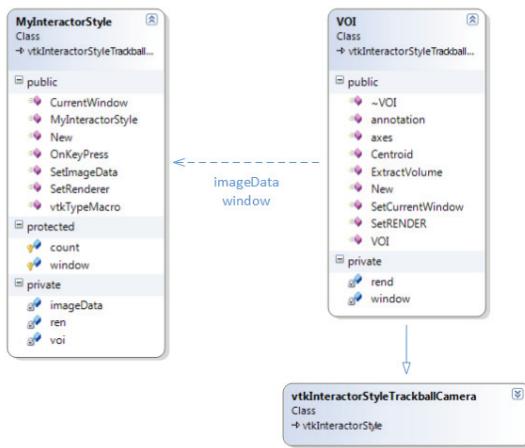


Figure 3.19 Diagramme UML pour la programmation de l'extraction du volume d'intérêt.

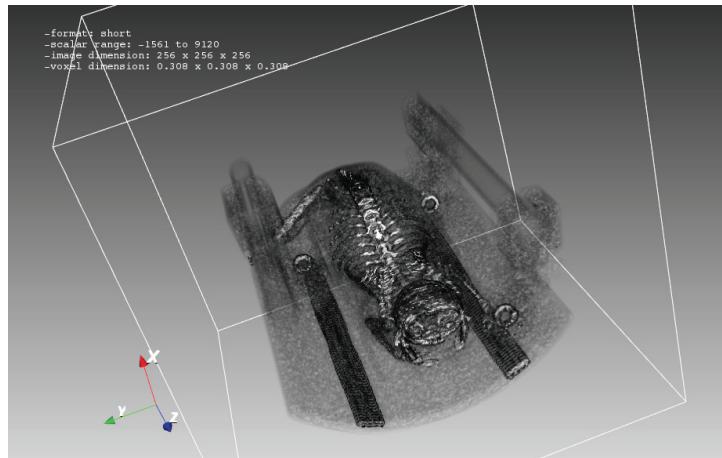


Figure 3.20 Image TDM d'une souris avec 3 marqueurs fiduciaux.

3.4.1 Calcul des centroïdes

Le calcul des centroïdes a été utilisé pour l'identification des positions des marqueurs fiduciaux pour la TEP, la TDM et l'IRM. Cette méthode localise la moyenne des valeurs

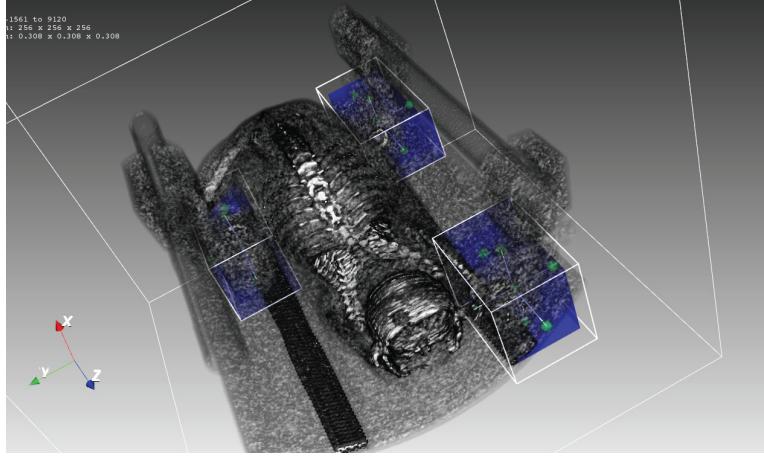


Figure 3.21 Positionnement des *BoxWidgets* sur les 3 marqueurs fiduciaux.

scalaires des voxels pondérés dans l'espace selon l'équation suivante :

$$\begin{aligned}\bar{x} &= \frac{\sum_{i=1}^n w_i x_i}{\sum_{i=1}^n w_i}, \\ \bar{y} &= \frac{\sum_{i=1}^n w_i y_i}{\sum_{i=1}^n w_i}, \\ \bar{z} &= \frac{\sum_{i=1}^n w_i z_i}{\sum_{i=1}^n w_i},\end{aligned}\tag{3.1}$$

où w_i est l'intensité des pixels i [Fitzpatrick et Sonka, 2009]. Le calcul du centroïde se fait suite au positionnement des *BoxWidgets* autour des marqueurs fiduciaux. L'utilisateur doit confirmer la validité des marqueurs fiduciaux par un signal du clavier géré par la classe *MyInteractorStyle*. Suite au signal, il y aura une extraction automatique des volumes des marqueurs fiduciaux ainsi que le calcul des positions des trois marqueurs fiduciaux. Les positions doivent absolument être affichées sur la fenêtre d'extraction de volume. Dans le cas contraire, il ne peut y avoir de recalage en raison de manque de données qui sont dues fort probablement à un mauvais placement d'un *BoxWidget*. Par exemple ; un placement à l'extérieur des frontières du volume d'image.

3.4.2 Calcul de transformation rigide

Le calcul de la transformation rigide se fait suite à l'obtention des positions spatiales, c'est à dire les centroïdes P_1 , P_2 et P_3 , des marqueurs fiduciaux pour l'image source et l'image cible comme on peut le voir à la figure 3.22. À partir de ces positions, il faut trouver les vecteurs unitaires formant une triade pour l'image source et une triade pour l'image

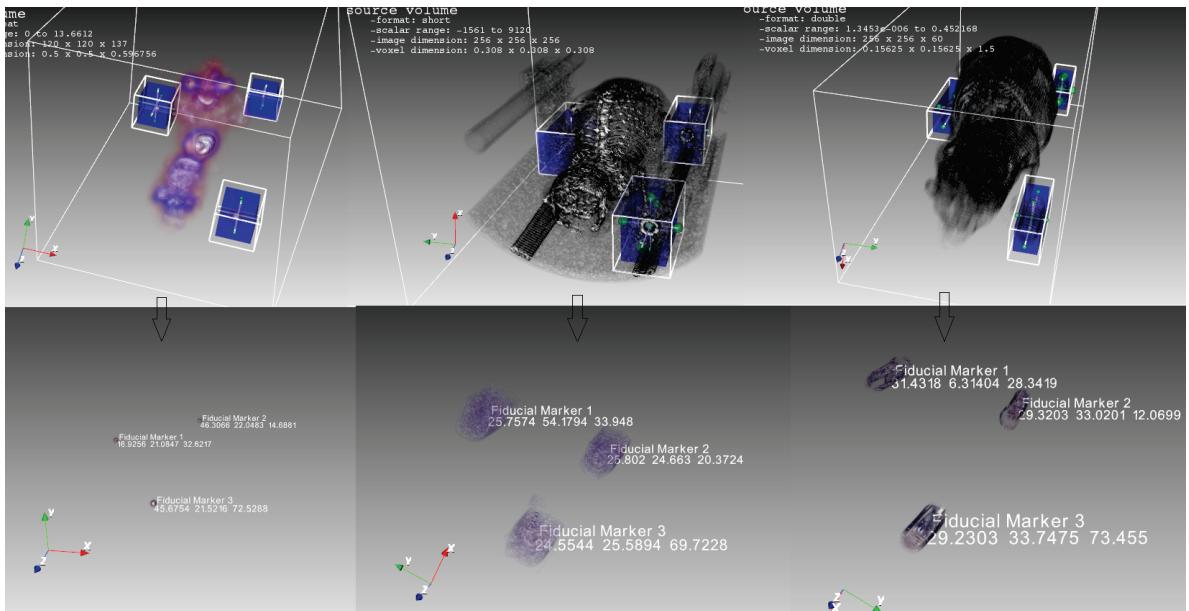


Figure 3.22 Extraction des volumes des marqueurs fiduciaux de la TEP (droite), TDM (milieu) et l'IRM (gauche). Ces images proviennent d'une même fenêtre (source) du logiciel réalisé.

cible comme le montre à la figure 3.23. Voici les calculs exécutés pour trouver les vecteurs unitaires \hat{x}_s , \hat{y}_s et \hat{z}_s de l'image source. Le même calcul est répété pour l'image cible.

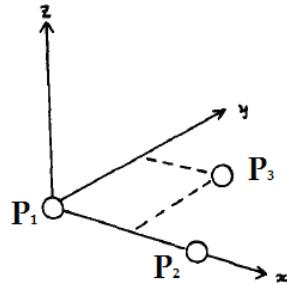


Figure 3.23 Triade formée en fonction des positions des marqueurs fiduciaux.

Calcul de la matrice de rotation

x_s est calculé comme suit :

$$x_s = P_2 - P_1. \quad (3.2)$$

Le vecteur unitaire le long de l'axe x s'obtient par :

$$\hat{x}_s = \frac{x_s}{\|x_s\|}. \quad (3.3)$$

Pour calculer y_s , il suffit de procéder comme suit :

$$y_s = (P_3 - P_1) - [(P_3 - P_1) \cdot \hat{x}_s] \hat{x}_s. \quad (3.4)$$

Le vecteur unitaire le long de l'axe y est donné par :

$$\hat{y}_s = \frac{y_s}{\|y_s\|}. \quad (3.5)$$

Le vecteur unitaire le long de l'axe z est obtenu par le produit vectoriel de \hat{x}_s et \hat{y}_s .

$$\hat{z}_s = \hat{x}_s \times \hat{y}_s. \quad (3.6)$$

Suite à l'obtention des vecteurs unitaires, il est possible de trouver la matrice M_s de passage du référentiel de VTK au référentiel de l'image source. Cette matrice a simplement comme colonnes les composantes des vecteurs \hat{x}_s , \hat{y}_s et \hat{z}_s comme suit :

$$M_s = |\hat{x}_s \hat{y}_s \hat{z}_s|. \quad (3.7)$$

et la matrice de passage du référentiel de VTK au référentiel de l'image cible, M_c comme le montre à la figure 3.24, est obtenue par

$$M_c = |\hat{x}_c \hat{y}_c \hat{z}_c|. \quad (3.8)$$

donc la matrice de rotation finale est obtenue comme suit :

$$R = M_c M_s^T. \quad (3.9)$$

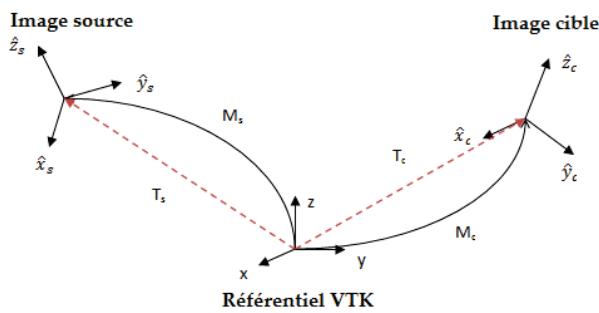


Figure 3.24 Changement de repère de l'image source à l'image cible.

Calcul du vecteur de translation

Le vecteur de translation entre les deux référentiels exprimé dans le référentiel VTK est obtenu par la somme vectorielle du vecteur de position d'un marqueur fiduciaux de l'image source par rapport à son homologue dans l'image cible.

$$T = T_c - T_s. \quad (3.10)$$

où T_s est égale au point P_1 de l'image source et T_c est égale au point P_1 de l'image cible. La programmation de la transformation rigide est présentée en détails dans l'annexe B et elle a nécessité l'utilisation de la classe *vtkLandmarkTransform* afin de calculer la matrice 4×4 de transformation de l'image source à l'image cible.

$$M_{4 \times 4} = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix}. \quad (3.11)$$

3.5 Recalage de surface

Le recalage de surface est utile lorsqu'on veut étudier le comportement surfacique de plusieurs modalités d'imagerie ou bien de comparer deux reliefs d'une même scène prise à différents moments. En ce qui concerne ce projet, le recalage de surface sera surtout utile pour jumeler des images profilométriques avec d'autres modalités d'imagerie comme la TEP.

Les images profilométriques du laboratoire sont obtenues par un profilomètre, un instrument à exploration progressive, qui par balayage d'une ligne laser et à l'aide d'une caméra, permet de mesurer un relief d'un objet, p.ex. un petit animal. Le résultat est présenté sous forme d'un maillage constitué de vertex et des cellules triangulaires comme le montre la figure 3.33 d'une petite souris. Pour aider au recalage dans certaines situations, l'utilisateur peut utiliser une interface graphique, appelée *Data Processing* qui a été programmée comme une option supplémentaire dans le logiciel réalisé dans ce projet. Cette Interface, présentée à la figure 3.25, est interne au programme et elle est conçue pour répondre au besoin de l'utilisateur en matière de traitement et de segmentation d'image. Elle permet à l'utilisateur de faire un pré-traitement sur des maillages ou des images afin qu'ils soient convenables au recalage. Il est préférable d'utiliser cette interface avant de faire un recalage en raison de forte présence de bruit ou d'artéfacts dans les images obtenues au laboratoire. Par exemple, certaines structures profilométriques de petite souris sont obtenues de façon séparées, c'est-à-dire qu'il y a deux fichiers qui représentent le côté gauche et le côté

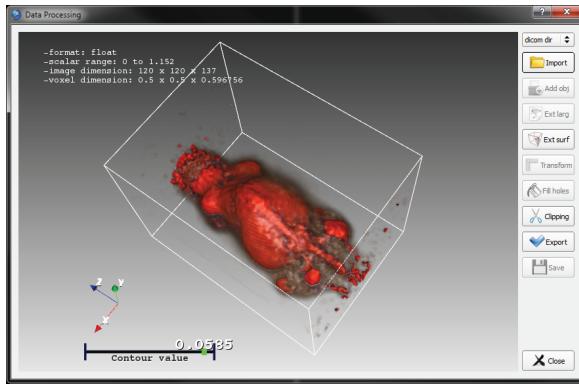


Figure 3.25 Interface graphique de *Data Processing*.

droit de la souris. À l'aide de cette interface graphique, il est possible de jumeler les deux côtés et appliquer une transformation géométrique à l'un des côtés permettant de réduire l'écart entre les deux côtés dû à l'imprécision de l'alignement des caméras durant la prise de mesure.

L'interface *Data Processing* permet aussi de faire l'extraction de volume d'intérêt. Celle-ci est faite de façon manuelle par l'utilisateur qui peut manipuler à l'aide d'un *BoxWidget*, comme le cas pour l'identification des marqueurs fiduciaux, un cube dont les dimensions géométriques seront utilisées pour fixer les limites cartésiennes du volume à extraire.

3.5.1 Structure de l'algorithme

Le recalage de surface suit une structure particulière comme le montre la figure 3.26 qui dépend du type d'image à recaler. Pour un recalage de deux profiles, l'approche choisie est de positionner au moins trois points, *landmarks*, correspondants suivant la logique du cas du recalage avec des marqueurs fiduciaux, afin de trouver la transformation qui permet d'aligner les bases des deux profiles avant de passer à l'algorithme d'ICP pour corriger l'erreur due aux positionnements des points correspondants (les *landmarks* permettent en fait de faire un pré-recalage qui est ensuite raffiné par ICP). Pour un recalage d'un profile avec une image 3D, il faut absolument extraire la surface de cette dernière avant de la comparer au profile. Cette extraction va maintenant être discutée.

3.5.2 Extraction de surface à partir d'une image 3D

Les images tomographiques obtenues du laboratoire sont généralement présentées soit en série de fichier Dicom comme le cas d'images TEP ou en un seul fichier de type .raw comme pour les images IRM et TDM. Ces images sont en fait des champs scalaires représentés sous

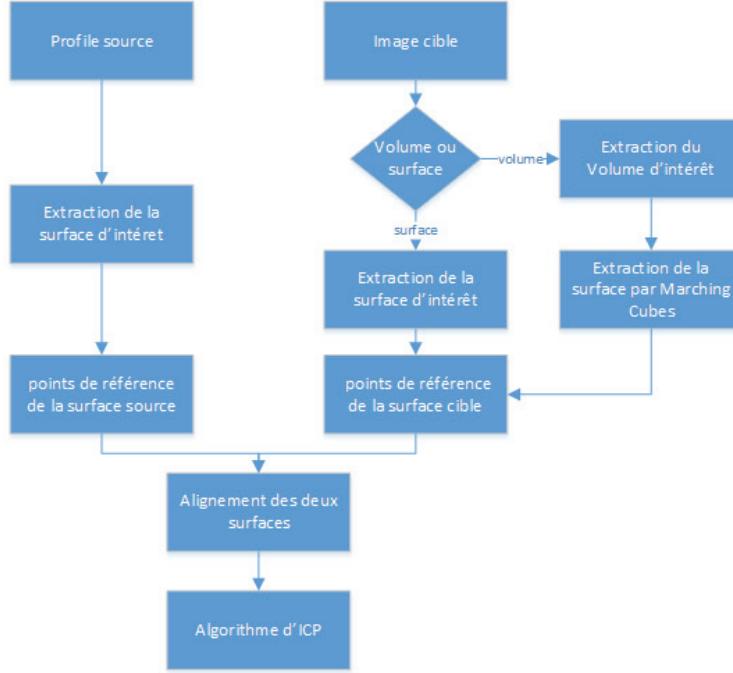


Figure 3.26 Algorithme de recalage de surface.

forme de densités associées à différents degrés de transparence. Différents seuils de densité définissent des couches d’isodensité qui sont obtenues selon une progression linéaire entre la densité maximale et la densité minimale des données. Un seuil doit être fixé par l’utilisateur dans les paramètres du programme afin de pouvoir extraire une couche d’isodensité et le seuil varie d’une modalité d’imagerie à une autre. La figure 3.27 présente une image TEP d’un petit animal dont nous avons affiché les couches d’isodensité en fonction des seuils que l’utilisateur fixe à l’aide d’un *sliderWidget*, ou curseur de défilement. Il est possible d’extraire des couches de façon successive comme le montre la figure 3.28.

La gestion de transparence est faite à l’aide d’une fonction puissance qui déterminera une valeur de gamma (taux d’opacité) pour chaque couche d’isodensité et une fonction permettant de fixer les niveaux de gris de chacune des couches afin de les repérer de façon visuelle sur la fenêtre de VTK. Cela est réalisé par les classes *vtkPiecewiseFunction* et *vtkColorTransferFunction* comme on peut le voir à la figure 3.12.

Algorithme des Marching Cubes

L’algorithme des Marching Cubes est très utilisé dans le domaine de la reconnaissance faciale, de la robotique mobile ainsi que dans la conception des jeux vidéo. Nécessitant un temps de calcul relativement faible, il permet d’extraire une surface de densité fixée en un maillage structuré et uniforme 3D. L’algorithme consiste en une discréétisation de

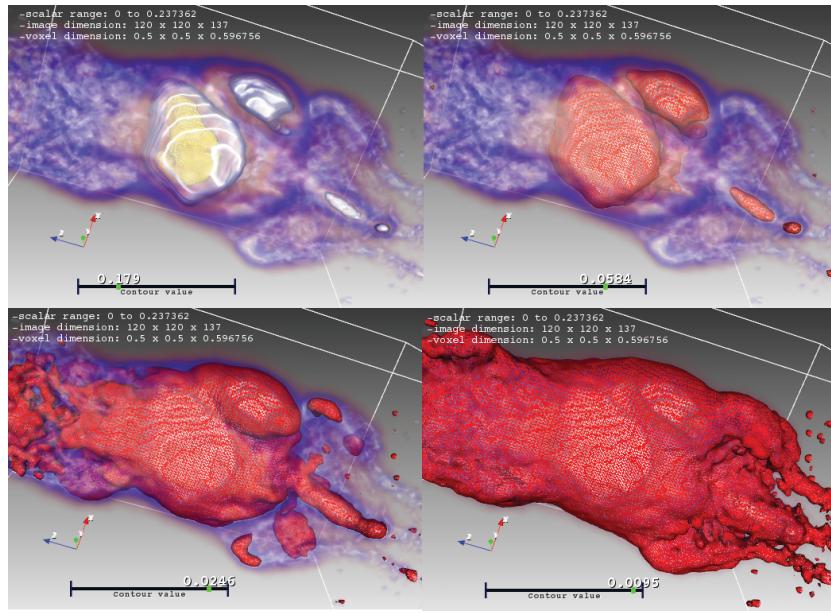


Figure 3.27 Couches d’isodensité d’une image PET en fonction d’une certaine valeur *Contour value* pour l’algorithme Marching Cubes.

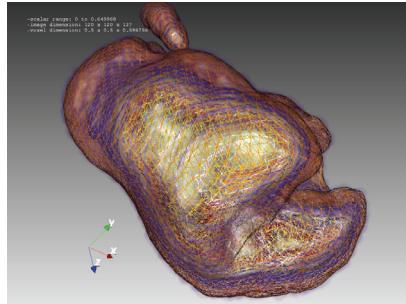


Figure 3.28 Extraction de plusieurs couches d’isodensité constituant un organe d’une souris.

la scène en cubes, une fixation d’un seuil de densité par l’utilisateur et un parcours cube par cube qui constitue l’espace de la scène. Pour chaque cube, on compte le nombre de sommets au dessus du seuil et on applique une topologie de triangulation à l’intérieur du cube [Lorensen et Cline, 1987].

Un cube comporte 8 sommets et 12 arêtes et chaque sommet peut prendre deux états ; il existe donc 256 configurations possibles. Grâce à des symétries, il est possible de réduire le nombre de configuration à 15. La figure 3.29 montre les 15 configurations de base que peut prendre la triangulation du cube. Chaque configuration correspond à un ensemble de facettes tracées à l’intérieur du volume et une reconstruction de la surface suivant le nombre de sommets se trouvant dans la couche. Les sommets en bleu indiquent un seuil au dessus du seuil fixée par l’utilisateur et on place un triangle au milieu de l’arrête sur

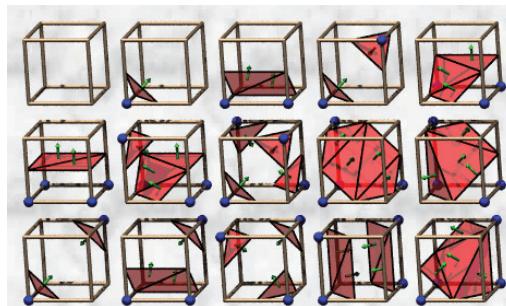


Figure 3.29 Différentes topologies de polygones servant à la reconstruction de surface. Image provenant de [Dod HPC Modernization program, s. d.].

laquelle est sensé se trouver le ou les sommets bleus. Ces triangulations de cube permettent au final de reconstituer la surface de la couche d'isodensité.

3.5.3 Positionnement des points de référence

Le positionnement des points de référence doit se faire de façon manuelle par l'utilisateur dans le logiciel développé dans le présent projet. L'utilisateur doit être en mesure de trouver trois points correspondants sans avoir une grande précision et insérer des *Landmarks* comme le montre la figure 3.30.



Figure 3.30 Structure graphique du point de référence pour le recalage de surface.

La figure 3.31 montre un exemple de positionnement par l'utilisateur des points de référence sur deux profils à géométrie simple. Dans ce cas, il est facile d'identifier les points correspondants puisque la structure n'est pas complexe. Cependant, pour certaines images, notamment dans le cas de la souris, il arrive parfois que les images à recaler n'ont pas grande similitude et cela risque d'amplifier l'erreur de l'utilisateur lors du positionnement des points de références. Les positions cartésiennes de ces points de référence sont dans un premier temps utilisées pour l'alignement des référentiels entre l'image source et l'image cible 3.31. Cet alignement est ensuite raffiné comme il sera maintenant expliqué.

3.6 Algorithme pour le raffinement des résultats

Suite à la transformation à l'aide des marqueurs fiduciaux, il est fortement suggéré d'appliquer un algorithme de recalage d'image afin de raffiner le résultat. Cela est du à l'in-

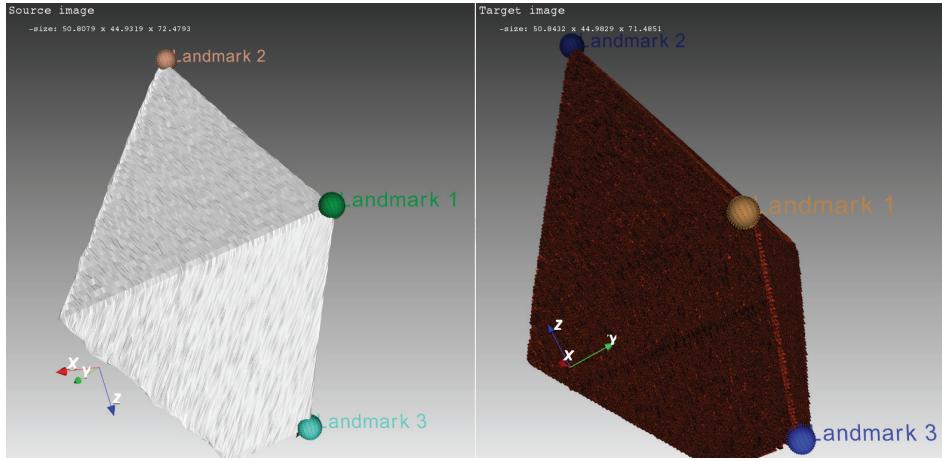


Figure 3.31 Positionnement des points de référence correspondant pour l'image source (droite) et cible (gauche).

exactitude de la mise en correspondance des marqueurs fiduciaux ou points de référence par l'utilisateur avec les *BoxWidgets* (figure 3.14) ou des landmarks (figure 3.30). En ce qui concerne le présent projet, afin de raffiner le recalage de surface, nous avons décidé d'appliquer l'algorithme de l'Iterative Closest Points. Pour le recalage à l'aide des marqueurs fiduciaux, nous avons décidé d'appliquer l'algorithme de l'information mutuelle.

3.6.1 Algorithme ICP (iterative closest points)

L'algorithme ICP a pour objectif de trouver une transformation géométrique permettant de minimiser l'erreur quadratique entre deux séries de points appartenant à une image 2D ou à une structure 3D. L'algorithme peut être utilisé dans la détection faciale comme un moyen de mesure de similarité entre deux profils de personne. Il peut être aussi utilisé pour estimer une position ou un paramètre entre deux points, par exemple ; en robotique, la transformation qui minimise l'erreur entre deux prises de mesure consécutives de l'environnement 3D du robot est proportionnelle au déplacement de celui-ci. Les étapes de l'algorithme d'ICP sont présentées à la figure 3.32 [Umeyama, 1991][Maurer Jr *et al.*, 1996].

Sélection des nuages de points

Pour le recalage avec marqueurs fiduciaux, la sélection de points pour les images source et cible est obtenue suite à la transformation du champ scalaire que compose l'image 3D en une structure géométrique 3D. Cela se fait par le passage des données de type *vtkImageData* au type de *vtkPolyData*.

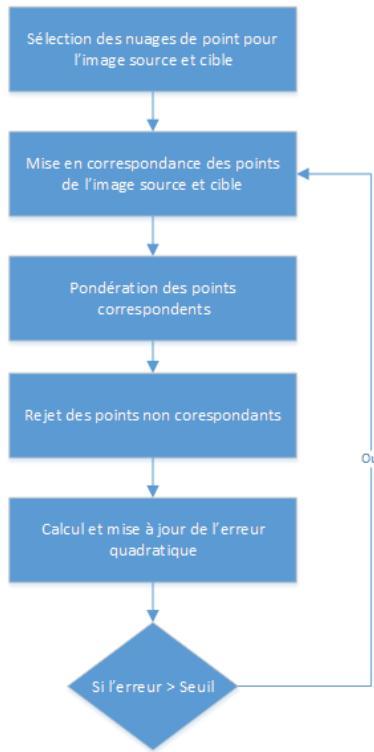


Figure 3.32 Algorithme Iterative Closest Points.

Pour un recalage de surface, le nuage de points correspond à l'ensemble des vertex qui constituent une image profilométrique ou une structure surfacique. Certaines images volumiques obtenues du laboratoire sont de grande taille (512x512x512) et nécessitent beaucoup de temps de calcul et cela peut nuire aux manipulations de l'image ou des *Box Widgets*. Il est préférable donc de retrancher la taille de l'image à (256x256x256), dont la résolution est encore acceptable, et extraire le volume ou la surface d'intérêt à l'aide des classes *vtkImageReslice* et *vtkExtractVOI*. Cela permet aussi de réduire le nombre de points à correspondre pour l'algorithme ICP, réduisant ainsi le temps d'itération et permettant aussi d'éviter d'avoir des points aberrants ou des artefacts qui peuvent nuire à la recherche de la transformation géométrique.

La figure 3.33 présente, à gauche, une extraction à l'aide d'un *BoxWidget* d'une image profilométrique d'une souris obtenue avec le système QOS ; à droite on représente le résultat de l'extraction.

Mise en correspondance des points

La mise en correspondance des points de l'image source et de l'image cible est cruciale pour l'algorithme d'ICP. En général, elle se fait par la méthode de la recherche du plus proche voisin. Cette méthode est utilisée dans de nombreux domaines tels que la reconnaissance

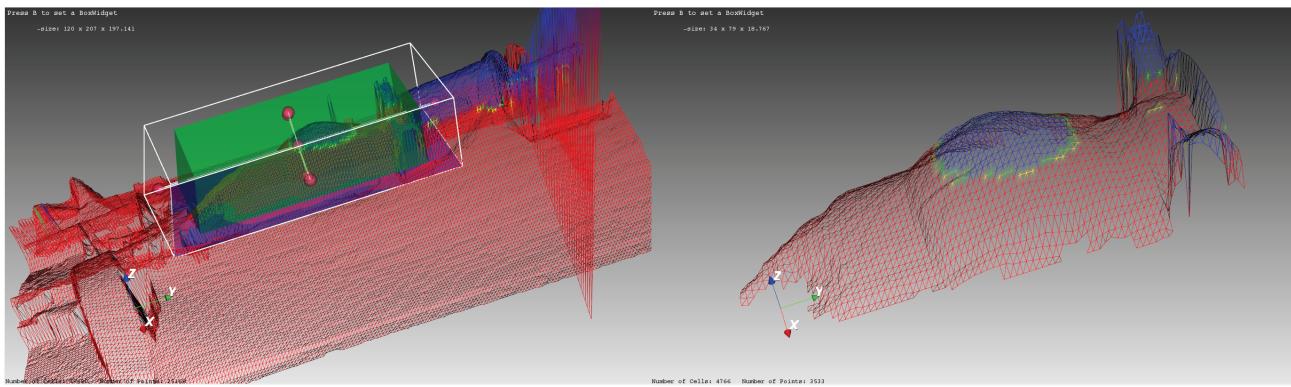


Figure 3.33 Profil d'une souris du système QOS à gauche et extraction du profil d'intérêt à droite.

de forme ou la compression des données géométriques 3D[Gupta *et al.*, 2002]. Pour le recalage, le calcul du plus proche voisin se fait d'abord en alignant les centroïdes de deux profils ou deux images 3D et ensuite, pour un point de l'image source, on calcule la distance euclidienne minimale le séparant avec les points de l'image cible. La recherche de point peut être accélérée en les représentant à l'aide d'une structure de données appelée *k-d tree*. Celle-ci permet d'accélérer la recherche de points en les partitionnant dans l'espace et en faisant une recherche par plages au lieu de parcourir le tableau de points linéairement.

Pondération des points correspondants

La mise en correspondance peut être d'une grande complexité surtout lorsqu'on a des images ou des profils ayant une grande quantité de points. Dans cette perspective, l'algorithme d'ICP permet d'ajouter une pondération sur les points ayant une distance très proche afin de favoriser leur alignement.

Rejet des points non correspondants

Dans la majorité des cas, le nombre de point de l'image source n'est pas égal au nombre de l'image cible. L'algorithme rejette dans les calculs les points d'une image n'ayant pas de point correspondant dans l'autre image. Il est possible aussi de rejeter les points correspondants ayant une grande distance comparativement à la moyenne des autres points.

Calcul de l'erreur quadratique

L'utilisateur peut choisir la transformation qu'il désire utiliser à l'aide de l'interface option interne au programme comme le montre la figure 3.34. L'interface présente les choix suivants de transformation ; rigide, similarité ou affine. Dans le cas d'une transformation

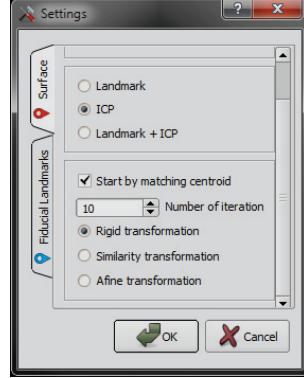


Figure 3.34 Interface d'options pour l'ICP.

rigide, l'erreur quadratique est donnée par

$$e^2(R, t) = \sum_{i=1}^N \| y_i - (Rx_i + t) \|^2, \quad (3.12)$$

où N est le nombre de paires de points considérés, R est la matrice de rotation et t le vecteur de translation entre les points de l'image cible y_i et ceux de l'image source x_i permettant de minimiser l'erreur quadratique. De la même manière, on a pour les transformations de similarité et affine les équations suivantes

$$e^2(c, R, t) = \sum_{i=1}^N \| y_i - (cRx_i + t) \|^2. \quad (3.13)$$

$$e^2(c, A, t) = \sum_{i=1}^N \| y_i - (cAx_i + t) \|^2. \quad (3.14)$$

Prenant l'exemple d'une transformation rigide 2D. Le principe de recherche de transformation se base d'abord à représenter les coordonnées des points de l'image source et cible en fonction de leurs centroïdes via les équations suivantes

$$\bar{x} = \frac{1}{N} \sum_{i=1}^N x_i,$$

$$\bar{y} = \frac{1}{N} \sum_{i=1}^N y_i, \quad (3.15)$$

On représente les points par rapport aux centroïdes afin de simplifier les calculs.

$$\begin{aligned} x'_i &= x_i - \bar{x}. \\ y'_i &= y_i - \bar{y}. \end{aligned} \quad (3.16)$$

On peut alors réécrire l'équation 3.12 sous la forme

$$e^2(R, t) = \sum_{i=1}^N \| y'_i - Rx'_i + (-R\bar{x} + \bar{y} - t) \|^2 \quad (3.17)$$

et si

$$t = -R\bar{x} + \bar{y}, \quad (3.18)$$

il suffit alors de minimiser :

$$\begin{aligned} &\sum_{i=1}^N \| y'_i - Rx'_i \|^2 \\ &= \sum_{i=1}^N \| y'_i \|^2 - 2\text{tr}(R \sum_{i=1}^N x'_i y'^T_i) + RR^T \sum_{i=1}^N \| x'_i \|^2 \\ &= \sum_{i=1}^N \| y'_i \|^2 - 2\text{tr}(R \sum_{i=1}^N x'_i y'^T_i) + \sum_{i=1}^N \| x'_i \|^2. \end{aligned} \quad (3.19) \quad (3.20) \quad (3.21)$$

Il suffit donc de maximiser le terme central de l'équation 3.21 afin de minimiser l'erreur quadratique. Il est possible de représenter le terme central en décomposition en valeurs singulières selon l'équation suivante

$$\sum_{i=1}^N x'_i y'^T_i = \begin{bmatrix} a_{xx} & a_{xy} & a_{xz} \\ a_{yx} & a_{yy} & a_{yz} \\ a_{zx} & a_{zy} & a_{zz} \end{bmatrix} = USV^T, \quad (3.22)$$

où U et V sont des matrices orthogonales et S une matrice diagonale composée de valeurs singulières. Il est possible de prouver que la matrice de rotation R est donnée par [Umeyama, 1991] [Besl et McKay, 1992] [Schonemann, 1966] [Farrell et Stuelpnagel, 1966] [Arun *et al.*, 1987]

$$R = VU^T. \quad (3.23)$$

Convergence de l'algorithme

L'algorithme d'ICP convergera à un minimum local et s'arrêtera lorsque la mise en correspondance ne change plus de façon appréciable, ceci étant spécifié par une valeur seuil. Afin de minimiser le temps de recherche des points correspondants, il est possible de jumeler avant toute transformation les centroïdes des deux nuages de points par l'intermédiaire de *Number of Iteration* présenté dans l'interface d'option d'ICP si l'algorithme prend un temps considérable pour atteindre la convergence.

3.6.2 Algorithme de l'information mutuelle

Il est possible d'appliquer l'algorithme de l'information mutuelle dans le but de raffiner le résultat obtenu par la transformation à l'aide des marqueurs fiduciaux. Cet algorithme suit l'architecture présentée à la figure 2.5. La figure 3.35 ainsi que l'annexe C présentent respectivement en détail la structure générale et le code informatique implémenté permettant d'obtenir la transformation matricielle entre les deux images. Le code est exécuté par la librairie ITK en utilisant les filtres *MutualInformationImageToImageMetric* permettant de calculer l'information mutuelle entre l'image source et cible et le filtre *VersorRigid3DTransformOptimizer* qui se base sur le calcul d'une descente de gradient (*gradient descent*) afin d'optimiser les angles de rotation et le vecteur de translation comme décrit à la section 2.5.4. Le filtre *VersorRigid3DTransform* permet d'appliquer la transformation rigide suite à l'optimisation et le filtre *LinearInterpolateImageFunction* permet une interpolation linéaire des voxels suite à la transformation [Ibanez *et al.*, 2005].

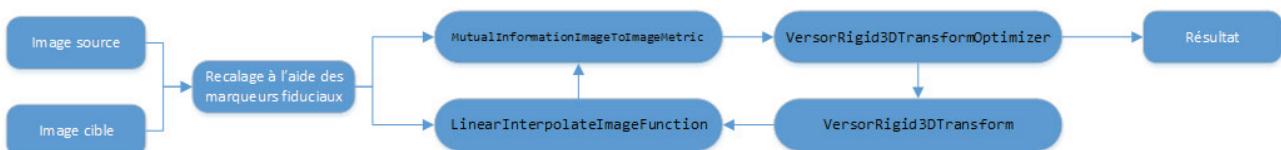


Figure 3.35 Structure de programmation suivie pour un recalage à l'aide des marqueurs fiduciaux et rafinage par l'algorithme de l'information mutuelle.

L'algorithme s'arrêtera lorsque la métrique *MutualInformationImageToImageMetric* atteindra une valeur stable ou qu'il atteindra un nombre maximal d'itération. Afin de réaliser cet algorithme, l'utilisateur doit spécifier dans l'interface d'option l'utilisation de l'algorithme d'information mutuelle suite à la transformation à l'aide des marqueurs fiduciaux.

3.7 Conclusion

Ce chapitre présente la conception informatique du projet de recherche à l'aide de Visual Studio et de Qt Designer en utilisant les librairies VTK et ITK. L'annexe D présente les propriétés techniques et fonctionnelles du logiciel permettant d'indexer l'avancement et la qualité du développement du code informatique. Dans ce chapitre, on présente aussi une description détaillée en ce qui concerne l'affichage des images source et cible. On démontre aussi comment le recalage rigide à l'aide des marqueurs fiduciaux a été implémenté ainsi que les algorithmes de raffinement ICP, utilisé pour le recalage de surface, et la métrique de l'information mutuelle, pour un recalage rigide multimodale d'images 3D.

CHAPITRE 4

RÉSULTATS ET DISCUSSION

Ce chapitre traitera des résultats obtenus lors des tests de recalage de surface et de recalage à l'aide des marqueurs fiduciaux. Un des objectifs visés lors de la conception du programme informatique était d'offrir le plus de degrés de liberté possibles à l'utilisateur en ce qui a trait au choix d'images à recaler. C'est à dire que l'utilisateur a le choix, peu importe le type de recalage, de la modalité considérée pour l'image source ou pour l'image cible. Dans le présent chapitre, en ce qui concerne le recalage de surface, on présentera les résultats pour un recalage à géométrie simple, un recalage à géométrie complexe ainsi qu'un recalage d'une image profilométrique avec l'imagerie TEP d'un petit animal. Pour le recalage à l'aide des marqueurs fiduciaux, nous avons décidé de présenter des résultats du recalage des modalités TEP avec CT et TEP avec IRM afin d'exploiter la théorie de l'information mutuelle.

4.1 Résultat du recalage de surface

4.1.1 Recalage surface à surface

Recalage de surface à géométries simples

La figure 4.1 présente la structure de programmation suivie pour faire un recalage de deux images profilométriques à géométries simples. L'image source a une géométrie d'un demi diamant tandis que l'image cible a une géométrie d'un diamant complet et elle est orientée à 180° par rapport à l'image source comme le montrent respectivement les parties supérieures gauche et droite de la figure 4.2. Nous avons donc recalé les deux structures à l'aide des positions des points de référence comme décrit à la section 3.4.2.

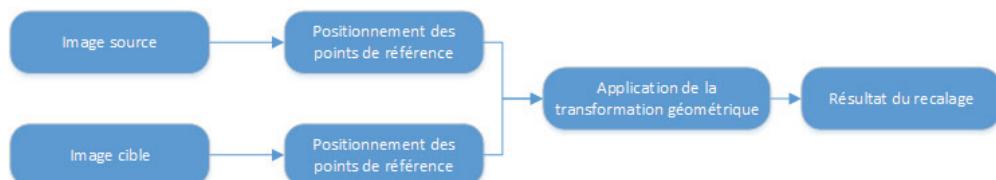


Figure 4.1 Structure de programmation suivie pour un recalage d'un profil à un autre profil à l'aide des points de référence.

Il est très important de placer les points de référence aux positions les plus correspondantes possibles pour les deux images. Plus l'utilisateur est précis dans les positionnements des points, meilleur sera le résultat du recalage. Pour des structures à géométries simples comme le cas de la figure 4.2, il n'est pas nécessaire d'être précis en raison de la grande similarité entre l'image source et l'image cible. Cependant, pour des tests avec des structures à géométries complexes comme le cas d'une souris de la figure 4.4, il est recommandé à l'utilisateur de porter une attention particulière sur l'emplacement de ces points de référence s'il désire obtenir un recalage le plus juste possible.

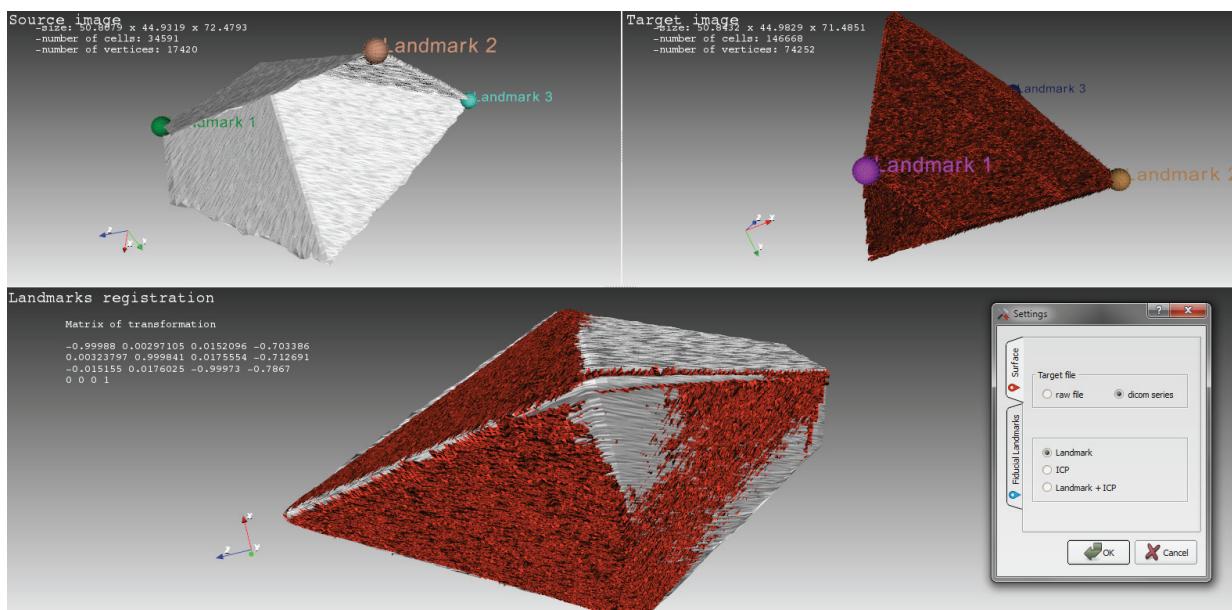


Figure 4.2 Résultat d'un recalage surface avec une autre surface d'un diamant avec 3 points de référence.

Le recalage a été exécuté sans l'ajout de l'algorithme d'ICP et le résultat ainsi que la matrice de transformation finale, qu'on a appliquée à l'image source, sont affichés au bas de la figure 4.2. Comme on peut le voir, le résultat semble tout à fait satisfaisant puisque l'image source est alignée de façon juste sur l'image cible.

Mesure du temps de recalage entre les deux diamants

Image	Nombre de points	Taille (mm)
Source	17420	50.8679 x 44.9319 x 72.4793
Cible	74252	50.8432 x 44.9829 x 71.4851

Temps du processus : moins d'une seconde

Tableau 4.1 Mesure expérimentale du temps d'exécution du code informatique pour un recalage de surface à géométries simples.

Recalage de surface à géométries complexes

Pour le recalage de surfaces à géométries complexes, nous avons décidé de recaler sans l'utilisation des points de référence une image source d'une souris et une image cible de la même souris à laquelle nous avons appliqué une transformation rigide afin de tester l'efficacité de l'algorithme ICP sur des structures complexes. La figure 4.3 présente la structure de programmation suivie pour ce type de test.

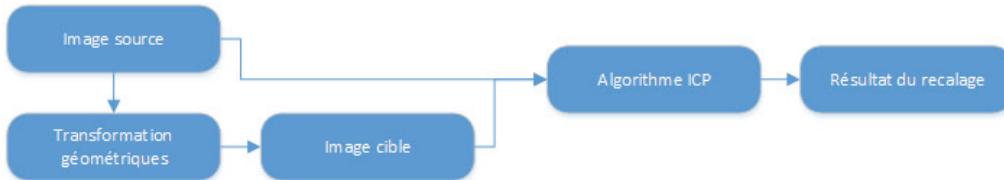


Figure 4.3 Structure de programmation suivie pour un recalage de surface d'une souris à l'aide de l'algorithme d'ICP.

La transformation rigide appliquée est présentée par la matrice théorique 4.2 qui consiste à une rotation de 30° autour de l'axe Y et une translation par le vecteur

$$25.4 \begin{bmatrix} 10 \\ 20 \\ 30 \end{bmatrix}. \quad (4.1)$$

$$T = \begin{bmatrix} 0.866 & 0 & -0.5 & 254 \\ 0 & 1 & 0 & 508 \\ 0.5 & 0 & 0.866 & 762 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.2)$$

La figure 4.4 présente le résultat obtenu. Comme on peut le constater, la matrice de transformation obtenue, présentée par l'équation 4.3, est très proche de la matrice théorique. Ceci prouve que l'algorithme ICP est très efficace comme moyen de raffinage pour un recalage de surface.

$$T = \begin{bmatrix} 0.867509 & -0.000857188 & -0.497421 & 254.673 \\ 0.0140718 & 0.999641 & 0.0228188 & 508.751 \\ 0.497222 & -0.0267952 & 0.867209 & 760.775 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.3)$$

Le graphique de la figure 4.5 présente la courbe de variation de la distance moyenne, en

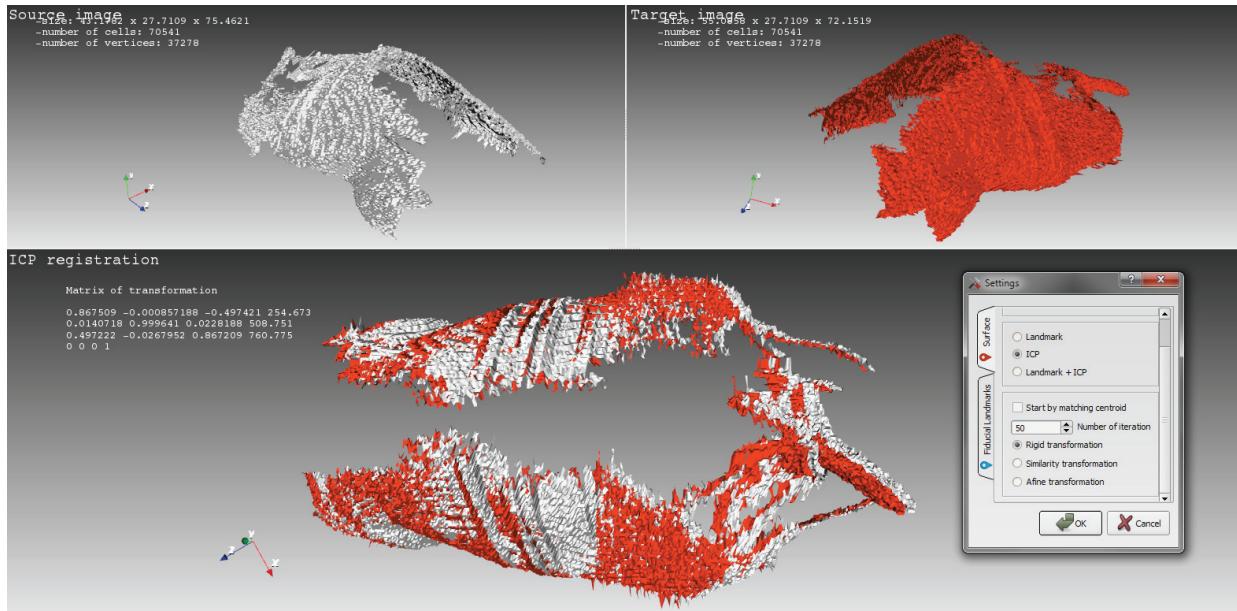


Figure 4.4 Résultat d'un recalage de surface d'une souris avec la même surface mais ayant subi une transformation rigide représentée par la matrice de l'équation 4.2.

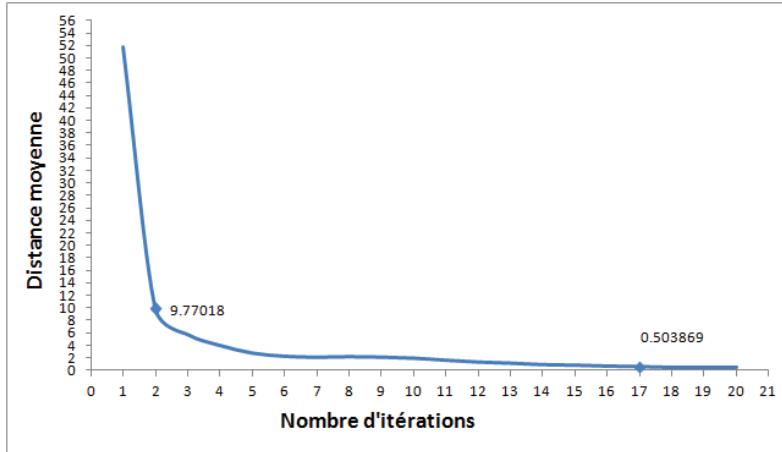


Figure 4.5 Graphique de la distance moyenne entre les points correspondants en fonction du nombre d'itérations.

millimètres, entre les points correspondants de l'image source et l'image cible en fonction du nombre d'itérations. On peut remarquer qu'après chaque itération, la distance entre les points diminue rapidement jusqu'à une valeur très petite et stable. Comme les deux images sont similaires, il est normal que la courbe tend à s'approcher le plus près possible de 0. Pour des structures ayant une faible similarité, la courbe se stabilisera à une certaine valeur non nulle. C'est pour cette raison que nous avons décidé de donner le choix à l'utilisateur de fixer le nombre maximal d'itérations au lieu de la distance moyenne puisque cela risquerait

de plonger le programme informatique dans une boucle infinie s'il n'atteint pas la distance voulue.

Mesure du temps de recalage entre les deux images

Image	Nombre de points	Taille (mm)
Source	70541	$43.1782 \times 27.7109 \times 75.4621$
Cible	70541	$55.0858 \times 27.7109 \times 72.1519$

Temps du processus : 1 sec pour 50 itérations

Tableau 4.2 Mesure expérimentale du temps d'exécution du code informatique pour un recalage avec ICP de surface à géométries complexes.

4.1.2 Recalage de surface avec volume tomographique

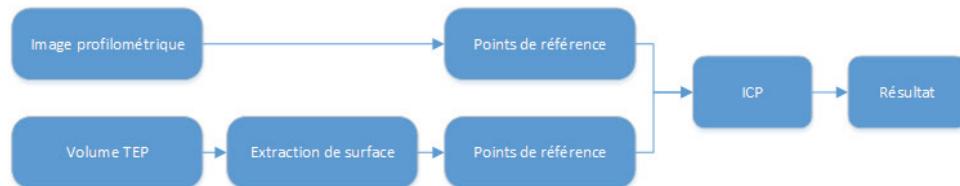


Figure 4.6 Structure de programmation suivie pour un recalage d'image profilométrique et d'un volume d'image TEP.

La figure 4.6 présente la structure de programmation suivie pour faire un recalage d'une image source profilométrique d'une souris et une image cible d'un volume TEP de la même souris. Tout d'abord, l'utilisateur doit procéder à l'extraction de surface de façon manuelle, comme le montre à la figure 4.7, à l'aide de l'interface *Data Processing* présentée à la figure 3.25.

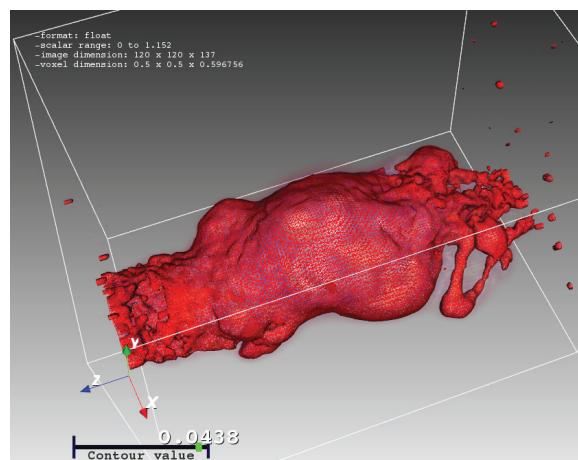


Figure 4.7 Extraction de surface d'un volume TEP dont la valeur d'isodensité est de 0.0438.

L'extraction de la surface doit être jugée valide ou non par l'utilisateur, c'est-à-dire si elle représente raisonnablement la surface de la souris. Pour ce recalage, il a été déterminé de façon empirique que la valeur 0.0438 d'isodensité est adéquate pour obtenir une surface du volume TEP suffisamment similaire à celle de l'image profilométrique. Suite à l'extraction de cette surface, nous avons d'abord commencé à recaler par points de référence et ensuite raffiner le résultat par l'algorithme d'ICP dont le nombre maximal d'itérations est fixé à 20. La figure 4.8 montre le résultat de l'alignement de l'image source sur le volume TEP. Le résultat semble largement satisfaisant puisque l'image profilométrique de la partie supérieure de la souris se positionne adéquatement sur la partie supérieure du volume TEP.

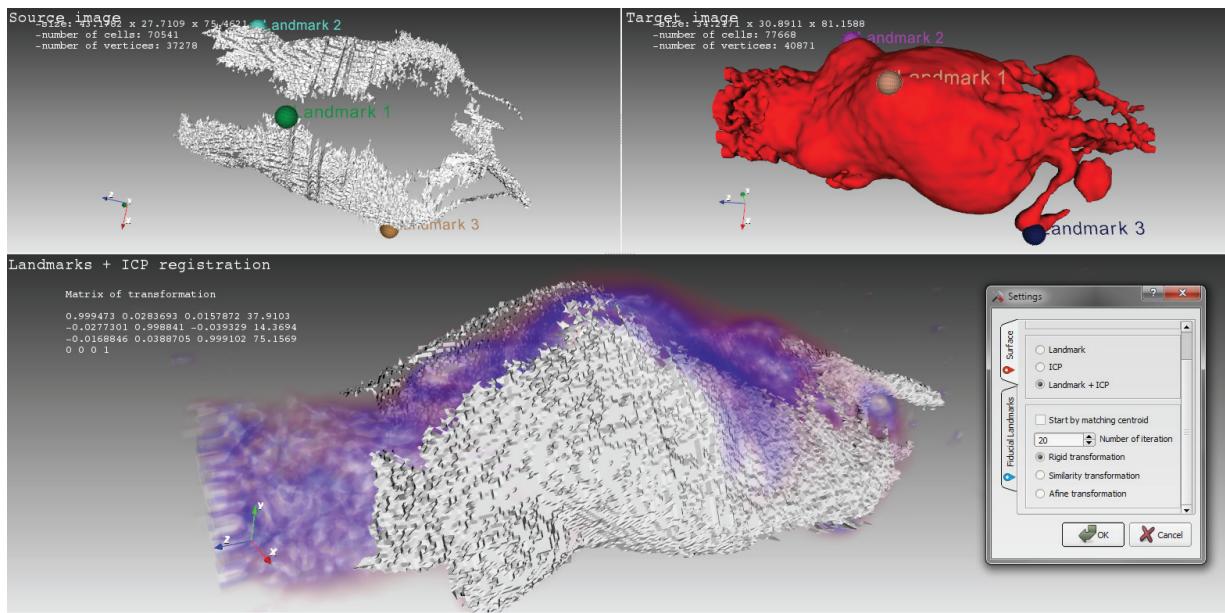


Figure 4.8 Résultat d'un recalage de surface d'une souris avec un volume TEP de la même souris.

Mesure du temps de recalage entre l'image profilométrique et l'image TEP

Image	Nombre de points	Taille (mm)
Source	70541	$43.1782 \times 27.7109 \times 75.4621$
Cible	77668	$34.2471 \times 30.8911 \times 81.1588$

Temps du processus : 1 sec pour 20 itérations

Tableau 4.3 Mesure expérimentale du temps d'exécution du code informatique pour un recalage avec ICP de surface avec volume tomographique.

Comme pour le cas du recalage précédent, le graphique de la figure 4.9 présente la distance moyenne des points correspondants de la surface profilométrique et la surface du volume TEP en fonction du nombre d'itérations suite à l'application de l'algorithme d'ICP. On

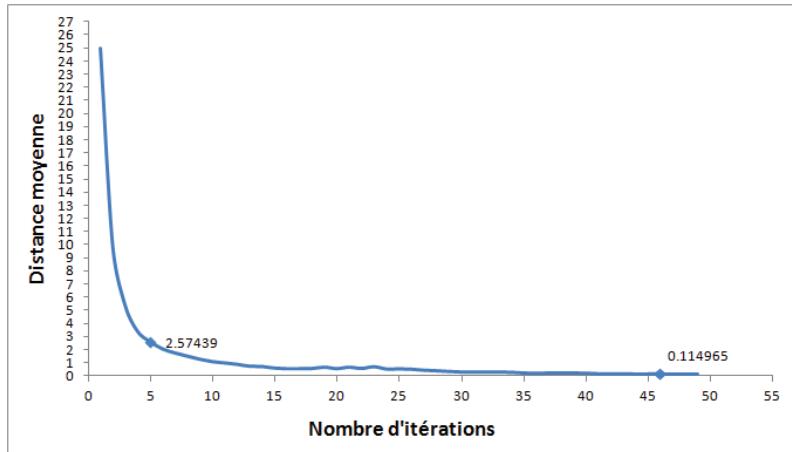


Figure 4.9 Graphique de la distance moyenne entre les points correspondants en fonction du nombre d’itérations.

peut conclure que malgré la complexité géométrique des images, la distance entre les points diminue rapidement après chaque itération jusqu’à une valeur très petite et proche de 0.

4.2 Résultat du recalage à l'aide des marqueurs fiduciaux

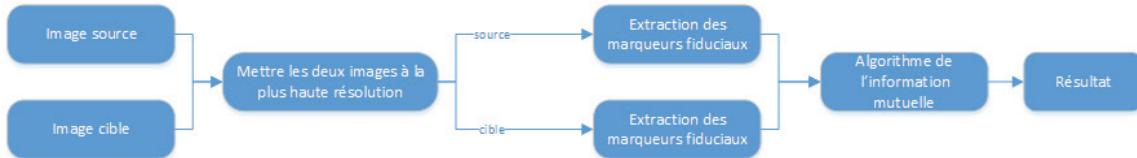


Figure 4.10 Structure de programmation suivie pour un recalage d'image à l'aide des marqueurs fiduciaux.

Le recalage à l'aide de marqueurs fiduciaux a été réalisé en suivant la structure de la figure 4.10. Nous avons décidé d'abord de mettre les deux images à la plus haute résolution et ensuite procéder au recalage à l'aide des marqueurs fiduciaux. Comme le montre la structure, pour le recalage d'image de différente modalités, nous avons utilisé l'information mutuelle afin de raffiner le résultat.

4.2.1 Recalage TEP avec TDM

Pour ce recalage, nous avons décidé que celui-ci doit être exécuté avec les marqueurs fiduciaux. L'image TEP sera l'image source à laquelle on appliquera la transformation

rigide afin de la superposer le plus justement possible avec l'image cible, la TDM. Ceci est dû à la résolution de l'image TDM qui est bien supérieure à celle de l'image TEP.

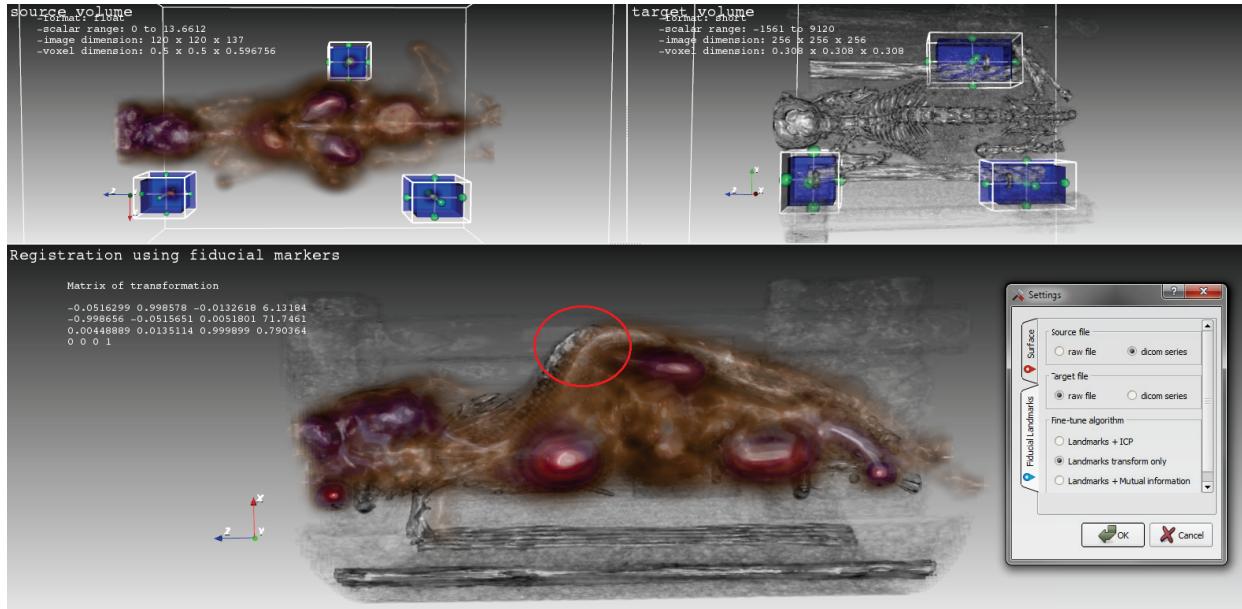


Figure 4.11 Résultat d'un recalage d'un volume de souris TEP (supérieur gauche) et d'un volume TDM (supérieur droit) à l'aide des marqueurs fiduciaux seulement.

La figure 4.11 présente les images des deux modalités dans lesquelles les marqueurs fiduciaux correspondants ont été entourés manuellement à l'aide des *BoxWidgets*. Comme on peut le voir au bas de la figure, l'image TEP semble s'enligner avec l'image TDM. Cependant, on peut remarquer que l'alignement n'est pas tout à fait au point puisqu'il y a une discordance au niveau des vertèbres entourées par le cercle rouge. Afin de corriger le problème, l'algorithme de l'information mutuelle, dont le nombre maximal d'itération est fixé à 100, est appliqué au résultat du recalage. Un excellent alignement est alors obtenu comme le montre à la figure 4.12 et 4.13. La figure 4.14 présente des coupes 2D du résultat du recalage sans (droite) et avec (gauche) raffinement par information mutuelle. À première vue, l'image TEP (colorée) semble s'aligner avec l'image TDM comme le montre le haut de la figure 4.14 où le marqueur fiduciaux, en pointillé, dans l'image TEP se trouve exactement au centre du marqueur fiduciaux de l'image TDM. Le changement semble très visible au bas à droite de la figure où on voit l'organe (coeur) dans l'image TEP s'enligner parfaitement à la position anatomique du coeur dans l'image TDM.

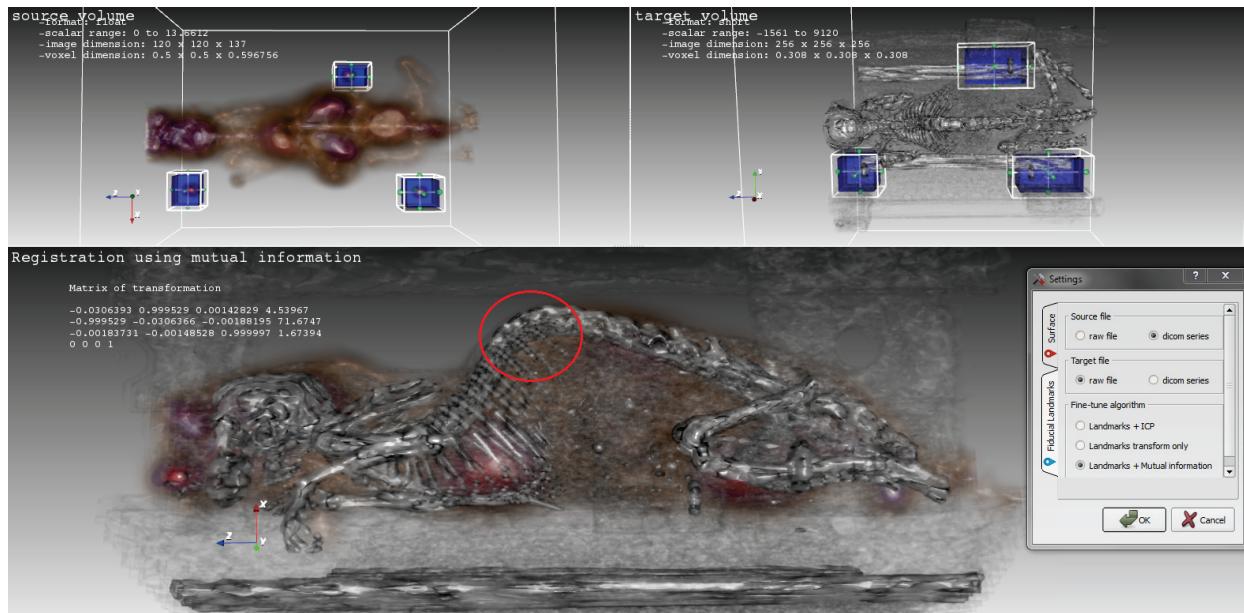


Figure 4.12 Résultat d'un recalage d'un volume de souris TEP (supérieur gauche) et d'un volume TDM (supérieur droit) à l'aide des marqueurs fiduciaux et avec un raffinement par l'algorithme de l'information mutuelle.

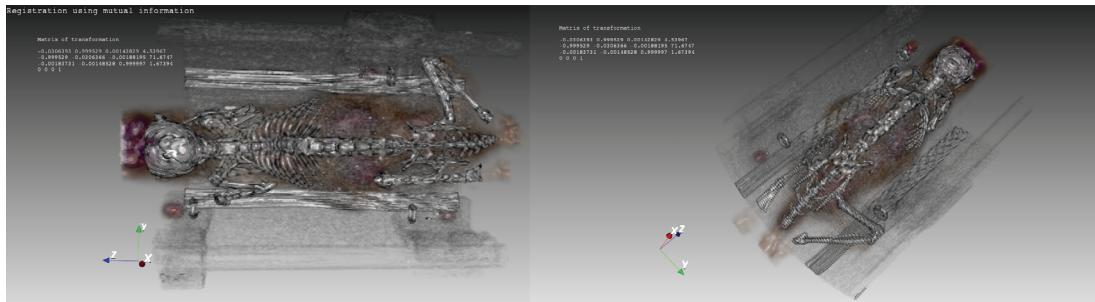


Figure 4.13 Vue du dessus (droite) et inclinée (gauche) du recalage TEP-TDM.

Mesure du temps de recalage TEP-TDM

Image	Résolution (pixels)
Source	$120 \times 120 \times 137$
Cible	$256 \times 256 \times 256$
Temps du processus : 1min et 30 sec (100 itérations)	

Tableau 4.4 Mesure expérimentale du temps d'exécution du code informatique pour un recalage TEP-TDM avec information mutuelle.

4.2.2 Recalage TEP avec IRM

Comme l'image IRM (à droite de la figure 4.15) a une plus grande résolution spatiale que l'image TEP (gauche de la figure), celle-ci sera représentée comme l'image cible. Le recalage

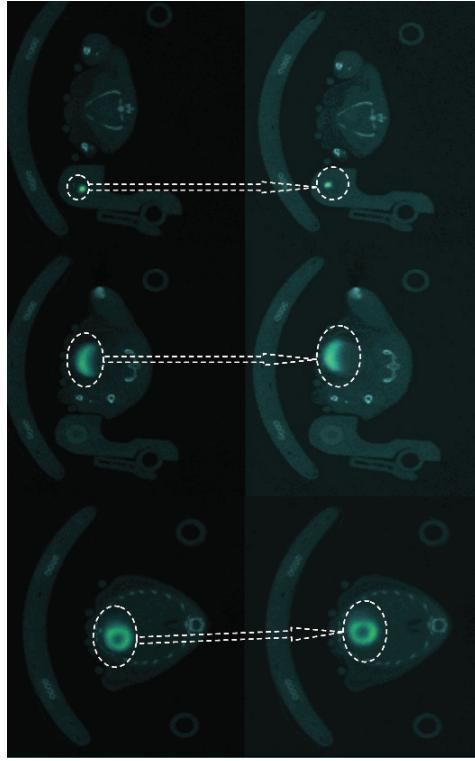


Figure 4.14 Vue de trois coupes 2D du résultat du recalage TEP-TDM avec seulement marqueurs fiduciaux (droite) et raffinement par information mutuelle (gauche).

est effectué suite à l'extraction du volume des marqueurs fiduciaux ainsi que l'application de l'algorithme de l'information mutuelle. Le résultat est présenté au bas de la figure et on remarque que l'image TEP semble bien être alignée sur l'image IRM. La figure 4.16

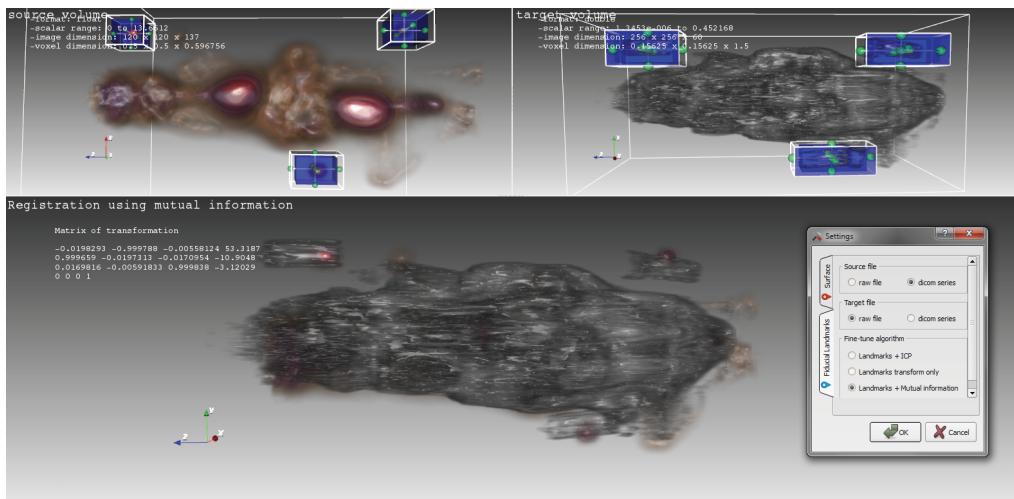


Figure 4.15 Résultat d'un recalage d'un volume de souris TEP (supérieur gauche) et d'un volume IRM (supérieur droit) à l'aide des marqueurs fiduciaux et raffinement par information mutuelle.

présente un recalage sans (droite) et avec (gauche) un raffinage par information mutuelle. On peut remarquer que les deux images sont presque semblables malgré le raffinement. Cela peut s'expliquer par le fait que seule la transformation rigide est considérée dans le projet. Alors on peut prétendre que l'animal a été déplacé d'une modalité à l'autre ayant pour conséquence d'appliquer une transformation non-linéaire.

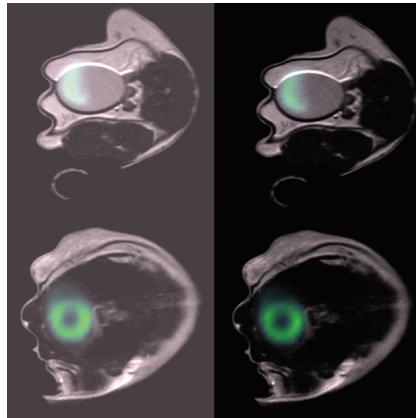


Figure 4.16 Vue de deux coupes 2D du résultat du recalage TEP-IRM avec seulement marqueurs fiduciaux (droite) et raffinement par information mutuelle (gauche).

Mesure du temps de recalage TEP-IRM

Image	Résolution (pixels)
Source	$120 \times 120 \times 137$
Cible	$256 \times 256 \times 60$
<hr/>	
Temps du processus :	1min (100 itérations)

Tableau 4.5 Mesure expérimentale du temps d'exécution du code informatique pour un recalage TEP-IRM avec information mutuelle.

4.3 Discussion

Le recalage de surface semble être satisfaisant malgré que les images profilométriques semblent ne pas constituer la totalité de la structure surfacique de la souris. Cela peut nuire au résultat et surtout lors de la mise en correspondance des points par l'algorithme d'ICP. En ce qui concerne le recalage à l'aide des marqueurs fiduciaux, il est préférable d'appliquer l'algorithme de l'information mutuelle afin de raffiner les résultats si les images source et cible sont de différentes modalités. Il faut souligner que l'algorithme de l'information mutuelle est appliqué sur toute l'image source et cible. Donc, il tient forcément compte de la présence des marqueurs fiduciaux et cela peut nuire au résultat final. Une des suggestions

afin d'améliorer les résultats est de soustraire les images source et cible par les mêmes images mais sans l'animal afin de pouvoir appliquer le calcul de l'information mutuelle que sur l'animal et non avec son environnement.

CHAPITRE 5

CONCLUSION

5.1 Sommaire

Le but de ce projet consistait à développer un programme informatique permettant de faire en premier lieu un recalage d'images tomographiques d'un petit animal dans un environnement 3D à l'aide de marqueurs fiduciaux. En seconde étape, le programme devait permettre de faire un recalage de surface de ces images tomographiques avec des structures profilométriques du même animal. Dans le présent document, le chapitre 2 décrit en détail la nature des modalités d'imagerie TEP, TDM et IRM ainsi que quelques notions mathématiques liées au recalage d'image. Le chapitre 3 décrit la conception informatique permettant d'exploiter les librairies VTK et ITK spécialisées dans la segmentation, le recalage et la visualisation d'image. Le chapitre 4 présente les résultats obtenus suite au recalage des images tomographiques à l'aide des marqueurs fiduciaux et le recalage superficiel avec des images profilométriques du laboratoire.

Le projet répond parfaitement aux objectifs fixés dans l'introduction, cependant les résultats du recalage dépendent fortement de la qualité de l'image source et de l'image cible. Malheureusement, le programme informatique ne permet pas de faire du filtrage de bruit ou autres traitements d'images complexes. Cependant, il permet avec succès d'accomplir les points suivants :

- Un recalage de deux surfaces de mêmes ou de différentes tailles.
- Un recalage de surface entre une image profilométrique avec une image tomographique (TEP, TDM, IRM ou TOD).
- Un recalage d'images tomographiques 3D par un algorithme basé sur l'intensité des voxels (maximisation de l'information mutuelle ou l'erreur quadratique) sans l'utilisation des marqueurs fiduciaux.
- Un recalage d'images tomographiques 3D à l'aide des marqueurs fiduciaux raffiné ou pas par un algorithme basé sur l'intensité des voxels.
- Permettre de faire du prétraitement et de la segmentation de base sur des profils ou des volumes avant de faire un recalage.

Bien que l'imagerie TOD n'a pas été testée dans ce projet, on prévoit comme même que le programme informatique n'aura aucun problème à afficher l'image et que l'utilisateur n'aura pas de difficulté à identifier les marqueurs fiduciaux.

5.2 Perspectives futures

Il est possible d'améliorer le programme informatique en exploitant en profondeur les librairies ITK et VTK, ou utiliser d'autres librairies spécialisées dans le traitement de données comme les librairies *Point Cloud* (PCL). En raison du manque de temps, on suggère quelques propositions qui pourraient être réalisées à cours ou à long terme.

- Développer un moyen d'interaction permettant à l'utilisateur d'extraire des volumes d'intérêt pour le recalage de n'importe quelles géométries au lieu d'utiliser les dimensions d'un *BoxWidget*.
- Programmer un moyen automatique de détection des marqueurs fiduciaux de n'importe quelle modalité d'imagerie.
- Programmer un moyen d'extraire l'animal et d'éliminer les marqueurs fiduciaux lors des algorithmes de raffinement.
- Programmer un algorithme de recalage pour des transformations non rigides si le sujet se déplace d'une modalité à une autre.
- Programmer un moyen plus poussé pour le pré-traitement (nettoyage) d'image profilométriques.

ANNEXE A

Exemple de code C++ pour afficher un maillage

```
1 #include "vtkPolyData.h"
2 #include "vtkPolyDataMapper.h"
3 #include "vtkActor.h"
4 #include "vtkInteractorStyleTrackballCamera.h"
5 #include "vtkRenderWindow.h"
6 #include "vtkRenderer.h"
7
8 {
9 vtkSmartPointer<vtkPolyData> maillage = vtkSmartPointer<vtkPolyData>::
10 New();
11 maillage->SetInput("adresse du fichier .obj");
12
12 vtkSmartPointer<vtkPolyDataMapper> Mapper = vtkSmartPointer<
13     vtkPolyDataMapper>::New();
13 Mapper->SetInput(maillage->GetOutput());
14
15 vtkSmartPointer<vtkActor> Actor = vtkSmartPointer<vtkActor>::New();
16 Actor->SetMapper(Mapper);
17
18 // Appliquer un interactor qui utilise le controle de la camera en mode
19 // TrackBall
20 vtkSmartPointer<vtkInteractorStyleTrackballCamera> style =
21     vtkSmartPointer<vtkInteractorStyleTrackballCamera>::New();
22 vtkSmartPointer<vtkRenderWindow> renderWindow = vtkSmartPointer<
23     vtkRenderWindow>::New();
24 vtkSmartPointer<vtkRenderer> renderer = vtkSmartPointer<vtkRenderer>::
25 New();
26 renderWindow->AddRenderer(renderer);
27 renderer->AddActor(Actor);
28 renderer->SetBackground(0.9,0.9,0.9);
29 renderer->GradientBackgroundOn();
30 // Position de la camera
31 renderer->GetActiveCamera()->Azimuth(30);
32 renderer->GetActiveCamera()->Elevation(30);
33 renderer->ResetCamera();
34 // Affichage en utilisant QVTKWidget (Qt/VTK)
35 qwin->SetRenderWindow(renderWindow);
36 qwin->GetInteractor()->GetInteractorStyle()->SetCurrentRenderer(renderer
    );
37 qwin->GetInteractor()->SetInteractorStyle(style);
38 qwin->GetRenderWindow()->Render();
39 qwin->update();
40 }
```


ANNEXE B

Programmation de la transformation rigide à l'aide de VTK

```
1 #include "vtkLandmarkTransform.h"
2 #include "vtkPoints.h"
3 #include "vtkMath.h"
4 #include "vtkMatrix4x4.h"
5 struct Frame
6 {
7     Frame(double p1[3], double p2[3], double p3[3]){
8         this->setOrigin(p1);
9         this->setXDirection(p2);
10        this->setYDirection(p3);
11        this->setZDirection();
12    }
13    double origin[3];
14    double XDirection[3];
15    double YDirection[3];
16    double ZDirection[3];
17    // Position de l'origine
18    void setOrigin(double o[3]){
19        this->origin[0] = o[0];
20        this->origin[1] = o[1];
21        this->origin[2] = o[2];
22    }
23    // Calcul du vecteur unitaire le long de l'axe x
24    void setXDirection(double direction[3]){
25        double a[3];
26        vtkMath::Subtract(direction, this->origin, a);
27        vtkMath::Normalize(a);
28        this->XDirection[0] = a[0];
29        this->XDirection[1] = a[1];
30        this->XDirection[2] = a[2];
31    }
32    // Calcul du vecteur unitaire le long de l'axe y
33    void setYDirection(double direction[3]){
34        double a[3];
35        vtkMath::Subtract(direction, this->origin, a);
36        double b;
37        vtkMath::Dot(this->XDirection, a, b);
38        vtkMath::MultiplyScalar(this->XDirection, b);
39        double c[3];
40        vtkMath::Subtract(a, this->XDirection, c)
41        vtkMath::Normalize(c);
42        this->YDirection[0] = c[0];
43        this->YDirection[1] = c[1];
44        this->YDirection[2] = c[2];
45    }
46    // Calcul du vecteur unitaire z
47    void setZDirection(){
48        double a[3];
49        vtkMath::Cross(this->XDirection, this->YDirection, a);
```

ANNEXE B. PROGRAMMATION DE LA TRANSFORMATION RIGIDE À L'AIDE
74 DE VTK

```
50     vtkMath::Normalize(a);
51     this->ZDirection[0] = a[0];
52     this->ZDirection[1] = a[1];
53     this->ZDirection[2] = a[2];
54 }
55 };
56 void Transformation_Rigide(Frame sourceFrame, Frame targetFrame,
57     vtkTransform* transform)
58 {
59     // Cette fonction permet d'identifier les triades a partir des points
60     // P1, P2 et P3
61     vtkSmartPointer<vtkLandmarkTransform> landmarkTransform =
62         vtkSmartPointer<vtkLandmarkTransform>::New();
63     // Calcul de la triade de l'image source
64     vtkSmartPointer<vtkPoints> sourcePoints = vtkSmartPointer<vtkPoints>::
65         New();
66     sourcePoints->InsertNextPoint(sourceFrame.origin);
67     double sX[3];
68     vtkMath::Add(sourceFrame.origin,sourceFrame.XDirection,sX);
69     sourcePoints->InsertNextPoint(sX);
70     double sY[3];
71     vtkMath::Add(sourceFrame.origin,sourceFrame.YDirection,sY);
72     sourcePoints->InsertNextPoint(sY);
73     double sZ[3];
74     vtkMath::Add(sourceFrame.origin,sourceFrame.ZDirection,sZ);
75     sourcePoints->InsertNextPoint(sZ);
76     // Calcul de la triade de l'image cible
77     vtkSmartPointer<vtkPoints> targetPoints = vtkSmartPointer<vtkPoints>::
78         New();
79     targetPoints->InsertNextPoint(targetFrame.origin);
80     double tX[3];
81     vtkMath::Add(targetFrame.origin,targetFrame.XDirection,tX);
82     targetPoints->InsertNextPoint(tX);
83     double tY[3];
84     vtkMath::Add(targetFrame.origin,targetFrame.YDirection,tY);
85     targetPoints->InsertNextPoint(tY);
86     double tZ[3];
87     vtkMath::Add(targetFrame.origin,targetFrame.ZDirection,tZ);
88     targetPoints->InsertNextPoint(tZ);
89     // Calcul de la matrice de passage de l'image source a l'image cible
90     landmarkTransform->SetSourceLandmarks(sourcePoints);
91     landmarkTransform->SetTargetLandmarks(targetPoints);
92     landmarkTransform->SetModeToRigidBody();
93     landmarkTransform->Update();
94
95     vtkMatrix4x4* M = landmarkTransform->GetMatrix();
96     transform->SetMatrix(M);
97 }
```

ANNEXE C

Programmation de l'algorithme de l'information mutuelle à l'aide d'ITK

Le présent code permet d'obtenir la matrice de transformation de l'information mutuelle par le biais de la classe *vtkMatrix4x4*. La fonction *MutualInformation* prend en paramètre le lien de l'image source, le lien de l'image cible, le nombre d'itération ainsi que les types de pixels des deux images.

```
1 template<typename pixelType1, typename pixelType2>
2 vtkSmartPointer<vtkMatrix4x4> MutualInformation(const string source,
3   const string target,int iteration,pixelType1,pixelType2)
4 {
5   typedef itk::Image< pixelType1, 3 > FixedImageType;
6   typedef itk::Image< pixelType2, 3 > MovingImageType;
7
8   typedef itk::VersorRigid3DTransform< double > TransformType;
9   typedef itk::VersorRigid3DTransformOptimizer
10    OptimizerType;
11   typedef itk::MutualInformationImageToImageMetric< FixedImageType ,
12     MovingImageType > MetricType;
13   typedef itk::LinearInterpolateImageFunction< MovingImageType , double >
14    InterpolatorType;
15   typedef itk::ImageRegistrationMethod< FixedImageType , MovingImageType
16    > RegistrationType;
17
18   MetricType::Pointer metric = MetricType::New();
19   OptimizerType::Pointer optimizer = OptimizerType::New();
20   InterpolatorType::Pointer interpolator = InterpolatorType::New();
21   TransformType::Pointer transform = TransformType::New();
22
23   RegistrationType::Pointer registration = RegistrationType::New();
24   registration->SetMetric( metric );
25   registration->SetOptimizer( optimizer );
26   registration->SetInterpolator( interpolator );
27   registration->SetTransform( transform );
28
29   metric->SetFixedImageStandardDeviation(4);
30   metric->SetMovingImageStandardDeviation(4);
31
32   typedef itk::ImageFileReader<FixedImageType> FixedImageReaderType;
33   typedef itk::ImageFileReader<MovingImageType> MovingImageReaderType;
34
35   FixedImageReaderType::Pointer fixedImageReader =
36     FixedImageReaderType::New();
37   MovingImageReaderType::Pointer movingImageReader =
38     MovingImageReaderType::New();
39   fixedImageReader->SetFileName(source.c_str());
40   movingImageReader->SetFileName(target.c_str());
41
42   registration->SetFixedImage(fixedImageReader->GetOutput());
43   registration->SetMovingImage( movingImageReader->GetOutput() );
44 }
```

ANNEXE C. PROGRAMMATION DE L'ALGORITHME DE L'INFORMATION
MUTUELLE À L'AIDE D'ITK

```
37
38     try{
39         fixedImageReader->Update();
40     }catch(itk::ExceptionObject & excp){
41         std::cerr << excp << std::endl;
42     }
43     registration->SetFixedImageRegion(fixedImageReader->GetOutput()->
44                                         GetBufferedRegion() );
45
46     typedef itk::CenteredTransformInitializer<TransformType , FixedImageType
47             , MovingImageType> TransformInitializerType;
48     TransformInitializerType::Pointer initializer =
49         TransformInitializerType::New();
50
51     initializer->SetTransform(transform);
52     initializer->SetFixedImage(fixedImageReader->GetOutput());
53     initializer->SetMovingImage(movingImageReader->GetOutput());
54     initializer->MomentsOn();
55     initializer->InitializeTransform();
56
57     typedef TransformType::VersorType VersorType;
58     typedef VersorType::VectorType VectorType;
59     VersorType rotation;
60     VectorType axis;
61     axis[0] = 0.0;
62     axis[1] = 0.0;
63     axis[2] = 1.0;
64     const double angle = 0;
65     rotation.Set( axis, angle );
66     transform->SetRotation( rotation );
67
68     registration->SetInitialTransformParameters( transform->GetParameters
69         () );
70
71     typedef OptimizerType::ScalesType OptimizerScalesType;
72     OptimizerScalesType optimizerScales( transform->GetNumberOfParameters
73         () );
74     const double translationScale = 1.0 / 1000.0;
75     optimizerScales[0] = 1.0;
76     optimizerScales[1] = 1.0;
77     optimizerScales[2] = 1.0;
78     optimizerScales[3] = translationScale;
79     optimizerScales[4] = translationScale;
80     optimizerScales[5] = translationScale;
81     optimizer->SetScales( optimizerScales );
82     optimizer->SetMaximumStepLength( 0.2000 );
83     optimizer->SetMinimumStepLength( 0.00000000000000000000000000000001 );
84     optimizer->SetNumberOfIterations( iteration );
85
86     CommandIterationUpdate::Pointer observer = CommandIterationUpdate::New
87         ();
88     observer->GetTomo( this->in );
89     optimizer->AddObserver(itk::IterationEvent(), observer);
90
91     try{
92         registration->Update();
93         std::cout << "Optimizer stop condition: " << registration->
94             GetOptimizer()->GetStopConditionDescription() << std::endl;
95     }
```

```

89     catch( itk::ExceptionObject & err ){
90         std::cerr << "ExceptionObject caught !" << std::endl;
91         std::cerr << err << std::endl;
92     }
93
94     OptimizerType::ParametersType finalParameters = registration->
95         GetLastTransformParameters();
96     const double versorX                      = finalParameters[0];
97     const double versorY                      = finalParameters[1];
98     const double versorZ                      = finalParameters[2];
99     const double finalTranslationX          = finalParameters[3];
100    const double finalTranslationY         = finalParameters[4];
101    const double finalTranslationZ         = finalParameters[5];
102    const unsigned int numberofIterations = optimizer->GetCurrentIteration
103        ();
104    const double bestValue = optimizer->GetValue();
105
106    std::cout << std::endl << std::endl;
107    std::cout << " Result = " << std::endl;
108    std::cout << " versor X      = " << versorX << std::endl;
109    std::cout << " versor Y      = " << versorY << std::endl;
110    std::cout << " versor Z      = " << versorZ << std::endl;
111    std::cout << " Translation X = " << finalTranslationX << std::endl;
112    std::cout << " Translation Y = " << finalTranslationY << std::endl;
113    std::cout << " Translation Z = " << finalTranslationZ << std::endl;
114    std::cout << " Iterations      = " << numberofIterations << std::endl;
115    std::cout << " Metric value    = " << bestValue           << std::endl;
116
117    transform->SetParameters( finalParameters );
118    TransformType::MatrixType matrix = transform->GetMatrix();
119    TransformType::OffsetType offset = transform->GetOffset();
120    std::cout << "Matrix = " << std::endl << matrix << std::endl;
121    std::cout << "Offset = " << std::endl << offset << std::endl;
122
123    vtkSmartPointer<vtkMatrix4x4> matrixOfTransformation = vtkSmartPointer
124        <vtkMatrix4x4>::New();
125    for(unsigned int i=0;i<3;i++){
126        for(unsigned int j=0;j<3;j++){
127            matrixOfTransformation->SetElement(i,j,matrix(i,j));
128        }
129    }
130    matrixOfTransformation->SetElement(0,3,finalTranslationX - offset[0]);
131    matrixOfTransformation->SetElement(1,3,finalTranslationY - offset[1]);
132    matrixOfTransformation->SetElement(2,3,finalTranslationZ - offset[2]);
133    matrixOfTransformation->SetElement(3,0,0);
134    matrixOfTransformation->SetElement(3,1,0);
135    matrixOfTransformation->SetElement(3,2,0);
136    matrixOfTransformation->SetElement(3,3,1);
137    /*Obtention de la matrice finale de transformation par information
       mutuelle*/
138    return matrixOfTransformation;
139 }
140 /*Classe permettant le controle du processus iterative de la
   registration*/
141 class CommandIterationUpdate : public itk::Command
142 {
143 public:
144     typedef CommandIterationUpdate Self;
145     typedef itk::Command Superclass;

```

```

143  typedef itk::SmartPointer<Self> Pointer;
144  itkNewMacro(Self);
145  protected:
146  CommandIterationUpdate();
147  public:
148  typedef itk::VersorRigid3DTransformOptimizer OptimizerType;
149  typedef const OptimizerType *OptimizerPointer;
150
151  void Execute(itk::Object *caller, const itk::EventObject & event){
152  Execute((const itk::Object *)caller, event);
153 }
154
155  void Execute(const itk::Object *object, const itk::EventObject & event)
156  {
157  OptimizerPointer optimizer = dynamic_cast< OptimizerPointer >( object
158  );
159  if(!itk::IterationEvent().CheckEvent(&event)){
160  return;
161  }
162  cout<<optimizer->GetCurrentIteration()<<"      ";
163  cout<<optimizer->GetValue()<<"      ";
164  cout<<optimizer->GetCurrentPosition()<<endl;
165  double r = optimizer->GetCurrentIteration();
166  double g = optimizer->GetNumberOfIterations();
167  double ration = r/g;
168  this->TomoProgress->SetProgressBar(ration);
169
170  void GetTomo(TomoRegistration* a){
171  this->TomoProgress = a;
172 }
173 private:
174 TomoRegistration* TomoProgress;
175 };

```

ANNEXE D

La métrique du logiciel

Tableau D.1 La métrique du logiciel.

Métrique	Valeur
Nombre de fichiers	47
Nombre de lignes de code programmées	8448
Nombre de déclarations	5659
Porcentage de commentaires	0.4%
Définition de classes	33
Nombre de méthodes moyen par classe	7.07
Déclaration moyenne par méthode	17.2
Complexité moyenne	2.46
Complexité maximale	30
L'inheritance moyenne par classe	1.63
L'inheritance maximale	8

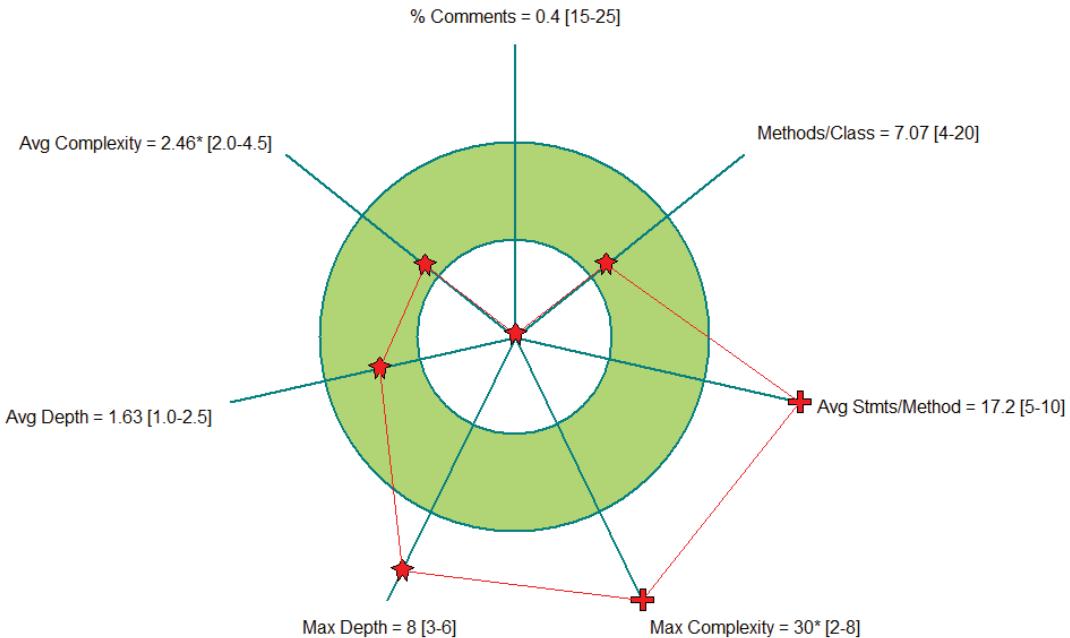


Figure D.1 Représentation de la métrique par le diagramme de Kiviat.

Le tableau D.1 présente la métrique du programme informatique du projet de recherche et la figure D.1 présente le diagramme de Kiviat obtenu à l'aide du logiciel *SourceMonitor* [SourceMonitor, s. d.]. Ce diagramme affiche les valeurs métriques obtenues du code du projet ainsi que les valeurs, entre crochés, auxquelles on s'attend à respecter et qui sont conformes à l'état de l'art de conception d'un logiciel informatique. On peut remarquer

que le pourcentage de lignes de commentaires (*% Comments*) est très faible. La complexité moyenne (*Avg Complexity*), l'inheritance moyenne par classe (*Avg Depth*) ainsi le nombre de méthode moyen par classe (*Methods/Class*) repectent la norme de conception. Cependant pour les valeurs d'inheritance maximale (*Max Depth*), la complexité maximale (*Max Complexity*) et le nombre moyen de déclaration par méthode (*Avg Stmt/Method*) sont à l'extérieurs de la zone verte du diagramme.

LISTE DES RÉFÉRENCES

- Arun, K. S., Huang, T. S. et Blostein, S. D. (1987). Least-squares fitting of two 3D point sets. *IEEE Trans. Pattern Anal. Mach. Intell.*, volume 9, numéro 5, p. 698–700.
- Barillot, C. (1999). *Fusion de Données et Imagerie 3D en Médecine*. Thèse de doctorat, Université de Rennes 1, Rennes, France, 136 p.
- Besl, P. J. et McKay, N. D. (1992). A method for registration of 3-D shapes. *IEEE, Transactions on pattern analysis and machine intelligence*, volume 14, numéro 2, p. 239–256.
- Boas, D., Brooks, D., Miller, E., DiMarzio, C., Kilmer, M., Gaudette, R. et Zhang, Q. (2001). Imaging the body with diffuse optical tomography. *IEEE, Signal Processing Magazine*, volume 18, numéro 6, p. 57–75.
- Boffety, M. (2010). *Étude quantitative de la tomographie optique diffuse de luminescence. Application à la localisation de sources en imagerie moléculaire*. Thèse de doctorat, École Centrale Paris, Paris, France, 136 p.
- Brooks, R. A. et Chiro, G. D. (1975). Theory of Image Reconstruction in Computed Tomography. *US National Library of Medicine National Institutes of Health*, volume 117, numéro 3.
- Cover, T. M. et Thomas, J. A. (1991). *Elements of Information Theory*. Wiley-Interscience, 576 p.
- Desgrez, A., Bittoun, J., Idy-Peretti, I. et Kellersohn, C. (1994). *Bases physiques de l'IRM*, volume 2. Paris, Masson, Englewood Cliffs, NJ, USA, 106 p.
- Dod HPC Modernization program (s. d.). *DAAC Data Analysis and Assessment Center*. http://daac.hpc.mil/gettingStarted/Marching_Cubes.html (page consultée le 1 2014).
- Farrell, J. L. et Stuelpnagel, J. C. (1966). A least squares estimate of satellite attitude. *SIAM Rev.*, volume 8, numéro 3, p. 384–386.
- Fitzpatrick, J. M. et Sonka, M. (2009). *Handbook of Medical Imaging, Volume 2. Medical Image Processing and Analysis*. SPIE Publications, 1108 p.
- Fonseca, L. M. et Manjunath, B. (1996). Registration Techniques for Multisensor Remotely Sensed Imagery. *American Society for Photogrammetry and Remote Sensing*, volume 62, numéro 9, p. 1049–1056.
- Goldman, L. W. (2007). Principles of CT and CT technology. *Journal of Nuclear Medicine Technology*, volume 35, numéro 3, p. 115–128.
- Gonzalez, R. C. et Woods, R. E. (2007). *Digital Image Processing*. Prentice Hall, 976 p.

- Groller, E., Hadwiger, M., Buhler, K. et Beyer, J. (2009). *GPU-based Multi-Volume Rendering of Complex Data in Neuroscience and Neurosurgery* (Rapport technique). Vienna University of Technology, 131 p.
- Gupta, S., Sengupta, K. et A.Kassim, A. (2002). Compression of Dynamic 3D Geometry Data Using Iterative Closest Point Algorithm. *Computer Vision and Image Understanding*, volume 87, numéro 3, p. 116–130.
- Hervé, L., Koenig, A., Silva, A. D., Berger, M., Boutet, J., Dinten, J., Peltié, P. et Rizo, P. (2007). Tomographie optique diffuse de fluorescence de la souris. *GRETSI, Groupe d'Etudes du Traitement du Signal et des Images*, p. 113–116.
- Hill, D. L., Hawkes, D. J., Harrison, N. A. et Ruff, C. F. (1993). A strategy for automated multimodality image registration incorporating anatomical knowledge and imager characteristics. *Lecture Notes in Computer Science*, p. 182–196.
- Ibanez, L., Schroeder, W., Ng, L., Cates, J. et the Insight Software Consortium (2005). *The ITK Software Guide*, volume 2. Kitware inc, 836 p.
- Johnson, A. E. et Hebert, M. (1997). Surface Registration by Matching Oriented Points. *Proceedings of the International Conference on Recent Advances in 3-D Digital Imaging and Modeling*, p. 121–128.
- Kak, A. C. et Slaney, M. (2001). *Principles of Computerized Tomographic Imaging*, volume 33. Society for Industrial and Applied Mathematics, West Lafayette, Indiana, 327 p.
- Kitware (s. d.). *Visualization Toolkit*. <http://www.vtk.org/VTK/resources/software.html> (page consultée le 1 janvier 2014).
- Kitware (s. d.). *Insight Toolkit*. <http://www.itk.org/ITK/resources/software.html> (page consultée le 1 janvier 2014).
- Kitware (s. d.). *CMake*. <http://www.cmake.org/cmake/project/about.html> (page consultée le 1 janvier 2014).
- Kostelec, P. J. et Periaswamy, S. (2003). Image Registration for MRI. *MSRI, Modern Signal Processing*, volume 46, p. 151–184.
- Lorensen, W. E. et Cline, H. E. (1987). Marching Cubes : a high resolution 3D surface construction algorithm. *Computer Graphics*, volume 21, numéro 4, p. 163–169.
- Maes, F., Collignon, A., Vandermeulen, D., Marchal, G. et Suetens, P. (1997). Multimodality Image Registration by Maximization of Mutual Information. *IEEE, Transactions On Medical Imaging*, volume 16, numéro 2, p. 187–198.
- Maintz, J. B. A. et Viergever, M. A. (1998). A survey of medical image registration. *Medical Image Analysis*, volume 2, numéro 1, p. 1–36.
- Maurer Jr, C. R., Aboutanos, G. B., Dawant, B. M., Maciunas, R. J. et Fitzpatrick, J. M. (1996). Registration of 3-D Images Using Weighted Geometrical Features. *IEEE Transactions on Medical Imaging*, volume 15, numéro 6, p. 836–849.

- Nahrendorf, M., Keliher, E., Marinelli, B., Waterman, P., Feruglio, P. F., Fexon, L., Pivovarov, M., Swirski, F. K., Pittet, M. J., Vinegoni, C. et Weissleder, R. (2010). Hybrid PET-optical imaging using targeted probes. *Proceedings of the National Academy of Sciences of the United States of America*, volume 107, numéro 17, p. 7910–7915.
- Noblet, V. (2006). *Recalage non rigide d'images cérébrales 3D avec contrainte de conservation de la topologie*. Thèse de doctorat, Université Louis Pasteur, Strasbourg, France, 228 p.
- Qt (s. d.). *Qt Project*. <http://qt-project.org/downloads> (page consultée le 1 janvier 2014).
- Romans, L. E. (2010). *Computed Tomography for Technologists : A Comprehensive Text*. Lippincott Williams & Wilkins, 400 p.
- Saha, G. B. (2010). *Basics of PET Imaging : Physics, Chemistry, and Regulations*, volume 2. Springer, 314 p.
- Schonemann, P. H. (1966). A generalized solution of the orthogonal Procrustes problem. *Psychometrika*, volume 31, numéro 1, p. 1–10.
- Schroeder, W. J., Martin, K. M. et Lorensen, W. E. (1996). The design and implementation of an object-oriented toolkit for 3D graphics and visualization. *IEEE, Computer Society Press*, p. 93–ff.
- SourceMonitor (s. d.). *Campwood Software*. <http://www.campwoodsw.com/sm20.html> (page consultée le 1 janvier 2014).
- Studholme, C., Hill, D. L. et Hawkes, D. J. (1999). An overlap invariant entropy measure of 3D medical image alignment. *Pattern Recognition*, volume 32, numéro 1, p. 71–86.
- Townsend, D. (2004). Physical principles and technology of clinical PET imaging. *Annals-Academy of Medicine Singapore*, volume 33, numéro 2, p. 133–145.
- Umeyama, S. (1991). Least-Squares Estimation of Transformation Parameters Between Two Point Patterns. *IEEE, Transactions on pattern analysis and machine intelligence*, volume 13, numéro 4, p. 376–380.
- Université de Rennes (s. d.). *Laboratoire d'informatique médicale*. http://www.med.univ-rennes1.fr/cerf/edicerf/BASES/BA004_idx.html (page consultée le 1 janvier 2014).
- Unlu, M. B., Lin, Y., Birgul, O., Nalcioglu, O. et Gulsen, G. (2008). Simultaneous in vivo dynamic magnetic resonance-diffuse optical tomography for small animal imaging. *Journal of Biomedical Optics*, volume 13, numéro 6, p. 060501–060501.
- Viola, P. A. (1995). *Alignement by Maximization of Mutual Inforamtion* (Rapport technique). Massachusetts Institute of Technology, 156 p.
- Viola, P. A., Schraudolph, N. N. et Sejnowski, T. J. (1996). Empirical Entropy Manipulation for Real-World Problems. *Advances in Neural Information Processing Systems*, volume 8, p. 851–857.

- Wang, M. Y., Fitzpatrick, J. M., et Maurer, C. R. (1995). Design of fiducials for accurate registration of CT and MR volume images. *Medical Imaging 1995 : Image Processing*, volume 2434, p. 96–108.
- Wang, M. Y., Maurer Jr, C. R., Fitzpatrick, J. M. et Maciunas, R. J. (1996). An automatic technique for finding and localizing externally attached markers in CT and MR volume images of the head. *IEEE Trans Biomed Engineering*, volume 43, numéro 6, p. 627–637.
- Wernick, M. N. et Aarsvold, J. N. (2004). *Emission Tomography : The Fundamentals of PET and SPECT*. Academic Press, 596 p.
- Xiao, T. (2012). *Co-registration of fluorescence diffuse optical tomography (fDOT) with Positron emission tomography(PET) and development of multi-angle fDOT*. Thèse de doctorat, Université Paris Sud, Paris, France, 148 p.
- Yao, J., Ruggeri, M. R., Taddei, P. et Sequeira, V. (2011). Robust surface registration using N-points approximate congruent sets. *EURASIP Journal on Advances in Signal Processing*, volume 2011, numéro 1, p. 1–22.

