Mise en place de l'imagerie Cerenkov 3D

Arnaud Bertrand

UNIVERSITÉ DE STRASBOURG

ÉCOLE DOCTORALE Physique et Chimie- Physique

IPHC/UMR7178

THÈSE présentée par :

Arnaud BERTRAND

soutenue le : **06 Novembre 2015**

pour obtenir le grade de : **Docteur de l'université de Strasbourg**

Discipline/ Spécialité : Physique

Mise en place de l'imagerie Cerenkov 3D

THÈSE dirigée par :

Mr LAQUERRIERE Patrice	Professeur, université de Strasbourg

RAPPORTEURS :

Mme SARDA-MANTEL Laure	Professeur, INSERM
Mr HECHT Frederic	Professeur, UPMC

AUTRES MEMBRES DU JURY :

Mr PRUD'HOMME Christophe	Professeur, IRMA

Remerciements

Je tiens à remercier en premier lieu David Brasse et Patrice Laquerriere pour m'avoir permis de rejoindre le groupe ImaBio qui fut pour moi un lieu convivial et très enrichissant dans de nombreuses disciplines.

Je souhaite remercier Bruno Jessel, Lionel Thomas et Patrice Marchand pour les journées, parfois compliquées, à réaliser de l'imagerie dans la salle d'expérimentation. Pour la partie mécanique, je remercie Jacques Wurtz et Virgile Baekert ainsi que pour leurs conseils divers.

Enfin, je remercie mes camarades Cécile Bopp, Harold Barquero, Benjamin Auer, Regina Rescigno et Yusuf Karakaya qui partage ou partageait mon bureau pour toutes les discussions, professionnelles ou non, que nous avons pu avoir au cours de ces 3 dernières années.

Abbréviations

CLI : Imagerie Cerenkov 2D (de l'anglais Cerenkov Luminescence Imaging)

CLT : Tomographie Cerenkov (de l'anglais Cerenkov Luminescence Tomography)

FLI : Imagerie de fluorescence 2D (de l'anglais Fluorescence Luminescence Imaging)

FLT : Tomographie de fluorescence (de l'anglais Fluorescence Luminescence Tomography)

BLI : Imagerie de bioluminescence 2D (de l'anglais BioLuminescence Imaging)

BLT : Tomographie de bioluminescence (de l'anglais BioLuminescence Tomography)

DOT : Tomodiffusion (de l'anglais Diffuse Optical Tomography)

IRM : Imagerie par Résonnance Magnétique

TDM : TomoDensitoMétrie

TEP : Tomographie par Emission de Positons

TEMP : Tomographie par Emission MonoPhotonique

FDG : FluoroDéoxyGlucose

ATP : Adénosine Tri-Phosphate

QDot : Quantum Dots

CYRCE : CYclotron pour la ReCherche et l'Enseignement

AMISSA : A Multimodality Imaging System for Small Animal

Table des matières

Introduction

L'imagerie moléculaire est un pendant de l'imagerie ayant pour but d'étudier les mécanismes biologiques chez le vivant. La recherche en biologie, le diagnostic ou encore le suivi thérapeutique sont des applications de l'imagerie moléculaire [1]. L'imagerie moléculaire se décline en un grand nombre de techniques d'imagerie, appelées modalités. Les ultrasons, l'IRM (Imagerie par Résonnance Magnétique), la TDM (TomoDensitoMètrie), les méthodes d'imagerie optique ou les méthodes d'imagerie nucléaire comme la TEP (Tomographie par Emission de Positons) et la TEMP (Tomographie par Emission MonoPhotonique) sont des exemples de modalités utilisées dans le cadre de l'imagerie moléculaire.

Parmi ces modalités, certaines sont utilisées chez l'homme en routine clinique (TDM, TEP, TEMP, IRM, US) comme en préclinique. Les études précliniques visent à développer des outils (traceurs, techniques d'imagerie, etc) pour l'imagerie clinique et permettent aussi de mener des études sur le vivant. L'imagerie optique, peu utilisée en clinique à cause de sa profondeur de pénétration dans les tissus limitée à quelques centimètres, est très utilisée pour des études précliniques sur des cellules et des petits animaux. L'imagerie de fluorescence (FLI, Fluorescence Luminescence Imaging), de bioluminescence (BLI, BioLuminescence Imaging) et leurs version tomographique (FLT, Fluorescence Luminescence Tomography, et BLT, BioLuminescence Tomography, respectivement) ou encore la tomographie optique diffuse (DOT) sont un exemple des modalités propres à l'imagerie préclinique sur petit animal. Parmi toutes ces modalités, certaines peuvent apporter une information anatomique ou fonctionnelle et se complète mutuellement. L'imagerie multimodale consiste à utiliser plusieurs modalités pour croiser les informations. Cette complémentarité peut se résumer à superposer les images mais il est aussi possible d'avoir un couplage entre les différentes techniques afin d'améliorer les images obtenues comme l'utilisation d'informations anatomiques *a priori*. Chacune de ces méthodes a ses atouts et ses contraintes (fig. 1, tableau 1). Le coût d'un examen, la fréquence des examens, le coût de l'appareil, les contraintes liées au sujet à imager, ce que l'on souhaite imager (organe/corps entier, taille, rendu 3D) sont autant de paramètres à considérer.

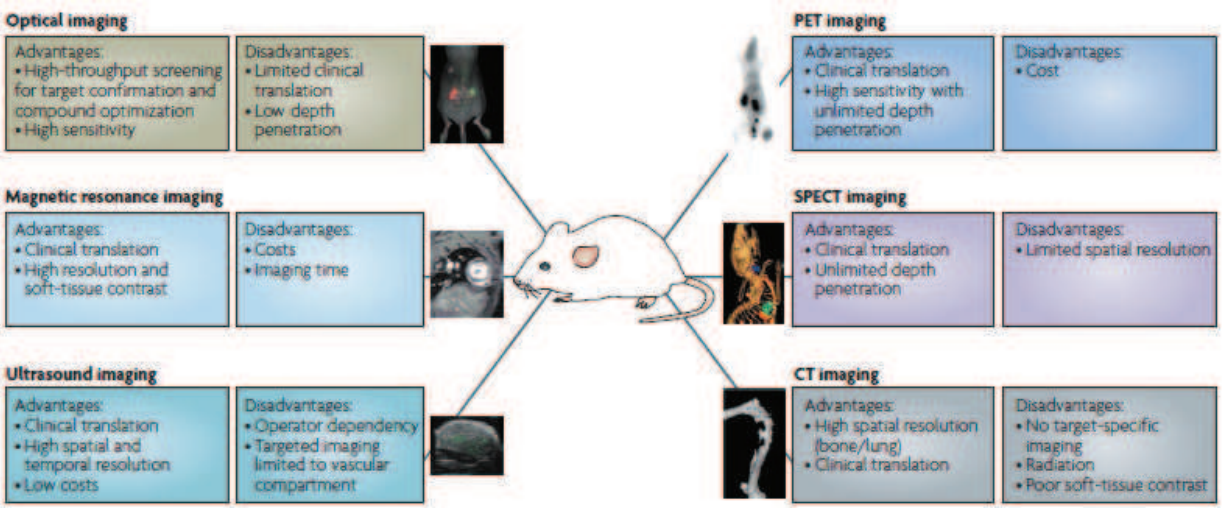

FIGURE 1 – Schéma présentant les modalités majeures d'imagerie moléculaire (tiré de [2]).

Modalité	Sonde(s)	Phénomène physique	Profondeur de pénétration	Résolution
IRM	Noyaux de spin non nul	Résonnance magnétique nucléaire	Pas de limite en clinique	10-100 μm
TDM	Tissus atténuants ou produit de contraste	Atténuation des rayons X	Pas de limite en clinique	50 μm
Ultrason	Changement de densité	Réflexion des ultrasons	Pas de limite en clinique	50 μm
TEP	Emetteur β^+	Radioactivité	Pas de limite en clinique	1-2 mm
TEMP	Emetteur γ	Radioactivité	Pas de limite en clinique	1-2 mm
FMT	Molécule fluorescente	Fluorescence	$<$ 10 cm	1 mm
BLT	Luciférase	Réaction chimique	Quelques cm	Quelques mm

TABLE 1 – Tableau présentant les modalités majeures d'imagerie moléculaire.

Les modalités TEP et TEMP permettent d'imager des isotopes radioactifs. Les isotopes émetteurs de rayonnement γ (TEMP) et β^+ (TEP) peuvent être imagés *in vivo*. Toutefois, aucune de ces techniques ne permet d'imager des isotopes émetteurs de β^-.

En 2009, les premières publications [3, 4] montrent que l'utilisation du rayonnement Cerenkov dans le cadre de l'imagerie moléculaire est possible. L'imagerie Cerenkov regroupe plusieurs modalités exploitant le rayonnement Cerenkov pour réaliser de l'imagerie moléculaire [5]. Le rayonnement Cerenkov est le signal optique produit par l'effet Cerenkov qui survient lorsqu'une particule chargée se déplace dans un milieu avec une vitesse supérieure à celle de la lumière dans ce milieu. Ainsi, certains isotopes émetteurs de rayonnement β (β^+ ou β^-) peuvent être imagés avec l'imagerie Cerenkov. L'imagerie Cerenkov est actuellement la seule modalité d'imagerie à pouvoir imager des isotopes émetteurs de β^- *in vivo*.

Toutefois, le signal optique interagit fortement avec les tissus biologiques et la reconstruction tomographique est délicate.

Ma thèse consiste à développer l'imagerie Cerenkov 3D dans le cadre de l'imagerie moléculaire chez la souris au sein du groupe ImaBio. L'objectif est d'utiliser des images Cerenkov avec l'information spectrale du rayonnement Cerenkov ainsi que 4 vues optiques de l'objet afin de reconstruire la distribution du traceur. Mon travail implique la mise en place des protocoles expérimentaux et du processus de reconstruction liés à la tomographie Cerenkov. Dans un premier temps, j'expliquerai la physique liée à l'imagerie Cerenkov ainsi que les méthodes d'imagerie utilisant le rayonnement Cerenkov pour l'imagerie moléculaire. Je décrirai ensuite la plateforme d'imagerie AMISSA (A Multimodality Imaging System for Small Animal) présente au sein du groupe ImaBio. Puis je parlerai des problématiques liées à la reconstruction 3D d'une source optique dans un organisme vivant et des méthodes de reconstruction tomographique. Je présenterai et discuterai les résultats obtenus concernant l'imagerie Cerenkov 2D et l'imagerie Cerenkov 3D. Je finirai par une conclusion de ce travail et sur des perspectives.

Chapitre 1

Imagerie Cerenkov

L'effet Cerenkov a été découvert en 1934 par Pavel A. Cerenkov [6]. L'imagerie Cerenkov définit l'ensemble des applications de l'effet Cerenkov dans le cadre de l'imagerie moléculaire. L'effet Cerenkov se caractérise par la production de photons optiques lors du déplacement de particules chargées. Les particules chargées émises lors de la désintégration d'isotopes sont susceptibles d'émettre un rayonnement Cerenkov. Les isotopes émetteurs de rayonnement β sont à la base de l'imagerie Cerenkov et font appel aux notions d'imagerie nucléaire. D'autre part, le rayonnement Cerenkov se compose de photons optiques, ce qui fait que l'enregistremet du signal et son traitement sont du domaine de l'imagerie optique. Le caractère hybride de l'imagerie Cerenkov entre imagerie nucléaire et imagerie optique est décrit dans cette partie ainsi que les applications de cette modalité dans le cadre de l'imagerie moléculaire.

1.1 Effet Cerenkov

L'effet Cerenkov se produit lorsqu'une particule chargée se déplace avec une vitesse supérieure à la vitesse de phase de la lumière. La vitesse de la lumière dépend du milieu dans lequel la lumière se propage. Dans le vide, la vitesse de la lumière vaut c. Dans un matériau, l'indice optique n décrit comment la vitesse de la lumière va être modifiée dans le matériau. La vitesse de la lumière dans un matériau devient alors v=c/n. Pour une particule chargée donnée et en fonction du milieu, il existe une énergie cinétique minimale E telle que l'effet Cerenkov est possible. Ce seuil dépend uniquement de l'indice du milieu n et de la masse de la particule m.

$$1 \geq \frac{1}{\beta n} \tag{1.1}$$

Avec :

$$\beta = \sqrt{1 - \left(\frac{1}{\frac{E}{mc^2} + 1}\right)^2} \tag{1.2}$$

D'où :

$$E = mc^2 \left(\frac{1}{\sqrt{1 - \frac{1}{n^2}}} - 1 \right) \tag{1.3}$$

Particule	Seuil Cerenkov dans l'eau (n = 1.33)
Electron	264 keV
Positon	264 keV
Proton	484 MeV
Noyau d'hélium	1.92 GeV

TABLE 1.1 – Seuil d'émission du rayonnement Cerenkov dans l'eau en fonction de la particule

Une émission de photons est observée lorsque l'effet Cerenkov survient. Physiquement, c'est le milieu de propagation qui émet les photons suite aux changements rapides de polarisation causés par le déplacement de la particule. L'émission de photons cause des interférences destructives dans le cas où la particule ne dépasse pas le seuil mais peut devenir constructive si le seuil en énergie est dépassé (tab. 1.1). Le nombre de photons émis peut être calculé en utilisant la formule de Frank et Tamm [7]. Cette équation décrit le nombre de photons émis par un électron avec une énergie E parcourant une distance l dans un milieu d'indice n et ce, dans une bande du spectre comprise entre les longueurs d'onde λ_1 et λ_2.

$$N = 2\pi\alpha l \left(\frac{1}{\lambda_1} - \frac{1}{\lambda_2} \right) \left(1 - \frac{1}{\beta^2 n^2} \right) \tag{1.4}$$

Avec :
– α est la constante de structure fine valant 1/137

– N est le nombre de photons générés dans la bande λ_1 à λ_2

Cette émission de photons présente plusieurs caractéristiques. Le spectre d'émission est une bande allant du domaine ultraviolet au domaine infrarouge de manière continue. Le nombre de photons émis à une longueur d'onde donnée varie en fonction de la longueur d'onde. L'allure du spectre est proportionnelle au carré de l'inverse de la longueur d'onde (fig. 1.1). Le spectre énergétique est, quand à lui, proportionnel au cube de l'inverse de la longueur d'onde. Le nombre de photons émis dépend du matériau, ainsi que de l'énergie et de la charge de la particule.

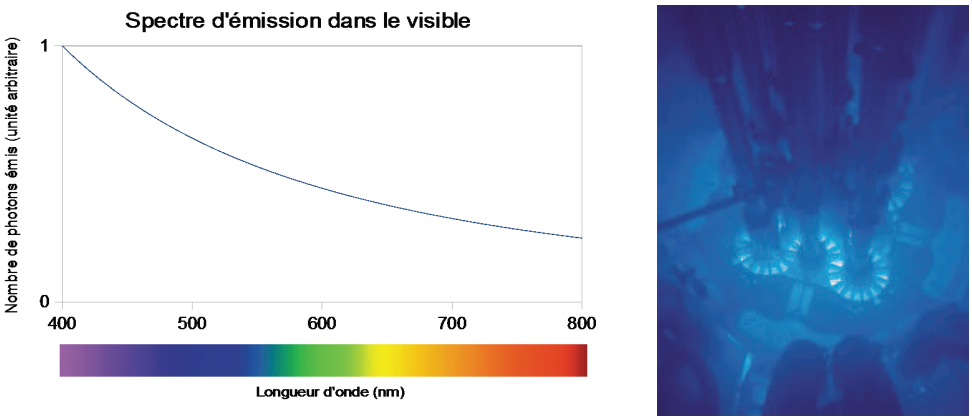

FIGURE 1.1 – Spectre du rayonnement Cerenkov.

$$cos(\phi) = \frac{1}{\beta n} \tag{1.5}$$

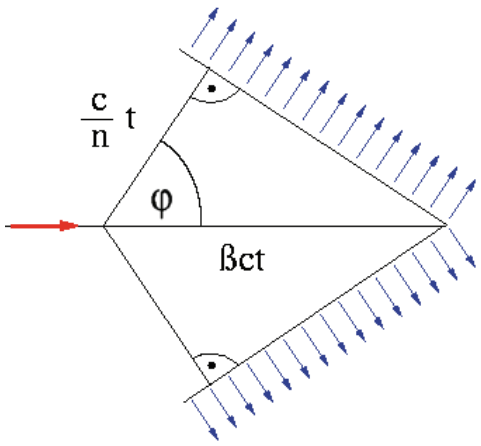

FIGURE 1.2 – Schéma de l'émission du cône Cerenkov.

L'émission se fait suivant un cône dont l'angle ϕ dépend de l'énergie cinétique et du milieu (fig. 1.2). Les rayonnements ionisants ainsi que les noyaux radioactifs sont susceptibles de créer de l'effet Cerenkov si les énergies cinétiques mises en jeu sont suffisantes pour dépasser le seuil Cerenkov dans le milieu (tab. 1.2). En général, seules les particules chargées de masse faible sont directement à l'origine de l'effet Cerenkov. L'effet Cerenkov

produit par certains rayonnements ionisants provient des particules chargées générées lors de l'interaction du rayonnement avec la matière.

Rayonnement	Gamme d'énergie	Cerenkov possible
Visible	1-10 eV	Non
UV	10-100 eV	Non
X/γ (ionisant)	100 eV - qques MeV	Oui (à partir d'un certain seuil) (indirect)
β +	0-3 MeV	Oui
β -	0-3 MeV	Oui
Proton (accélérateur pour le médical)	0-240 MeV	Oui (indirect)
α	0-9 MeV	Oui (indirect)

TABLE 1.2 – Possibilité d'observer de l'effet Cerenkov suivant le rayonnement

1.2 Imagerie nucléaire

1.2.1 Imagerie PET/SPECT

En imagerie nucléaire, on distingue la TEP et la TEMP. Les techniques d'imagerie nucléaire TEP et TEMP se basent sur les propriétés nucléaires de certains noyaux à émettre des rayonnements lors de leur désintégration. L'imagerie TEMP utilise des isotopes émetteurs de photons γ tandis que la TEP utilise des émetteurs de positons. Ces isotopes sont liés à une molécule qui va se propager dans l'organisme à imager et dont la localisation va dépendre de l'effet biologique de cette molécule. Cette molécule est appelée radiotraceur.

Les photons γ émis, soit par désintégration d'un noyau soit par l'annihilation d'un positon avec un électron, sont très peu absorbés par les tissus. En conséquence, les techniques TEP et TEMP sont applicables indifféremment chez l'homme ou chez le petit animal. Le principe de reconstruction consiste à déterminer la direction des photons dans l'espace afin de définir une ligne de réponse. Chaque ligne de réponse étant associée à une désintégration ou annihilation, il est possible de reconstruire en 3D la source.

L'imagerie TEMP utilise un seul photon γ pour obtenir une ligne de réponse. La TEP détermine une ligne de réponse en détectant 2 photons en coïncidence issus de l'annihilation d'un positon avec un électron.

Ces modalités d'imagerie permettent de réaliser de l'imagerie moléculaire et présentent l'avantage d'obtenir la distribution du radiotraceur en 3D sur du petit animal comme chez l'homme.

Les performances de ces techniques varient suivant l'appareil et le processus de reconstruction associé.

1.2.2 Radiotraceur

Le radiotraceur est un élément important en imagerie nucléaire puisque ses propriétés physico-chimiques vont déterminer comment le traceur va se distribuer dans l'organisme. D'autre part, l'isotope radioactif utilisé va déterminer quelle modalité va être utilisée pour l'imagerie.

Le radiotraceur est composé de 2 éléments : le marqueur et le traceur. Le marqueur est l'isotope radioactif utilisé pour pouvoir détecter la molécule. Le traceur, aussi appelé vecteur, est la molécule à laquelle le marqueur est lié. Le traceur est la partie qui va permettre la spécificité du marquage du radiotraceur dans l'organisme. Le traceur permet l'acheminement du marqueur jusqu'à une cible. Suivant le traceur, le marqueur peut se retrouver fixé à la cible ou juste être accumulé.

L'utilisation de radiotraceurs présente plusieurs contraintes. La synthèse des radiotraceurs est un processus souvent complexe puisqu'il nécessite d'associer un atome radioactif avec une molécule. La synthèse doit permettre d'obtenir une molécule marquée avec un isotope radioactif et contenue dans une solution non toxique. Cela nécessite de travailler avec des équipements de radioprotection. D'autre part, la synthèse doit être compatible avec la demi-vie de l'isotope utilisé.

Enfin, le comportement du traceur dans l'organisme dépend de beaucoup de paramètres biologiques difficiles à évaluer. La propagation du traceur est un phénomène continu qui implique un changement de la distribution du traceur en fonction du temps. L'évolution de la distribution du radiotraceur dans l'organisme en fonction du temps est appelée biodistribution.

L'imagerie Cerenkov *in vivo* utilise des radiotraceurs avec des isotopes émetteurs de β. Elle permet donc l'utilisation d'isotopes et traceurs déjà existants en TEP mais surtout des radiotraceurs utilisés en autoradiographie ou pour la thérapie et d'ouvrir la voie à de nouveaux radiotraceurs. Le tableau 1.3 liste les isotopes utilisés dans le domaine médical et leur intérêt pour l'imagerie Cerenkov [4, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17].

Isotope	Demi-vie	Emission(s)	Energie(s)	PET/SPECT
^{131}I	8 jours	β^-, γ	606 keV, 364 keV	SPECT
^{13}N	9.97 min	β^+	1200 keV	PET
^{11}C	20.3 min	β^+	960 keV	PET
^{15}O	2 min	β^+	1732 keV	PET
^{18}F	109.77 min	β^+	633 keV	PET
^{64}Cu	12.7 heures	β^+, β^-	653 keV, 578 keV	PET
^{68}Ga	67.7 min	β^+	1899 keV	PET
^{74}As	17.7 jours	β^+, β^-	2540 keV, 1350 keV	PET
^{89}Zr	78.4 heures	β^+	897 keV	PET
^{124}I	4.18 jours	β^+	2135 keV	PET
^{32}P	14.29 jours	β^-	1709 keV	
^{86}Rb	18.6 jours	β^-	1776 keV	
$^{89/90}$Sr	50.5 jours	β^-	546 keV	
^{90}Y	64 heures	β^-	2284 keV	
^{114}In	71.9 secondes	β^-	1989 keV	
^{177}Lu	6.65 jours	β^-	498 keV	
^{198}Au	2.7 jours	β^-	960 keV	

Isotope	Demi-vie	Emission(s)
^{212}At	7.21 heures	α
^{225}Ac	10 jours	α
$^{210/212/213}$Bi	1 h/45.6 min	α
^{210}Pb	22,2 ans	α
^{230}U	18.72 jours	α

TABLE 1.3 – Liste des isotopes utilisés dans le domaine médical produisant du rayonnement Cerenkov. Dans le cas des émetteurs α, l'émission du rayonnement Cerenkov est indirect.

1.3 Imagerie optique

Dans le cadre de l'imagerie moléculaire, l'imagerie optique dispose de certains avantages et se ramifie en plusieurs modalités [18]. Le rayonnement visible est peu invasif car non ionisant, facile à manipuler avec des outils optiques et il dispose d'un capteur très répandu qu'est l'oeil ainsi que des capteurs très sensibles comme les capteurs CMOS. L'acquisition d'images planaires peut nécessiter des temps inférieurs à la minute. Toutefois, la lumière visible traverse difficilement les tissus biologiques, limitant la majeure partie de ses applications au petit animal et à l'étude cellulaire.

Chez le petit animal, les modes d'acquisition en optique sont nombreux et permettent d'obtenir des images différentes. Des images peuvent être acquises avec une source d'illumination pour observer la géométrie des structures alors que sans illumination, la luminescence peut être observée. Plusieurs angles de vues peuvent être observés simultanément soit en utilisant plusieurs détecteurs, soit avec un seul et un système de miroirs. Il est aussi possible de faire tourner l'objet devant le détecteur mais les images ne sont plus acquises simultanément. L'utilisation d'un filtre permet de sélectionner une plage de longueur d'onde à observer ou à exciter. Les détecteurs optiques disposent d'un rendement quantique élevé pouvant dépasser 90% et permettent d'obtenir des images exploitables en quelques secondes. Cela permet d'acquérir des images pour suivre l'évolution dans le temps.

L'imagerie optique chez le petit animal comprend plusieurs modalités. La fluorescence et la bioluminescence sont actuellement les 2 modalités les plus utilisées en imagerie optique moléculaire du petit animal *in vivo* [18] (fig. 1.3).

La fluorescence se base sur le principe d'excitation par une source d'excitation externe et de réémission d'une molécule. Cette modalité dispose d'un grand nombre de molécules fluorescentes et de QDot (QuantumDot), nanocristaux de semi-conducteurs. Toutefois, la source d'excitation optique doit d'abord traverser les tissus une fois avant d'atteindre sa cible. Il est nécessaire de séparer le signal dû à la source d'excitation et le signal de fluorescence. Cette séparation est généralement faite en choisissant des longueurs d'onde différentes pour la source d'excitation et le signal de fluorescence.

La bioluminescence utilise majoritairement l'émission de lumière obtenue par réaction enzymatique entre la luciférase et la luciférine. Cette technique permet de s'affranchir d'une source d'excitation. Les cellules mammifères ne produisant aucun de ces composés, il est nécessaire de modifier les gènes pour obtenir une production localisée de luciférase. L'injection de luciférine dans l'animal provoquent une émission de lumière par réaction avec la luciférase dans les tissus. Le spectre d'émission de cette réaction est jaune-vert (540-600 nm). Le signal de bioluminescence est plus faible de plusieurs ordres de grandeur que le signal de fluorescence.

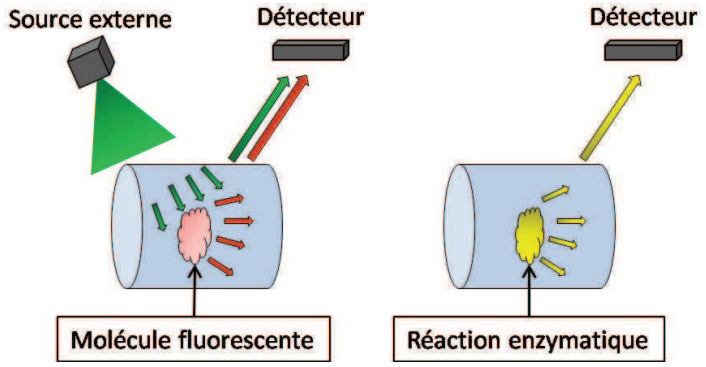

FIGURE 1.3 – Principe de l'imagerie de fluorescence (à gauche) et de bioluminescence (à droite).

Les techniques de bioluminescence et fluorescence permettent aussi de réaliser de la tomographie (BLT et FLT) pour reconstruire la source optique. Ces techniques de tomographie sont difficiles à réaliser puisque les phénomènes optiques au sein des tissus biologiques sont nombreux. En particulier, les diffusions multiples sont fréquentes.

L'imagerie Cerenkov s'apparente beaucoup à la bioluminescence (fig. 1.4). Ces 2 modalités ne nécessitent pas de source externe. La source est interne à l'animal. Mais la source optique diffère entre ces 2 modalités, en particulier le spectre d'émission. Le rendement de production de photons en imagerie Cerenkov est dépendant de l'isotope et de l'activité alors qu'en bioluminescence, le signal dépend de la réaction chimique entre la luciférine et la luciférase (tab. 1.4).

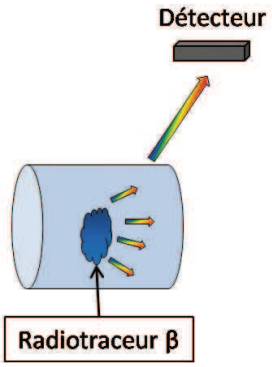

FIGURE 1.4 – Principe de l'imagerie Cerenkov utilisant des radiotraceurs.

Modalité	Mécanisme d'activation	Sonde optique	Signal
FLI	Source d'excitation externe	Molécule fluorescente, QDots	Dépend de la quantité de la sonde et de l'intensité du faisceau d'excitation
BLI	Réaction chimique	Luciférase	Dépend des quantités de matières
CLI	Radioactivité β	Radiotraceur	Dépend de l'activité, du milieu et de l'énergie des β

TABLE 1.4 – Présentation des modalités optiques en imagerie moléculaire.

L'imagerie optique permet aussi l'accès à une information anatomique. La tomographie optique diffuse (DOT) est une modalité permettant de déterminer les coefficients optiques dans un volume [19, 20]. Elle permet de récupérer une information anatomique en volume en reconstruisant les coefficients de propagation optiques dans les tissus. C'est la seule technique qui existe pour connaître les coefficients optiques *in vivo*. Certaines techniques se limitent à obtenir la surface d'un objet en 3D [21, 22, 23] mais ne permettent pas d'avoir d'informations internes à un animal.

1.4 Intérêts et applications de l'imagerie Cerenkov

L'une des premières applications de l'effet Cerenkov a été la détection de particules à hautes énergies qui génèrent un grand nombre d'électrons secondaires dépassant le seuil d'émission Cerenkov. La première application biologique date de 1969 avec le suivi de la cinétique d'absorption du potassium en solution, contenant du rubidium-86, par des racines d'orge [24]. En 1971, l'effet Cerenkov produit par du phosphore-32 a servi pour un diagnostic de tumeur présente dans l'oeil d'un lapin [25]. Ces applications permettaient de suivre l'activité par comptage du rayonnement Cerenkov. Mais ce n'est qu'avec le développement de détecteurs optiques ultrasensibles tels que les capteurs CCD que les applications de l'effet Cerenkov se sont ouvertes sur l'imagerie, particulièrement dans le domaine médical. Depuis 2009, un grand nombre d'applications dans l'imagerie médicale ont émergé [5, 26, 27, 28, 29].

1.4.1 Imagerie Cerenkov planaire

L'imagerie Cerenkov planaire dispose des mêmes avantages que l'imagerie optique [10]. Le temps d'acquisition d'une image 2D est très rapide (pouvant être de juste quelques secondes) grâce à des détecteurs très sensibles et avec des résolutions pouvant être inférieures à 50 μm. Le champ de vue en optique peut permettre d'imager un grand nombre de souris simultanément (> 5). L'acquisition de plusieurs vues de l'animal peut se faire en n'utilisant

que un seul détecteur.

La première réalisation d'une image Cerenkov 2D *in vivo* a été réalisée avec du ^{18}F-FDG sur des souris [4]. Une solution de ^{18}F-FDG a été injectée à des souris possédant chacune une tumeur sous-cutanée. Une image optique des souris a été enregistrée avec un imageur optique Xenogen IVIS 200 et une image PET a été réalisée pour chaque souris. Les images optiques ainsi obtenues sont cohérentes avec l'image obtenue par PET. Cette étude a démontré la faisabilité de l'imagerie Cerenkov et la possibilité d'utiliser des radiotraceurs émetteurs de β^+. Le ^{18}F-FDG a été utilisé dans beaucoup d'études [10, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40] (fig. 1.5). D'autres radiotraceurs utilisant du ^{18}F ont été montré comme utilisables pour l'imagerie Cerenkov comme le ^{18}F-FHBG [32], le ^{18}FNa [10], le ^{18}F-FLT [41] ou encore le ^{18}F-FCP [42] (tab. 1.5).

D'autres études ont montré que d'autres isotopes β^+ comme le ^{68}Ga [43, 40, 44], le ^{124}I [45] ou le ^{89}Zr [11, 46] peuvent servir pour l'imagerie Cerenkov.

Une approche originale proposée est d'observer les structures absorbantes [43]. Ici, le rayonnement Cerenkov sert de signal de fond pour faire apparaître les structures absorbantes comme les vaisseaux sanguins. Des vaisseaux de moins de 50 μm ont été observés avec cette technique permettant l'utilisation de l'imagerie Cerenkov pour l'angiographie. Le tableau 1.6 fait la liste des isotopes et radiotraceurs déjà utilisés dans l'imagerie Cerenkov sur souris.

Isotope	Radiotraceur	Fonction
^{18}F	FDG	dérivé du glucose, très consommé par le cerveau, le coeur, les tumeurs, le tissu adipeux marron et les yeux
^{18}F	F-CP-118,954	spécifique d'une molécule liée à la maladie d'Alzheimer
^{18}F	FHBG	spécifique d'un gène (HSV1)
^{18}F	FNa	se décompose en F^-, se fixe principalement sur les os
^{18}F	FLT	spécifique d'un mécanisme de la multiplication cellulaire

TABLE 1.5 – Rôles des molécules au ^{18}F utilisées en imagerie Cerenkov sur souris.

Isotope	Radiotraceur	Activité (MBq)	Cible	Réf.
^{18}F	FDG	10	Tumeur, vessie	[4]
^{18}F	FDG	13	Coeur, tumeur, vessie	[30]
^{18}F	FDG	10,4-11,3	Tumeur, coeur	[10]
^{18}F	FDG	6,4-7,5	Vessie, tumeur, cerveau	[41]
^{18}F	FDG	30	Vessie, coeur	[34, 33]
^{18}F	FDG	20	Cerveau, tumeur	[35]
^{18}F	FDG	11,1	Tumeur	[36]
^{18}F	FDG	20	Cerveau, tumeur	[37]
^{18}F	FDG	30	Cerveau, tumeur	[38]
^{18}F	FDG	10,3	Tissu adipeux marron	[39]
^{18}F	FDG	3,7	Coeur, cerveau, tumeur	[40]
^{18}F	F-CP-118,954	35	Cerveau	[42]
^{18}F	FHBG	10-11	Tumeur, vessie	[32]
^{18}F	FNa	5,3-5,7	Os (colonne, crâne, genoux)	[10]
^{18}F	FLT	7,3-8	Vessie, tumeur	[41]
^{68}Ga	$GaCl_3$	17	~Vaisseaux sanguins ~	[43]
^{68}Ga	Ga-3PRGD$_2$	3,7	Intestin, estomac, tumeur	[40]
^{124}I	I	7,4	Thyroïde/ Tumeur	[45]
^{124}I	I-Herceptin	3,3	Thyroïde	[14]
^{89}Zr	Zr-DFO-J591	11	Tumeur	[11]
^{89}Zr	Zr-DFO-trastuzumab	4	Tumeur	[46]
^{89}Zr	Zr-rituximab	2,6	Foie, rate	[47]
^{32}P	ATP	10	Vessie, foie	[37]
^{32}P	ATP	10	Coeur, cerveau, côtes, intestin	[48]
^{131}I	NaI	2,2-2,3	Thyroïde, vessie	[10]
^{131}I	I	7,4	Thyroïde	[45]
^{131}I	NaI	16,6	Thyroïde	[49]
^{90}Y	YCl_3	1,8-2	Foie, vessie	[10]
^{90}Y	RGD-BBN	2,6-3,3	Vessie, tumeur	[10]
^{198}Au	nanocage	1,4	Tumeur, rate, foie	[50]

TABLE 1.6 – Applications de l'imagerie Cerenkov 2D existantes sur souris

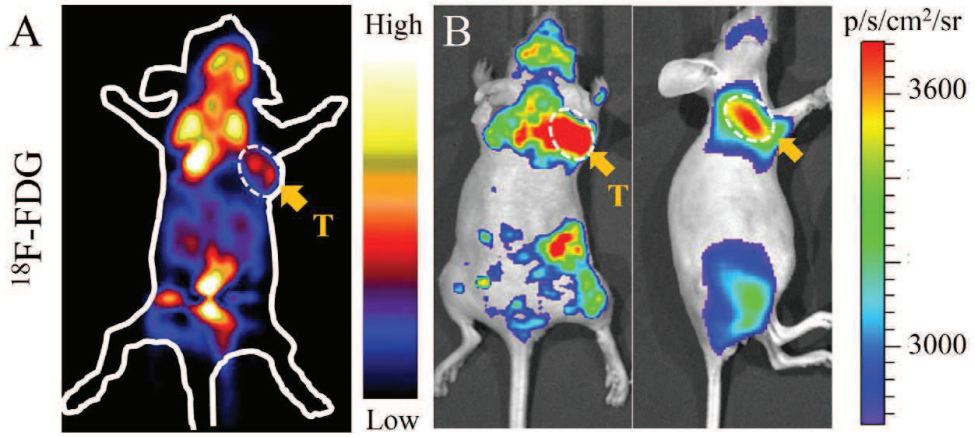

FIGURE 1.5 – Image d'une souris injectée avec 3,7 MBq de ^{18}F-FDG dans la veine de la queue, réalisée avec un IVIS Spectrum acquise pendant 5 min [40]. Les images PET (à gauche) et Cerenkov (à droite) ont été réalisées après 1h de biodistribution. La flèche indique la position de la tumeur.

Les imageurs optiques utilisés pour ces études sont des imageurs optiques Xenogen IVIS (100, 200, Spectrum ou Kinetic) (tab. 1.7). Les cas où un système non commercial est utilisé est toujours dans l'objectif de réaliser de la tomographie Cerenkov.

Système IVIS	100	200	Kinetic	Spectrum
Nombre de pixels	2048 x 2048	1920 x 1920	2024 x 2024	2048 x 2048
Efficacité quantique CCD	85% 500-700nm, >30% 400-900nm	85% 400-700nm, >50% 350-900nm	85% 500-700nm, >30% 400-900nm	85% 500-700nm, >30% 400-900nm
Objectif	f/0.95-f/16, 50mm	f/1-f/8	f/0.95-f/16, 50mm	f/1-f/8

TABLE 1.7 – Comparaison des caractéristiques des détecteurs des différents systèmes Xenogen IVIS

1.4.1.1 Suivi thérapeutique

Le suivi thérapeutique, en particulier pour des études sur le cancer, peut être réalisé grâce à des radiotraceurs. Il a été montré que l'imagerie Cerenkov est quantitative pour des sources peu profondes. En préclinique, des radiotraceurs peuvent ainsi être utilisés sans

nécessiter des imageurs TEP ou TEMP [36, 41, 51], en permettant l'utilisation d'imageurs optiques.

1.4.1.2 Suivi chirurgical

Le signal optique Cerenkov peut être observé en temps réel *in vivo*. Ce signal peut servir à guider un acte chirurgical grâce à la grande affinité des radiotraceurs. La faisabilité de cette application a été validé sur souris [46]. Des images Cerenkov ont été acquises à chaque étape (avant incision au niveau de la tumeur, après incision, après extraction de la tumeur et après avoir suturé). L'imagerie Cerenkov a permis de localiser la tumeur et de prouver que la tumeur a été intégralement enlevée avant de refermer les tissus.

1.4.1.3 Cerenkov Radiation Energy Transer (CRET)

L'obectif de cette technique est de récupérer le signal Cerenkov absorbé en utilisant une molécule fluorescente pour améliorer l'imagerie Cerenkov. Le signal enregistré par l'imagerie Cerenkov planaire peut être amélioré via l'utilisation de molécules absorbantes dans les faibles longueurs d'onde du spectre Cerenkov et réémettant à des longueurs d'onde plus grandes dans le rouge IR. Le signal Cerenkov ainsi absorbé par les tissus peut alors être mis à profit et le signal réémis peut être différencié du signal Cerenkov pour une même longueur d'onde [31, 52].

1.4.1.4 Applications cliniques

L'imagerie Cerenkov 2D montre déjà des résultats cliniques. Cette technique d'imagerie a été utilisé pour du diagnostic pour des problèmes de thyroïde avec de l'iode-131 [53] (fig. 1.6) et de ganglions lymphatiques avec du ^{18}FDG [54]. Cette imagerie est toutefois limitée à la détection de sources peu profondes.

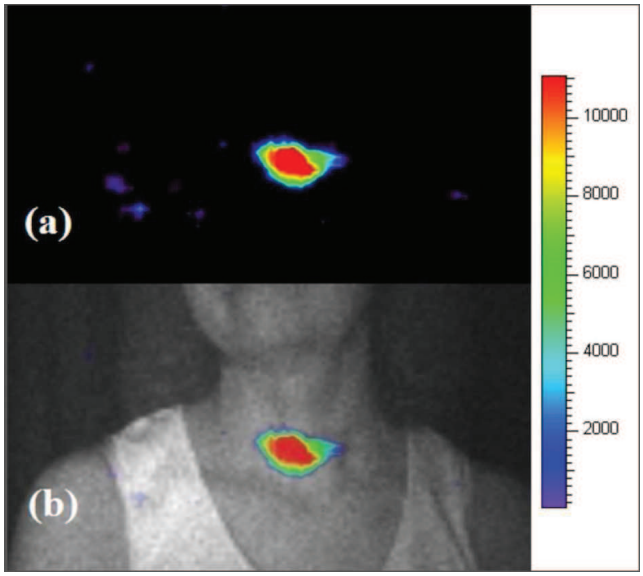

FIGURE 1.6 – Imagerie Cerenkov chez l'homme [53]. Le patient a été injecté avec 550 MBq de ^{131}I. L'image a été réalisée 24h après injection pendant 2 min avec un dispositif comprenant un EMCCD (512 x 512, efficacité quantique maximum : 90%) et un objectif (f/1.4, 8mm).

1.4.2 Endoscopie Cerenkov

Le signal Cerenkov émis par des tissus à une profondeur de plusieurs centimètres n'atteint pas le détecteur à cause de la forte absorption des tissus. Pour palier à cette limite, l'endoscopie peut être utilisée en combinaison de radiotraceurs émettant du signal Cerenkov. L'imagerie Cerenkov par endoscopie est réalisable aussi bien chez le petit animal [55, 56] que chez l'homme [57]. Chez le petit animal, des fibres optiques ont été introduites dans l'animal vivant pour observer le signal présent dans différents organes (cerveau, coeur, foie, reins, tumeur et muscle) (fig. 1.7). Cette technique a permis la quantification de l'activité de ^{18}F-FDG dans chaque organe pour des activités aussi faibles que 1 μCi (37 kBq). L'inconvénient principal est l'aspect invasif de cette méthode à cause de l'insertion de fibres optiques dans l'animal. Chez l'homme, cette technique a été développée pour le diagnostic de maladie gastro-intestinale [57].

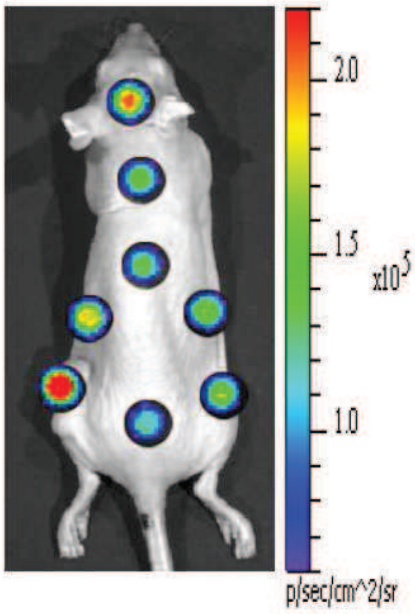

FIGURE 1.7 – Imagerie Cerenkov par endoscopie [55]. L'animal a été injecté de 35 MBq de ^{18}F-FDG dans la veine de la queue. L'image a été réalisée 90 min après injection via des fibres optiques de 6 mm introduites dans l'animal jusqu'aux organes. L'animal est euthanasié avant imagerie.

1.4.3 Source d'excitation interne

L'effet Cerenkov peut servir de source d'excitation pour l'imagerie de fluorescence. Cette technique permet de s'affranchir de source d'excitation externe et des problèmes pour exciter les molécules fluorescentes profondes dans les tissus biologiques pour l'imagerie de fluorescence. De plus, en fluorescence, il est possible d'avoir des molécules absorbant et émettant à des longueurs d'onde différentes et peuvent donc être facilement distinguées. Couplé à une excitation de type Cerenkov qui se produit dans tout le visible, il est possible d'exciter plusieurs molécules avec un même radiotraceur et de pouvoir différencier le signal des molécules fluorescentes présentes [58, 59, 60] (fig. 1.8).

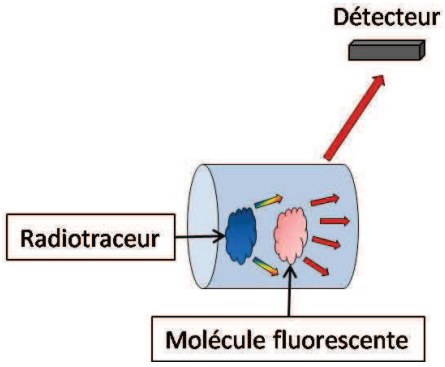

FIGURE 1.8 – Principe de l'imagerie de fluorescence utilisant le rayonnement Cerenkov comme source d'excitation.

1.4.4 Scanner par luminescence par excitation Cerenkov

Les rayons X produits par un accélérateur linéaire sont susceptibles de provoquer du rayonnement Cerenkov en traversant les tissus. En excitant des tissus par un faisceau X dépassant plusieurs centaines de keV, il est possible de créer des photons optiques sur le trajet du faisceau X. Ce procédé est exploité pour scanner une souris possédant une molécule fluorescente de manière optique [61, 62]. Le rayonnement Cerenkov est utilisé pour exciter la molécule fluorescente. La molécule fluorescente est excitée dans une fine tranche de tissu via un faisceau plan. En scannant l'animal suivant des plans dans les 3 directions de l'espace, il est possible de localiser la molécule fluorescente au sein de l'animal en 3D.

1.4.5 Tomographie Cerenkov

Un des développements possibles pour l'imagerie Cerenkov est la reconstruction de la distribution du radiotraceur dans les tissus *in vivo* [30, 48, 49, 63, 64, 65, 66, 67]. Cette technique est appelée tomographie du rayonnement Cerenkov ou juste tomographie Cerenkov. Toutefois, le signal optique interagit énormément avec les tissus biologiques. Les phénomènes de diffusion et d'absorption ne sont pas négligeables et rendent difficile la connaissance du trajet emprunté par un photon. Pour reconstruire la distribution de la source, les images planaires du signal Cerenkov seules sont insuffisantes et l'utilisation d'informations supplémentaires est nécessaire. La géométrie et les propriétés optiques de l'objet sont généralement nécessaires pour aboutir à une bonne résolution en tomographie Cerenkov.

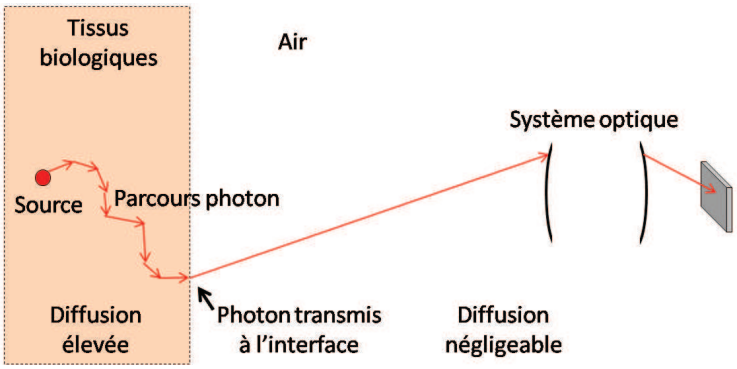

FIGURE 1.9 – Schéma du problème de reconstruction du parcours des photons optiques.

Les reconstructions en tomographie Cerenkov se déroulent souvent en considérant le problème de propagation optique dans les tissus et dans le système optique différemment puisque le nombre d'interactions est plus important dans les tissus (fig. 1.9). Une étape intermédiaire dans la tomographie Cerenkov est la reconstruction du flux optique à la frontière entre les tissus biologiques et l'air (fig. 1.10). Pour cette étape, il faut considérer le système optique et la surface de l'animal. Le flux reconstruit dépend de l'angle de vue avec lequel on observe l'animal et est valable dans une certaine bande du spectre optique. Dans un second temps, l'information du flux en surface permet de reconstruire la source au sein des tissus. Il faut utiliser un modèle de propagation pour décrire le comportement des photons dans les tissus. L'équation de transfert radiatif (ou équation de Boltzmann) décrit intégralement la propagation des photons dans un milieu. Les phénomènes optiques dépendent du milieu et de la longueur d'onde.

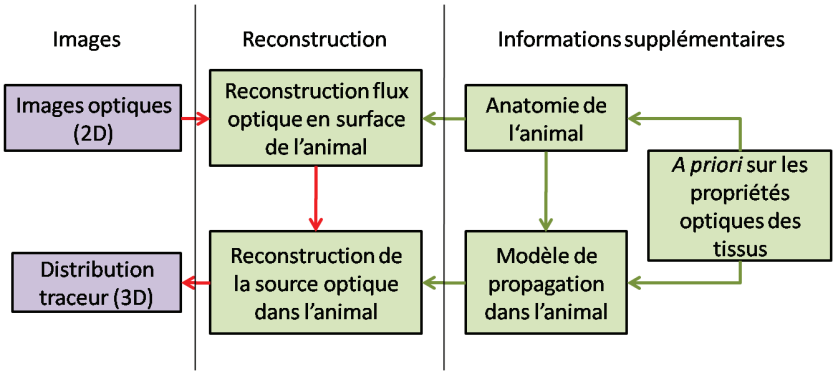

FIGURE 1.10 – Principe de la tomographie Cerenkov.

Plusieurs approches ont déjà montré la faisabilité de cette technique (tab. 1.8 et 1.9). Une première approche [30] a été de mesurer le flux optique en surface grâce aux vues latérales de la souris. Les images optiques ont été enregistrées avec l'imageur optique Xenogen Ivis 100. Les images latérales de l'animal ont été enregistrées simultanément avec un filtre passe-bande entre 695-770 nm pendant 15 min. L'anatomie de l'animal a été obtenue par un TDM annexe à l'imageur optique. La reconstruction de la source est réalisée en employant une méthode par éléments finis pour résoudre l'équation de diffusion, un modèle simplifié de l'équation de Boltzmann valable si la diffusion est prédominante par rapport à l'absorption. Un maillage homogène est généré à partir du volume obtenu par TDM. L'algorithme de reconstruction est un algorithme PCG (Preconditioned Conjugate Gradient) utilisant une régularisation de Tikhonov. Cette méthode a été testée sur une souris injectée avec 12 MBq de ^{18}F-FDG. La reconstruction en imagerie Cerenkov 3D corps entier montre la présence du produit dans le coeur et la vessie ce qui est cohérent avec la reconstruction PET (fig. 1.11).

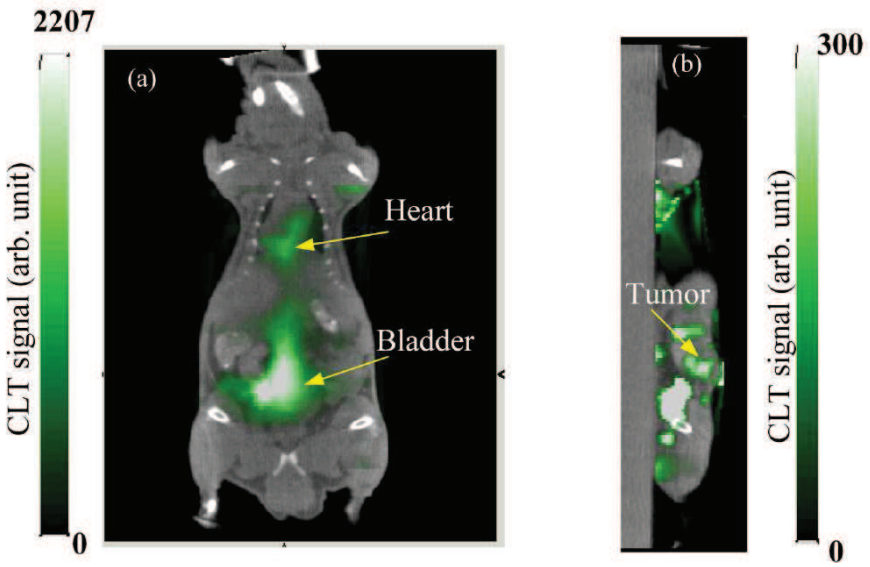

FIGURE 1.11 – Résultats de reconstruction tomographique Cerenkov [30]. La souris a été injectée avec 12 MBq de ^{18}F-FDG dans la veine. La source optique est reconstruite dans la vessie, le coeur et la tumeur.

Une autre approche consiste à utiliser l'information spectrale du rayonnement Cerenkov pour reconstruire la source [48]. Un imageur Xenogen Ivis 200 a été utilisé pour enregistrer une vue dorsale de l'animal avec 4 filtres passe-bande étroits (longueur d'onde centrale : 600, 620, 640, 660 nm ; largeur des filtres : 20 nm). L'anatomie de l'animal a été acquise par l'imageur optique par projection d'une lumière structurée sur l'animal afin de reconstruire la surface de l'animal. Le volume est considéré homogène. Dans un milieu homogène, une solution de l'équation de diffusion est donnée par le modèle de Green. Ce modèle est utilisé pour reconstruire la distribution de la source. L'algorithme de reconstruction est un algorithme des moindres carrées à valeurs positives utilisant une régularisation de Tikhonov. Cette méthode a été réalisée sur une souris injectée avec 10 MBq de ^{32}P-ATP. La reconstruction en imagerie Cerenkov 3D a été superposée à une image IRM de l'animal pour localiser la source reconstruite dans la souris (fig. 1.12). La souris a ensuite été disséquée pour vérifier que l'ATP s'est distribué comme le montre la fusion de l'image Cerenkov 3D et IRM. L'intestin, le cerveau, le coeur et les côtes ont été imagés *ex vivo* par imagerie Cerenkov 2D. La reconstruction en imagerie Cerenkov 3D coïncide avec le signal enregistré *ex vivo*.

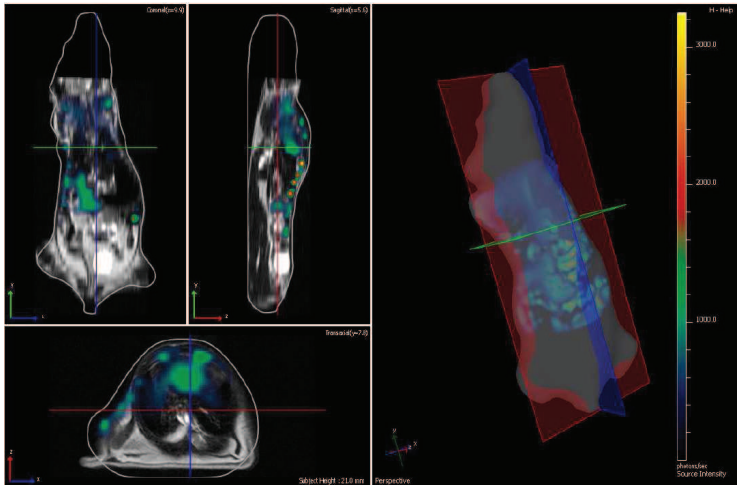

FIGURE 1.12 – Résultats de reconstruction tomographique Cerenkov [48]. La souris a été injectée avec 12 MBq de ^{32}P-ATP dans la veine. La source optique est reconstruite dans l'intestin, le coeur et le cerveau.

La nécessité d'avoir une information anatomique *a priori* demande une seconde modalité d'imagerie. Considérer l'hétérogénéité de la souris permet d'avoir plus d'informations *a priori*. Toutefois, la seule modalité optique *in vivo* qui permet d'obtenir l'anatomie en volume est la DOT mais elle est longue et délicate à réaliser.

Une méthode a consisté à utiliser 4 vues, à 90° chacune de l'animal, sans filtre, grâce à un système rotatif pour l'animal [64, 65]. L'imageur optique utilisé se compose d'un capteur CCD camera (Princeton Instruments VersArray 1300B, Roper Scientific, Trenton, NJ) possédant 1340 x 1300 pixels. Le volume TDM est utilisé pour générer un maillage hétérogène. La résolution de l'équation de transfert radiatif est effectué avec une méthode par éléments finis avec l'approximation SP_3. L'algorithme de reconstruction est un algorithme GRS (Geometric Row Scaling) avec terme de régularisation $L_{1/2}$. Des souris injectées avec du ^{18}F-FDG ont été utilisées pour valider cette approche. Les reconstructions Cerenkov 3D sont cohérentes avec l'imagerie PET (fig. 1.13).

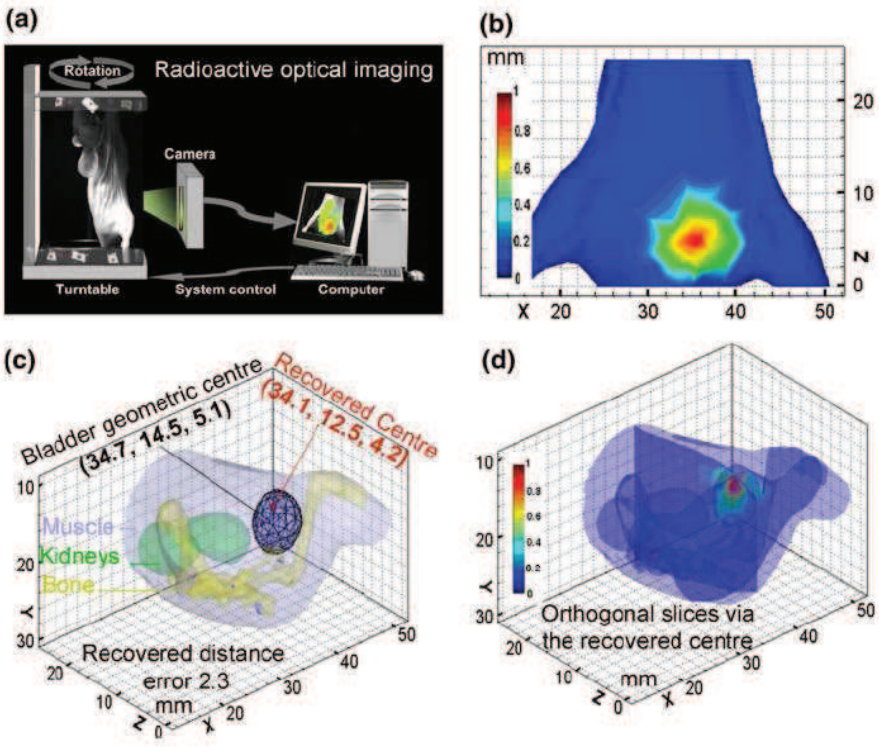

FIGURE 1.13 – Résultats de reconstruction tomographique Cerenkov [65]. La souris a été injectée avec 11,1 MBq de ^{18}F-FDG dans la veine de la queue. La reconstruction permet de localiser la source dans la vessie.

Un groupe réalise de l'imagerie Cerenkov 3D à partir de systèmes combinant imagerie optique et TDM. Un système utilisé est le système nommé ZKKS-Direct 3D [49, 63, 66]. Il utilise un support rotatif pour l'animal. Le système d'imagerie optique utilise un capteur CCD (Princeton Instruments PIXIS 2048B, Roper Scientific, Trenton, NJ) fournissant des images 2048 x 2048, 16-bit. Il est couplé à un objectif (Nikon Normal Macro 55 mm f/2.8). L'équation de diffusion est résolue avec une méthode par éléments finis avec un algorithme adaptative hp-FEM [68]. Les essais effectués avec de l'iode-131 avec 4 vues et 1 filtre ou avec 4 filtres et une vue donnent de bons résultats pour les reconstructions Cerenkov 3D cohérents avec l'imagerie SPECT (fig. 1.14).

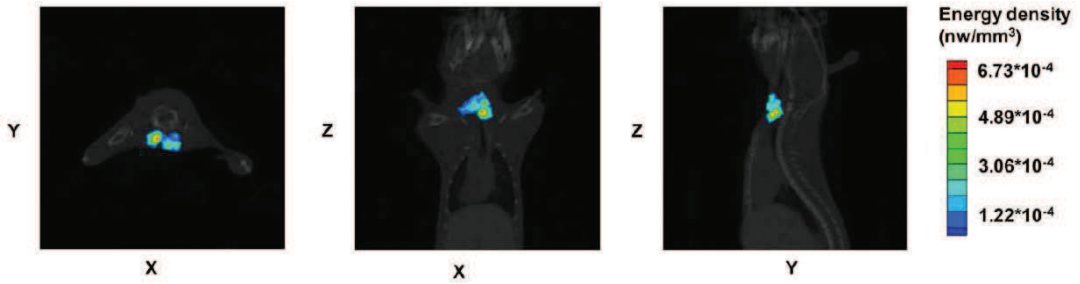

FIGURE 1.14 – Résultats de reconstruction tomographique Cerenkov [49]. La souris a été injectée avec 16,6 MBq de ^{131}I dans la veine de la queue. La reconstruction fait apparaître la thiroïde de la souris.

Référence	[30]	[65]	[64]	[48]
Source optique	^{18}F-FDG injecté dans la veine	^{18}F-FDG injecté dans la veine	^{18}F-FDG injecté dans la veine	^{32}P-ATP injecté dans la veine
Activité (MBq)	12	11,1	14,5	10
Imageur optique	Ivis 100	Système d'imagerie moléculaire bi-modalité optique-TDM	Système d'imagerie moléculaire bi-modalité optique-TDM	Ivis 200
Acquisition optique	2 (Latérales gauche et droite simultanément) ; 1 (695-770 nm)	4 (Dos, gauche, ventre et droite de l'animal séquentiellement) ; Sans filtre	4 (Dos, gauche, ventre et droite de l'animal séquentiellement) ; Sans filtre	1 (Vue dorsale) ; 4 (Longueur d'onde centrale : 600, 620, 640, 660 nm, largeur à mi-hauteur de 20 nm)
Temps d'acquisition optique	15 min	4x1x2 min = 8 min	4x1x2 min = 8 min	1x4x5 min = 20 min
A priori anatomique	TDM, volume homogène maillé	TDM, volume hétérogène segmenté maillé	TDM, volume hétérogène segmenté maillé	Optique, surface reconstruite et échantillonnage voxelisé
Modèle de propagation dans les tissus	Equation de diffusion	SP_3 approximation	SP_3 approximation	Modèle de Green
Algorithme de reconstruction	PCG avec regularisation de Tikhonov	GRS avec terme de régularisation	GRS avec terme de régularisation	Moindres carrées à valeur positive

TABLE 1.8 – Détails des principaux procédés de tomographie Cerenkov existants

Référence	[63]	[66]	[49]
Source optique	Cylindre de $Na^{131}I$ placé dans l'abdomen	^{131}I-NGR	$Na^{131}I$ injecté dans la veine
Activité (MBq)	22,2	22,2	16,6
Imageur optique	Système d'imagerie moléculaire bi-modalité optique-TDM (ZKKS-Direct 3D)	Système d'imagerie moléculaire bi-modalité optique-TDM (ZKKS-Direct 3D)	Système d'imagerie moléculaire bi-modalité optique-TDM (ZKKS-Direct 3D)
Acquisition optique	4 (Dos, gauche, ventre et droite de l'animal séquentiellement) ; 1 (695-770 nm)	4 (Dos, gauche, ventre et droite de l'animal séquentiellement) ; 1 filtre (695-770 nm)	1 (Dos) ; 4 filtres (Longueur d'onde centrale : 580, 620, 660, 700 nm, bande de 40 nm) Sans filtre
Temps d'acquisition optique	4x1x5 min = 20 min	4x1x5 min = 20 min	1x4x5 min = 20 min
A priori anatomique	TDM, volume hétérogène segmenté maillé	TDM, volume hétérogène segmenté maillé	TDM, volume hétérogène segmenté maillé
Modèle de propagation dans les tissus	Equation de diffusion	Equation de diffusion	Equation de diffusion
Algorithme de reconstruction	Hp-FEM [68]	Hp-FEM [68]	Hp-FEM [68]

TABLE 1.9 – Détails des principaux procédés de tomographie Cerenkov existants (suite)

1.4.5.1 Discussion

La tomographie Cerenkov s'attache à localiser la source optique. Sa réalisation se rapproche de la tomographie en bioluminescence et les procédés développés en CLT ont généralement déjà été éprouvés en BLT. Dans le cas de l'imagerie *in vivo*, la source est diffusée dans tout l'organisme et des modalités comme la TEMP et TEP doivent être utilisées pour comparer les résultats.

Les activités injectées aux souris sont entre 11 et 15 MBq pour le ^{18}F (β^+, E_{max} = 633 keV), entre 15 et 20 MBq pour le ^{131}I (β^-, E_{max} = 606 keV) et de 10 MBq pour le ^{32}P (β^-, E_{max} = 1709 keV).

Les systèmes d'imagerie utilisés sont majoritairement des développements au sein des groupes permettant d'obtenir l'information anatomique par TDM dans le même dispositif que l'imagerie optique.

L'utilisation de 4 vues est récurrent mais n'utilise qu'un filtre ou pas de filtre. Dans les cas où une unique vue est utilisée, l'information spectrale est exploitée via 4 filtres pour réaliser de la reconstruction multispectrale.

Le TDM est utilisé dans tous les cas, sauf un, et un maillage hétérogène est extrait de cette modalité. Dans le cas où le TDM n'est pas utilisé, seule la surface est acquise pour reconstruire en faisant l'approximation d'un milieu homogène. La reconstruction est superposée à une image IRM. Les volumes obtenus pour la reconstruction Cerenkov et IRM n'ont pas la même géométrie du à la manipulation de l'animal. Souvent, l'acquisition de l'anatomie est effectuée dans le même appareil que celui servant à l'acquisition optique.

Le modèle de propagation utilisé est soit l'équation de diffusion, soit l'approximation SP_3. Les algorithmes utilisent un terme de régularisation pour résoudre le problème mal posé même dans le cas multispectral.

Les reconstructions obtenues en CLT se limitent souvent à reconstruire un point source, souvent la source la plus intense. Les sources sont généralement sous-cutanées ou à faible profondeur. Actuellement, la tomographie Cerenkov demeure une technique encore en développement sur les différentes étapes du processus. Les points clés de la tomographie Cerenkov sont :

- l'isotope et radiotraceur choisis
- le nombre de vues et l'information spectrale pour les images optiques
- les *a priori* anatomiques utilisés
- le processus de reconstruction utilisé

Chapitre 2

Approche utilisée pour l'imagerie Cerenkov 3D

L'objectif est de réaliser de l'imagerie Cerenkov 3D.

Pour cela, nous disposons d'un imageur optique (PhotonImager RT de BiospaceLab) qui nous permet de récupérer 4 vues simulatnément et d'acquérir des images avec différents filtres optiques. Un TDM conçu à ImaBio permet d'obtenir une image anatomique en 3D. Un TEP

Expérimentalement, nous avons des contraintes liées d'une part à la manipulation de produits radioactifs, d'autre part à la manipulation d'animaux vivants. Par conséquent, cela implique de définir des protocoles expérimentaux rigoureux. Le cyclotron CYRCE (CYclotron pour la ReCherche et l'Enseignement) peut actuellement produire du ^{18}F et faire la synthèse de certains radiotraceurs au ^{18}F. Une animalerie est présente à ImaBio.

Le dernier point consiste à mettre en place la chaîne de reconstruction pour traiter les données acquises lors de l'imagerie.

Le procédé complet est détaillé dans le schéma 2.1.

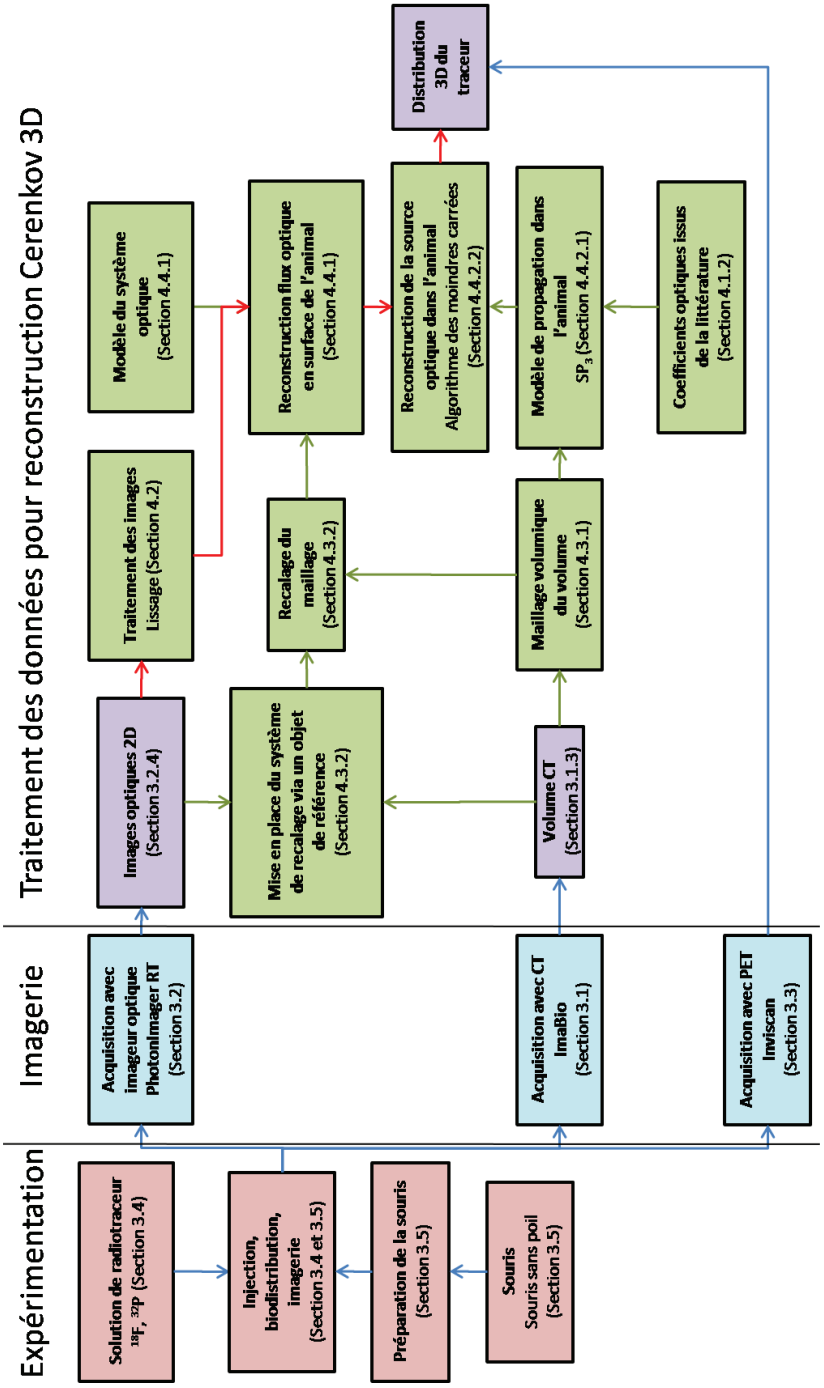

FIGURE 2.1 – Schéma des étapes nécessaires à la reconstruction Cerenkov 3D dans le cadre de cette thèse.

Chapitre 3

Plateforme d'imagerie ImaBio : AMISSA

AMISSA (A Multimodality Imaging System for Small Animal) est une plateforme d'imagerie multimodale dédiée au petit animal. Elle se compose d'une animalerie pouvant accueillir jusqu'à 200 souris, d'un TDM-X (TomoDensitoMètre à Rayons X), d'un SPECT, d'un PET Inviscan et d'un PhotonImager RT de BiospaceLab. Le cyclotron CYRCE est accessible pour la production d'isotopes radioactifs et la synthèse de radiotraceurs.

3.1 TDM-X

3.1.1 Présentation générale

Le TDM-X (TomoDensitoMètre à rayon X) est un instrument développé au sein du groupe ImaBio (fig. 3.1). Il permet d'effectuer une image anatomique 3D sur petit animal. L'acquisition se fait avec une source de rayons X projetant un faisceau conique sur un détecteur. La source est un tube à rayons X (L9181-02 de HAMAMATSU) et le détecteur (C7942CA-22 de HAMAMATSU) est un capteur composé d'un cristal scintillant (CsI (Tl)) et de CMOS. Chaque image enregistrée par le détecteur est appelée projection. Pour reconstruire un volume (image 3D), des projections sont acquises sur 360° autour de l'animal puis ces projections sont utilisées par un algorithme de reconstruction.

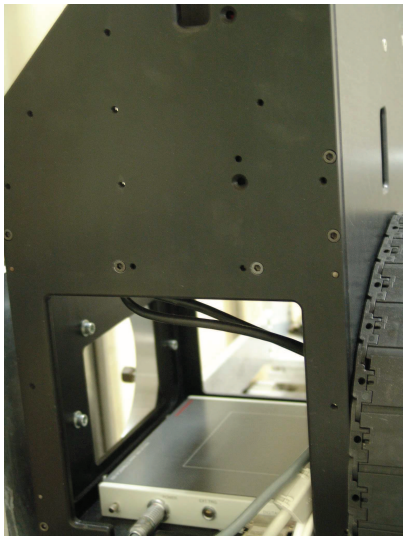

FIGURE 3.1 – Photographie du TDM-X conçu à ImaBio.

3.1.2 Protocole d'acquisition

L'animal est placé sur un lit qui est positionné entre la source et le détecteur. L'ensemble source et détecteur tourne autour de l'animal et les projections sont faites à différents angles. Le nombre de projections et l'intervalle angulaire entre elles permettent de reconstruire des images avec des résolutions différentes (tab. 3.1). Les paramètres de la source et du détecteur peuvent être modifiés pour l'acquisition.

Nombre de projections	Binning (informatique)	Temps d'acquisition	Temps de reconstruction	Taille voxel
768	1x1	5 min 58 s	pendant l'acquisition	0,1x0,1x0,1 mm^3
768	2x2	2 min 56 s	pendant l'acquisition	0,1x0,1x0,1 mm^3
360	4x4	42 s	pendant l'acquisition	0,2x0,2x0,2 mm^3

TABLE 3.1 – Exemples des modes d'acquisition possibles avec le TDM-X.

3.1.3 Reconstruction d'image

L'image finale est un volume voxelisé dans lequel la valeur de chaque voxel est la valeur de l'atténuation reconstruite (fig. 3.2). La reconstruction est effectué par un algorithme

FBP (Filtered Back Projection).

La taille du volume est de 5.12 x 3.84 x 7.2 cm^3. Le nombre de voxels dépend du protocole d'acquisition et des paramètres de reconstruction choisis.

FIGURE 3.2 – Image CT obtenue après reconstruction.

3.2 PhotonImager RT

3.2.1 Présentation générale

L'imageur optique utilisé est un PhotonImager RT de BiospaceLab (fig. 3.3). Il permet d'acquérir des images optiques par comptage du nombre de photons qui arrivent sur chaque pixel du détecteur qui est une matrice CCD. Les images peuvent être obtenues avec ou sans source d'illumination (illuminateur). Dans le cas où la source est allumée, l'image obtenue permet de prendre une photographie de l'objet. Si l'illuminateur est éteint, on observe le signal optique qui est émis par l'objet, qui ne doit pas être confondu avec le signal émis par la source au sein de l'objet. Il est possible de voir chacune des images et de les superposer, l'image photo étant représentée en nuances de gris et le signal optique en couleur. Une roue à filtre est installée pour sélectionner la gamme de longueurs d'onde observée. L'imageur est aussi équipé d'une source de longueur d'onde variable pour l'imagerie de fluorescence. Il dispose de plusieurs équipements pour l'imagerie de souris comme des becs pour une anesthésie gazeuse ou encore un module 4 vues qui permet d'observer une souris sous 4 angles de vues différents : dessus, dessous et sur les 2 côtés gauche et droite (fig. 3.4). Un picoprojecteur permet, combiné au module 4 vues, d'obtenir une représentation 3D de la surface de l'animal par projection d'une lumière structurée sur l'objet.

FIGURE 3.3 – Photographie du Photon Imager RT de Biospace Lab.

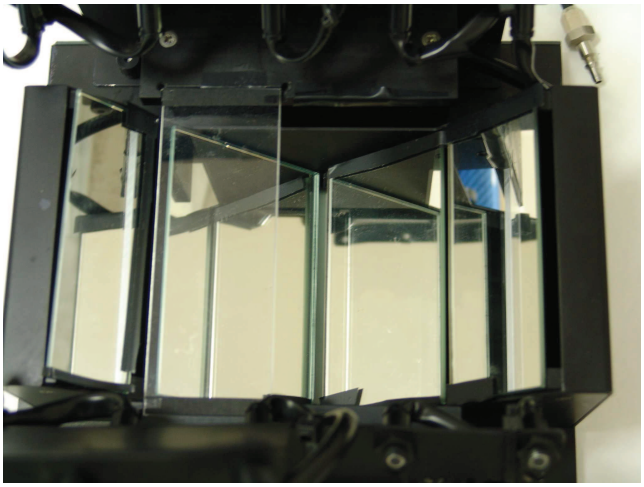

FIGURE 3.4 – Photographie du module 4 vues.

3.2.2 Chaîne de détection

Le signal émis n'est pas directement collecté par le capteur CCD, les photons subissent plusieurs traitements avant (fig. 3.5). Dans un premier temps, les photons rencontrent un filtre situé sur la roue à filtre. Les photons transmis sont alors récupérés par un objectif (Canon EF-L 24 mm) qui focalise le signal sur le tube intensificateur de lumière.

Le tube intensificateur est composé de plusieurs parties dont l'ensemble permet de passer d'un photon d'énergie quelconque à un grand nombre de photons dans une bande étroite du spectre. Pour cela, le tube dispose d'une photocathode qui convertit les photons en électrons. Une galette de microcanaux (MCP) amplifie le signal électronique puis un écran phosphorescent génère des photons à partir des électrons incidents. Les photons sont ensuite collectés par le capteur CCD.

Etant donné la présence du tube dans la chaîne de détection, l'information qui nous intéresse n'est pas le nombre de photons incidents sur le capteur CCD mais les photons incidents sur le tube. Il est possible de remonter au nombre de photons incidents sur le tube via un gain. Un photon incident sur le tube est appelé « coups ». Le « détecteur » désignera l'ensemble tube et capteur CCD. Un coup sur le détecteur signifie qu'un photon a été récupérer sur le tube.

Entre l'objet et le détecteur, le photon ne subit que des changements de direction propres à la géométrie du dispositif et à l'objectif à l'exception du filtre qui induit une variation sur la photométrie. Le tableau 3.2 compare le PhotonImager aux systèmes IVIS.

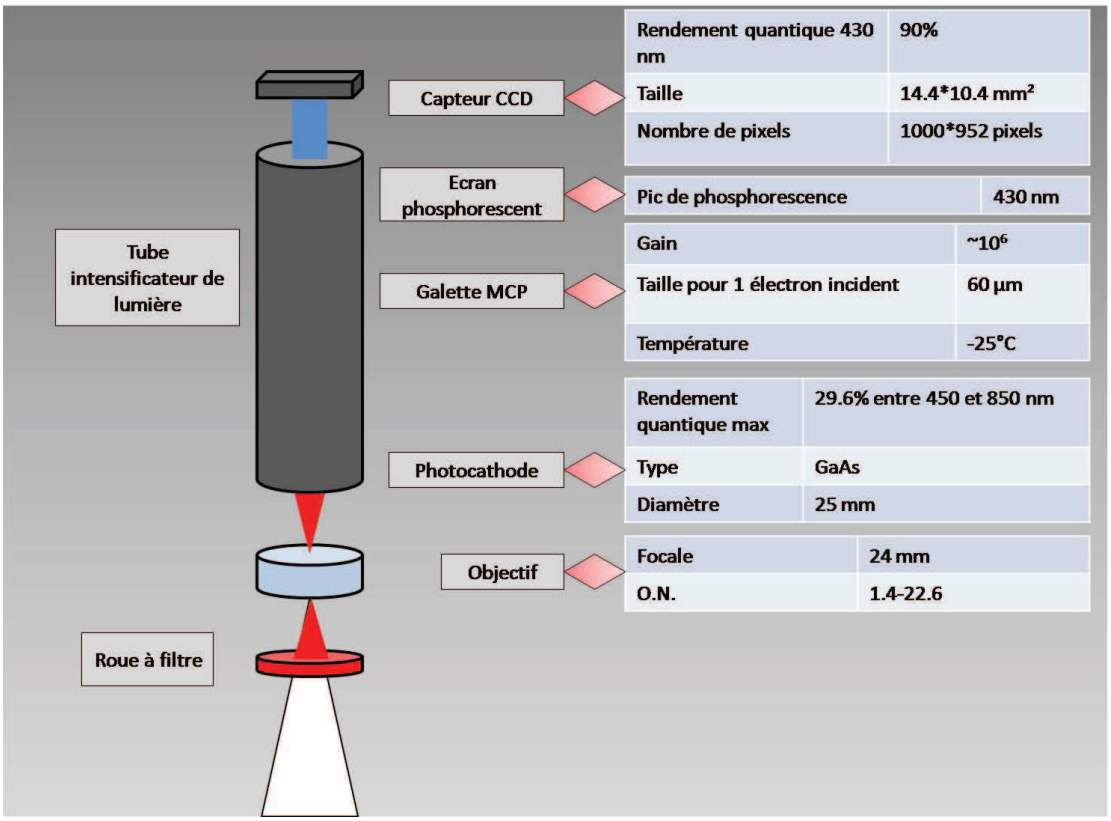

FIGURE 3.5 – Chaîne de détection d'un photon optique.

Système	IVIS 100	IVIS 200	PhotonImager RT
Nombre de pixels	2048 x 2048	1920 x 1920	1000 x 752
Efficacité quantique	85% 500-700nm, >30% 400-900nm	85% 400-700nm, >50% 350-900nm	29,6% 450-850nm, >4% 400-900nm
Objectif	f/0,95-f/16 ; 50mm	f/1-f/8	f/1,4-f/22,6 ; 24mm

TABLE 3.2 – Comparaison des caractéristiques des systèmes Xenogen IVIS avec le PhotonImager

3.2.3 Géométrie du système optique

La connaissance de la géométrie permet d'associer un pixel du détecteur à une zone en surface de l'objet. Les photons émis par l'objet interagissent avec 2 parties au moins : le filtre et l'objectif. Les miroirs jouent aussi un rôle lors de l'utilisation du module 4 vues.

La roue à filtre contient 7 emplacements pour 7 filtres. L'un des emplacements est vide pour des acquisitions sans filtre. Les autres sont équipés de filtre passe-haut à 615 nm, 660 nm, 700 nm et 770 nm ; et de 2 filtres passe-bande dans les gammes 755-805 nm et 790-840 nm.

L'objectif possède une distance focale de 24 mm et d'une ouverture numérique allant de 1.4 à 22.6 réglable via un diaphragme. Le focus de l'objectif est réglable entre 280 mm et l'infini.

L'objet est positionné sur le plateau ou sur le module 4 vues. La distance entre le plateau et l'objectif est réglable entre 440 mm (position basse) et 280 mm (position haute). Lors de l'utilisation du module 4 vues, l'objet est situé 80 mm au dessus du plateau.

3.2.4 Images obtenues

Une image est la combinaison de 2 acquisitions optiques, l'une réalisée avec l'illuminateur comme source d'éclairement (mode photo) et l'autre sans éclairage extérieur à l'objet (mode bioluminescence). L'image photographique est représentée en nuance de gris (fig. 3.6). L'image acquise en mode bioluminescence est représentée en couleur (fig. 3.7). Il est possible d'avoir les images en fonction du temps lors d'une acquisition en mode bioluminescence. Il est possible de cumuler ces images pour obtenir un meilleur ratio signal sur bruit.

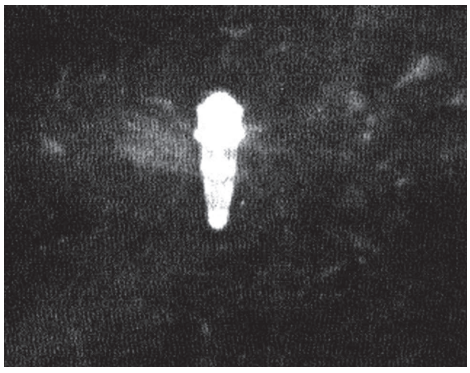

FIGURE 3.6 – Image photographique classique d'un tube eppendorf.

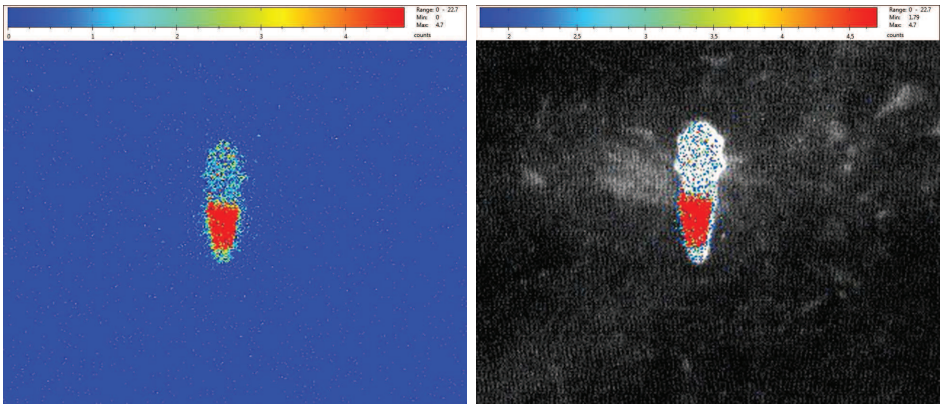

FIGURE 3.7 – Image en mode bioluminescence brut (à gauche) et superposition de l'image bioluminescence sur l'image photographique (à droite).

L'image en mode bioluminescence peut être traitée avec le logiciel de visualisation. Seules les images brutes sont utilisées pour réaliser de l'imagerie Cerenkov 3D. Les images en mode bioluminescence traitées par le logiciel sont lissées puis seuillées (fig. 3.8). Ces images traitées donnent une information qualitative et permettent d'observer le signal Cerenkov et la photo de l'objet simultanément.

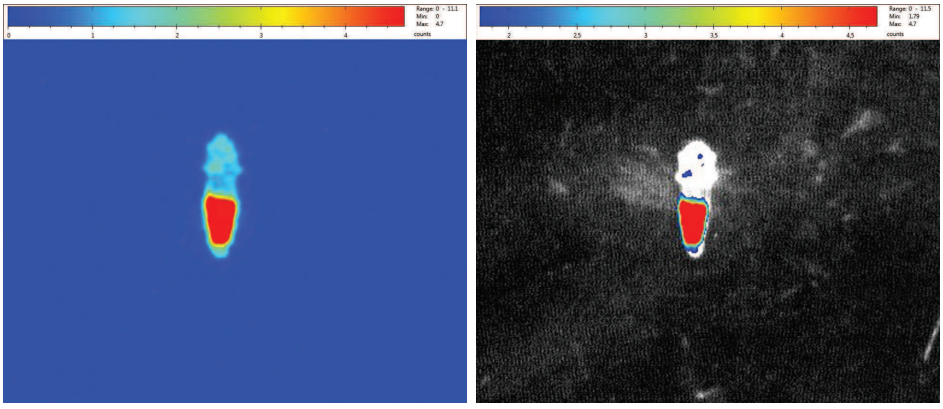

FIGURE 3.8 – Image en mode bioluminescence lissée (à gauche) et superposition de l'image bioluminescence lissée sur l'image photographique (à droite).

Dans le cas de l'utilisation du module 4 vues, une image contient plusieurs angles de vue de l'objet. Les vues de dessus, dessous et latérales sont alors visibles sur une seule image acquise (fig. 3.9).

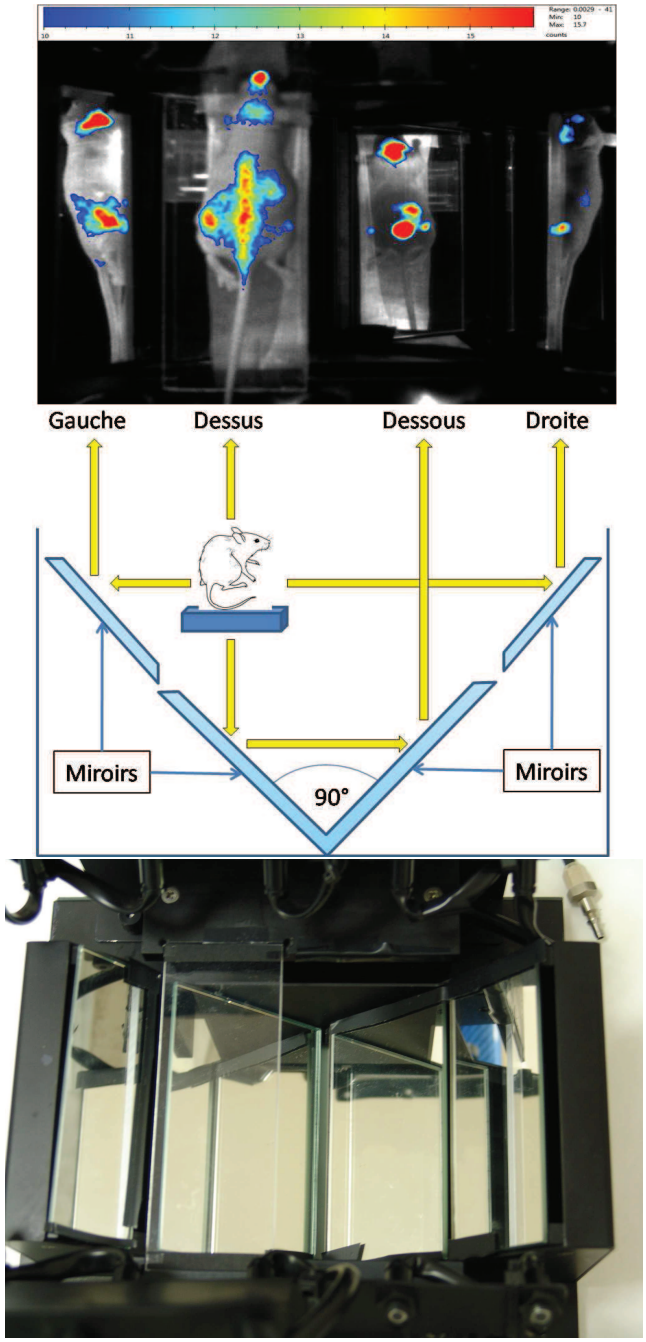

FIGURE 3.9 – Image obtenue en utilisant le module 4 vues.

3.2.5 Principe de reconstruction du PhotonImager

Un logiciel permet de réaliser des reconstructions 3D de la distribution d'une source optique à partir d'images acquises en mode 4 vues. Les 4 vues sont découpées pour former les vues de dessus (0°), dessous (180°), gauche (90°) et droite (270°). La surface de l'objet est acquise en utilisant un pico-projecteur. Le flux optique en surface est reconstruit en projetant parallèlement les images de chaque vue. La résolution de l'équation de diffusion en considérant le milieu comme étant de l'eau permet d'obtenir la distribution de la source optique.

Toutefois, ce procédé de reconstruction est imprécis sur plusieurs points et donne des résultats erronés.

Une première approximation faite est que les 4 vues sont exactement à 90° autour de l'animal. Ceci est faux puisque l'animal n'est jamais placé directement sur l'axe optique et les vues latérales ne sont pas à 90° par rapport à la vue de dessus. Des objets symétriques (fig. 3.10) ne sont pas reconstruits de la même manière pour chaque côté (fig. 3.11). De plus, le pico projecteur ne permet d'acquérir que la surface visible du dessus. La surface du dessous est considérée comme plane (fig. 3.11). Ce qui n'est pas forcément le cas dans le cas de parties qui ne touchent pas le support comme les oreilles si la souris est sur le ventre, ou les pattes si la souris est sur le dos.

Pour finir, la reconstruction de la source optique fait l'approximation d'un milieu homogène rempli d'eau pour la souris. L'approximation de tissus biologiques par un milieu transparent comme l'eau fausse énormément la quantification de la source reconstruite. De plus, l'approximation d'un milieu homogène est une limite de cette technique puisque le pico-projecteur ne permet pas d'obtenir des informations anatomiques internes à l'animal.

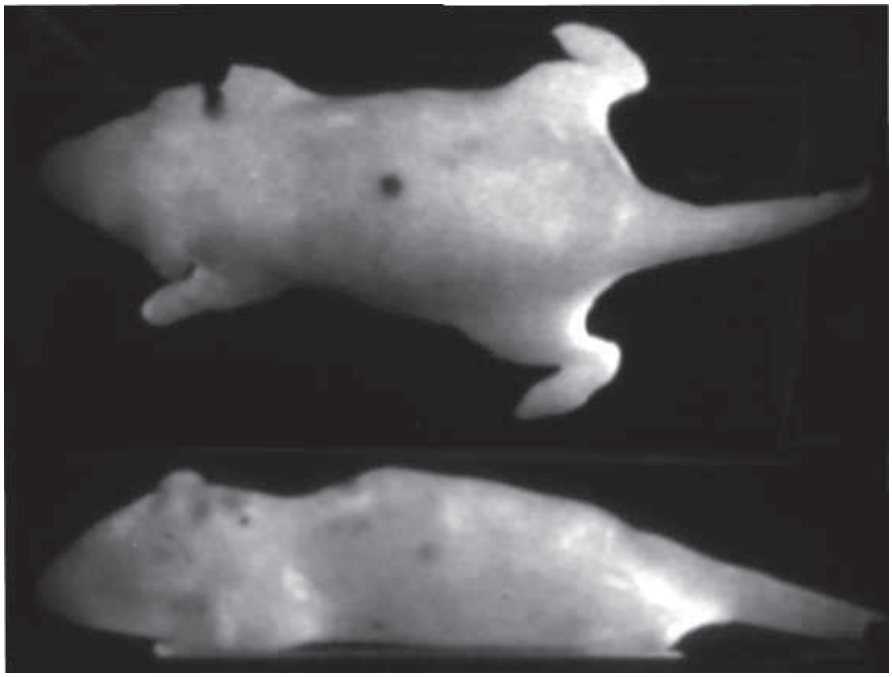

FIGURE 3.10 – Photographie du modèle de souris.

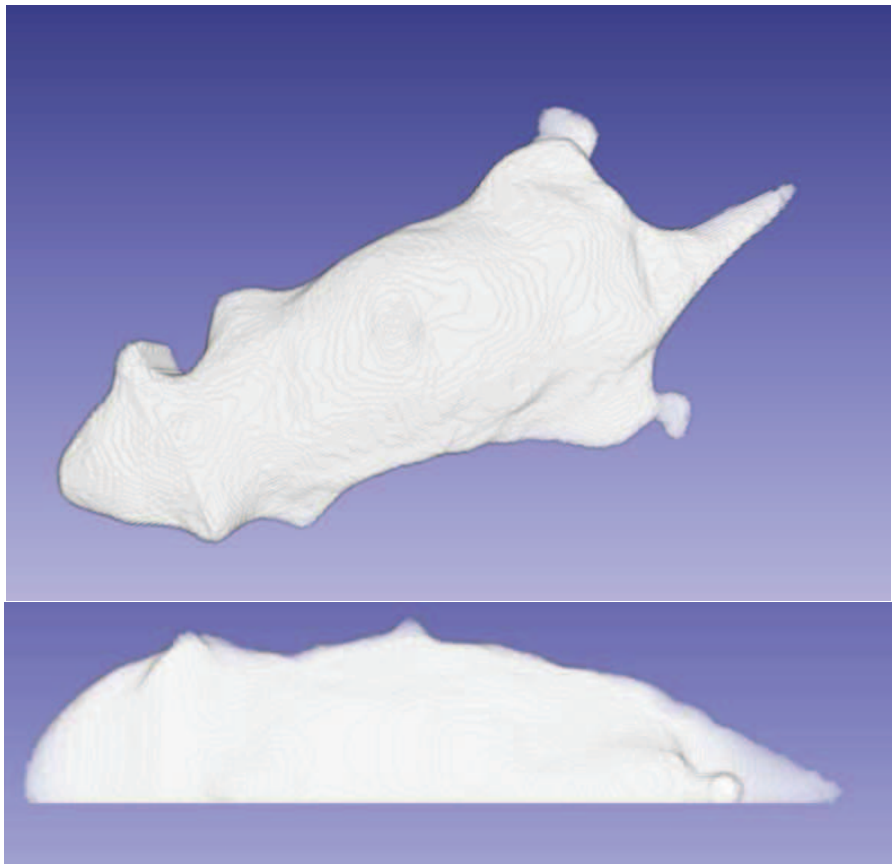

FIGURE 3.11 – Reconstruction de la surface réalisé par le PhotonImager RT avec le mode 4 vues et le picoprojecteur.

3.3 PET Inviscan

Un TEP Inviscan est présent à la plateforme d'imagerie d'ImaBio (fig. 3.12). Il est constitué de 2 anneaux contenant chacun 4 détecteurs fixes. Il permet d'obtenir une image 3D corps entier de la distribution d'un radiotraceur émetteur de positons. Les caractéristiques du TEP sont listées dans le tableau 3.3. Pour obtenir une image TEP, il est important de ne pas saturer les détecteurs. Cette saturation se produit si l'activité est supérieure à 10 MBq.

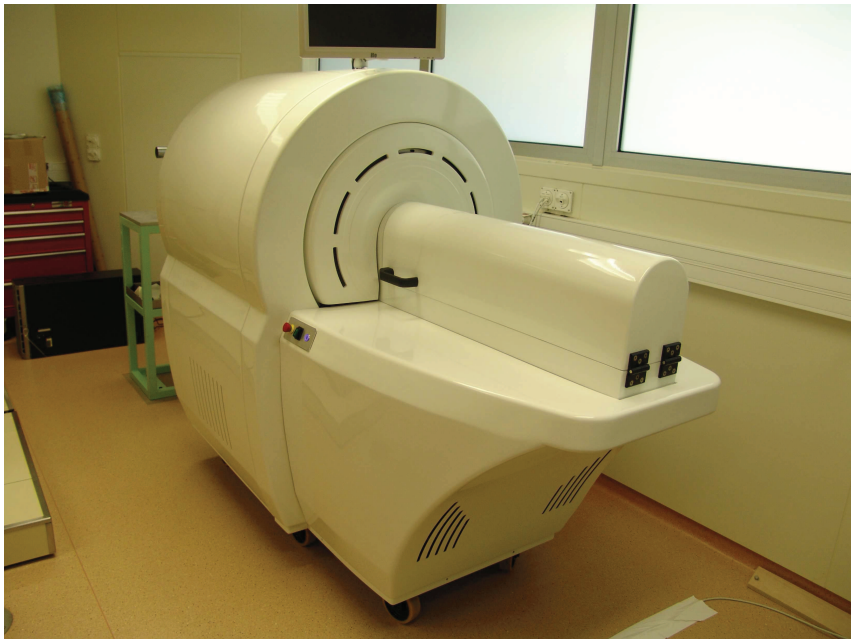

FIGURE 3.12 – Photographie du PET Inviscan.

Sensitivité	9.8 % (entre 250 keV et 750 keV)
Résolution spatiale	1.1 mm
Champ de vue axial	94 mm
Champ de vue trans-axial	80 mm
Résolution en énergie	14 %
Résolution temporelle	1.4 ns

TABLE 3.3 – Tableau des caractéristiques du PET Inviscan

3.4 Isotopes et traceurs utilisés

3.4.1 Règlementation liée à l'utilisation de sources radioactifs

La manipulation de sources radioactives est réglementée par l'ASN (Autorité de Sureté Nucléaire).

Lors de la manipulation avec des sources radioactives, des consignes de radioprotection sont mises en place pour protéger les personnes et le matériel. Les activités que l'on peut posséder sont limitées en fonction de l'isotope. Les déchets contaminés doivent attendre 10 périodes avant de pouvoir être jetés suivant la même procédure que les déchets propres.

Dans le cadre de ma thèse, j'ai manipulé du ^{18}F et du ^{32}P décrits dans la suite. La majeure partie de mon travail a été réalisé principalement avec du ^{18}F car celui-ci est produit directement par le cyclotron CYRCE ou commercial. Il est possible de manipuler des activités importantes (au delà de 20 MBq par injection) et les déchets générés peuvent être considérés comme propre au bout de 20h. A l'inverse, il ne nous est possible de possèder que 37 MBq au maximum de ^{32}P et les déchets doivent être laissés en décroissance pendant plusieurs mois (20 semaines) avant de pouvoir être jetés.

3.4.2 Fluor-18

Le fluor-18 est un isotope du fluor présent naturellement à l'état de trace. L'isotope courant du fluor est le fluor-19. Le ^{18}F se désintègre en oxygène-18 par émission β^+ à 96.9 % avec une période de demi-vie de 109.77 min. L'énergie des positons émis est de 633 keV au maximum. La désintégration des positons engendre des photons gamma à une énergie de 511 keV. Ces rayonnements demandent de manipuler avec des équipements de radioprotection. Les solutions sont manipulées dans des contenants plombés. Les seringues sont protégées avec des caches plombés. La manipulation des souris après injection est à contrôler pour limiter l'exposition.

Le ^{18}F et les radiotraceurs utilisés, à l'exception du FDG qui est commercial, proviennent du cyclotron CYRCE.

3.4.2.1 FNa

Le FNa est un sel qui se dissout en ions F^- et Na^+. Dans un organisme vivant, le F^- a une grande affinité pour les os. Les articulations captent davantage de F^- car l'os est renouvelé régulièrement. Il traverse facilement les tissus et le temps de biodistribution chez une souris est de 40 à 60 min suivant le mode d'injection.

3.4.2.2 FDG

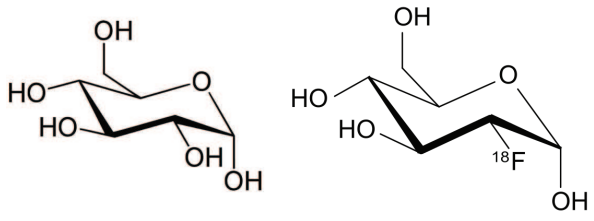

FIGURE 3.13 – Molécule de glucose à gauche et molécule de fluorodéoxyglucose à droite marquée au fluor-18.

Le FDG (FluoroDéoxyGlucose) est un dérivé du glucose (fig. 3.13). Le glucose est un sucre utilisé par les cellules pour produire de l'ATP (Adénosine TriPhosphate). L'ATP est la source d'énergie d'une majorité de processus cellulaires. Le FDG est capté par les cellules de la même manière que le glucose mais la modification chimique du FDG empêche une complète assimilation du FDG par les cellules et le fluor-18 reste piégé dans la cellule. Le temps de biodistribution chez une souris est de 60 à 90 min.

3.4.3 Phosphore-32

Le phosphore-32 est un isotope du phosphore présent naturellement à l'état de trace. L'isotope courant du phosphore est le phosphore-31. Le ^{32}P se désintègre en soufre-32 par émission β^- avec une période de demi-vie de 14.29 jours. L'énergie des électrons émis est de 1709 keV au maximum. Il est utilisé dans le domaine médical pour la thérapie et le diagnostic.

3.4.3.1 ATP

L'ATP (Adénosine TriPhosphate) est une molécule utilisée par les cellules comme source d'énergie directe (fig. 3.14). La réaction produisant l'énergie provient des groupements phosphate. L'ATP peut réagir pour chaque groupement phosphate.

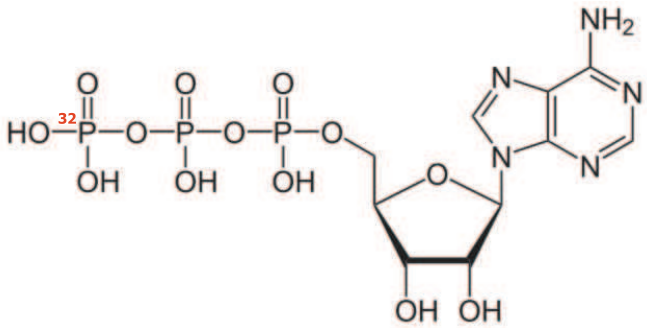

FIGURE 3.14 – Molécule d'ATP Gamma marquée avec un phosphore-32.

Le ^{32}P-ATP est une molécule d'ATP marqué avec un atome de ^{32}P. Ce marquage peut être effectué sur un ou plusieurs des phosphates. Nous avons utilisé de l'ATP Gamma ^{32}P où le phosphore du premier groupe phosphate à réagir est un atome de ^{32}P.

3.5 Expérimentation animal

3.5.1 Comité d'éthique et souris utilisées

Toutes les manipulations réalisées sur les souris sont en accord avec le comité d'éthique (AL/05/12/02/13). Les expérimentations animales ont été réalisées sur des souris sans poils ou des souris immuno-déficientes ne disposant pas de poils afin de s'affranchir des problèmes optiques dus aux poils.

3.5.2 Anesthésie

Pour manipuler et imager les souris, celles-ci sont anesthésiées. Plusieurs types d'anesthésie peuvent être utilisés. L'anesthésie par injection d'un mélange de kétamine/xylazine permet d'endormir une souris pour une durée, dépendant de la quantité injectée, pouvant aller jusqu'à une heure. Ce type d'anesthésie présente l'intérêt de pouvoir manipuler l'animal que ce soit pour injecter un produit, faire de la chirurgie ou pour l'imager. Toutefois, elle présente un risque pour la survie de l'animal si l'anesthésie est trop importante et est à éviter pour des anesthésies longues ou répétées. La seconde méthode d'anesthésie consiste à faire respirer à la souris un gaz contenant de l'isoflurane, dit anesthésie gazeuse. La souris doit respirer en continu le gaz pour dormir. Si elle arrête de respirer l'isoflurane, elle se réveille en moins d'une minute. Ce type d'anesthésie est inoffensive pour la souris mais l'anesthésie n'est pas assez forte pour bloquer la douleur de l'animal et cette anesthésie est principalement utilisée pour l'imagerie.

3.5.3 Injection solution radioactive

Les injections pratiquées sur les souris sont réalisées par injection intraveineuse (IV) ou injection intrapéritonale (IP). Le mode d'injection influe sur la manière de distribuer le produit injecté. La distribution dans tout l'animal est plus rapide dans le cas d'une injection IV que pour une injection IP. Il est possible d'injecter un plus grand volume en IP qu'en IV où 250 μL est le maximum injectable. La solution à injecter ne doit pas être toxique.

3.5.4 Biodistribution

Le temps de biodistribution est le temps biologique pour que la distribution de la molécule injectée atteigne un palier pendant lequel la distribution varie peu. Ce temps est dépendant de la méthode d'injection et de la molécule injectée. Le temps de biodistribution est plus faible dans le cas d'une injection IV que d'une injection IP. Les molécules de petites tailles se distribuent plus vite car elles peuvent passer plus facilement les membranes. Les molécules plus imposantes sont souvent dépendantes de mécanismes biologiques pour pouvoir traverser les membranes.

3.5.5 Imagerie

Pour réaliser de la tomographie Cerenkov *in vivo*, un protocole expérimental doit être mis en place. La manipulation des souris et la partie imagerie nécessite de garder l'animal

endormi pendant une longue période. Plusieurs paramètres peuvent être ajustés pour l'imagerie optique. En particulier, le choix du temps d'acquisition est important. En imagerie Cerenkov, les facteurs limitants sont la biobistribution et la demi-vie de l'isotope. Dans le cas du ^{18}F, son temps de demi-vie est de 109,7 min. Afin d'avoir une activité que l'on peut considérer constante, il faut un temps d'acquisition court par rapport au temps de demi-vie. De plus, des images sont acquises pour chaque filtre. Le temps d'acquisition choisi sera à multiplier par 7. Le temps d'acquisition pour du ^{18}F est choisi au maximum à 5 min. Au bout de 5 min, l'activité a diminué de 3,1 %. Au bout de 35 min, soit après acquisition avec chaque filtre, l'activité a diminué de 19,9 %. Pour le ^{32}P, son temps de demi-vie (14,3 jours) n'est pas limitant comparé à la biodistribution. Le temps d'acquisition peut être ajusté.

3.5.6 Protocole expérimental

La souris est d'abord anesthésiée par injection IP pour la préparer à l'injection du radiotraceur. La souris est alors mis à part le temps de la biodistribution. Après biodistribution, elle est gardée endormie pendant la phase d'imagerie. Dans certains cas où la souris se réveille durant la biodistribution, afin de limiter la manipulation de souris injectées avec un produit radioactif, la souris est euthanasiée. Le tableau 3.4 présente les étapes expérimentales pour la réalisation de l'imagerie Cerenkov avec les temps asssociés par étape.

Protocole expérimental classique	
Préparation seringue	5 min
Préparation souris	5 à 15 min
Injection, récupération seringue et souris	5 min
Biodistribution	40 à 90 min (après injection)
Acquisition CT	1 min (acquis pendant la biodistribution)
Acquisition optique	7 x 5 min (après biodistribution)
Total	120 min minimum

TABLE 3.4 – Etapes expérimentales pour réaliser de l'imagerie Cerenkov sur souris

Chapitre 4

Optique

4.1 Phénomènes optiques

Le rayonnement Cerenkov s'étendant dans le domaine visible, les problématiques associées sont celles qui se posent en optique. Cette partie vise dans un premier temps à rappeler les notions d'optiques puis à décrire les méthodes qui permettent de modéliser les phénomènes optiques.

4.1.1 Propagation dans un milieu

La lumière se déplaçant dans le vide n'interagit pas et possède une vitesse de 299 792 458 $m.s^{-1}$. Lorsqu'elle se trouve dans un milieu, sa propagation est altérée. La vitesse de la lumière diminue et les photons peuvent interagir avec le milieu. Le changement de vitesse de la lumière est caractérisé par l'indice optique du milieu n. La vitesse de la lumière vaut alors v=c/n.

La lumière peut interagir de 2 manières différentes : absorption ou diffusion. La lumière est transmise si elle n'interagit pas. Lorsque des photons sont absorbés, leur énergie est transmise au milieu et les photons disparaissent. Des photons I_0 traversant une épaisseur homogène d sont partiellement absorbés et le nombre de photons transmis I est donné par la loi de Beer-Lambert :

$$I(x) = I_0 * e^{(-\mu_a.d)} \tag{4.1}$$

μ_a est le coefficient d'absorption du milieu. Il représente la probabilité qu'un photon soit absorbé par unité de longueur. Ce modèle ne tient pas compte de la diffusion.

Les photons diffusés conservent leur énergie mais leur direction de propagation change. La diffusion diffère suivant la taille des particules qui la provoque. Si la taille des particules est bien inférieure à la longueur d'onde du photon, la diffusion est une diffusion de Rayleigh. Si la taille des particules est du même ordre de grandeur que la longueur d'onde, la diffusion suis une loi de Mie. Dans le cas où les particules sont bien plus grandes que la longueur d'onde, on considère que la lumière change de milieu, les phénomènes qui se produisent sont

alors dominés par la géométrie de l'interface et par la différence d'indice optique. Alors que la diffusion de Mie fluctue peu en fonction de la longueur d'onde, la diffusion de Rayleigh est très dépendante de la longueur d'onde. La diffusion est généralement anisotrope (fig. 4.1). A l'instar de l'absorption, on peut définir un coefficient de diffusion μ_s qui représente la probabilité qu'un photon soit diffusé par unité de longueur.

La nouvelle direction d'un seul photon ne peut pas être déterminée mais pour un flux de photons, il est possible d'avoir une distribution des photons diffusés en fonction de l'angle. Pour un flux unidirectionnel, la diffusion est un phénomène à symétrie cylindrique par rapport à la direction de propagation. La distribution peut alors s'exprimer en fonction de l'angle entre le faisceau incident et le faisceau diffusé.

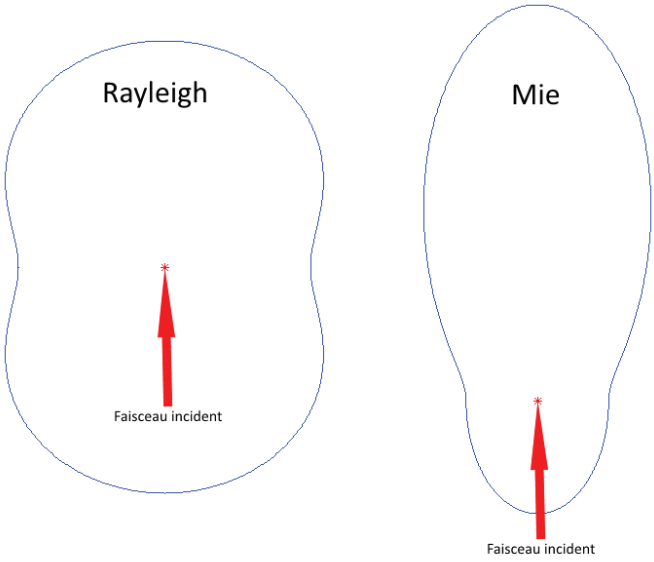

FIGURE 4.1 – Profils de diffusion de Rayleigh (à gauche) et de Mie (à droite).

La propagation de la lumière dans un milieu donné est entièrement décrite par l'équation de transfert aussi appelée équation de Boltzmann [69] :

$$\frac{\partial L(\vec{r},\hat{s},t)/c}{\partial t} = -\hat{s}.\nabla L(\vec{r},\hat{s},t) - \mu_t L(\vec{r},\hat{s},t) + \mu_s \int_{4\pi} L(\vec{r},\hat{s},t)P(\hat{s}'.\hat{s})d\Omega' + S(\vec{r},\hat{s},t) \quad (4.2)$$

Avec :
– L représente la radiance en $\mathrm{W.m^{-2}}$

– μ_t est la somme de μ_a et de μ_s

- S représente la génération des photons en chaque point du milieu en W.m^{-3}
- ŝ représente la direction de propagation
- Ω' représente l'angle solide en sr
- $\frac{\partial L(\vec{r},\hat{s},t)/c}{\partial t}$ représente la variation de l'intensité du flux L en un point $\vec{r}$ et dans une direction donnée $\hat{s}$
- $-\hat{s}.\nabla L(\vec{r},\hat{s},t)$ représente la propagation du flux optique
- $\mu_t L(\vec{r},\hat{s},t)$ représente les pertes dûes à l'absorption et la la diffusion des photons du flux
- $\mu_s \int_{4\pi} L(\vec{r},\hat{s},t)P(\hat{s}'.\hat{s})d\Omega'$ représente les photons incidents en $\vec{r}$ et diffusés dans la direction $\hat{s}$
- $S(\vec{r},\hat{s},t)$ représente les photons générés en $\vec{r}$ dans la direction $\hat{s}$

La fonction de phase P décrit la probabilité pour un photon incident de direction ŝ subissant une diffusion d'être diffusé dans la direction ŝ'. Elle décrit le caractère anisotrope de la diffusion.

Cette équation prend en compte les phénomènes d'absorption et de diffusion, ainsi que le profil de la diffusion. Ces paramètres sont dépendants du milieu dans lequel on se place.

4.1.2 Propagation dans des tissus biologiques

Visuellement, les tissus biologiques sont rarement transparents. Les tissus biologiques sont composés d'un grand nombre de structures de tailles variables allant de la molécule ($\tilde{}10^{-10}$ m) à celle d'un organe (10^{-3} m pour une souris, plus d'un cm pour le foie). La forme de ces structures est complexe. De plus, ces structures ne sont pas fixes et peuvent bouger ou varier de forme dans un organisme vivant. Il en résulte que les propriétés optiques sont très hétérogènes dans un milieu biologique.

Des méthodes pour déterminer les coefficients optiques *in vitro* existent [70]. Le couplage entre absorption et diffusion lors des mesures implique généralement d'évaluer ces phénomènes de manière simultanée.

Chaque méthode n'utilise pas le même protocole expérimental ni le même modèle de propagation.

La manière de préparer l'échantillon à mesurer influe aussi sur la mesure.

De plus, les propriétés optiques varient suivant la longueur d'onde.

Tous ces points rendent difficile l'accès à l'information des paramètres optiques pour des tissus biologiques *in vitro*.

Le problème diffère dans le cas *in vivo* puisque la géométrie n'est pas contrôlable. On

trouve un grand nombre de résultats expérimentaux pour la détermination des coefficients optiques dans la littérature [70, 71, 72] dont certaines essaient de trouver un modèle déterminant ces coefficients pour chaque longueur d'onde à partir de données expérimentales incomplètes [72, 73]. Ces modèles s'appuient souvent sur les propriétés optiques du sang pour l'absorption qui est présent dans tous les tissus mous et est le principal absorbant. L'eau (fig. 4.2), l'hémoglobine et l'oxyhémoglobine (fig. 4.3) sont les principaux composants du sang. D'autres éléments très absorbants sont la peau et les poils.

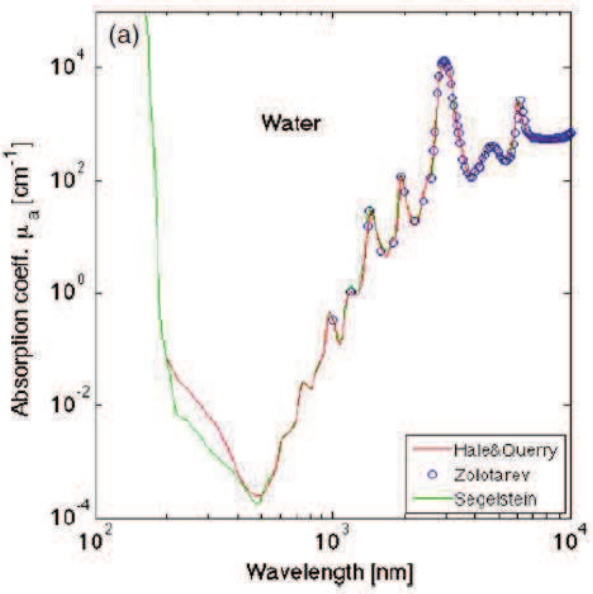

FIGURE 4.2 – Spectre d'absorption de l'eau (tiré de [72]).

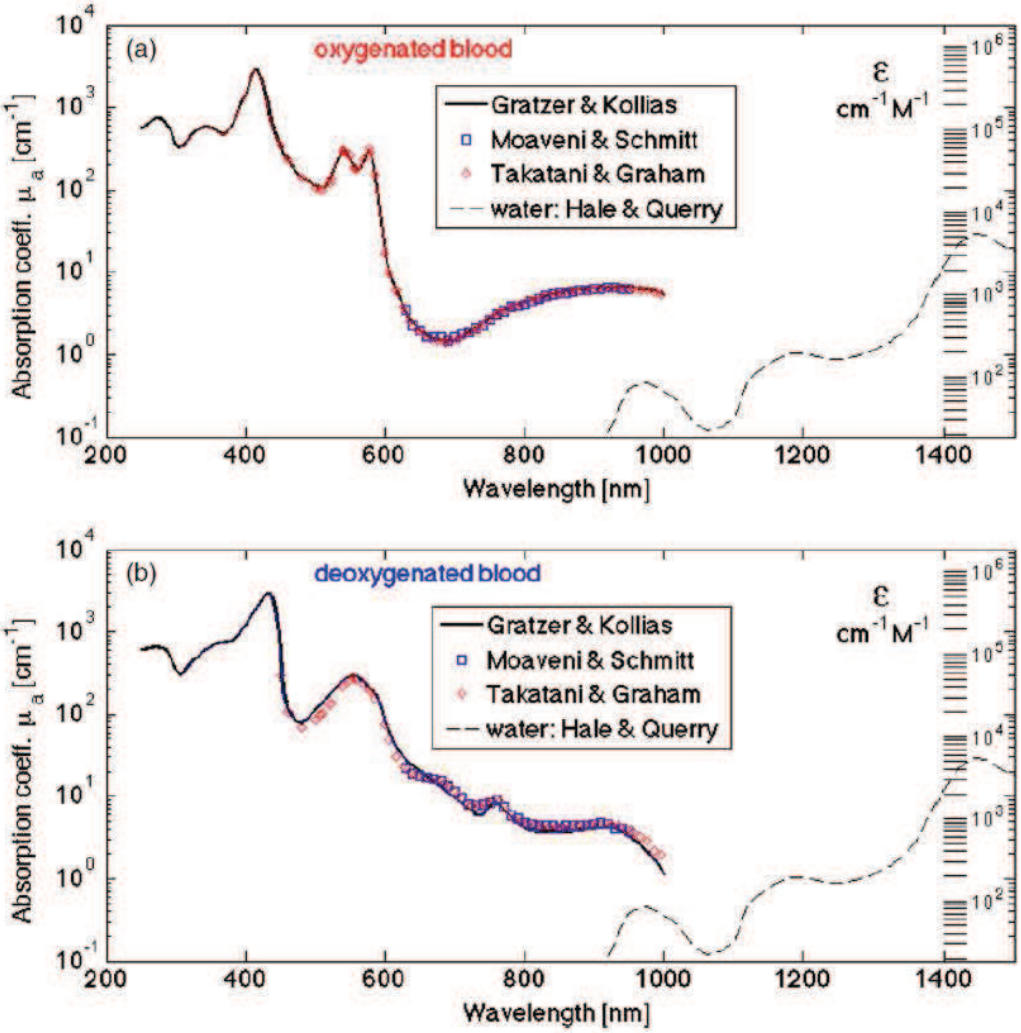

FIGURE 4.3 – Spectre d'absorption des composants absorbants du sang, l'hémoglogine et l'oxyhémoglobine (tiré de [72]).

La diffusion est due à la taille des particules. Dans les longueurs d'onde auxquelles la lumière n'est pas trop absorbée, la diffusion de Rayleigh est négligeable. Le modèle de Henyey-Greenstein est alors utilisé pour décrire la diffusion dans les tissus [74, 75].

$$P(cos(\theta)) = \frac{1 - g^2}{4\pi(1 + g^2 - 2g.cos(\theta))^{\frac{3}{2}}} \tag{4.3}$$

Où θ est l'angle de diffusion et g = <cosθ> est le coefficient d'anisotropie variant entre

-1 et 1. La valeur 0 représente une diffusion isotrope, les valeurs positives une diffusion majoritaire dans le sens du faisceau incident et les valeurs négatives une diffusion majoritaire dans le sens opposé au faisceau incident. La valeur de g, dépendant du type de diffusion, varie en fonction de la longueur d'onde (fig. 4.4).

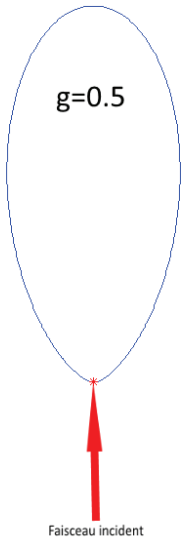

FIGURE 4.4 – Profil de diffusion du modèle de Henyey Greenstein avec g = 0,5.

Il est impossible expérimentalement de mesurer μ_s. La mesure de la diffusion dépend du caractère de la diffusion donc de g. Il est possible de mesurer expérimentalement μ_s' où μ_s'=μ_s.(1-g) (fig. 4.5). La valeur de μ_s ne peut être retrouvé qu'à partir de la connaissance de g.

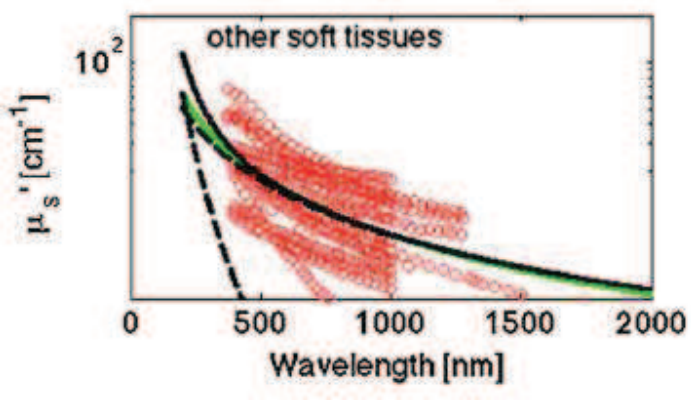

FIGURE 4.5 – Spectre de diffusion des tissus mous (tiré de [72]). En rouge, les données expérimentales. En tirets noirs, la contribution de la diffusion de Rayleigh ($<$ 500nm) et de Mie. En vert, un modèle ne considérant que la diffusion de Mie et en noir continu, un modèle général tenant compte des diffusions de rayleigh et Mie.

4.1.2.1 *A priori* sur les coefficients optiques

Les valeurs des coefficients optiques μ_a, μ_s' et g sont tirées de la littérature. En particulier, pour connaître ces valeurs à une longueur d'onde donnée, un modèle général [72] est utilisé :

$$\mu_a(\lambda) \simeq BS\mu_{a.oxy}(\lambda) + B(1-S)\mu_{a.deoxy}(\lambda) + W\mu_{a.water}(\lambda) \tag{4.4}$$

Avec :
- B est la fraction de volume sanguin dans le tissu
- W est la fraction de volume d'eau dans le tissu
- S est le taux de saturation de l'oxyhémoglobine dans le sang, en pourcentage de la quantité d'hémoglobine totale
- $\mu_{a.oxy}(\lambda)$ est le coefficient d'absorption de l'oxyhémoglobine à la longueur d'onde λ
- $\mu_{a.deoxy}(\lambda)$ est le coefficient d'absorption de la deoxyhémoglobine à la longueur d'onde λ
- $\mu_{a.water}(\lambda)$ est le coefficient d'absorption de l'eau à la longueur d'onde λ

Le modèle complet fait aussi intervenir des composés comme la mélanine, la graisse, le β-carotène et la bilirubine. Ces composés sont spécifiques à certains tissus comme la peau.

Pour un tissu mou, la quantité de ces composés est négligée. En particulier, aucune donnée expérimentale n'est fourni sur ces composés pour les tissus autre que la peau.

$$\mu'_s(\lambda) \simeq a' \left(f_{Ray} \left(\frac{\lambda}{500} \right)^{-4} + (1 - f_{Ray}) \left(\frac{\lambda}{500} \right)^{-b_{Mie}} \right) \tag{4.5}$$

Avec :
- a' est égal à $\mu'_s(500nm)$
- f_{Ray} est la fraction de diffusion Rayleigh
- b_{Mie} est un terme représentant le pouvoir de diffusion par la diffusion de Mie

$$g(\lambda) \simeq cste \tag{4.6}$$

Par la suite, pour la mise en place du modèle direct, les coefficients ont été calculés et l'animal a été modélisé par un milieu homogène composé de muscle. Ces valeurs ont été calculées pour chaque bande spectrale (tab. 4.1).

Muscle	Bande 1 : 615-665 nm	Bande 2 : 665-700 nm	Bande 3 : 700-770 nm	Bande 4 : 755-805 nm	Bande 5 : 790-840 nm
μ_a	0,93	0,50	0,24	0,25	0,18
μ_s	9,90	8,23	6,61	5,59	4,94
g	0,9	0,9	0,9	0,9	0,9

TABLE 4.1 – Valeurs des coefficients optiques calculés pour les différentes bandes pour du muscle de souris

4.1.3 Phénomènes aux interfaces

Lors d'un changement de milieu, la lumière peut être transmise ou réfléchie. Ces phénomènes apparaissent lorsque le changement de milieu implique un changement d'indice optique.

Le flux optique est généralement différent de part et d'autre d'une interface. Si la source se trouve dans un des deux milieux, le flux dans le second milieu provient uniquement de la transmission du flux à l'interface. Au niveau de l'interface du premier milieu, le flux est égale à la l'éventuelle source plus le flux réfléchi [76] :

$$\Phi = \begin{cases} S + R.\Phi & \text{, dans le milieu} \\ T.\Phi = (1 - R)\Phi & \text{, en dehors du milieu} \end{cases} \tag{4.7}$$

Avec :
- S est la génération de photons dans le milieu à l'interface, non présente en dehors di milieu
- R est le pourcentage de réflexion
- T est le pourcentage de transmission, complémentaire avec R

La réflexion dépendant de l'angle des indices optiques et de l'angle, on considère la loi de réfraction :

$$R(cos(\theta_i)) = \begin{cases} \frac{1}{2}\left(\frac{n_i cos(\theta_s) - n_s cos(\theta_i)}{n_i cos(\theta_s) + n_s cos(\theta_i)}\right)^2 + \frac{1}{2}\left(\frac{n_i cos(\theta_i) - n_s cos(\theta_s)}{n_i cos(\theta_i) + n_s cos(\theta_s)}\right)^2 & \text{, si } \theta_i < \theta_c \\ 1 & \text{, si } \theta_i \geqslant \theta_c \end{cases} \tag{4.8}$$

Où les indices optiques et angles obéissent à l'équation de réfraction :

$$n_i sin(\theta_i) = n_s sin(\theta_s) \tag{4.9}$$

Avec :
- i et s font référence au milieu du faisceau incident et au second milieu dans lequel le faisceau est transmis respectivement
- n_i et n_s sont les indices optiques des différents milieux
- θ_i et θ_s sont les angles des faisceaux dans chacun des milieux
- θ_c est l'angle critique de réflexion totale

Cette condition de passage permet de décrire le flux de part et d'autre d'une interface. En particulier, elle permet d'exprimer le flux sortant J_+ dans une direction Ω à la surface, de normale sortante $\vec{n}$, d'un objet via la connaissance du flux interne Φ à l'objet :

$$J_+(r) = \int_{\Omega.\vec{n}>0} (1 - R(\Omega.\vec{n}))(\Omega.\vec{n})\Phi(r,\Omega)d\Omega \tag{4.10}$$

4.1.4 Optique géométrique

L'optique géométrique fait référence à l'ensemble des procédés qui modifie la distribution d'un flux mais ne modifie pas l'intensité totale du flux. Les miroirs et lentilles en font partie. Les phénomènes de surface comme la réflexion et la réfraction sont utilisé dans le cas de l'optique géométrique.

En pratique, les pertes dans ces dispositifs ne sont pas nulles mais sont souvent négligés. L'absorption du système optique est négligeable devant celle liée aux tissus biologiques.

4.1.4.1 Miroirs

Les miroirs plans ne déforment pas l'image. Ils reproduisent l'image dans une autre direction avec le même champ de vue par réflexion. De plus, 2 miroirs avec un angle de 90° permettent de récupérer la lumière avec la même direction que la lumière incidente. Le taux de réflexion n'est jamais de 100 % et varie avec la longueur d'onde. Toutefois, l'effet des miroirs est aussi négligé.

4.1.4.2 Objectif

Les caractéristiques optiques pour une lentille simple sont connues et une lentille est caractérisée par un seul paramètre : sa distance focale noté f'. Toutefois, des aberrations sont aussi observées et généralement considérés comme négligeables si on se place dans certaines conditions que sont les conditions de Gauss. Les conditions de Gauss sont que les rayons doivent arriver sur la lentille proche du centre optique et que l'angle par rapport à l'axe optique doit être faible. Ainsi, pour un objet donné, une image formée et la position et la taille de l'image peuvent être déterminé grâce à la formule de conjugaison.

Les aberrations sont de plusieurs types :

- Aberration sphérique (moins de matière traversée)(lentille mince)
- Aberration de coma (distance du centre)(Gauss)
- Astigmatisme (angle d'arrivée suivant l'axe)(Gauss)
- Aberration chromatique (longueur d'onde)

Un objectif est un dispositif composé d'au moins une lentille dont le but est de former une image dans un champ de vue donné d'un objet. Dans les appareils d'optiques classiques, la zone de formation de l'image est souvent l'oeil et l'objectif permet de visualiser des objets à champ de vue très faible comme dans le cas d'une loupe ou d'un télescope. On peut remplacer l'oeil humain par un détecteur afin de pouvoir enregistrer l'image et la traiter.

Les objectifs à champ de vue large sont souvent composés de plusieurs lentilles pour palier à l'apparition d'aberrations puisque l'on s'éloigne des conditions de Gauss. Si l'on souhaite récupérer un maximum de signal, il est utile de prendre un maximum de photons arrivant sur la lentille même si ceux-ci sont loin du centre optique. Le champ large signifie que les angles d'entrée peuvent être grands.

Les objectifs sont caractérisés par leur distance focale et par leur ouverture numérique qui donne une information sur le champ de vue maximum.

Un objectif est souvent modélisé par un système optique analogue à une lentille comprenant des plans principaux objet et image.

4.2 Traitements des images optiques

Le signal Cerenkov est de très faible intensité. Les images enregistrées présentent souvent de grandes variations de valeurs pour des pixels adjacents (fig. 4.6).

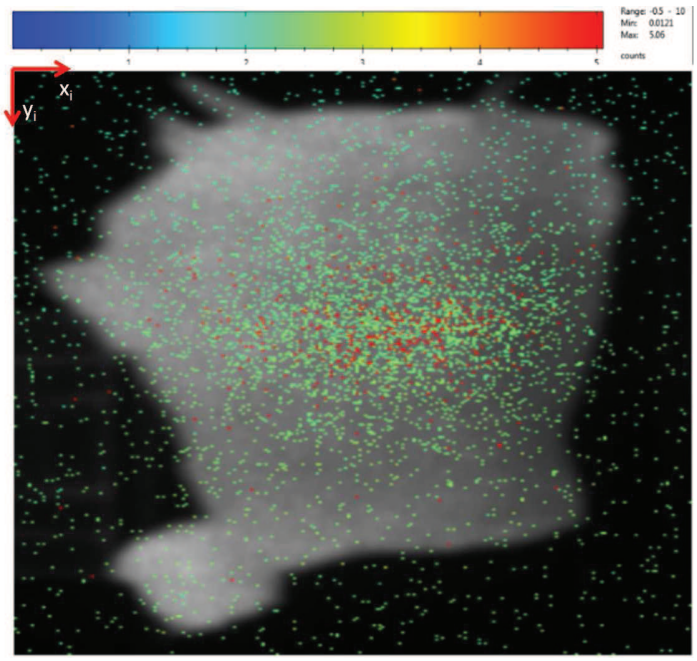

FIGURE 4.6 – Image Cerenkov d'un morceau de muscle avec 2 MBq de ^{18}FNa injecté.

Pour une image donnée, on définit les axes x_i et y_i qui sont les axes propres à l'image 2D (fig. 4.6). L'axe x_i parcourt les différentes vues de l'objet à y_i fixé et l'axe y_i parcourt l'objet à x_i fixé, c'est-à-dire pour une vue donnée. L'origine est situé dans le coin en en haut à gauche de l'image. L'axe x_i est dirigé vers la droite et l'axe y_i dirigé vers le bas.

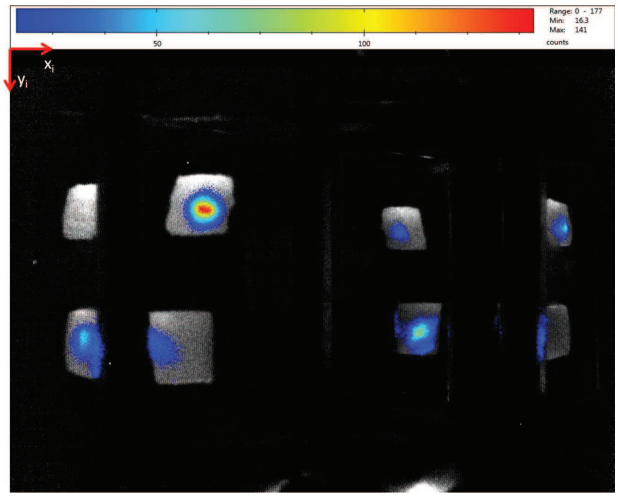

FIGURE 4.7 – Image Cerenkov de 2 morceaux de muscle injecté de ^{18}F acquis en 4 vues.

Pour lisser les images, un filtre gaussien ($\sigma = 2$ pixels) est appliqué aux images optiques (fig. 4.7). Ce lissage permet d'éliminer les forts contrastes présents localement dans l'image (fig. 4.8). Ce lissage est appliqué indifféremment sur chacune des vues de l'image. La valeur est choisie suffisamment grande pour éliminer les forts contrastes mais un lissage trop important tend à générer des tâches optiques circulaires. Cette valeur a été prise car visuellement, elle convient à ce compromis.

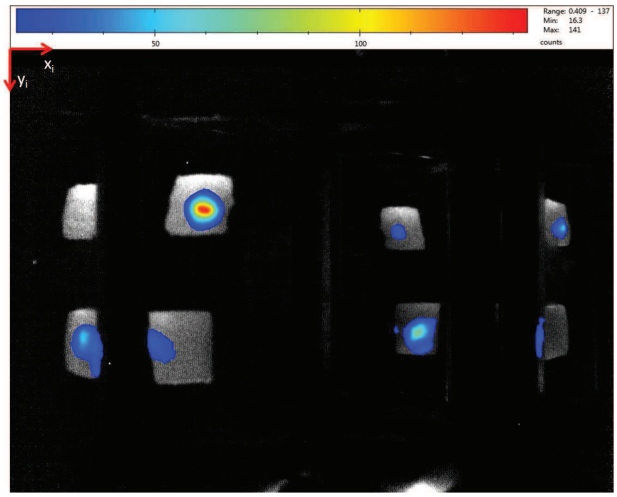

FIGURE 4.8 – Image précédente lissée.

Pour séparer les vues, on définit une zone de signal (fig. 4.9) délimitée pour chaque vue manuellement. L'intérêt est de pouvoir sélectionner le signal optique sur de petites zones pour diminuer la quantité d'information à traiter. De plus, aucune information ne permet de différencier de manière automatique les différentes vues. Les coordonnées relatives de ces zones par rapport à l'axe optique seront utilisées mais pas leur coordonnées absolues. Chaque zone ne peut pas être traitée sans le reste de l'image par la suite.

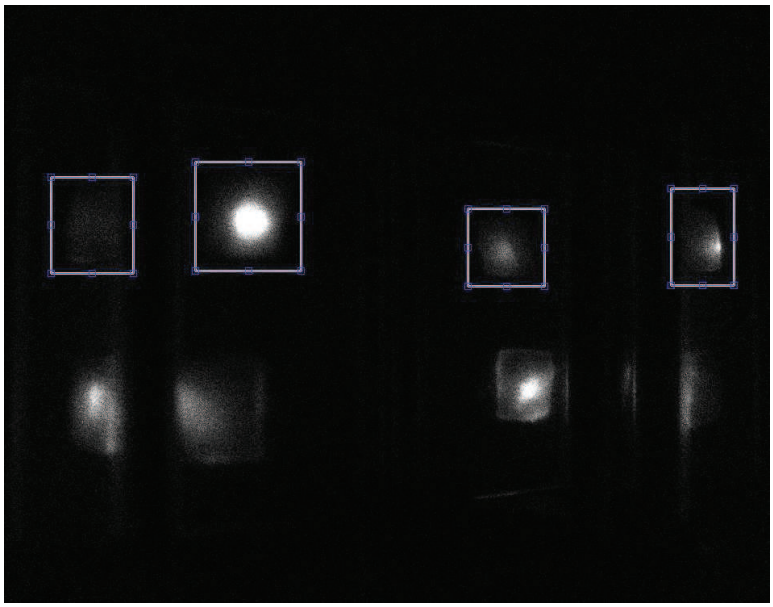

FIGURE 4.9 – Image Cerenkov brute avec zones définies pour chaque vue pour un objet.

Les valeurs dans l'image ne sont pas modifiées par rapport au filtre. L'information spectrale du rayonnement Cerenkov est considérée plus tard dans le processus. La soustracion des images causent des soucis avec ou sans lissage qui conduisent à faire apparître des zones contrastées voire négatives.

4.3 Insertion d'un *a priori* anatomique

4.3.1 Recalage

Le recalage permet d'utiliser les informations du volume CT dans la chaine de reconstruction en tomographie Cerenkov en mettant les informations anatomiques et optiques dans un même repère. Cette étape est réalisée en utilisant un objet de référence qui sera placé à côté de l'objet lors des acquisitions TDM-X et optiques.

Les sources phosphorescentes ou luminescentes saturent rapidement le détecteur si l'ouverture numérique est maximum et il est nécessaire de réduire l'ouverture numérique lors de

leur utilisation. Le signal Cerenkov est enregistré avec l'ouverture numérique la plus élevée pour récupérer le maximum de signal. C'est pourquoi le recalage est réalisé grâce à l'image photographique et non sur l'image de luminescence. Le recalage est effectué à partir d'un objet de référence (fig. 4.10). L'objet de référence est transparent et possède des contours visibles sur les images optiques et dans le volume CT.

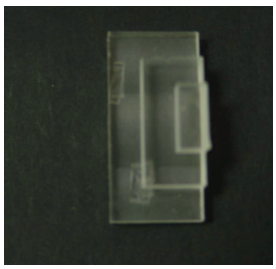

FIGURE 4.10 – Photographie de l'objet de référence utilisé pour le recalage Optique-TDM.

Le recalage permet de placer le maillage dans le référentiel optique 3D. Une matrice de transformation du référentiel CT au référentiel optique. Cette transformation est appliquée au maillage précédemment obtenu. Précisément, la transformation est appliquée sur chaque point du maillage. Les autres éléments restent identiques.

4.3.2 Maillage

L'implémentation d'une méthode par éléments finis nécessite de définir un volume. Un volume est délimité par une surface fermée. Un maillage sert à représenter des surfaces ou des volumes. Un maillage est un ensemble de points qui forme une base pour la modélisation d'un objet. Les points servent à décrire des éléments tels que des segments, polygones ou des polyèdres. Un segment est représenté comme un ensemble de 2 points, un polygone comme un ensemble de 3 segments (triangle) ou plus et un polyèdre comme un ensemble de 4 polygones (tétraèdre) ou plus. La surface comprenant un volume peut être représentée par un ensemble de polygones et le volume peut être décrit comme un ensemble de polyèdres. Les formes les plus basiques sont le triangle et le tétraèdre. Il est important pour un maillage de contrôler la densité du maillage mais aussi la forme des structures [77]. Les segments, faces et volumes décrits ne doivent pas se chevaucher. Plusieurs paramètres servent de critère pour contrôler l'aspect du maillage obtenu. Ces paramètres peuvent être inclus lors de la génération du maillage ou sur un maillage déjà existant. Les critères généralement utilisés pour caractériser un maillage sont les conditions de Delaunay.

Par la suite, on notera N_{points}, le nombre de points total d'un maillage, N_{face} le nombre de faces, N_{tetra} le nombre de tétraèdres et N_{surf} le nombre de points en surface.

Pour chaque point est associé des coefficients optiques différents. Un maillage dit homogène est un maillage généré à partir d'une seule surface fermée et dont chaque point à les mêmes propriétés optiques. Un maillage hétérogène est généré à partir des interfaces entre

les différents milieux considérés. Les propriétés optiques sont assignés au sein de chaque volume.

Pour reconstruire le flux de photons au niveau de la surface en 3D de l'objet, il est nécessaire de connaître la surface 3D de l'objet. Cette surface est extraite à partir du volume CT que nous donne le TDM-X (fig. 4.11).

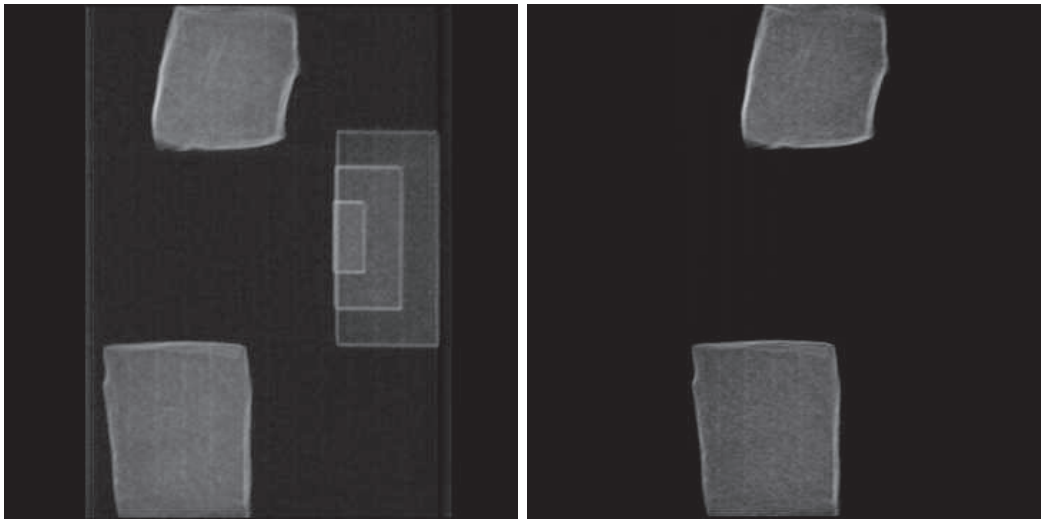

FIGURE 4.11 – Image TDM reconstruite sans traitement (à gauche) et Image TDM reconstruite avec support et objet référence supprimé (à droite).

Le volume TDM sans support est utilisé comme base pour générer un maillage surfacique et volumique de l'objet (fig. 4.12). Le maillage surfacique est un ensemble de faces triangulaires délimitant un volume. Le maillage volumique est constitué de la surface qui le délimite et de tétraèdres (tab. 4.2). Le support de l'objet est supprimé de l'image CT avant maillage. La densité du maillage est ajustée en fonction de la résolution spatiale des images qui dépend de la vue.

FIGURE 4.12 – Maillages surfacique (à gauche) et volumique (à droite) générés à partir de l'image TDM.

Information maillage	
Nombre de points total	1709
Nombre de faces	1478
Nombre de tétraèdres	8085

TABLE 4.2 – Tableau récapitulatif des informations du maillage.

4.4 Modélisation et reconstruction

4.4.1 Système optique

4.4.1.1 Modélisation d'un détecteur

Dans un premier temps, il est possible de reconstruire la radiance sur la surface 3D de l'objet à partir des images optiques 2D (fig. 4.13 et 4.14). Cette reconstruction peut être réalisée indépendamment des problèmes de propagation optique dans les tissus. Le problème de diffusion des photons est négligeable et simplifie cette étape.

Dans un système optique, le trajet d'un rayon peut être reconstruit grâce au principe du retour inverse de la lumière. Il est possible de modéliser un objectif sans considérer le parcours des rayons dans l'objectif mais seulement en tenant compte des rayons entrant et sortant [78, 79, 80].

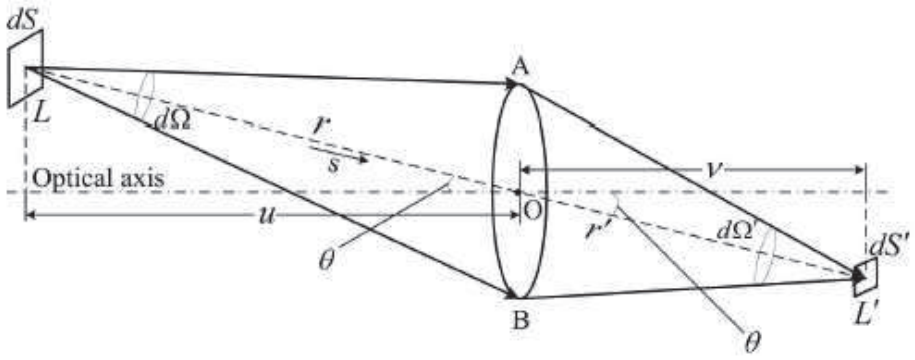

FIGURE 4.13 – Reconstruction du flux en surface d'un objet à partir du signal enregistré par un système optique (tiré de [80]).

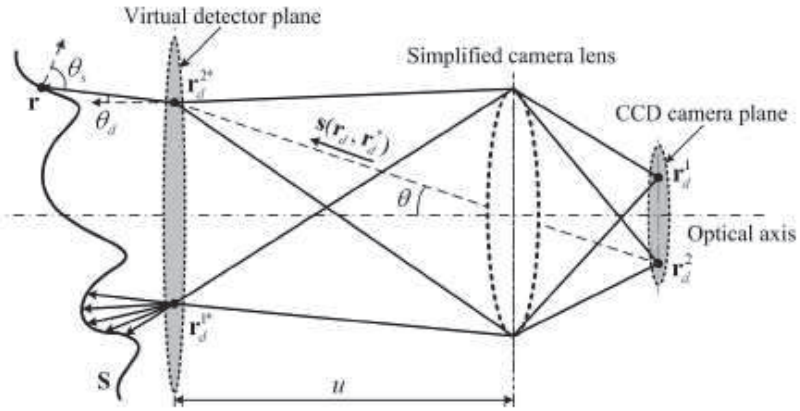

FIGURE 4.14 – Reconstruction du flux en surface d'un objet à partir du signal enregistré par un système optique (tiré de [80]).

$$P(\vec{r_d}) = \frac{1}{\pi}\int_S J_+(\vec{r})\varepsilon(\vec{r},\vec{r_d})\frac{\cos\theta_s\cos\theta_d}{\left|\vec{r_d}-\vec{r}-\frac{tu'^2}{f\cos\theta}\vec{s}\right|}\frac{dA_d}{t^2}dS \tag{4.11}$$

Avec :
– $\vec{r}$ est la position du rayon sur la surface de l'objet

– $\vec{r_d}$ est la position du rayon sur le plan du détecteur

- $P(\vec{r_d})$ est la puissance enregistrée sur le détecteur
- J_+ est le flux optique en surface de l'objet
- $\varepsilon(\vec{r}, \vec{r_d})$ est une fonction qui élimine les points de la surface de l'objet $\vec{r}$ non visibles sur le détecteur en $\vec{r_d}$
- $\cos\theta_s$ est l'angle entre l'axe optique et le rayon entre l'objet et le plan du détecteur virtuel
- $\cos\theta_d$ est l'angle entre le rayon et la normale en un point de la surface
- t est coefficient de grandissement de l'objectif simplifié
- u' est la distance de l'objet par rapport à l'objectif simplifié
- A_d est la surface du détecteur
- S est la surface de l'objet

Le flux sortant J_+ d'une surface observé par un système optique peut être reconstruit à partir des images enregistrées et de la modélisation du système optique.

4.4.1.2 Calibration 4 vues et séparation vues

Expérimentalement, le module 4 vues est mobile et sa position est variable d'une expérience à l'autre. Toutefois, la position du module varie uniquement suivant l'axe x_i et doit être calibrée pour fixer l'origine au milieu des deux miroirs à 90°.

Le système optique contenant l'objectif et le détecteur est modélisé par son centre et un vecteur donnant l'orientation du système (fig. 4.15).

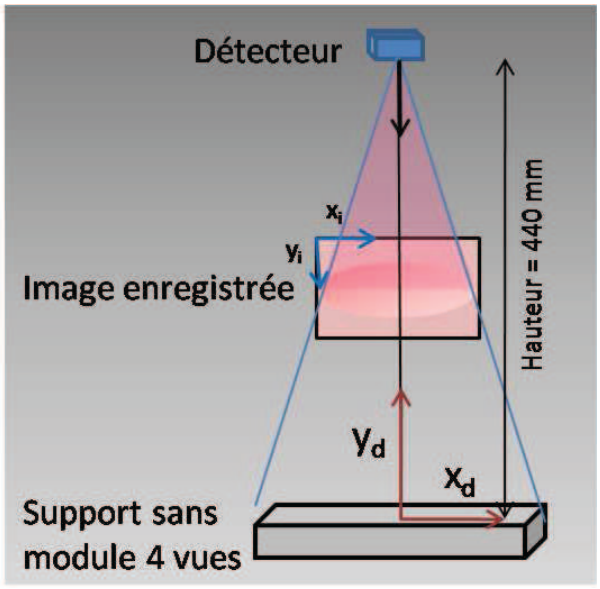

FIGURE 4.15 – Modélisation du système optique réel du PhotonImager.

Pour exploiter les informations de chaque angle de vue, on modélise l'ensemble du système optique et modules 4 vues par un système de 4 systèmes optiques virtuels. Parmi ces 4 systèmes optiques virtuels, l'un représente le système optique réel, les autres sont les images du système optique réel par le système de miroir du module 4 vues. La position de chaque système optique virtuel est différente et représente chacune des vues de l'objet. Pour réaliser cette étape, on travaille dans le plan contenant l'axe optique du système optique réel et contenant les normales des miroirs (fig. 4.15). Les axes de ce plan sont notés x_d et y_d où y_d est orienté selon l'axe optique du système optique réel et x_d est le second axe caractéristique du plan. L'origine de ce système est positionnée en bas du module 4 vues et au milieu des 2 miroirs permettant la récupération de l'image de dessous (fig. 4.16).

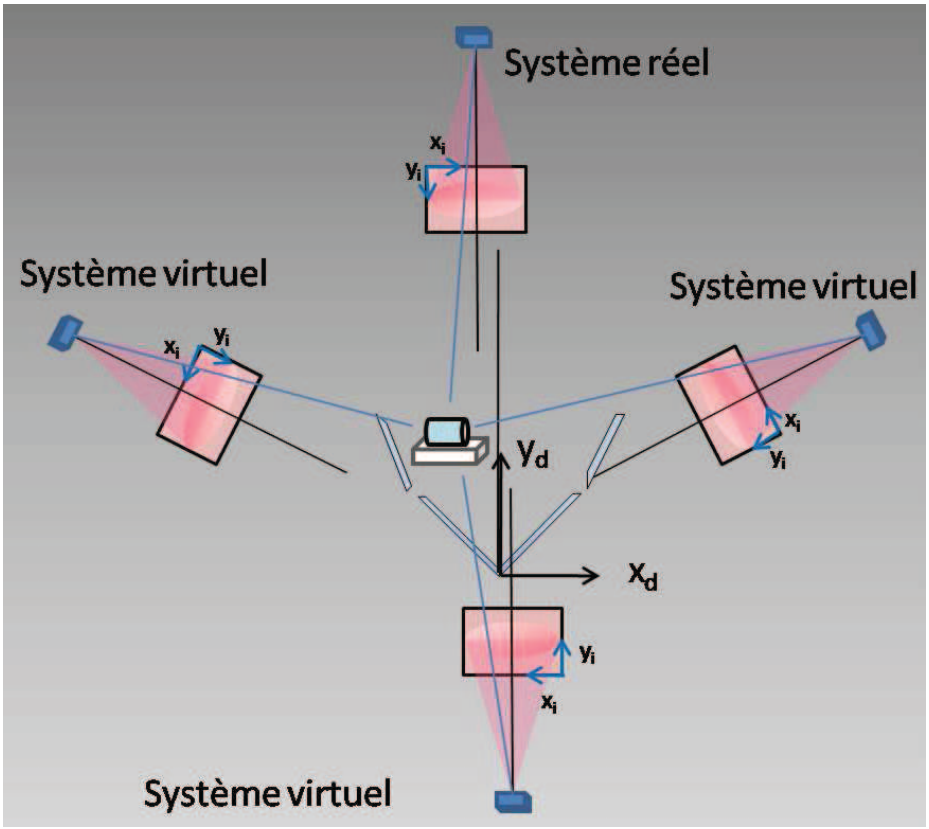

FIGURE 4.16 – Modélisation des différentes vues du PhotonImager obtenues avec le module 4 vues.

Dans cette représentation, il n'est pas nécessaire d'extraire les différentes vues de l'image. Chaque système observe l'image complète mais suivant un angle de vue différent. Pour chaque position dans l'image, il est possible de connaître la direction du rayon incident. La taille que représente un pixel dépend de la distance entre l'objet et le système. Le module 4 vues modifiant les distances entre l'objet et les systèmes virtuels, la résolution spatiale dépend de l'angle de vue (tab. 4.3).

Vue	Distance de l'objet (mm)	Dimension pixel (mm)
Sans module 4 vues	440	0,250
Vue dessus avec module 4 vues	360	0,205
Vue dessous avec module 4 vues	510	0,290
Vue gauche avec module 4 vues	409	0,232
Vue droite avec module 4 vues	470	0,267

TABLE 4.3 – Variation de la résolution spatiale des images suivant la vue

Le repère optique 3D (x,y,z) utilisé est défini par rapport au module 4 vues. L'origine est la même que celle du repère des détecteurs. L'axe x est identique à x_d, l'axe z est identique à y_d et l'axe y est choisi pour former une base orthonormale directe. Pour un pixel (x_i,y_i) et pour un système donné caractérisé dans (x_d,y_d), il est possible de connaître le rayon incident dans l'espace.

4.4.1.3 Reconstruction flux surfacique

A 440 mm du détecteur, la zone observée par un pixel est de 0.25 x 0.25 mm^2. Comme la distance entre le détecteur et l'objet varie suivant la vue, la taille de la zone observée varie aussi (fig. 4.17).

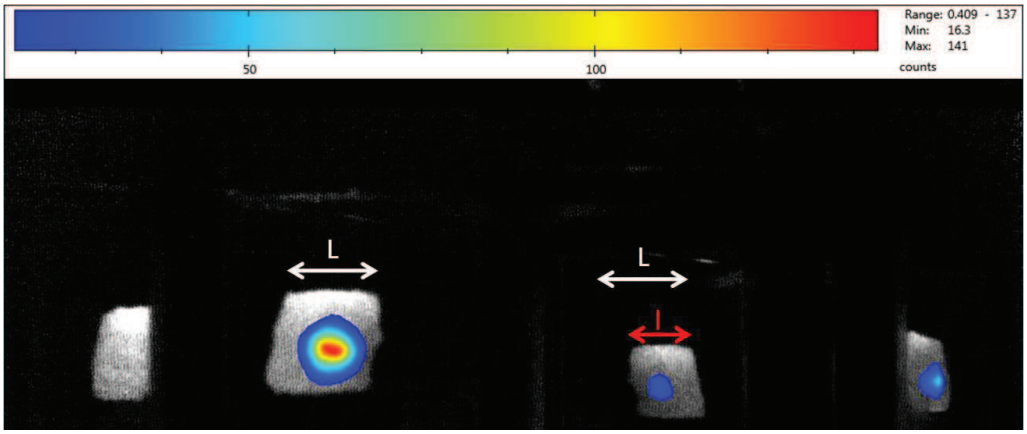

FIGURE 4.17 – Différence de taille due à la distance suivant la vue.

Le maillage est généré de sorte que pour une face donnée, on ait plusieurs pixels qui correspondent.

4.4.2 Dans les tissus

4.4.2.1 Modèle direct

4.4.2.1.1 Simulation Monte-Carlo

Les simulations types Monte-Carlo consistent à simuler les interactions de chaque particule individuellement pour connaître le comportement de la particule. Cette méthode est très pratique pour suivre les particules ainsi que leurs propriétés pendant le trajet. Toutefois, cette méthode nécessite de simuler chaque particule. Cette méthode est donc très coûteuse en temps de calcul surtout pour l'imagerie optique qui fait intervenir un grand nombre de photons. Le comportement des photons est modélisé par les coefficients optiques μ_a, μ_s, g et n pour une longueur d'onde donnée. Elle reste une méthode directe précise. MOSE (Molecular Optical Simulation Environment, [81]) est un logiciel gratuit permettant la simulation de photons optiques par Monte-Carlo dans des volumes homogènes ou hétérogènes [82, 83]. Chaque volume est délimité par un maillage surfacique.

4.4.2.1.2 Equation de transfert radiatif

Solutions approchées

La modélisation de la propagation optique via l'équation de transfert radiatif est un problème complexe [84, 85, 86, 87, 88, 89, 90, 91, 92, 93]. Il n'existe pas de solution analytique directe à l'équation de transfert radiatif. Il est nécessaire de faire des approximations pour pouvoir obtenir des solutions approchées. Cette équation dépend des coefficients optiques qui dépendent de la longueur d'onde. Par conséquent, ce modèle est valable pour une longueur d'onde donnée. Dans le cas des tissus biologiques, la diffusion est prédominante par rapport à l'absorption ($\mu_a \ll \mu'_s$). Dans ce cas, l'équation de transfert se simplifie en une équation appelée équation de diffusion :

$$\mu_a.\Phi(\vec{r}) - \nabla(D.\nabla\Phi(\vec{r})) = S(\vec{r}) \tag{4.12}$$

Avec :
- Φ représente le flux optique en W.m^{-2}
- μ_a est le coefficient d'absorption
- D est le terme de diffusion valant $1/(3(\mu_a+\mu_s'))$

Le modèle de Green est solution de l'équation de diffusion si on suppose que D ne dépend pas de la position, c'est-à-dire que le milieu est homogène. On définit alors un coefficient prenant en compte les effets de l'absorption et de la diffusion $\mu_{eff} = \sqrt{(3\mu_a(\mu_a + \mu'_s))}$. Si on souhaite travailler en milieu hétérogène ou dans des cas plus généraux sans faire

l'approximation d'un phénomène de diffusion prédominant, des solutions existent avec un ordre N, entier positif impair, donnant l'ordre de développement de la solution [94, 95]. S_N (approximation des "discrete ordinates" à l'ordre N) [94, 96], P_N (approximation des harmoniques sphériques à l'ordre N) [94, 97, 98, 99] ou encore SP_N (approximation des harmoniques sphériques simplifiées à l'ordre N) [100, 101, 102, 103, 104, 105] sont les modèles existants (tab. 4.4). Les approximations S_N et P_N convergent vers la solution exacte quand l'ordre du développement N augmente. L'approximation SP_N ne converge pas avec N croissant [76].

Modèle	Nombre d'équations à résoudre	Avantages	Limites
DE	1	Dispose d'une solution analytique dans le cas homogène (modèle de Green)	Ne peut pas s'appliquer à certaines longueurs d'onde ou certains tissus
S_N	N(N+2)	Converge vers la solution exacte quand N augmente	Souffre du « ray effect », long
P_N	$(N+1)^2$	Converge vers la solution exacte quand N augmente	Long (plus que S_N)
SP_N	(N+1)/2	Moins d'équations à résoudre, pas de problème de « ray effect », utilisation d'algorithme de résolution de diffusion standard	Ne converge pas vers la solution exact quand N augmente

TABLE 4.4 – Comparaison des modèles d'approximation de l'équation de transfert.

Ces méthodes ont déjà été éprouvées dans le cadre de la tomographie optique [73, 76, 106, 107, 108, 109, 110, 111, 112, 113, 114].

En connaissant le flux d'un côté d'une interface, on peut connaître le flux de l'autre côté. Les conditions aux frontières peuvent être reformulées suivant l'approximation et l'ordre utilisés [84, 76].

Méthode par éléments finis

Implémentation

Dans le cadre de la propagation de la lumière, on souhaite caractériser le flux de photons en fonction de la source optique présente dans le volume. La propagation de la lumière peut être résolue localement. Sur la base d'un maillage, il est possible de décrire la propagation d'un flux de photons entre 2 points liés [115]. En établissant l'ensemble des relations existantes pour chaque liaison, il est possible de modéliser la propagation des photons dans l'ensemble du domaine du maillage. Le flux en un point est lié au flux des voisins de ce point ainsi qu'à la source présente en ce point. Il est alors possible d'établir un système liant la source présente en chaque point et le flux en chaque point. Ce système est un système linéaire. Il prend en compte les phénomènes optiques en volume que sont l'absorption et la diffusion. En chaque point i, on définit sa fonction ν_i propre à une base linéaire par morceau. On décompose le flux en une somme de fonctions linéaires définies pour chaque point du maillage [115] :

$$\phi(\vec{r}) = \sum_{i=1}^{N} \phi_i.\nu_i(\vec{r}) \tag{4.13}$$

$$S(\vec{r}) \simeq \sum_{i=1}^{N} x_i.\nu_i(\vec{r}) \tag{4.14}$$

Modèle SP_3

Développement mathématique de l'approximation SP_3 à partir de l'équation 4.2 [76]. La radiance est exprimée en fonction des moments de Legendre ϕ_1 et ϕ_2 :

$$\phi_1(\vec{r}) \simeq \sum_{i=1}^{N} \phi_{1,i}.\nu_i(\vec{r}) \tag{4.15}$$

$$\phi_2(\vec{r}) \simeq \sum_{i=1}^{N} \phi_{2,i}.\nu_i(\vec{r}) \tag{4.16}$$

Les équations suivantes décrivent le modèle de propagation au sein d'un milieu :

$$\begin{cases} -\nabla.\dfrac{1}{3\mu_{a1}}\nabla\phi_1 + \mu_a\phi_1 = S + \dfrac{2\mu_a}{3}\phi_2 \\ -\nabla.\dfrac{1}{7\mu_{a3}}\nabla\phi_2 + (\dfrac{4}{9}\mu_a + \dfrac{5}{9}\mu_{a2})\phi_2 = -\dfrac{2}{3}S + \dfrac{2}{3}\mu_a\phi_1 \end{cases} \tag{4.17}$$

Avec :

$$\mu_{an} = \mu_a + \mu_s - \mu_s g^n \tag{4.18}$$

Les équations des conditions aux frontières sont exprimées de la manière suivante :

$$\begin{cases} \left(\frac{1}{2} + A_1\right)\phi_1 + \left(\frac{1+B_1}{3\mu_{a1}}\right) n.\nabla\phi_1 = \left(\frac{1}{8} + C_1\right)\phi_2 + \left(\frac{D_1}{\mu_{a3}}\right) n.\nabla\phi_2 \\ \left(\frac{7}{24} + A_2\right)\phi_2 + \left(\frac{1+B_2}{7\mu_{a3}}\right) n.\nabla\phi_2 = \left(\frac{1}{8} + C_2\right)\phi_1 + \left(\frac{D_2}{\mu_{a1}}\right) n.\nabla\phi_1 \end{cases} \tag{4.19}$$

Le modèle peut ainsi être exprimé de la manière suivante :

$$\begin{cases} M_{11}\phi_1 + M_{12}\phi_2 = BS \\ M_{21}\phi_1 + M_{22}\phi_2 = -\frac{2}{3}BS \end{cases} \tag{4.20}$$

Avec :

$$\begin{cases} M_{11jk} = \int_\Omega \left(\frac{1}{3\mu_{a1}}\nabla v_j(r).\nabla v_k(r) + \mu_a v_j(r)v_k(r)\right)dr - \int_{\partial\Omega} Z_{11}v_j(r)v_k(r)dr \\ M_{12jk} = \int_\Omega \frac{2\mu_a}{3}v_j(r)v_k(r)dr + \int_{\partial\Omega} Z_{12}v_j(r)v_k(r)dr \\ M_{21jk} = \int_\Omega \frac{2\mu_a}{3}v_j(r)v_k(r)dr - \int_{\partial\Omega} Z_{21}v_j(r)v_k(r)dr \\ M_{22jk} = \int_\Omega \left(\frac{1}{7\mu_{a3}}\nabla v_j(r).\nabla v_k(r) + (\frac{4}{9}\mu_a + \frac{5}{9}\mu_{a2})v_j(r)v_k(r)\right)dr - \int_{\partial\Omega} Z_{22}v_j(r)v_k(r)dr \\ B_{jk} = \int_\Omega v_j(r)v_k(r)dr \end{cases} \tag{4.21}$$

Et :

$$\begin{cases} Z_{11} = \frac{1}{3}\left(D_1(\frac{1}{8}+C_2) - \frac{1+B_2}{7}(\frac{1}{2}+A_1)\right) \Big/ \left(\frac{(1+B_1)(1+B_2)}{21} - D_1D_2\right) \\ Z_{12} = \frac{1}{3}\left(\frac{1+B_2}{7}(\frac{1}{8}+C_1) - D_1(\frac{7}{24}+A_2)\right) \Big/ \left(\frac{(1+B_1)(1+B_2)}{21} - D_1D_2\right) \\ Z_{21} = \frac{1}{7}\left(\frac{1+B_1}{3}(\frac{1}{8}+C_2) - D_2(\frac{1}{2}+A_1)\right) \Big/ \left(\frac{(1+B_1)(1+B_2)}{21} - D_1D_2\right) \\ Z_{22} = \frac{1}{7}\left(D_2(\frac{1}{8}+C_1) - \frac{1+B_1}{3}(\frac{7}{24}+A_2)\right) \Big/ \left(\frac{(1+B_1)(1+B_2)}{21} - D_1D_2\right) \end{cases} \tag{4.22}$$

Les coefficients A_i, B_i, C_i et D_i sont développés dans [76]. Les moments composés peuvent être retrouvés avec :

$$\begin{cases} \phi_1 = [M_{12}^{-1}M_{11} - M_{22}^{-1}M_{21}]^{-1}[M_{12}^{-1} + \frac{2}{3}M_{22}^{-1}]BS \\ \phi_2 = [M_{11}^{-1}M_{12} - M_{21}^{-1}M_{22}]^{-1}[M_{11}^{-1} + \frac{2}{3}M_{21}^{-1}]BS \end{cases} \tag{4.23}$$

On peut ainsi exprimer le flux optique sortant par :

$$J_+ = \beta_1\phi_1 + \beta_2\phi_2 = MS \tag{4.24}$$

Avec :

$$\begin{cases} \beta_1 = \frac{1}{4} + J_0 - (0.5+J_1)Z_{11} - J_3Z_{21} \\ \beta_2 = -\frac{1}{16} - \frac{2J_0}{3} + \frac{J_2}{3} - (0.5+J_1)Z_{12} - J_3Z_{22} \end{cases} \tag{4.25}$$

Où les coefficients J_i sont décrits dans [76].

On obtient alors la matrice carrée A de taille N_{points} x 2 qui permet d'exprimer le flux optique en surface J_+ à partir de la source S :

$$J_+ = M.S \tag{4.26}$$

4.4.2.2 Problème de reconstruction

L'information que l'on connait n'est pas le flux en tout point mais le flux au niveau de la surface. Le flux interne à l'animal est inconnu. La reconstruction de la source optique à

partir des données partielles du flux est alors un problème mal posé puisque l'on dispose de plus d'inconnues que de relations déterminant ces inconnues. En intégrant les conditions de passage représentées par la matrice B, de taille N_{surf} x N_{points}, il est possible d'exprimer le flux sortant J_+ au niveau de la surface en fonction de la source :

$$J_+(\partial\vec{r}) = B.A.S(\vec{r}) = M.S(\vec{r}) \tag{4.27}$$

Dans un problème classique avec suffisamment d'information, on cherche à résoudre $J_+(\partial\vec{r})$-M.S($\vec{r}$) = 0. Dans un problème mal posé, la résolution de cette équation aboutit soit à une infinité de solution soit aucune. Pour aboutir à une solution, on ajoute un terme de régularisation $R(\vec{r})$. On cherche à minimiser la somme :

$$E(S(\vec{r})) = |J_+(\partial\vec{r}) - M.S(\vec{r})| + R(\vec{r}) \tag{4.28}$$

Le choix de la fonction de régularisation ainsi que celui de l'algorithme utilisé pour effectuer la minimisation sont des points délicats qui influeront sur le résultat et le temps de convergence vers ce résultat. Le facteur de régularisation est généralement de la forme k^2.|S(r)| avec k facteur de régularisation.

D'autres méthodes ont été mises en oeuvre [116, 117, 118, 119, 120] (tab. 4.5).

Référence	Algorithme utilisé	Particularité
[119]	« incomplete variables truncated conjugate gradient method » (IVTCG)	
[118]	Reconstruction en parallèle	Reconstruction sur plusieurs parties du maillage
[68, 116]	Adaptative-FEM	Affinage du maillage pendant la reconstruction
[120]	Max-flow /min-cut	Résolution en terme de flux et non en terme de source. Maximise le flux sur un minimum de segments

TABLE 4.5 – Présentation de quelques algorithmes de recontruction utilisés en BLT.

Le système M et les données J_+ varient en fonction de la longueur d'onde. Il est possible de réaliser des reconstructions pour chaque donnée $J_+(\lambda)$ en prenant le système $M(\lambda)$ associé. Une même source peut émettre dans un spectre donné et pour chaque longueur d'onde de ce spectre, il est possible de résoudre ce problème. Mais ces différents systèmes doivent mener au même résultat puisque la source est la même. Il est alors possible de résoudre le problème de reconstruction en utilisant les informations spectrales de la source

[48, 121, 122, 111, 123, 124, 125]. Dans le cadre d'une modélisation par éléments finis, cela revient à modifier le système M et prendre les données J_+ de plusieurs images.

$$J_+(\partial\vec{r}) = M.S(\vec{r}) \tag{4.29}$$

avec

$$J_+ = \begin{pmatrix} J_+(\lambda_1) \\ J_+(\lambda_2) \\ \vdots \\ J_+(\lambda_n) \end{pmatrix} \tag{4.30}$$

et

$$M = \begin{pmatrix} M(\lambda_1) \\ M(\lambda_2) \\ \vdots \\ M(\lambda_n) \end{pmatrix} \tag{4.31}$$

Le système mal posé est alors amélioré par ajout d'informations spectrales dans le modèle de propagation. La résolution d'un modèle multispectrale converge plus vite mais le système à résoudre est plus important.

Chapitre 5

Résultats et discussion

5.1 Effet Cerenkov

5.1.1 Signal Cerenkov

Une première partie de mon travail a consisté en la caractérisation expérimentale du signal Cerenkov. Le signal Cerenkov émis par une source de fluor-18 a été quantifié et la relation de proportionnalité entre activité de la source et signal Cerenkov a été vérifié. Pour cela, 50μL de solution de fluor-18 ont été mis dans un tube eppendorf de 100μL (fig. 5.1). L'activité initiale est mesurée au temps t_0 et l'activité à un instant t est déterminée par :

$$A(t) = A(\mathrm{t_0})\mathrm{e}^{-\lambda.t} \tag{5.1}$$

Où λ est la constante de décroissance radioactive, liée au temps de demi-vie par :

$$T_{\frac{1}{2}} = \frac{ln(2)}{\lambda} \tag{5.2}$$

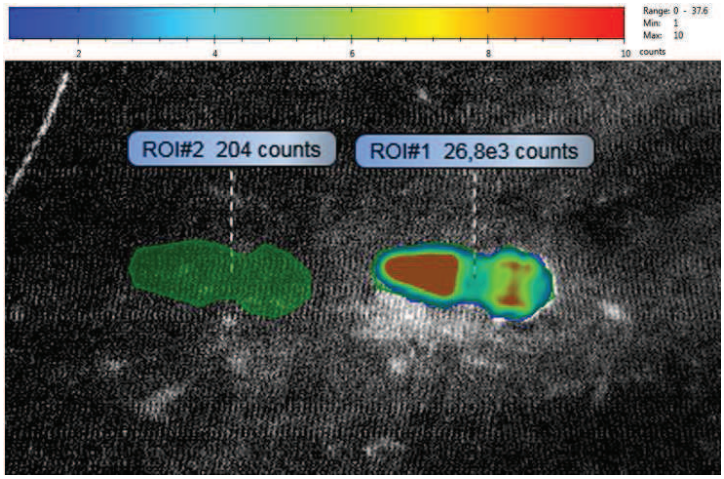

FIGURE 5.1 – Tube eppendorf avec 4,8 MBq de ^{18}F dans 50 μL. L'image a été acquise **sans filtre pendant 3 min**. Le signal provenant du tube eppendorf est de 26,8.10^3 coups et le signal de fond est de 204 coups.

Pour déterminer le lien entre activité et photon détecté, des images ont été enregistrées à différentes activités. Le signal collecté doit être proportionnel à l'activité. Un moyen de faire varier l'activité est de suivre la décroissance radioactive. Une acquisition optique du tube eppendorf contenant une solution aqueuse de ^{18}F a été réalisée pendant 30 min sans filtre (tab. 5.1 et fig. 5.2).

Acquisition optique	
Filtres	Sans filtre
Durée par filtre	30 min
Module 4 vues ?	non
Focus	90 %
O.N.	1,4

TABLE 5.1 – Acquisition optique réalisée sur le tube eppendorf pour étude de la décroissance.

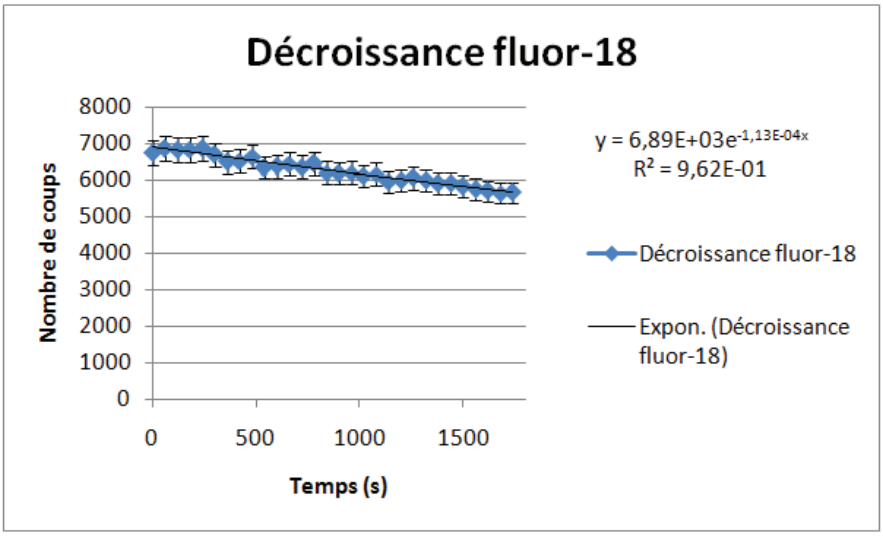

FIGURE 5.2 – Signal Cerenkov acquis pendant **30 min sans filtre** du tube eppendorf précédent. Les valeurs sont obtenues en comptant le nombre de coups par minute. L'utilisation d'un modèle exponentiel pour retrouver la loi de décroissance du ^{18}F donne une constante de radioactivité $\lambda = 1{,}13.10^{-4}$ s^{-1}.

Le temps de demi-vie du ^{18}F est de 109,77 min. Expérimentalement, on trouve une valeur de 102,2 min ± 9 min. Le signal optique enregistré est donc proportionnel à l'activité.

Le nombre de désintégrations entre les instants t et t+T est :

$$N_{\text{dés}} = \int_t^{t+T} A(t)dt = A(t).\frac{(1-e^{-\lambda.T})}{\lambda} \tag{5.3}$$

Le nombre de coups par désintégration peut être calculé en intégrant le nombre de coups pendant la durée d'acquisition divisé par le nombre de désintégration dans cet intervalle. Cette valeur permet de remonter aux nombres de photons Cerenkov émis dans l'échantillon par désintégration. Il faut considérer l'angle solide dans lequel est observé l'échantillon puisque seule une partie des photons est récupérée.

Le nombre de photons émis par désintégration dans toutes les directions est déterminé par la relation :

$$\frac{\text{photons émis}}{\text{nombre désintégrations}} = \frac{\text{nombre coups colléctés}}{N_{\text{dés}}}\frac{4\pi}{\Omega} \tag{5.4}$$

Avec Ω angle solide du détecteur à 440 mm considérée constante pour chaque point de l'image.

Pour le ^{18}F, on trouve une valeur de 0,7 ph/désintégration. Le même processus a été réalisé avec une solution de ^{32}P-ATP. La valeur trouvée est de 12,6 ph/désintégration soit 18 fois plus que le ^{18}F.

5.1.2 Spectre Cerenkov

La caractérisation du spectre du signal optique permet de valider que le signal optique est du rayonnement Cerenkov. Pour cela, des acquisitions avec les différents filtres ont été réalisées (tab. 5.2). Ces résultats sont comparés au spectre Cerenkov théorique. Le spectre Cerenkov théorique est calculé en fonction de la nature du filtre et de la réponse spectrale de la photocathode. Le tableau (tab. 5.3) présente la fraction du signal enregistré avec un filtre par rapport à une acquisition sans filtre.

Acquisition optique	
Filtres	Sans filtre puis avec chaque filtre
Durée par filtre	3 min
Module 4 vues ?	non
Focus	90 %
O.N.	1,4

TABLE 5.2 – Acquisition optique réalisée sur le tube eppendorf pour étude du spectre.

La comparaison entre le spectre du signal optique expérimental et le spectre Cerenkov est montré en figure 5.3 ainsi que la similarité des courbes en figure 5.4.

	Pourcentage du signal enregistré par rapport à une acquisition dans filtre
Sans filtre	100 %
Filtre passe-haut 615 nm	36 %
Filtre passe-haut 665 nm	27 %
Filtre passe-haut 700 nm	20 %
Filtre passe-haut 770 nm	15 %
Filtre passe-bande 755-805 nm	4 %
Filtre passe-bande 790-840 nm	3 %

TABLE 5.3 – Signal Cerenkov enregistré avec chaque filtre.

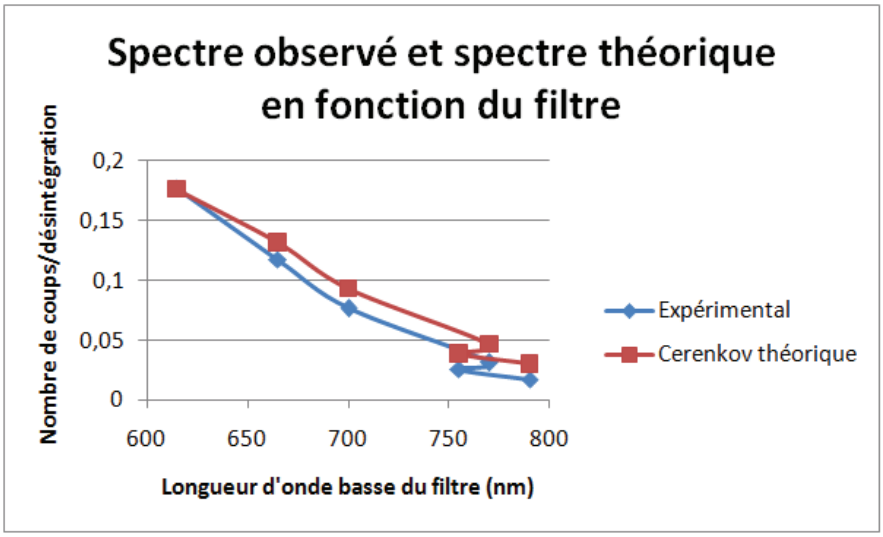

FIGURE 5.3 – Comparaison entre le spectre Cerenkov théorique calculé et le spectre mesuré expérimentalement.

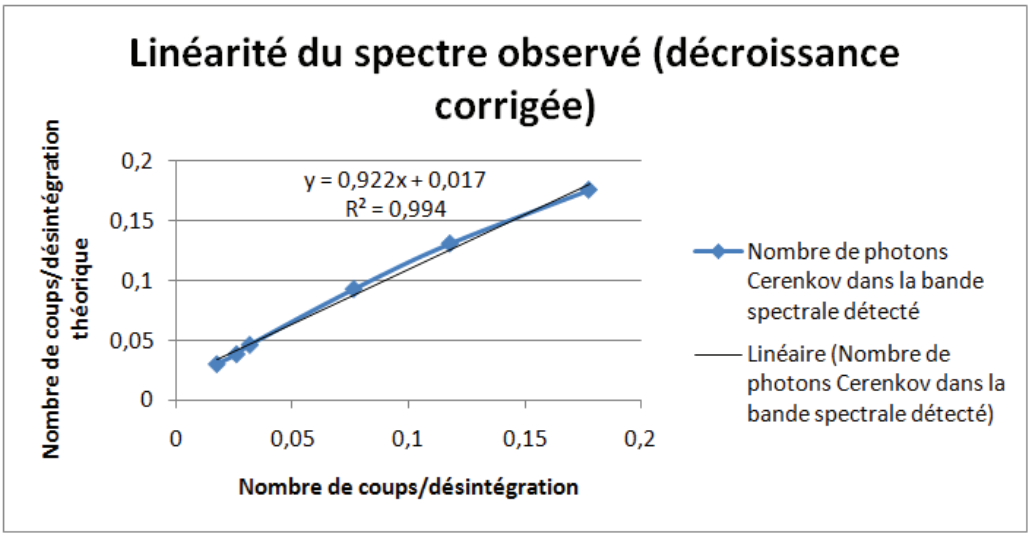

FIGURE 5.4 – Corrélation entre le spectre calculé et expérimental.

Les courbes présentent la même allure avec un coefficient de corrélation de 0,994 et le facteur de proportionnalité entre les 2 courbes est de 0,92. Ce facteur est très proche de 1. Cette légère différence peut s'expliquer par la difficulté à connaître la réponse spectrale de la photocathode en dehors de la plage constante 450-850 nm.

5.2 Imagerie Cerenkov 2D

5.2.1 *In vitro*

L'imagerie Cerenkov dans des tissus biologiques implique de travailler dans des milieux très absorbants et diffusants.

Le changement d'absorption des tissus organiques en fonction de la longueur d'onde altère le spectre Cerenkov observé. L'étude du spectre Cerenkov dans les tissus biologiques a été réalisée en injectant du ^{18}F directement dans des organes de souris (tab. 5.4 et 5.5).

Protocole expérimental	
Produit	^{18}F, 1,86 MBq/10μL
Injection	10 μL dans chaque organe
Acquisition optique	7 x 5 min (après acquisition TEP)
Comptage de l'activité pour chaque organe	

TABLE 5.4 – Protocole expérimental utilisé pour imager différents organes de souris avec du ^{18}F.

Acquisition optique	
Filtres	Sans filtre puis avec chaque filtre
Durée par filtre	5 min
Module 4 vues ?	non
Focus	90 %
O.N.	1,4

TABLE 5.5 – Acquisition optique réalisée sur les organes.

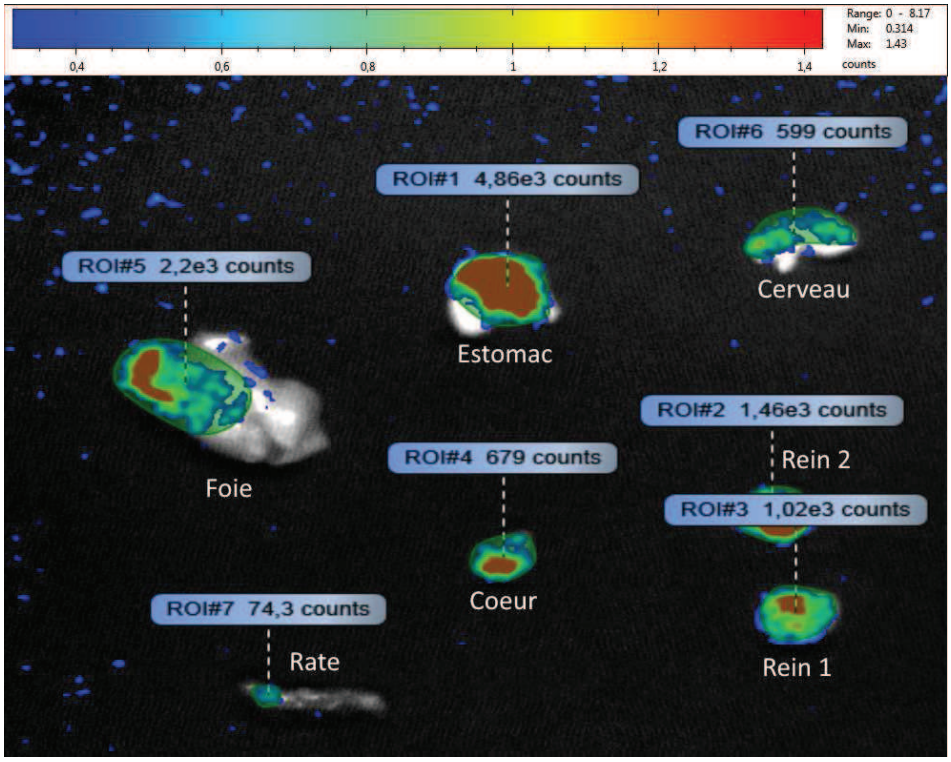

FIGURE 5.5 – Image Cerenkov d'organes de souris. Chaque organe est injecté avec 1,86 MBq de ^{18}F. L'image a été acquise pendant **5 min sans filtre**

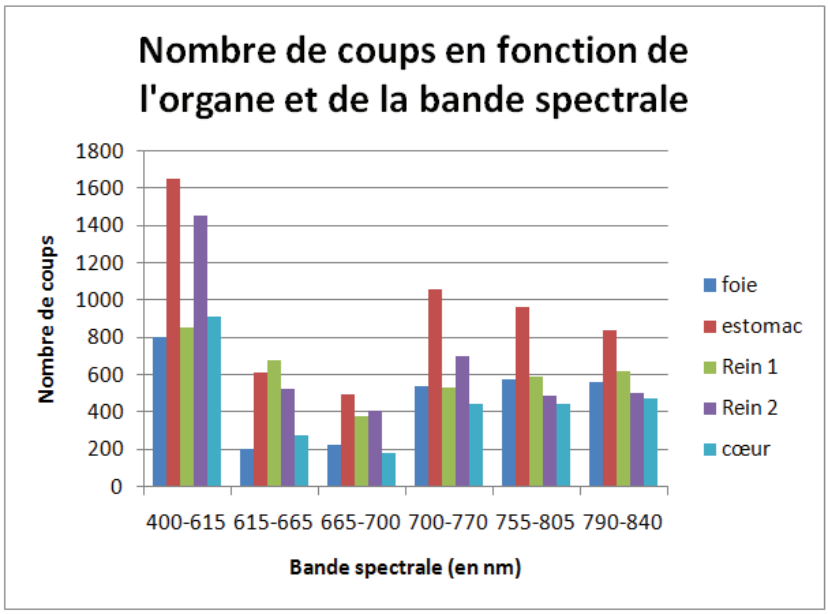

FIGURE 5.6 – Spectre Cerenkov dans différents organes de souris.

On remarque que le signal Cerenkov enregistré pour chaque organe est différent (fig. 5.5 et 5.6). En particulier, l'estomac transmet plus de photons que les autres organes. Cela est cohérent avec le fait qu'il contient moins de sang que les autres organes. On constate aussi que l'on collecte plus de signal Cerenkov dans les longueurs d'onde plus grande que 700 nm. Ceci coïncide avec le modèle des coefficients d'absorption des tissus biologiques. Toutefois, les différences de signal Cerenkov par organe ne sont pas identiques. Dans les bandes 615-665 nm et 665-700 nm, le signal Cerenkov du coeur et du foie est au moins 2 fois plus faible que celui des reins alors qu'il est du même ordre dans les autres bandes. Ces 2 organes sont très vascularisés. Cet écart peut provenir de l'absorption du sang dans cette gamme de longueur d'onde.

Dans le cadre de la tomographie Cerenkov, la qualité des images 2D est importante afin de pouvoir loacilser la source. Des images ont été enregistrées sur des morceaux de muscle (viande rouge, pour simuler une souris vivante) de 1 cm d'épaisseur avec 3 MBq de ^{18}F injecté dans 10 μL (pour avoir un point source le plus petit possible). Les figures 5.7 à 5.13 montrent que le signal Cerenkov enregistré est trop faible pour obtenir une tâche optique nette d'un point source. Seules les images enregistrées avec les filtres passe-haut disposent d'un rapport signal sur bruit suffisant pour distinguer le signal Cerenkov. Le signal enregistré est toutefois très éparse pour être utilisé pour des reconstructions 3D avec l'activité utilisée.

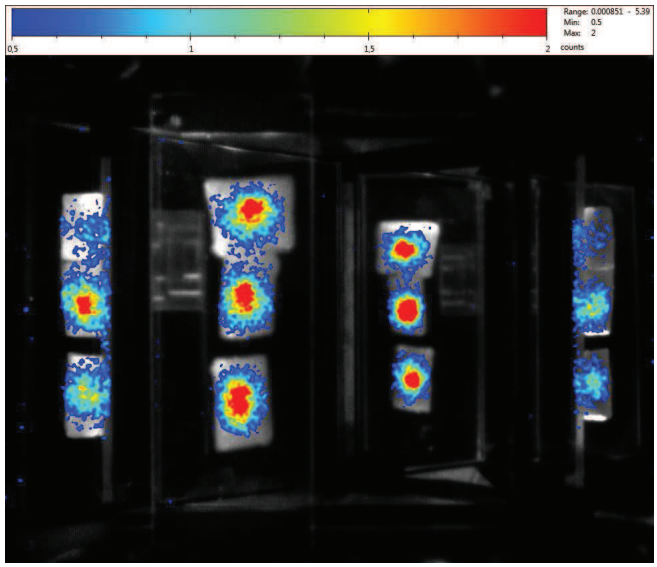

FIGURE 5.7 – Image enregistrée de morceaux de muscle de viande rouge injecté avec 3 MBq de ^{18}F. Image acquise pendant **5 min sans filtre**.

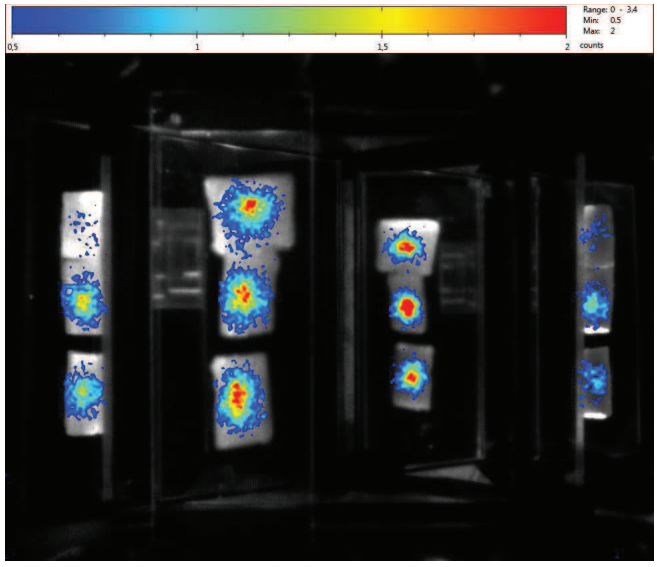

FIGURE 5.8 – Image enregistrée de morceaux de muscle de viande rouge injecté avec 3 MBq de ^{18}F. Image acquise pendant **5 min avec filtre passe-haut à 615 nm**.

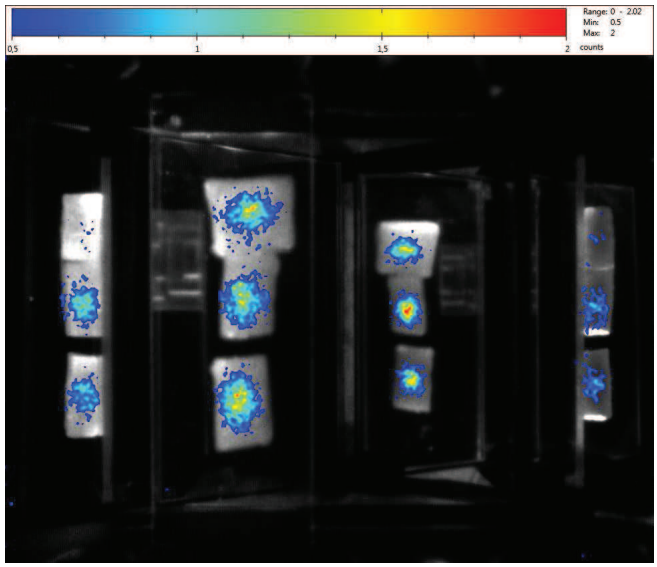

FIGURE 5.9 – Image enregistrée de morceaux de muscle de viande rouge injecté avec 3 MBq de ^{18}F. Image acquise pendant **5 min avec filtre passe-haut à 665 nm**.

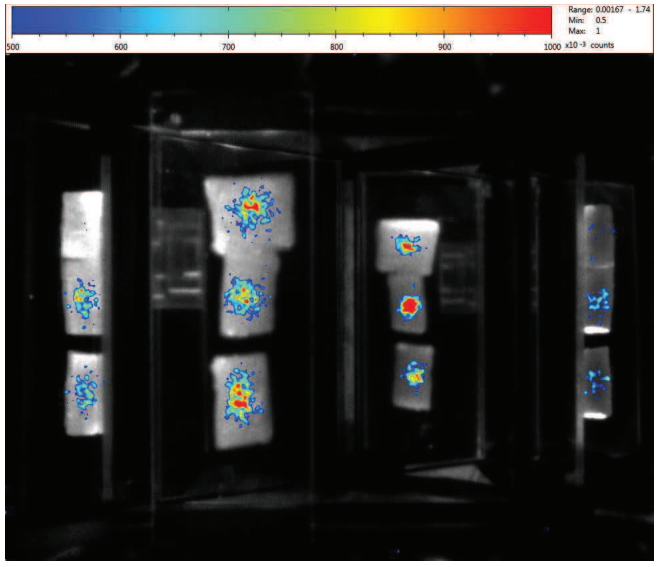

FIGURE 5.10 – Image enregistrée de morceaux de muscle de viande rouge injecté avec 3 MBq de ^{18}F. Image acquise pendant **5 min avec filtre passe-haut à 700 nm**.

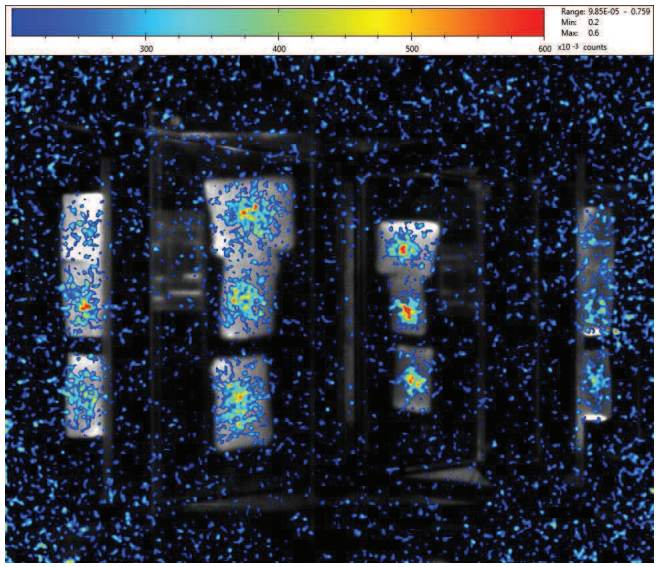

FIGURE 5.11 – Image enregistrée de morceaux de muscle de viande rouge injecté avec 3 MBq de ^{18}F. Image acquise pendant **5 min avec filtre passe-haut à 770 nm**.

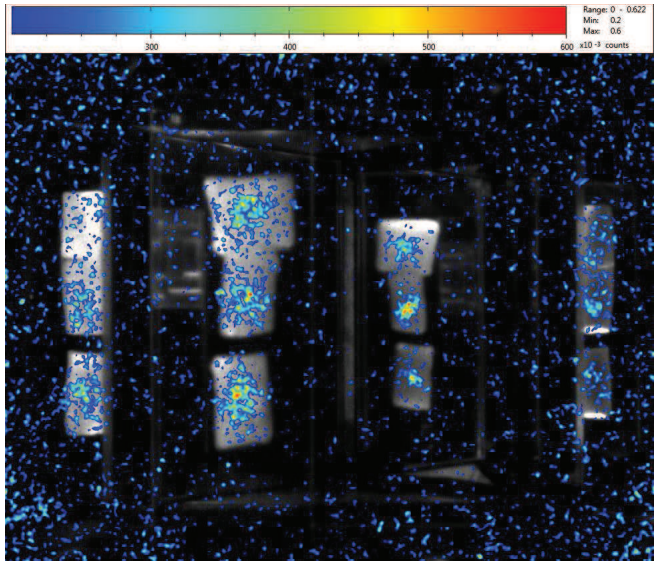

FIGURE 5.12 – Image enregistrée de morceaux de muscle de viande rouge injecté avec 3 MBq de ^{18}F. Image acquise pendant **5 min avec filtre passe-bande 755-805 nm**.

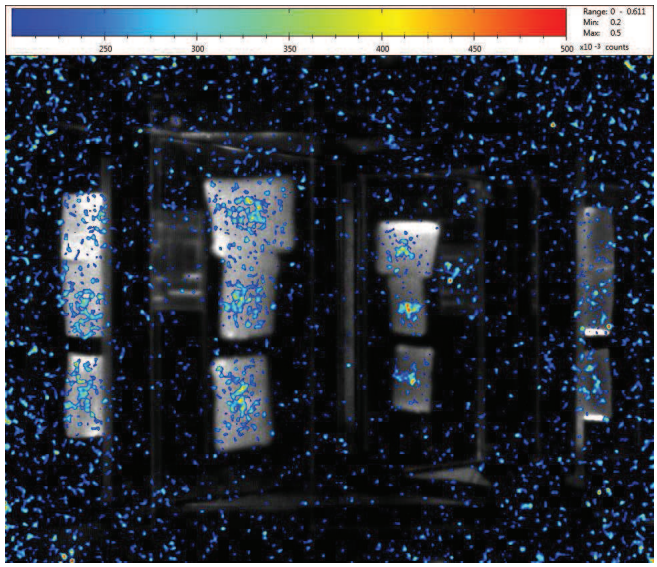

FIGURE 5.13 – Image enregistrée de morceaux de muscle de viande rouge injecté avec 3 MBq de ^{18}F. Image acquise pendant **5 min avec filtre passe-bande 790-840 nm**.

5.2.2 Imagerie Cerenkov 2D sur souris

L'imagerie Cerenkov 2D sur du petit animal permet d'avoir une information sur la localisation du radiotraceur. Cette partie présente des images Cerenkov obtenues avec différentes molécules marquées au ^{18}F et les images PET associées ainsi qu'avec du ^{32}P-ATP.

5.2.2.1 FNa

Les images présentées en figures 5.14 ont été obtenues avec le protocole expérimental présenté ci-dessous (tab. 5.6 et 5.7) en utilisant du ^{18}FNa.

Protocole expérimental	
Produit	^{18}FNa, 8,7 MBq
Injection	IP
Biodistribution	45 min
Acquisition TDM	42 s
Acquisition TEP	15 min (après biodistribution)
Acquisition optique	7 x 5 min (après acquisition TEP)

TABLE 5.6 – Protocole expérimental utilisé pour imager une souris injectée avec du ^{18}FNa.

Acquisition optique	
Filtres	Sans filtre puis avec chaque filtre
Durée par filtre	5 min
Module 4 vues ?	Oui
Focus	90 %
O.N.	1,4

TABLE 5.7 – Acquisition optique réalisée sur souris injectée avec du ^{18}FNa.

L'image optique (fig.5.14) montre du signal au niveau du crâne, de la colonne vertébrale et des genoux. Le reste du signal est trop diffus pour l'assigner à une partie anatomique. L'image PET (fig.5.15) confirme la présence de ^{18}F dans la colonne vertébrale et les genoux. Toutefois, l'image PET montre une forte activité au niveau de la mâchoire et non un niveau du crâne.

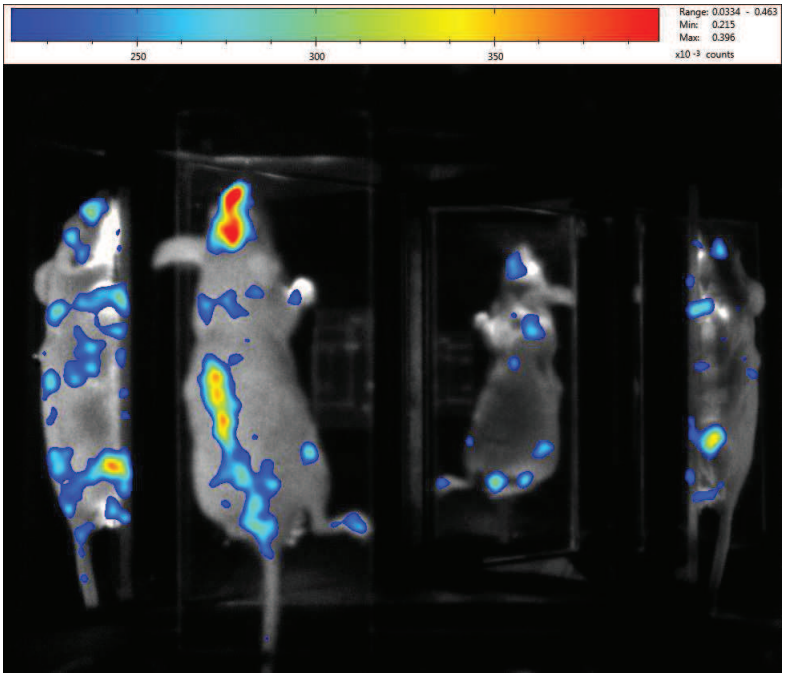

FIGURE 5.14 – Image d'une souris injectée avec 8,7 MBq de FNa par IP. La souris a été laissée 45 min en biodistribution après injection. L'image a été acquise **sans filtre pendant 5 min**.

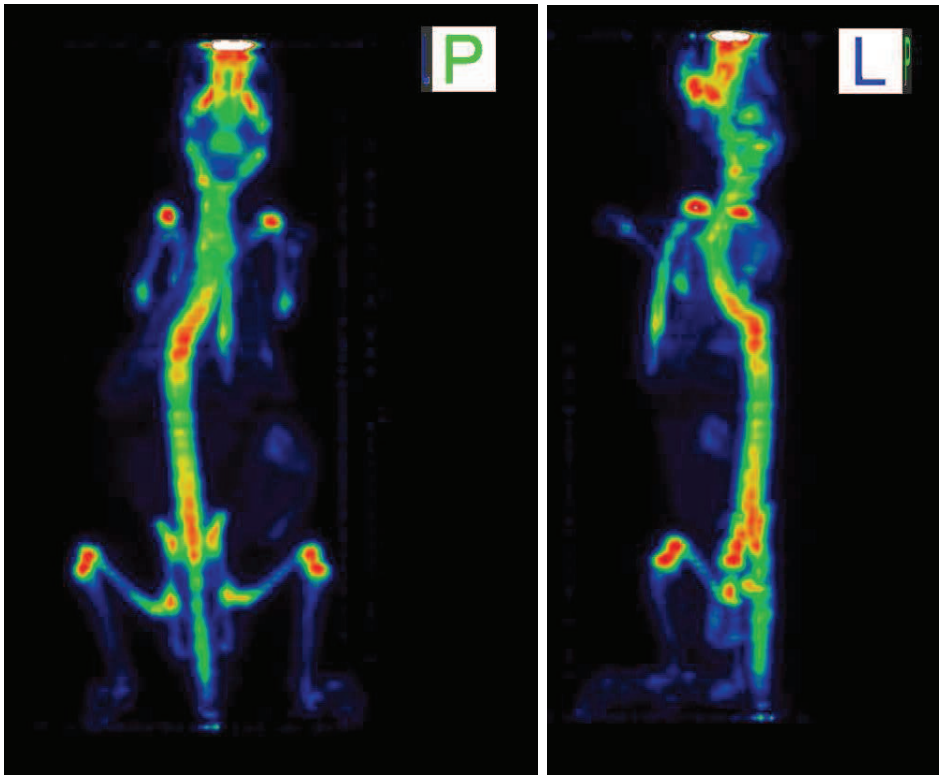

FIGURE 5.15 – Image PET réalisée après biodistribution. La souris a été imagée 15 min.

5.2.2.2 FDG

L'image présentée en figure 5.16 a été obtenue avec le protocole expérimental présenté ci-dessous (tab. 5.8 et 5.9) en utilisant du ^{18}F-FDG.

Protocole expérimental	
Produit	^{18}F-FDG, 7,4 MBq
Injection	IV
Biodistribution	45 min
Acquisition TDM	42 s
Acquisition TEP	15 min (après biodistribution)
Acquisition optique	7 x 5 min (après acquisition TEP)

TABLE 5.8 – Protocole expérimental utilisé pour imager une souris injectée avec du ^{18}FDG.

L'images Cerenkov (fig. 5.16) montre une zone de signal dans le crâne dont les yeux.

Acquisition optique	
Filtres	Sans filtre puis avec chaque filtre
Durée par filtre	5 min
Module 4 vues ?	Oui
Focus	90 %
O.N.	1,4

TABLE 5.9 – Acquisition optique réalisée sur souris injectée avec du ^{18}FNa.

Une deuxième zone se situe dans le cou sur le dessus de la souris. Enfin, une zone de signal très intense au niveau de la vessie. Sur les images PET (fig. 5.17), l'activité se situe dans les yeux, à l'arrière du cou et dans l'abdomen. Au niveau du cou, les souris disposent d'un tissu adipeux qui capte le FDG.

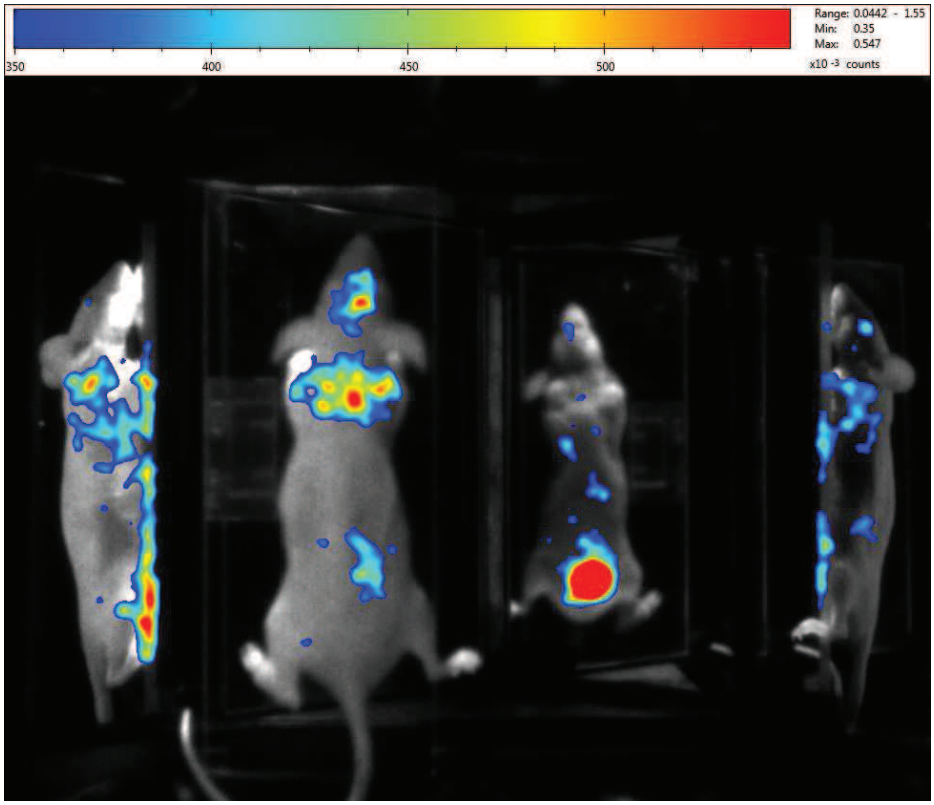

FIGURE 5.16 – Image d'une souris injectée avec 7,4 MBq de FDG par IV. La souris a été laissée 45 min en biodistribution après injection. L'image a été acquise **sans filtre pendant 5 min**.

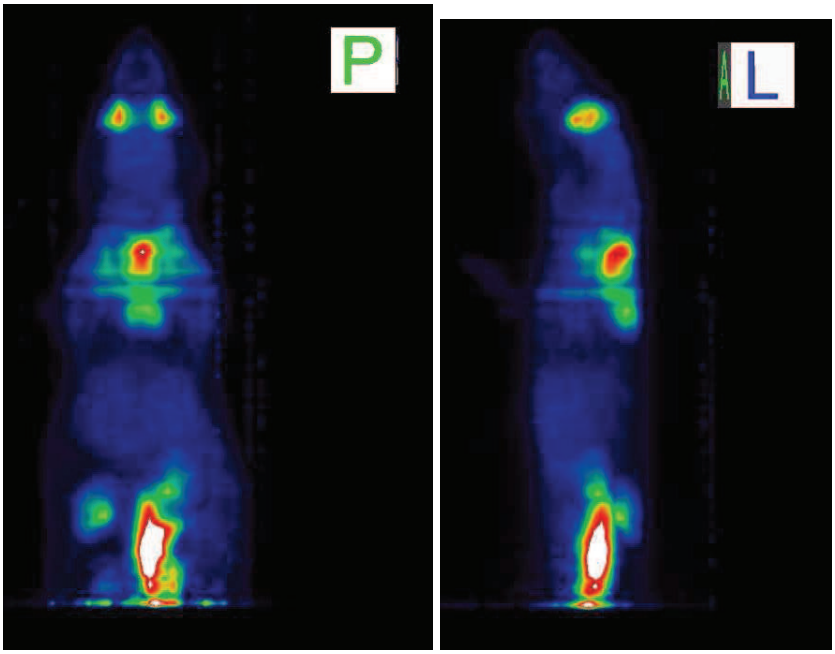

FIGURE 5.17 – Image PET réalisée après biodistribution. La souris a été imagée 15 min.

5.2.2.3 ATP

L'image présentée en figure 5.18 a été obtenue avec le protocole expérimental présenté ci-dessous (tab. 5.10 et 5.11) en utilisant de l'ATP marqué au phosphore-32. Le ^{32}P étant un émetteur β^-, il n'est pas possible de l'imager en PET. Des images Cerenkov ont été enregistrées. La souris est une souris sans poil avec une tumeur sur la patte arrière gauche. La souris est gardée endormie pendant la phase d'imagerie par voie gazeuse.

Protocole expérimental	
Produit	^{32}P-ATP, 3 MBq
Injection	IV jugulaire
Biodistribution	90 min
Acquisition TDM	42 s
Acquisition optique	7 x 5 min (après biodistribution)

TABLE 5.10 – Protocole expérimental utilisé pour imager une souris injectée avec du ^{32}P-ATP.

L'image Cerenkov montre du signal dans le museau, le cou, la colonne vertébrale, la vessie, les genoux, la tumeur et le point d'injection de l'anesthésie pratiqué pour l'injection

Acquisition optique	
Filtres	Sans filtre puis avec chaque filtre
Durée par filtre	5 min
Module 4 vues ?	Oui
Focus	90 %
O.N.	1,4

TABLE 5.11 – Acquisition optique réalisée sur souris injectée avec du ^{32}P-ATP.

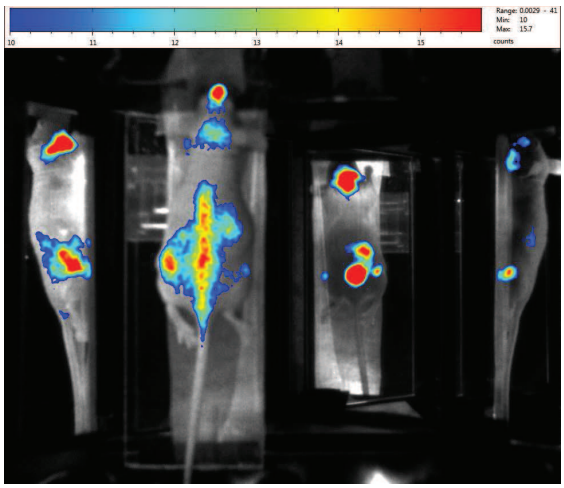

FIGURE 5.18 – Image d'une souris injectée avec 3 MBq de ATP par IP. La souris a été laissée 90 min en biodistribution après injection. L'image a été acquise **sans filtre pendant 5 min**.

de la solution d'ATP (au dessus de la vessie sur l'image du dessous de la souris). Le long de la colonne vertébrale, le signal enregistré provient peut être de la vessie mais qui est moins absorbé et diffusé par les os car moins vascularisé. Le signal situé au niveau du coup provient du point d'injection du produit dans la jugulaire. La tumeur est bien visible sur les vues de dessus et gauche de la souris.

5.3 Tomographie Cerenkov

5.3.1 Reconstruction à partir de données simulées

Des flux surfaciques ont été générés par méthode Monte-Carlo en utilisant le logiciel libre MOSE (Molecular Optical Simulation Environment)([81]). Les simulations ont été effectuées à partir d'un maillage surfacique. Le volume est homogène et les coefficients optiques ont été fixés pour différentes longueurs d'onde. La source optique est placée à différentes profondeurs pour chaque simulation. La source génère 10 millions de photons.

Les reconstructions ont été réalisées sur le même maillage et avec les mêmes coefficients optiques utilisés lors des simulations Monte-Carlo (tab. 5.12 et 5.13). Une méthode des éléments finis a été mise en place. Les coefficients optiques sont ceux déterminés en section 4.1.2.

Informations maillage	
Nombre de points total	16311
Nombre de faces	10306
Nombre de tétraèdres	83402

TABLE 5.12 – Tableau récapitulatif des informations du maillage utilisé dans le cadre des simulations avec MOSE et pour les reconstructions à partir des données simulées.

Informations reconstruction	
Bande spectrale	1, 2 et 3
Facteur de régularisation	k=30
Algorithme	Moindre-carrée à valeurs positives

TABLE 5.13 – Processus de recontruction utilisé dans le cadre de données simulées avec MOSE.

Les images de reconstruction en figure 5.19 à 5.28. Sur les images de reconstruction, les valeurs d'intensités reconstruites sont normalisées par rapport à l'intensité de la source utilisée pour les simulations. Les résultats sont présentés dans le tableau 5.14.

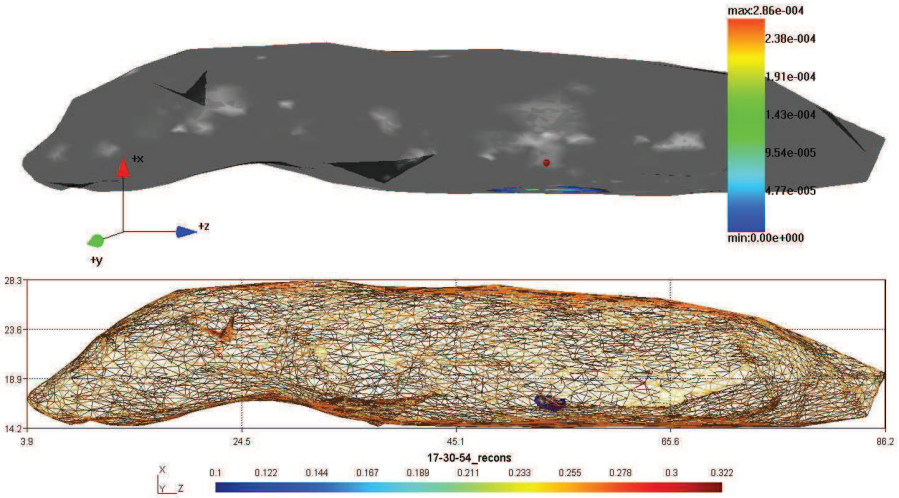

FIGURE 5.19 – Reconstruction obtenue avec **une source simulée en (17,30,54)** avec MOSE.

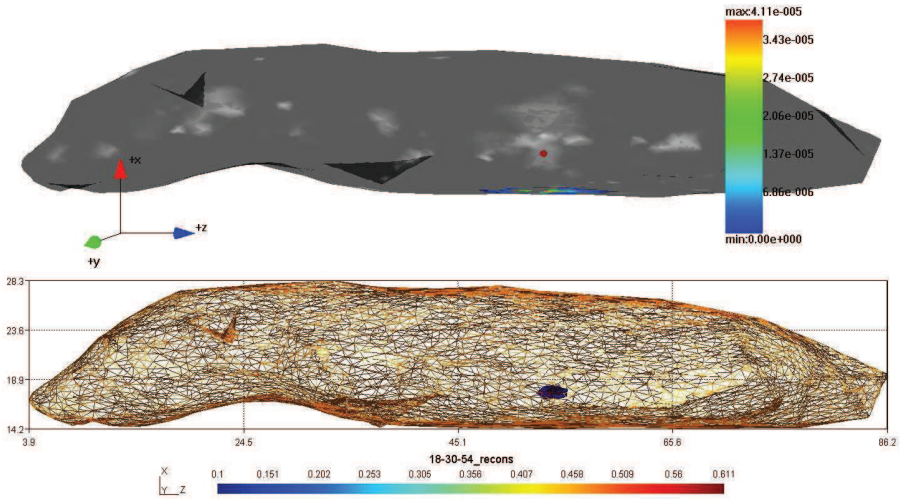

FIGURE 5.20 – Reconstruction obtenue avec **une source simulée en (18,30,54)** avec MOSE.

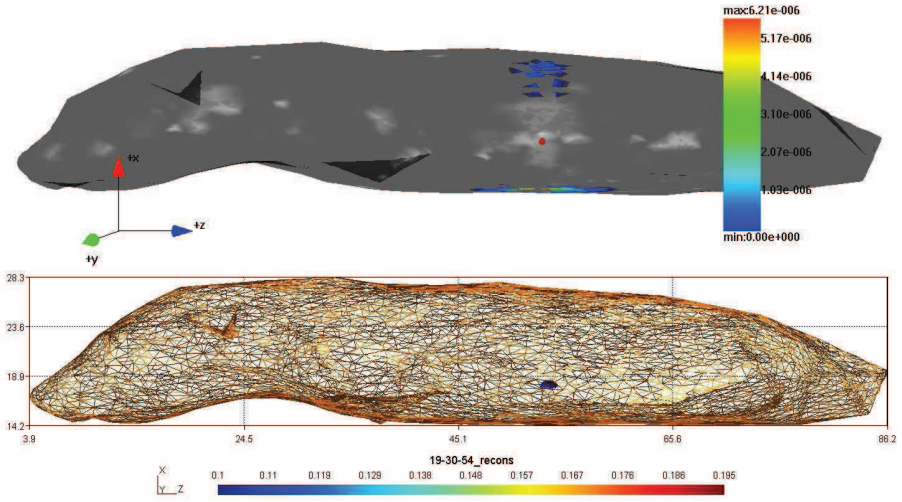

FIGURE 5.21 – Reconstruction obtenue avec **une source simulée en (19,30,54)** avec MOSE.

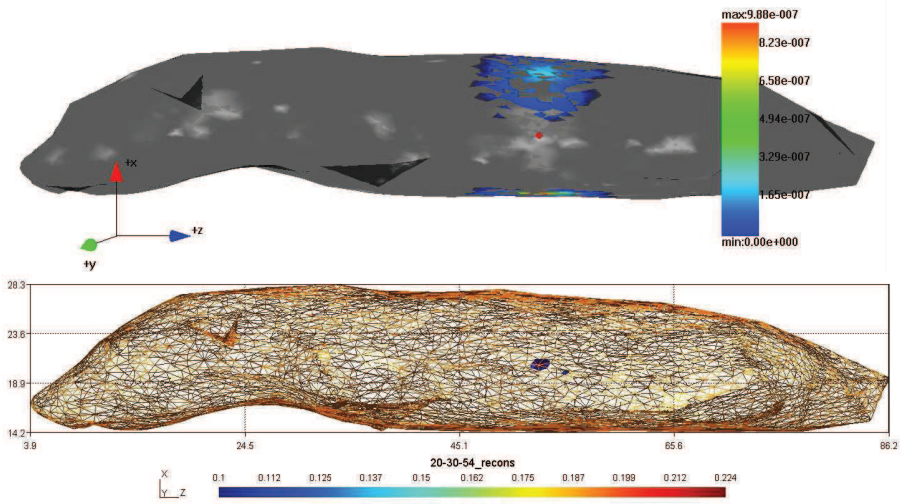

FIGURE 5.22 – Reconstruction obtenue avec **une source simulée en (20,30,54)** avec MOSE.

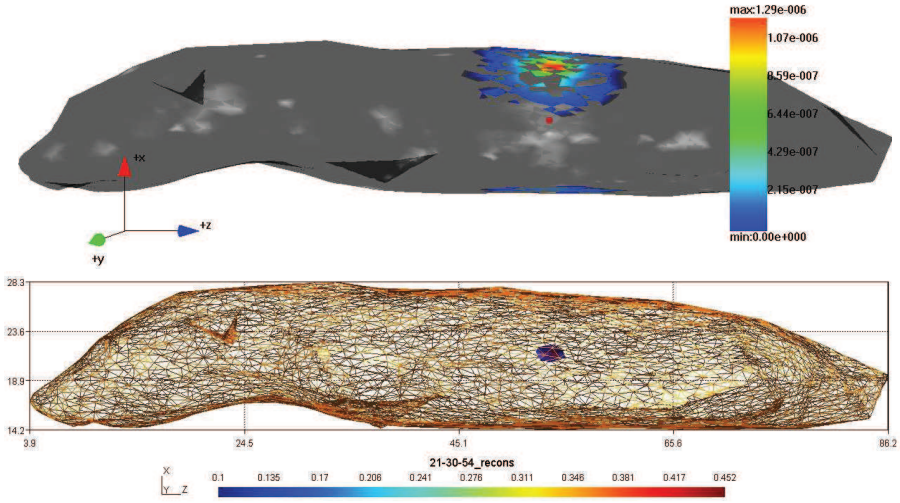

FIGURE 5.23 – Reconstruction obtenue avec **une source simulée en (21,30,54)** avec MOSE.

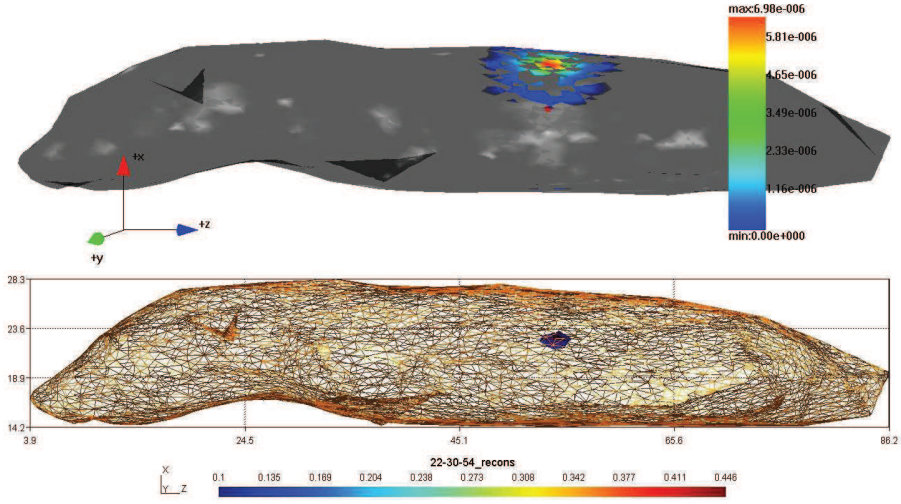

FIGURE 5.24 – Reconstruction obtenue avec **une source simulée en (22,30,54)** avec MOSE.

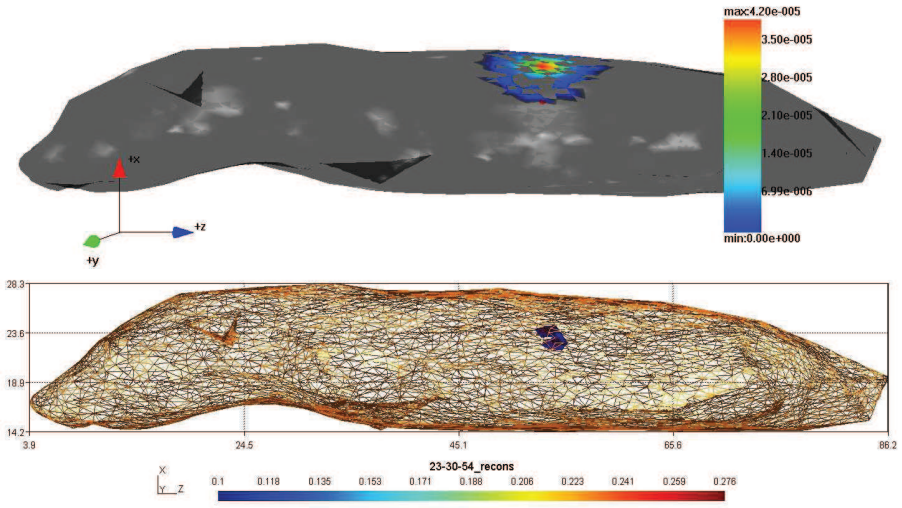

FIGURE 5.25 – Reconstruction obtenue avec **une source simulée en (23,30,54)** avec MOSE.

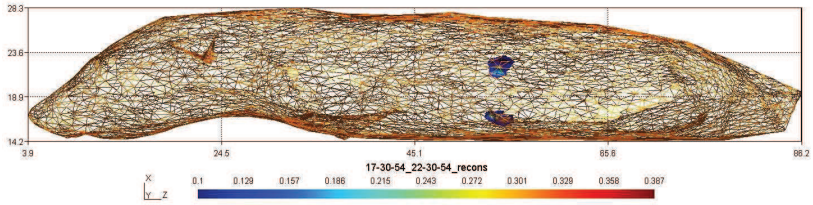

FIGURE 5.26 – Reconstruction obtenue avec **2 source simulées en (17,30,54) et (22,30,54)** avec MOSE.

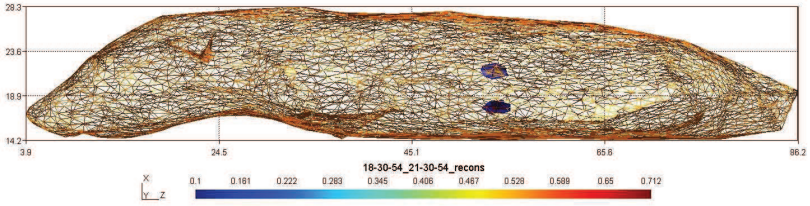

FIGURE 5.27 – Reconstruction obtenue avec **2 source simulées en (18,30,54) et (21,30,54)** avec MOSE.

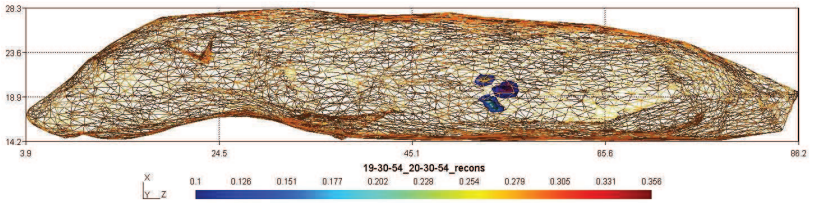

FIGURE 5.28 – Reconstruction obtenue avec **2 source simulées en (19,30,54) et (20,30,54)** avec MOSE.

Point source simulée	Position source reconstruite	Intensité reconstruite normalisée par rapport à la source simulée
(17,30,54)	(17,30,54)	32,2 %
(18,30,54)	(18,30,54)	61,1 %
(19,30,54)	(18,30,54)	19,5 %
(20,30,54)	(20,30,54), présence d'artefact	22,4 %
(21,30,54)	(21,30,54)	45,2 %
(22,30,54)	(22,30,54)	44,6 %
(23,30,54)	(23,30,54), présence d'artefact	27,6 %
(17,30,54) et (22,30,54)	(17,30,54) et (22,30,54)	22 % et 37 %
(18,30,54) et (21,30,54)	(18,30,54) et (21,30,54)	53 % et 71 %
(19,30,54) et (20,30,54)	(19,30,54) et (20,30,54), présence d'artefact	27 % et 20 %

TABLE 5.14 – tableau récapitulatif des résultats de reconstruction obtenus à partir de données simulées avec MOSE.

Les positions reconstruites coïncident avec la position de la source. Dans le cas de 2 sources, les positions des 2 sources sont reconstruites. On constate la présence d'artefact dans certaines images. De plus, la source reconstruite est plus faible que la source utilisée pour les simulations.

Le modèle utilisé ainsi que l'algorithme employé permettent d'effectuer des reconstructions précises de sources ponctuelles. Le modèle tient compte de l'information multispectrale de 3 bandes. Lors de l'utilisation de moins de bande, les reconstructions sont plus lentes

à converger et parfois n'aboutissent pas. La source reconstruite dans ces cas est moins profonde. L'erreur faite sur la profondeur peut aller jusqu'à 2 mm.

5.3.2 Reconstruction *in vitro*

Un morceau de viande rouge a été injecté avec 18,8 MBq de ^{18}F et utilisé pour réaliser de la tomographie Cerenkov (tab. 5.15 et 5.16). Une activité élevée de ^{18}F a été utilisée afin d'obtenir des images avec une tâche optique propre à un point source.

Protocole expérimental	
Produit	^{18}FNa, 18,8 MBq
Injection	25 μL dans un morceau de muscle de viande rouge
Acquisition TDM	42 s
Acquisition optique	7 x 5 min

TABLE 5.15 – Protocole expérimental utilisé pour imager un morceau de muscle de viande rouge injecté avec un point source de ^{18}FNa.

Acquisition optique	
Filtres	Sans filtre puis avec chaque filtre
Durée par filtre	5 min
Module 4 vues ?	Oui
Focus	90 %
O.N.	1,4

TABLE 5.16 – Acquisition optique réalisée sur un morceau de muscle de viande rouge injecté avec un point source de ^{18}FNa.

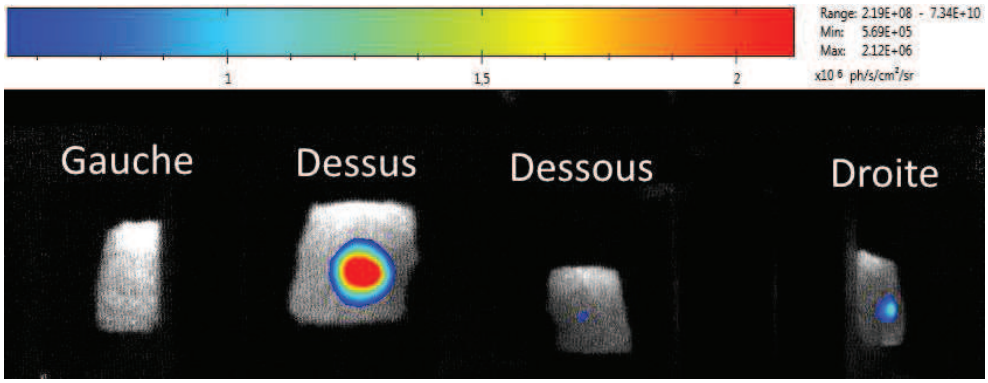

FIGURE 5.29 – Image 4 vues d'un morceau de muscle injecté avec 18,8 MBq de ^{18}FNa dans 25 μL. L'image a été acquise pendant **5 min sans filtre**.

Une des images Cerenkov obtenues est présentée en figure 5.29. La reconstruction du flux surfacique est réalisée en utilisant les 4 vues de l'objet. Le maillage utilisé et le processus de recontruction sont décrits dans les tableaux 5.17 et 5.18

Informations maillage	
Nombre de points total	1709
Nombre de faces	1478
Nombre de tétraèdres	8085

TABLE 5.17 – Tableau récapitulatif des informations du maillage généré à partir du volume TDM du morceau de muscle.

Informations reconstruction	
Vues utilisées	4 vues utilisées
Facteur de régularisation	k=30
Algorithme	Moindre-carrée à valeurs positives

TABLE 5.18 – Processus de recontruction utilisé dans le cadre d'un point source de ^{18}FNa dans du muscle.

La figure 5.30 représente la reconstruction du flux optique en surface à partir des 4 vues et sans filtre.

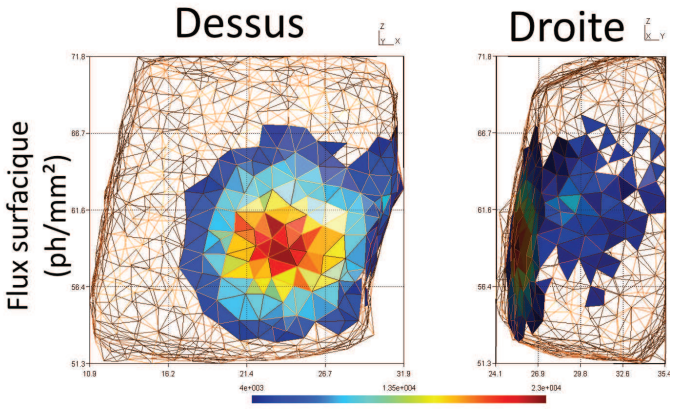

FIGURE 5.30 – Reconstruction du flux optique transmis en surface à partir des 4 vues et sans filtre.

Les reconstructions Cerenkov 3D sont effectuées pour chaque bande (1 à 5, fig. 5.31 à 5.35) et en multispectrale (fig. 5.36 et 5.37).

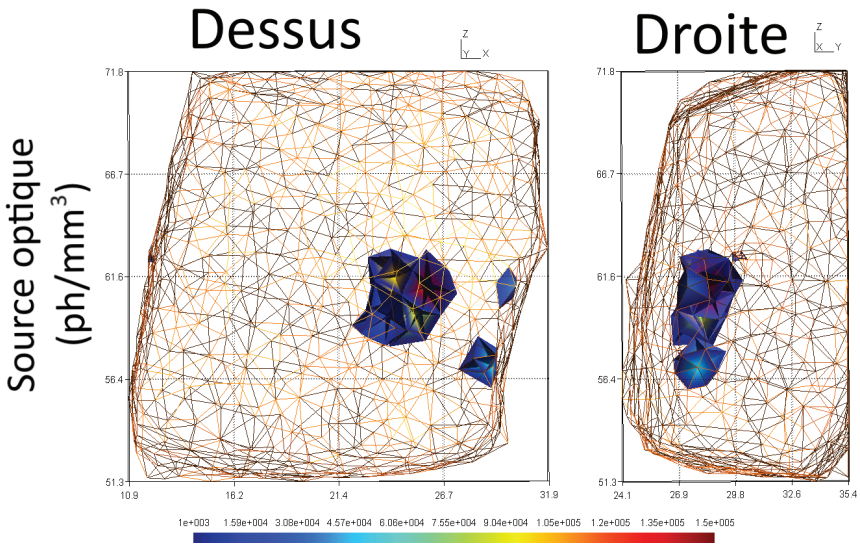

FIGURE 5.31 – Reconstruction de la source optique à partir de la **bande spectrale 1**.

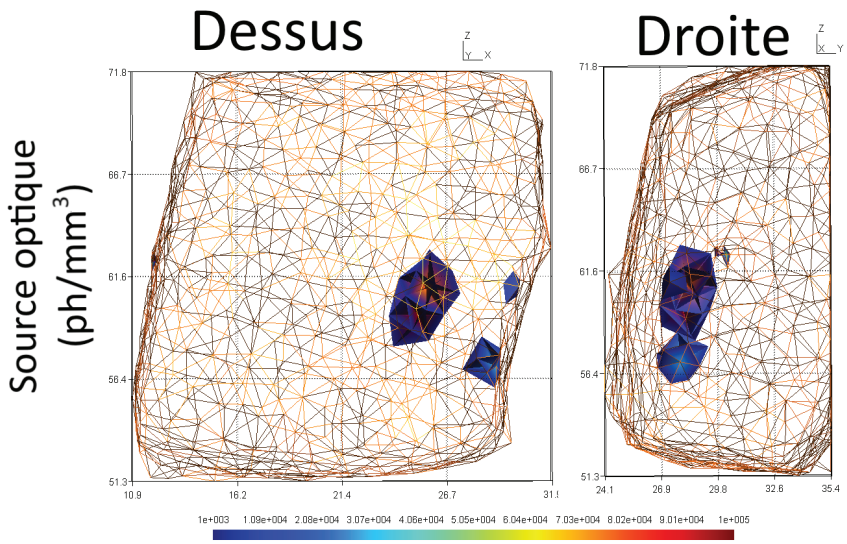

FIGURE 5.32 – Reconstruction de la source optique à partir de la **bande spectrale 2**.

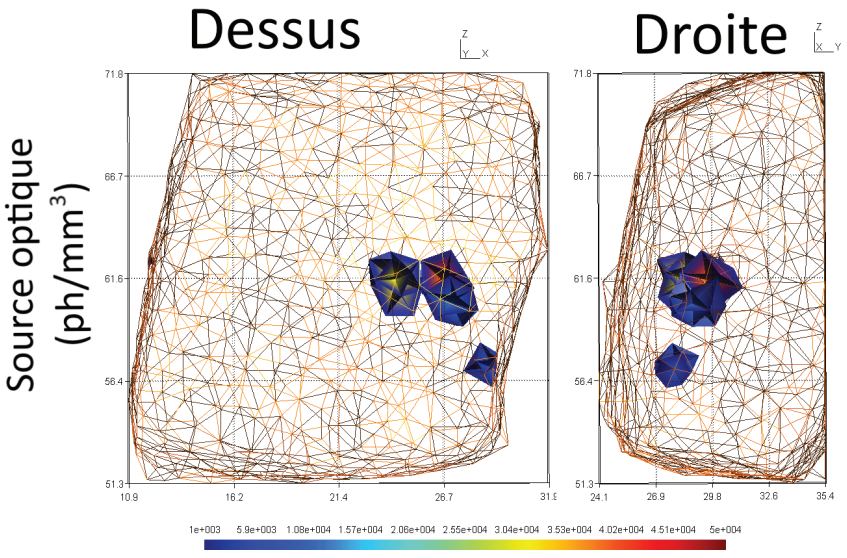

FIGURE 5.33 – Reconstruction de la source optique à partir de la **bande spectrale 3**.

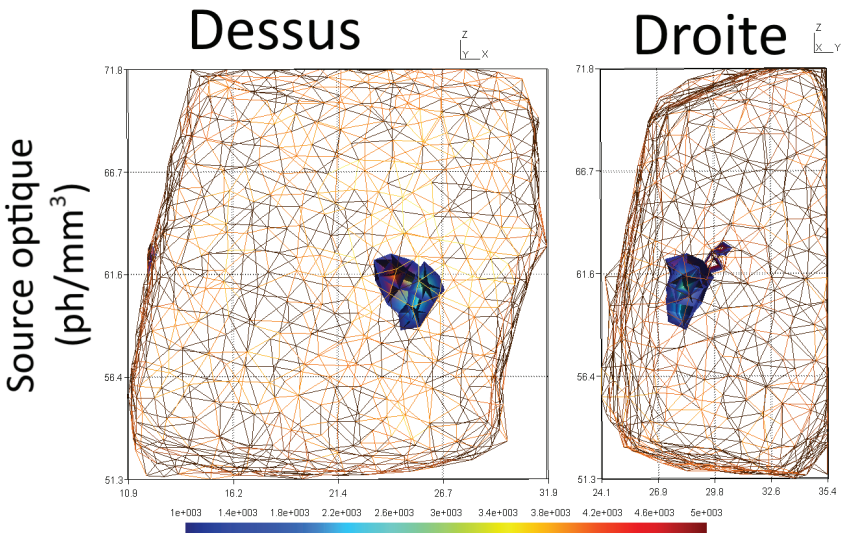

FIGURE 5.34 – Reconstruction de la source optique à partir de la **bande spectrale 4**.

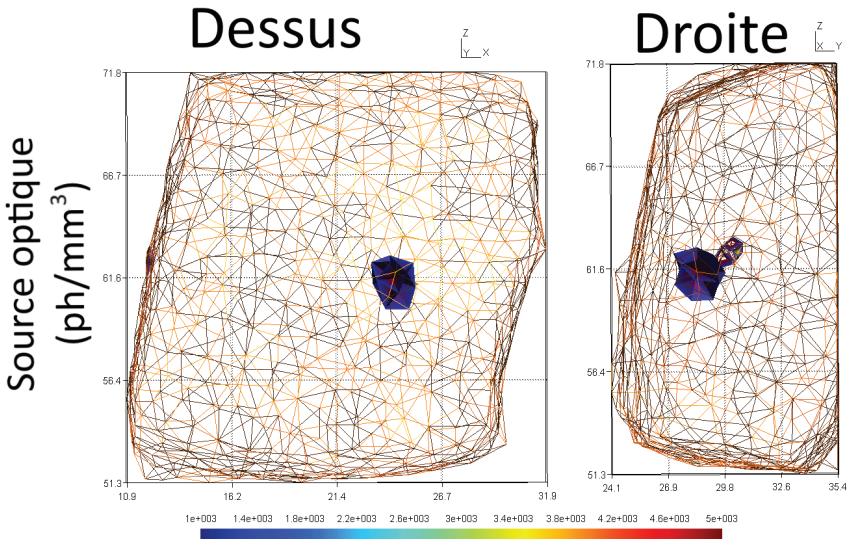

FIGURE 5.35 – Reconstruction de la source optique à partir de la **bande spectrale 5**.

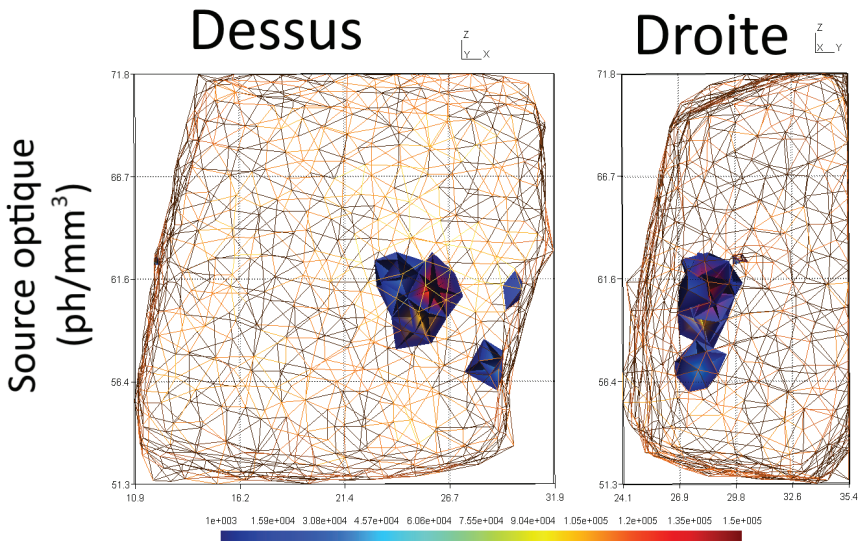

FIGURE 5.36 – Reconstruction de la source optique à partir de la **bande spectrale 1 et 2**.

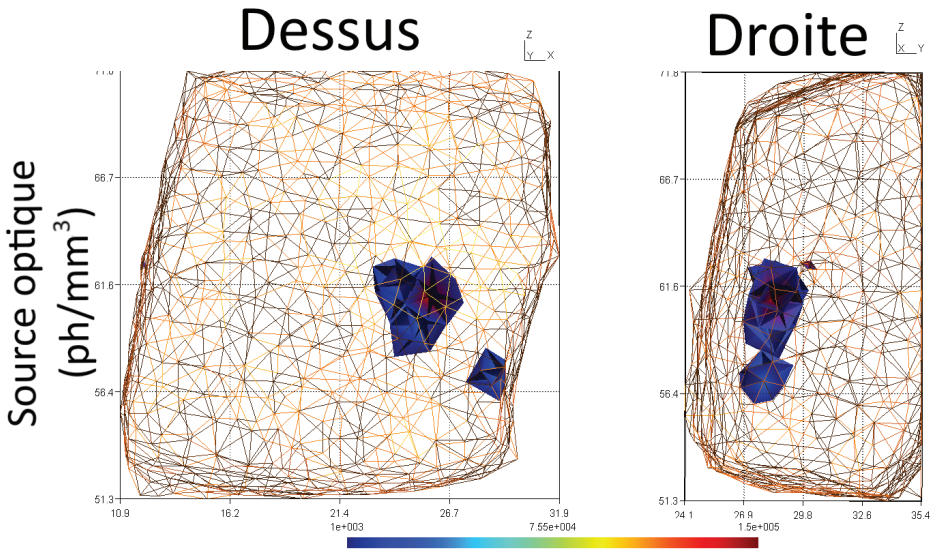

FIGURE 5.37 – Reconstruction de la source optique à partir de la **bande spectrale 1 et 3**.

La position de la source reconstruite coïncide avec les maximums du flux surfacique

reconstruit. Certaines images possèdent des artefacts (reconstructions avec les bandes 1 à 3). Toutefois, les reconstructions avec les bande 4 et 5 (filtre passe-bande) ne possèdent pas d'artefact. L'origine de ces résultats meilleurs est peut être due à l'absorption plus faible des tissus dans les bande 4 et 5.

5.3.3 Reconstruction *in vivo*

Dans le cadre des reconstruction *in vivo*, les images avec du ^{18}F n'ont pas assez de signal pour les activités utilisées. Les reconstructions ont été réalisées sur les images avec le ^{32}P (section 5.2.2.3)(fig. 5.38). Les reconstructions *in vivo* ont été effectués avec du ^{32}P-ATP avec les images réalisés sur la souris possédant une tumeur.

Pour les reconstructions Cerenkov 3D, seul l'abdomen a été utilisé pour le maillage (tab. 5.19). Le signal au niveau du cou provient du lieu de l'injection de la solution de radiotraceur. Cette zone n'est pas caractéristique de la biodistribution du traceur et n'est pas reconstruite.

Informations maillage	
Nombre de points total	1282
Nombre de faces	1234
Nombre de tétraèdres	5824

TABLE 5.19 – Tableau récapitulatif des informations du maillage généré à partir du volume TDM de la souris.

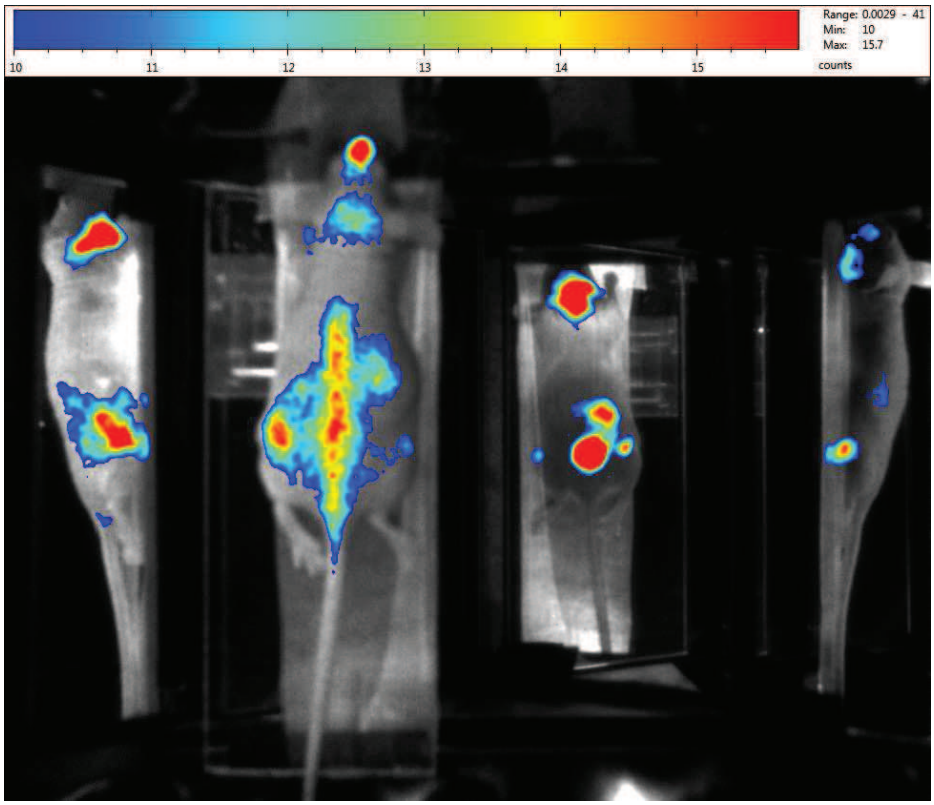

FIGURE 5.38 – Image d'une souris injectée avec 3 MBq de ATP par IP. La souris a été laissée 90 min en biodistribution après injection. L'image a été acquise **sans filtre pendant 5 min**.

La reconstruction du flux optique en surface a été réalisé pour différentes combinaisons de vues. Ici sont présentés les flux optiques obtenus avec la vue de dessus uniquement (fig. 5.39) et la vue de dessous uniquement (fig. 5.40).

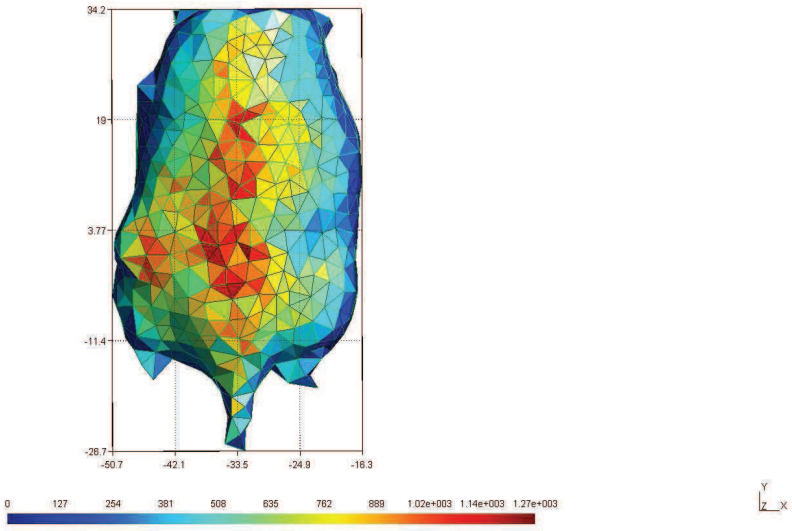

FIGURE 5.39 – Flux optique en surface reconstruit avec uniquement la vue de dessus.

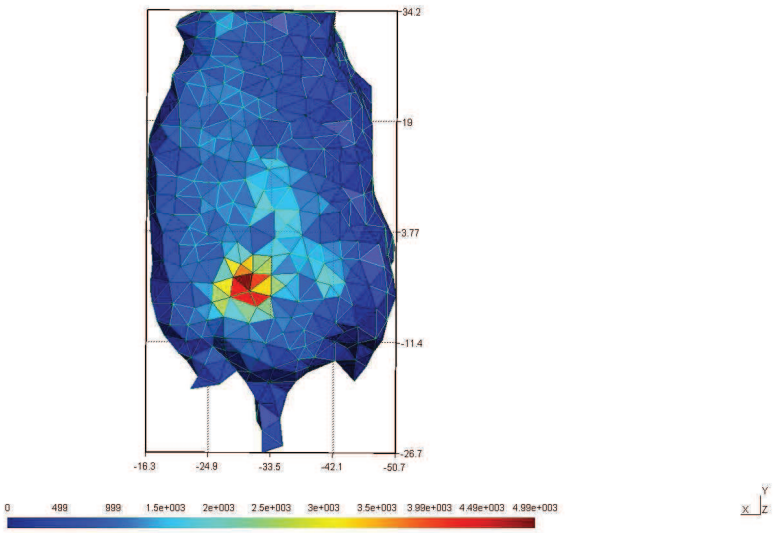

FIGURE 5.40 – Flux optique en surface reconstruit avec uniquement la vue de dessous.

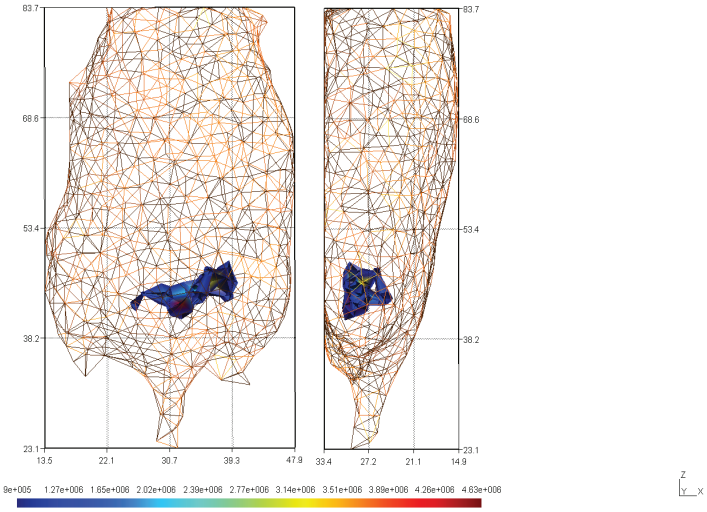

FIGURE 5.41 – Reconstruction obtenue en utilisant la **vue de dessous seulement** pour la reconstruction du flux optique en surface.

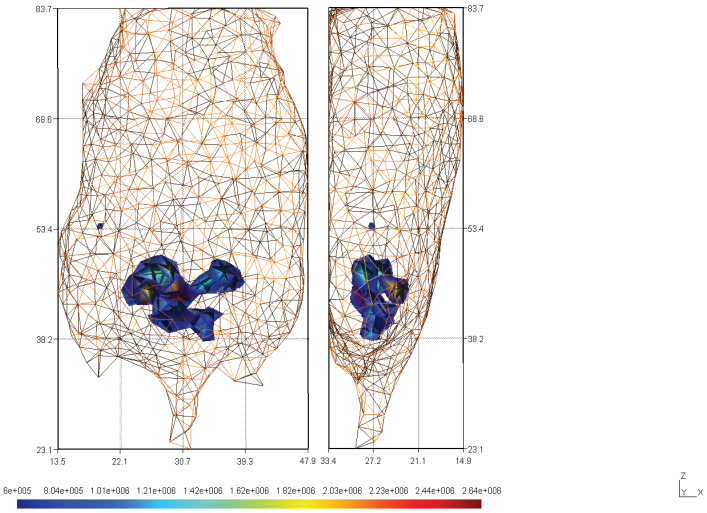

FIGURE 5.42 – Reconstruction obtenue en utilisant la **vue de dessus seulement** pour la reconstruction du flux optique en surface.

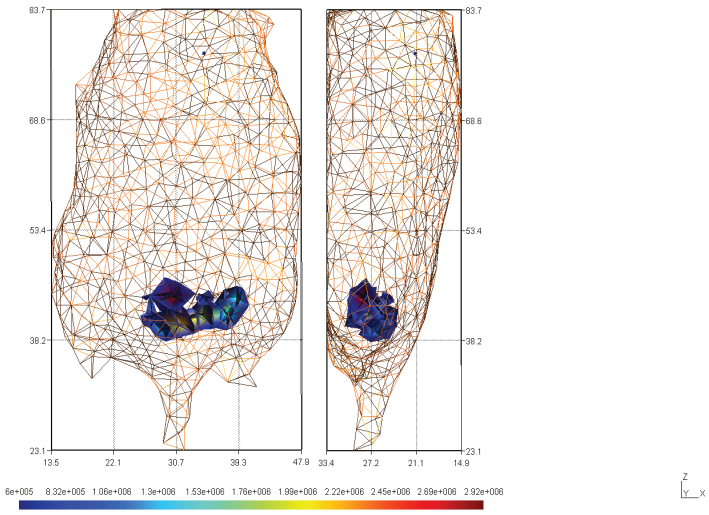

FIGURE 5.43 – Reconstruction obtenue en utilisant les **vues de dessous et de dessus** pour la reconstruction du flux optique en surface.

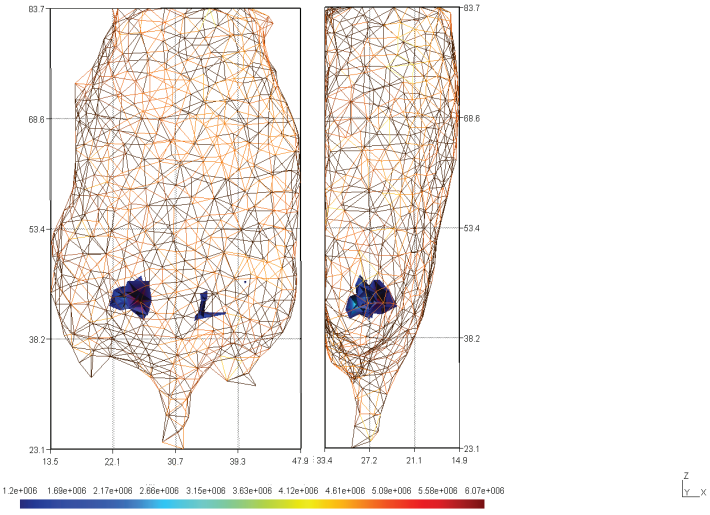

FIGURE 5.44 – Reconstruction obtenue en utilisant les **vues de dessus, gauche et droite** pour la reconstruction du flux optique en surface.

Les images précédentes montrent les reconstructions obtenues avec la bande 1 en utilisant la vue de dessous (fig. 5.41), la vue de dessus (fig. 5.42), les vues de dessus et dessous (fig. 5.43) et les vues de gauche, dessus et droite ensemble (fig. 5.44). La vessie est reconstruite dans chacun des cas mais la tumeur n'est pas visible. Des artefacts sont présents dans les reconstructions. Dans le cas de la reconstruction avec les vues de dessus, gauche et droite, la source reconstruite est décalée sur les côtés. Dans la reconstruction avec vue de dessus uniquement, un petit point est reconstruit au niveau de la tumeur (fig. 5.42).

5.4 Discussion

5.4.1 Imagerie

L'imageur optique utilisé est particulier à cause de la présence d'un tube amplificateur de lumière. Le rendement quantique de la photocathode est à comparer au rendement quantique d'un simple capteur CCD. Alors que les capteurs CCD peuvent avoir des rendements quantiques supérieurs à 90 % dans le visible, la photocathode a un rendement quantique de 29.6 % maximum. Toutefois, le rendement quantique de la photocathode est quasiment constant dans le visible ce qui facilite le traitement spectral des images enregistrées. Il n'empêche que ce rendement est 3 fois plus faible que celui des autres systèmes optiques utilisés en imagerie Cerenkov comme les systèmes IVIS.

La présence du tube amplificateur permet de s'affranchir du bruit du capteur CCD. Par contre, il génère un bruit qui n'est pas homogène. Le bruit est plus intense lorsque l'on s'éloigne du centre optique. Ainsi, le signal sur le bord des images dispose d'un rapport signal/bruit plus faible. Avec le module 4 vues, les vues latérales de l'objet ne disposent pas du même rapport signal sur bruit. Le signal provenant des vues latérales est surévalué. De plus, le bruit du tube amplifacateur varie davantage loin du centre optique ce qui rend difficile son élimination par seuillage.

Un autre point lié au module 4 vues est que les vues ne sont pas acquises suivant les mêmes conditions de distance par rapport au détecteur, par rapport à l'axe optique et ne fait pas intervenir le même nombre de miroirs. Mais les 4 vues sont acquises simultanément permettant de réduire le temps d'acquisition, ce qui est un avantage si des isotopes à courte demi-vie sont utilisés. Comparé au système de rotation de l'animal et de son support [49, 63, 64, 65, 66, 67], le module 4 vues présente un avantage en temps d'acquisition des images optiques mais les images obtenues demandent des traitements locaux dans l'image. Dans le cas du 4 vues, l'animal est juste posé sur le support alors que dans le cas du dispositif rotatif, l'animal est est disposé à la verticale dans le système rotatif et la souris est pendue par les pattes. Le dispositif rotatif demande plus de manipulations de la souris que le module 4 vues pour fixer l'animal ainsi qu'un temps d'imagerie proportionnel au nombre de vues. Mais le dispositif rotatif permet l'acquisition d'un TDM sans manipulation supplémentaire de l'animal. La prise simultanée des différentes vues permet aussi d'avoir les vues de l'animal dans la même position.

Le recalage entre les images TDM et optiques du PhotonImager est réalisé en pointant manuellement l'objet de référence sur l'image CT et sur une image optique. Cette étape est réalisée pour chaque objet à reconstruire. Cette étape est rapide. Le contraste entre certains tissus reste faible pour obtenir un modèle hétérogène rapidement à partir des images TDM.

5.4.2 Imagerie Cerenkov 2D

5.4.2.1 Constats

Le signal Cerenkov observé pour une source de ^{18}F est expérimentalement de 0,7 ph/désintégration. Cette valeur est identique à celle trouvé dans [126]. Le signal Cerenkov produit par du ^{32}P est théoriquement de 21 supérieur à celui du ^{18}F selon [126]. Ici, on trouve un

rapport de 18 entre le ^{32}P et le ^{18}F. Les résultats expérimentaux sont cohérents avec la théorie et avec d'autres résultats expérimentaux publiés [126].

Le signal Cerenkov provenant de sources dans des tissus biologiques est très faible avec les activités utilisées. Les images avec les filtres passe-bande ont un rapport signal sur bruit trop faible pour être exploité. Le signal à des longueurs d'onde inférieures à 700 nm est très absorbé par les tissus. L'utilisation de filtres étroits, bien que permettant de faire une moins grande erreur sur l'estimation des coefficients, ne peut que dégrader les images enregistrées.

Les images Cerenkov 2D obtenues dépendent des propriétés de l'isotope. Le ^{18}F est un émetteur de positons. Dans l'eau, la majeure partie des positons ne génèrent pas de rayonnement Cerenkov. De plus, le parcours des positons peut être abrégé par annihilation. Ainsi, avec le ^{18}F, il est nécessaire d'utiliser des activités importantes pour avoir un signal exploitable. Avec le ^{32}P, l'énergie d'émission est plus élevée et les électrons générés ne s'annihilent pas. Cela implique des libres parcours moyens différents en fonction de l'énergie et du type de rayonnement β. Le temps de demi-vie du ^{32}P est plus long et permet de faire des images sur une plus grande plage de temps. Toutefois, le ^{32}P, utilisé en thérapie, est plus invasif que le ^{18}F.

5.4.2.2 Traitement d'image

Les images Cerenkov 2D acquises sont lissées pour faire apparaître les tâches optiques en éliminant le bruit et les pixels vides. Toutefois, le bruit provenant du tube est inhomogène dans l'image. Cela impliquerait de faire du traitement d'image en fonction de la zone de l'image. De plus, le lissage introduit un biais puisque plus le lissage est important, plus le signal optique sera circulaire. Dans le cas d'un point source avec une surface plane et dans un milieu homogène, le signal optique est circulaire. Mais dans le cas d'un modèle complexe comme une souris, ce biais n'est généralement pas vrai.

De la même manière, le seuillage n'est pas appliqué sur l'image acquises lors du traitement puisque le bruit n'est pas homogène et que le signal observé pour chacune des vues n'est pas acquis dans les mêmes conditions. Le seuillage est utilisé sur les images pour une meilleure visualisation et interprétation des images 2D. Les images latérales par exemple ont un bruit plus élevé et moins de signal est enregistré. Le seuillage est donc omis lors du traitement quantitatif des images.

5.4.3 Tomographie Cerenkov

5.4.3.1 Maillage

Le maillage est généré arbitrairement pour obtenir un rendu des surfaces et une discrétisation du volume corrects avec un nombre de points et de faces du maillage minimal. Le critère pour générer le maillage est d'avoir des segments de longueur inférieure à 1,5 mm. Cette valeur permet d'avoir des faces dont la taille ne descend pas en dessous de la résolution des images 2D et d'avoir un échantillonnage du volume suffisant pour les tests expérimentaux. Pour des objets de faible volume comme dans le cas des tests *in vitro*, la densité de points est augmentée.

Des maillages hétérogènes peuvent être générés mais les surfaces internes présentent souvent beaucoup de petites structures (certains os comme les côtes par exemple). Cela implique d'augmenter l'échantillonnage pour avoir des éléments de maillage plus fins.

Ces choix ont été réalisés en tenant compte des contraintes de résolution 2D des images et de la résolution 3D suffisantes pour discriminer plusieurs sources.

5.4.3.2 Flux optique en surface

La reconstruction du flux transmis en surface est dépendante du modèle du système optique. Les vues étant acquises dans des conditions différentes, il est difficile de corréler deux vues entre elles. En particulier, chaque vue est décalée par rapport à l'axe optique et l'effet de perspective s'accentue. L'effet de perspective contribue à voir les sources sous un angle solide différent. Plus on s'éloigne du centre optique, plus l'objet est vu avec un angle solide faible. Chaque vue étant à des distances différentes de l'objet, il est délicat de reconstruire un flux surfacique observé sous deux angles de vue différents. Afin de minimiser cet effet, le maillage surfacique présente des faces suffisamment étendues pour que l'effet de perspective n'apparaisse pas.

Certaines sources ne sont pas visibles en fonction de l'angle de vue. L'utilisation de plusieurs vues permet d'enregistrer plus de signal. Sur les images 2D, plusieurs vues permettent déjà de discriminer une source sous-cutanée (visible seulement sur une vue) d'une source en profondeur (visible sur plusieurs vues).

La reconstruction du flux transmis en surface demande de tenir compte du problème de radiance à la surface en fonction de l'angle. Le modèle utilisé suppose une émission dans toutes les directions au niveau de la surface avec une variation d'intensité en fonction de l'angle par rapport à la normale. Les points précédents comme la perspective et les faces larges ne permettent pas de connaître l'angle précisément. En particulier, une même zone reconstruite avec deux vues ou une des deux ne donnera pas les mêmes résultats. Les vues proches de l'axe optique comme celles du dessus et du dessous posent moins de problèmes.

Un dernier problème est l'effet du volume qui intervient différemment sur chacune des vues. Pour un pixel correspond un angle solide. Cet angle solide délimite un volume si on le déplace suivant l'angle de vue. Pour un même point en surface, s'il est vu par deux angles de vue différents, le volume correspondant est différent. Par conséquent, une erreur est causée lors de la reconstruction du flux surfacique qui est dépendant de la source en volume dans l'objet. Or cette partie est traitée indépendamment des problèmes en volume.

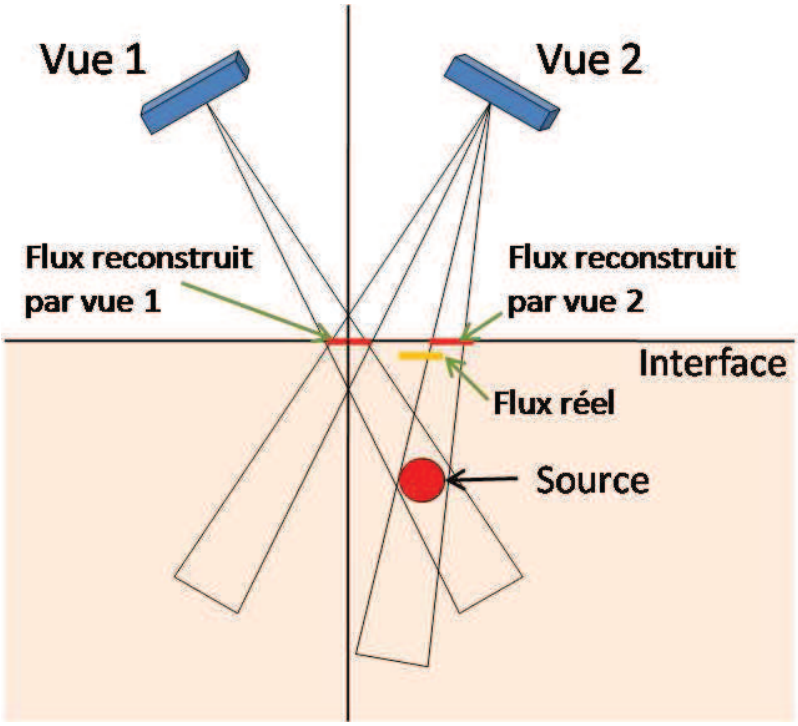

FIGURE 5.45 – Schéma de l'effet de volume intervenant dans la reconstruction du flux optique en surface.

Le moyen de limiter l'effet de volume serait de toujours enregistré le signal selon la normale à la surface. Mais cela implique de prendre des vues sur 360°.

Dans le cadre de ma thèse, les vues de dessus et dessous disposent d'une surface proche de ces conditions plus importantes que les vues latérales dans le cas d'une souris. Les vues de dessus et dessous sont donc à privilégier lors de la reconstruction du flux optique en surface.

Actuellement, seule la méthode proposé par Spinelli et al. [48] dispose d'un modèle corrélant l'information de la source avec les images 2D directement dans le processus de reconstruction mais n'utilise qu'une seule vue. Dans les autres cas, la reconstruction du flux optique en surface est une étape réalisée indépendamment des problèmes en volume. Toutefois, certains disposent d'un système permettant d'acquérir les images sur 360° pour palier à ce problème.

5.4.3.3 Modèle de propagation

5.4.3.3.1 Modèle SP_3

Le modèle utilisé en tomographie optique est souvent l'équation de diffusion. Toutefois,

cette équation n'est valable que pour des milieux où l'absorption est négligeable devant la diffusion. Les approximations $S_N$et P_N permettent de converger vers une solution exacte lorsque que N augmente. Mais ces méthodes demandent beaucoup de ressources informatiques (mémoire vive (RAM) en particulier) surtout quand l'ordre N augmente. L'approximation SP_N ne converge pas quand l'ordre N augmente mais le système à résoudre est plus compact car le traitement des diffusions multiples est simplifié.

Le choix d'utiliser la méthode SP_3 provient de plusieurs facteurs. Dans un premier temps, plusieurs publications comparant la méthode SP_N pour les ordres allant jusque 7 (1, 3, 5 et 7) montrent des résultats bien meilleurs pour la méthode SP_3 que pour la méthode SP_1. Les ordres supérieurs donnent parfois de meilleurs résultats mais moins significatifs que lors du passage de SP_1 à SP_3. De plus, les ordres 5 et 7 donnent parfois mieux, parfois moins bien que l'ordre 3. Par conséquent, la méthode SP_3 demeure un bon compromis entre précision, rapidité et ressources de calcul.

L'utilisation de la méthode SP_3 implique la mise en place d'un système de taille 2 fois le nombre de points du maillage. Ainsi avec l'implémentation réalisée, un trop grand nombre de points (> 8000) sature la mémoire lors de la mise en place du système et demande du SWAP (utilisation de la mémoire disque dur lors de la saturation de la RAM, beaucoup plus lente que la RAM) qui augmente considérablement le temps de calcul. Cette limite est propre à l'implémentation réalisée et impose une contrainte sur la taille des maillages. En particulier, les maillages hétérogènes générés (tissus mous et os uniquement) dépassent cette limite à cause da la présence de petites structures comme les côtes ou le dessus du crâne qui nécessitent un maillage beaucoup plus fin pour qu'elles soient représentées. L'implémentation est réalisée pour prendre en compte les hétérogénéités mais les maillages sont trop denses pour pouvoir être utilisés dans les reconstructions. Par conséquent, les maillages utilisés dans le cas *in vivo* sont homogènes malgré la possibilité d'extraire l'information sur les os ou les poumons (moins absorbants que les autres organes) des images TDM.

5.4.3.3.2 Coefficients optiques

Bien que les modèles S_N et P_N permettent de converger vers une solution exacte contre un coût de calcul important, la détermination des coefficients optiques est difficile surtout dans le cas *in vivo*. L'idéal serait de mesurer les coefficients optiques pour chaque animal. Mais la tomographie optique diffuse est une modalité d'imagerie tout aussi complexe. Les valeurs obtenues expérimentalement *in vitro* de ces coefficients sont dépendants du protocole expérimental, du dispositif expérimental et du modèle utilisé pour retrouver les valeurs [70].

Un autre point délicat est la variation de ces coefficients en fonction de la longueur d'onde. Comme les images sont enregistrées dans une bande de longueur d'onde, une seule valeur de coefficients peut être utilisée alors que plusieurs longueurs d'onde sont présentes. De plus, il est impossible de dire quelle longueur d'onde est majoritaire dans la bande pour sélectionner au mieux les valeurs des coefficients optiques. D'autant que dans le cas de l'imagerie Cerenkov, le maximum d'émission de photons se situe dans la zone la plus absorbante et n'est donc pas forcément majoritaire.

Bien que l'indice optique varie peu avec la longueur d'onde, il varie en fonction du tissu.

Sa valeur fluctue entre 1,33 et 1,4 dans les tissus biologiques. L'effet Cerenkov dépendant de l'indice optique, cette variation peut avoir une influence importante. Dans l'article [126], le rayonnement Cerenkov émis pour différents isotopes et pour différents indices à été simulé. Le changement d'indice de 1,33 à 1,4 fait varier le nombre de photons émis d'un facteur 2 pour du ^{18}F et d'un facteur 1,3 pour du ^{32}P. Le changement d'indice est donc un paramètre important pour la quantification du signal Cerenkov. Toutefois, la prise en compte exacte de ce phénomène implique de reconstruire le trajet de la particule. Dans le cas où le parcours moyen est important, des changements de milieux peuvent opérer après l'émission de la particule. De plus, les isotopes émettant le plus de rayonnement Cerenkov sont ceux émettant des rayonnements β très énergétiques et donc avec un parcours moyen élevé. Ce point est très dépendant de l'isotope utilisé.

5.4.3.4 Algorithme de reconstruction

Le problème de reconstruction vient du fait que le problème est mal posé. Le paramètre de régularisation ajoute une contrainte permettant de résoudre le problème. Toutefois, il introduit un biais dans les propriétés de la source à reconstruire. En particulier, minimiser la norme de la source signifie réduire l'étendue de la source. Cette régularisation tend à reconstruire des sources ponctuelles. Dans le cas *in vivo*, les sources n'ont pas de géométrie particulière. Les algorithmes utilisés en imagerie Cerenkov sont similaires à ceux utilisés en BLT. Ce sujet est encore un sujet de recherche d'actualité. Dans ce qui a été fait en tomographie Cerenkov, le groupe de Spinelli et al. [48] utilise un algorithme générique (moindres-carrées à valeurs positives) qui permet d'obtenir une reconstruction du système digestif car l'approche multispectrale permet de réduire la complexité du problème mal posé. Si le problème mal posé ne peut être amélioré, des algorithmes plus poussés [] permettent d'obtenir des reconstructions sur des sources ponctuelles ou très localisées (vessie, coeur, source implantée). Bien que l'algorithme utilisé a son importance, l'utilisation d'algorithmes poussés ne permet d'avoir une amélioration sur la précision sur les reconstructions aussi importante que lors de l'utilisation d'un modèle de propagation plus complet

Conclusions et perspectives

Dans le cadre de cette thèse, les protocoles et méthodes propres à l'élaboration de l'imagerie Cerenkov 3D ont été mis en place au sein de la plateforme d'imagerie AMISSA. Ce travail a consisté à mettre en place les protocoles expérimentaux propres à la manipulation de produits radioactifs et au travail sur petit animal *in vivo*. En plus de l'imagerie optique, l'information anatomique fournie par un TDM a été utilisé dans le processus de reconstruction. La méthode de reconstruction a été testée sur des données simulées par méthode Monte-Carlo avec succès. Les reconstructions *in vitro* donnent de bons résultats pour reconstruire un point source de ^{18}F. Mais une grande activité a du être utilisée pour aboutir à un résultat satisfaisant. Les reconstructions *in vivo* n'ont pu être réalisé qu'avec du ^{32}P qui fournit 18 fois plus de signal environ que du ^{18}F. Toutes les reconstructions se basent sur un modèle homogène ce qui n'est pas le cas dans les applications *in vivo*. De plus, d'autres algorithmes de reconstruction, plus complexes, existent pour diminuer l'impact du facteur de régularisation. Enfin, expérimentalement, d'autres isotopes sont susceptibles de fournir un meilleur signal Cerenkov. La dépendance de l'imagerie Cerenkov 3D d'une modalité d'imagerie annexe augmente l'erreur du à la manipulation.

Pour l'imagerie Cerenkov *in vivo*, les choix expérimentaux demandent de trouver un compromis entre l'énergie des rayonnements β et les problèmes liés aux hétérogénéités. Plus l'énergie des β est élevé, plus le nombre de photons Cerenkov émis est grand mais plus l'émission se fait loin de l'isotope. La spécificité du radiotraceur et l'étude animale associée influent sur la distribution du radiotraceur et donc sur la complexité de ce qui doit être reconstruit. L'isotope, le radiotraceur et l'activité utilisée vont sont des paramètres importants pour obtenir des images permettant de réaliser de la tomographie Cerenkov. Du côté de l'imageur optique, seuls les filtres et le temps d'acquisition demandent de trouver un compromis. Pour la tomographie Cerenkov, il est préférable d'utiliser des filtres le plus étroit possible mais cela dégrade la qualité des images. Pour le temps d'acquisition, il est limité soit par le temps de demi-vie de l'isotope, soit par la période biologique du radiotraceur. Bien qu'il soit préférable de faire une longue acquisition, l'imagerie Cerenkov impose des limites sur la durée de l'imagerie. L'acquisition de l'anatomie est un point important et divers modalités existent. Le choix de cette modalité ainsi que sa mise en correspondance avec l'imagerie optique est un point crucial. Ici, les modalités optique et CT ont été utilisées et sont 2 appareils distincts. La réalisation des images demande une intervention pour déplacer l'animal et le recalage des informations doit être réalisé pour chaque animal. L'idéal serait d'avoir un seul système regroupant les deux modalités. Le design d'un appareil tri-modalité a été communiqué comprenant les modalités TDM, PET

et optique [127]. Sur la partie reconstruction, l'idéal serait de mettre en place un seul modèle qui tient compte de l'émission des rayonnements β, de l'émission Cerenkov par ces particules β, de la propagation de ces photons dans l'animal, de la propagation de ces photons dans le système optique jusqu'au détecteur. En pratique, la diffusion importante des photons optiques par les tissus biologiques empêche de pouvoir prendre en compte les caractéristiques complètes de l'émission Cerenkov puisque la trajectoire est déjà, à cette étape de la reconstruction, impossible à connaître. Le modèle de propagation dans les tissus biologiques peut être changé mais l'impact sur les ressources informatiques et en temps de traitement peut être important. D'autant que la principale source d'erreur sur ce point ne vient pas du modèle mais de la sélection des coefficients optiques utilisés comme paramètres de ce modèle. Dans l'équation de transfert radiatif, la fonction de phase joue un rôle important. Toutefois, cette fonction n'est pas exactement connue et un modèle approché est utilisé. Cette fonction est compliquée à déterminer puisqu'elle varie en fonction des hétérogénéités.

Table des figures

Liste des tableaux

Bibliographie

[1] Ralph Weissleder and Umar Mahmood. Molecular imaging. *Radiology*, 219 :316–333, November 2001.

[2] Jürgen K. Willmann, Nicholas van Bruggen, Ludger M. Dinkelborg, and Sanjiv S. Gambhir. Molecular imaging in drug development. *Nature Reviews Drug Discovery*, 7 :591–607, July 2008.

[3] Jennifer S. Cho, Richard Taschereau, Sebastian Olma, Kan Liu, Yi-Chun Chen, Clifton K-F Shen, R. Michael van Dam, and Arion F. Chatziioannou. Cerenkov radiation imaging as a method for quantitative measurements of beta particles in a microfluidic chip. *Phys. Med. Biol.*, 54(22) :6757–6771, November 2009.

[4] Robbie Robertson, Melissa S. Germanos, C. Li, Gregory S. Mitchell, Simon R. Cherry, and Matthew D. Silva. Optical imaging of Cerenkov light generation from positron-emitting radiotracers. *Phys. Med. Biol.*, 54(16) :355–365, August 2009.

[5] Xiaowei Ma, Jing Wang, and Zhen Cheng. Cerenkov radiation : a multi-functional approach for biological sciences. *Frontiers in Physics*, January 2014.

[6] P. A. Cerenkov. Visible emission of clean liquids by action of γ radiation. *C. R.Dokl. Akad. Nauk. SSSR 2*, pages 451–454, 1934.

[7] J. V. Jelley. *Cerenkov radiation and its applications.* Pergamon, London, 1958.

[8] Pavel A. Cerenkov. Radiation of particles moving at a velocity exceeding that of light, and some of the possibilities for their use in experimental physics. *Nobel Lecture*, pages 426–440, December 1958.

[9] H. H. Ross. Measurement of β-emitting nuclides using cerenkov radiation. *Anal. Chem.*, 41(10) :1260–1265, August 1969.

[10] Hongguang Liu, Gang Ren, Zheng Miao, Xiaodong Tang Xiaofen Zhang and, Peizhen Han, Sanjiv S. Gambhir, and Zhen Cheng. Molecular optical imaging with radioactive probes. *Public Library of Science*, 5, March 2010.

[11] Alessandro Ruggiero, Jason P. Holland, Jason S. Lewis, and Jan Grimm. Cerenkov Luminescence Imaging of medical isotopes. *J. Nucl. Med.*, 51(7) :1123–1130, March 2010.

[12] Matthew A. Lewis, Vikram D. Kodibagkar, Orhan K. Oz, and Ralph P. Mason. On the potential for molecular imaging with Cerenkov luminescence. *Opt. Lett.*, 35(23) :3889–3891, December 2010.

[13] Gregory S. Mitchell, Ruby K. Gill, David L. Boucher, Changqing Li, and Simon R. Cherry. *In vivo* Cerenkov Luminescence Imaging : a new tool for molecular imaging. *Eur. J. Nucl. Med. Mol. Imaging*, 369 :4605–4619, 2011.

[14] Jeong Chan Park, Gwang Il An, Se-Il Park, Jungmin Oh, Hong Joo Kim, Yeong Su Ha, Eun Kyung Wang, Kyeong Min Kim, Jaetae Lee, Michael J. Welch, and Jeongsoo Yoo. Luminescence imaging using radionuclides : a potential application in molecular imaging. *Nucl. Med. Biol.*, 38 :321–329, September 2011.

[15] Yingding Xu, Hongguang Liu, and Zhen Cheng. Harnessing the power of radionuclides for optical imaging : Cerenkov Luminescence Imaging. *Journal of Nuclear Medicine*, 52 :2009–2018, November 2011.

[16] N. L. Ackerman and E. E. Graves. The potential for Cerenkov Luminescence Imaging of alpha-emitting radionuclides. *Phys. Med. Biol.*, 57 :771–783, January 2012.

[17] Bradley J. Beattie, Daniel L. J. Thorek, Charles R. Schmidtlein, Keith S. Pentlow, John L. Humm, and Andreas H. Hielscher. Quantitative modeling of Cerenkov light production efficiency from medical radionuclides. *Public Library of Science*, 7, February 2012.

[18] Vasilis Ntziachristos, Jorge Ripoll, Lihong V Wang, and Ralph Weissleder. Looking and listening to light : the evolution of whole-body photonic imaging. *Nature Biotechnology*, 23(3) :313–320, March 2005.

[19] S.R. Arridge. Optical tomography in medical imaging. *Inverse Problems*, 15 :R41–R93, November 1999.

[20] A. P. Gibson, J. C. Hebden, and S. R. Arridge. Recent advances in diffuse optical imaging. *Phys. Med. Biol.*, 50(4) :R1–R43, 2005.

[21] M. Takeda, H. Ina, , and S. Kobayashi. Fourier-transform method of fringe-pattern analysis for computer-based topography and interferometry. *J. Acoust. Soc. Am.*, 72 :156–160, 1982.

[22] D. C. Ghiglia and M. D. Pritt. *Two-dimensional phase unwrapping, theory, algorithms and software*. John Wiley and Sons, New York, 1998.

[23] Heiko Meyer, Anikitos Garofalakis, Giannis Zacharakis, Stylianos Psycharakis, Clio Mamalaki, Dimitris Kioussis, Eleftherios N. Economou, Vasilis Ntziachristos, and Jorge Ripoll. Noncontact optical imaging in mice with full angular coverage and automatic surface extraction. *Appl. Opt.*, 46(17) :3617–3627, 2007.

[24] A. Lauchli. Radioassay for beta-emitters in biological materials using cerenkov radiation. *Int. J. Appl. Radiat. Isot.*, 20(4) :265–270, 1969.

[25] W. M. Burch. Cerenkov light from 32P as an aid to diagnosis of eye tumours. *Nature*, 234(5328) :358, 1971.

[26] Giovanni Lucignani. Cerenkov radioactive optical imaging : a promising new strategy. *European Journal of Nuclear Medicine and Molecular Imaging*, 38 :592–595, December 2011.

[27] Daniel LJ. Thorek, Robbie Robertson, Wassifa A. Bacchus, Jaeseung Hahn, Julie Rothberg, Bradley J. Beattie, and Jim Grimm. Cerenkov imaging - a new modality for molecular imaging. *American Journal of Nuclear Medicine and Molecular Imaging*, 2(2) :163–173, April 2012.

[28] Patrick T. K. Chin, Mick M. Welling, Stefan C. J. Meskers, Renato A. Valdes Olmos, Hans Tanke, and Fijs W. B. van Leeuwen. Optical imaging as an expansion of nuclear medicine : Cerenkov-based luminescence vs fluorescence-based luminescence. *Eur. J. Nucl. Med. Mol. Imaging*, March 2013.

[29] Sudeep Das, Daniel L. J. Thorek, and Jan Grimm. Cerenkov imaging. *Advances in Cancer Research*, 124 :213–234, 2014.

[30] Changqing Li, Gregory S. Mitchell, and Simon R. Cherry. Cerenkov Luminescence Tomography for small animal imaging. *Opt. Lett.*, 35(7) :1109–1111, April 2010.

[31] Robin S. Dothager, Reece J. Goiffon, Erin Jackson, Scott Harpstrite, and David Piwnica-Worms. Cerenkov Radiation Energy Transfer (CRET) imaging : a novel method for optical imaging of PET isotopes in biological systems. *Public Library of Science*, 5, October 2010.

[32] Hongguang Liu, Gang Ren, Shuanglong Liu, Xiaofen Zhang, Luxi Chen, Peizhen Han, and Zhen Cheng. Optical imaging of reporter gene expression using a Positron-Emission-Tomography probe. *J. Biomed. Opt.*, 15(6) :060505, October 2010.

[33] Antonello E. Spinelli, Daniela D'Ambrosio, Laura Calderan, Mario Marengo, Andrea Sbarbati, and Frederico Boschi. Cerenkov radiation allows *in vivo* optical imaging of positron emitting radiotracers. *Phys. Med. Biol.*, 55 :483–495, December 2010.

[34] A. E. Spinelli, D. D'Ambrosio, L. Calderan, M. Marengo, A. Sbarbati, and F. Boschi. Reply to 'comments on "cerenkov radiation allows *in vivo* optical imaging of positron emitting radiotracers"'. *Phys. Med. Biol.*, 55(8) :L45–L49, 2010.

[35] Frederico Boschi, Laura Calderan, Daniela D'Ambrosio, Mario Marengo, Alberto Fenzi, Riccardo Calandrino, Andrea Sbarbati, and Antonello E. Spinelli. *In vivo* 18F-FDG tumour uptake measurements in small animals using Cerenkov radiation. *Eur. J. Nucl. Med. Mol. Imaging*, 38 :120–127, September 2011.

[36] Robbie Robertson, Melissa S. Germanos, Mark G. Manfredi, Peter G. Smith, and Matthew D. Silva. Multimodal imaging with 18F-FDG PET and Cerenkov Luminescence Imaging after MLN4924 treatment in a human lymphoma xenograft model. *J. Nucl. Med.*, 52(11) :1764–1769, November 2011.

[37] Antonello E. Spinelli and Frederico Boschi. Unsupervised analysis of small animal dynamic Cerenkov Luminescence Imaging. *J. Biomed. Opt.*, 16(12), December 2011.

[38] Antonello E. Spinelli, Frederico Boschi, Daniela D'Ambrosio, Laura Calderan, Mario Marengo, Alberto Fenzi, Marta Menegazzi, Andrea Sbarbati, Antonella Del Vecchio, and Riccardo Calandrino. Cherenkov radiation imaging of beta emitters : *in vitro* and *in vivo* results. *Nuclear Instruments and Methods in Physics Research A*, 648 :310–312, November 2011.

[39] Xueli Zhang, Chaincy Kuo, Anna Moore, and Chongzhao Ran. *In vivo* optical imaging of interscapular brown adipose tissue with 18F-FDG via Cerenkov Luminescence Imaging. *Public Library of Science*, 8, April 2013.

[40] Di Fan, Xin Zhang, Lijun Zhong, Xujie Liu, Yi Sun, Huiyun Zha, Bing Jia, Zhaofei Liu, Zhaohui Zhu, Jiyun Shi, and Fan Wang. 68Ga-labeled 3PRGD2 for dual PET and Cerenkov Luminescence Imaging of orthotopic human glioblastoma. *Bioconjug. Chem.*, 2015.

[41] Yingding Xu, Edwin Chang, Hongguang Liu, Han Jiang, Sanjiv Sam Gambhir, , and Zhen Cheng. Proof-of-concept study of monitoring cancer drug therapy with Cerenkov Luminescence Imaging. *J. Nucl. Med.*, 53(2) :312–317, February 2012.

[42] Dong Hyun Kim, Yearn Seong Choe, Joon Young Choi, Kyung-Han Lee, and Byung-Tae Kim. Binding of 2-[18f]fluoro-cp-118,954 to mouse acetylcholinesterase : micropet and *ex vivo* cerenkov luminescence imaging studies. *Nuclear Medicine and Biology*, 38 :541–547, November 2011.

[43] Jeffrey D Steinberg, Anandhkumar Raju, Prashant Chandrasekharan, Chang-Tong Yang, Karen Khoo, Jean-Pierre Abastado, Edward G Robins, and David W Townsend. Negative contrast Cerenkov Luminescence Imaging of blood vessels in a tumor mouse model using [68Ga]gallium chloride. *Eur. J. Nucl. Med. Mol. Imaging*, 4(15), 2014.

[44] Jiyun Shi, Di Fan, Liu Xujie, Li Yang, Ning Yan, Yi Sun, Huiyun Zhao, Bing Jia, Zhaohui Zhu, and Fan Wang. Dual PET and Cerenkov Luminescence Imaging of a kit-formulated integrin $\alpha v\beta 3$-selective radiotracer 68Ga-3PRGD2 in human glioblastoma xenografts and orthotopic tumors. *The Journal of Nuclear Medecine*, 54, 2013.

[45] Shin Young Jeong, Mi-Hye Hwang, Jung Eun Kim, Sungmin Kang, Jeong Chan Park, Jeongsoo Yoo, Jeoung-Hee Ha, Sang-Wo Lee, Byeong-Cheol Ahn, and Jaetae Lee. Combined Cerenkov Luminescence and Nuclear Imaging of radioiodine in the thyroid gland and thyroid cancer cells expressing sodium iodide symporter : initial feasibility study. *Endocr. J.*, 7 :575–583, 2011.

[46] Jason P. Holland, Guillaume Normand, Alessandro Ruggiero, Jason S. Lewis, and Jan Grimm. Intraoperative imaging of Positron Emission Tomographic radiotracers using Cerenkov luminescence emissions. *Mol. Imaging*, 3 :177–186, June 2011.

[47] Arutselvan Natarajan, Frezghi Habte, Hongguang Liu, Ataya Sathirachinda, Xiang Hu, Zhen Cheng, Claude M. Nagamine, and Sanjiv Sam Gambhir. Evaluation of 89Zr-rituximab tracer by Cerenkov Luminescence Imaging and correlation with PET in a humanized transgenic mouse model to image NHL. *Mol. Imaging Biol.*, 2013.

[48] Antonello E. Spinelli, Chaincy Kuo, Brad W. Rice, Riccardo Calandrino, Pasquina Marzola, Andrea Sbartbati, and Frederico Boschi. Multispectral Cerenkov Luminescence Tomography for small animal optical imaging. *Opt. Express*, 19(13) :12605–12618, June 2011.

[49] Zhenhua Hu, Xiaowei Ma, Xiaochao Qu, Jimin Liang, Jing Wang, and Jie Tian. Three-dimensional non invasive monitoring iodine-131 uptake in the thyroid using a

modified Cerenkov Luminescence Tomography approach. *Public Library of Science*, 7, May 2012.

[50] Yucai Wang, Yongjian Liu, Hannah Luehmann, Xiaohu Xia, Dehui Wan, Cathy Cutler, and Younan Xia. Radioluminescent gold nanocages with controlled radioactivity for real-time *in vivo* imaging. *Nano Lett.*, 13 :581–585, January 2013.

[51] Y. Xu, H. Liu, E. Chang, H. Jiang, and Z. Cheng. Cerenkov Luminescence Imaging (CLI) for cancer therapy monitoring. *Journal of Visualized Experiments*, (69) :e4341, 2012.

[52] Yann Bernhard, Bertrand Collin, and Richard A. Decreau. Inter/intramolecular Cherenkov Radiation Energy Transfer (CRET) from a fluorophore with a built-in radionuclide. *Chem. Commun.*, 50 :6711–6713, 2014.

[53] A. E. Spinelli, M. Ferdeghini, C. Cavedon, E. Zivelonghi, R. Calandrino, A. Fenzi, A. Sbarbati, and F. Boschi. First human Cerenkography, 2013.

[54] Daniel L.J. Thorek, Christopher C. Riedl, and Jan Grimm. Clinical Cerenkov Luminescence Imaging of 18F-FDG. *Journal of Nuclear Medicine*, 55 :95–98, 2014.

[55] Sri-Rajasekhar Kothapalli, Hongguang Liu, joseph C. Liao, Zhen Cheng, and Sanjiv Sam Gambhir. Endoscopic imaging of Cerenkov luminescence. *Biomedical Optics Express*, 3(6) :1215–1225, June 2012.

[56] Hongguang Liu, Colin M. Carpenter, Han Jiang, Guillem Pratx, Conroy Sun, Michael P. Buchin, Sanjiv Sam Gambhir, Lei Xing, and Zhen Cheng. Intraoperative imaging of tumors using Cerenkov luminescence endoscopy : a feasibility experimental study. *Journal of Nuclear Medicine*, 53(10) :1579–1584, October 2012.

[57] Hao Hu, Xin Cao, Fei Kang, Min Wang, Yenan Lin, Muhan Liu, Shujun Li, Liping Yao, Jie Liang, Jimin Liang, Yongzhan Nie, Xueli Chen, Jing Wang, and Kaichun Wu. Feasibility study of novel endoscopic Cerenkov Luminescence Imaging system in detecting and quantifying gastrointestinal disease : first human results. *Eur. Radiol.*, 2014.

[58] Hongguang Liu, Xiaofen Zhang, Bengang Xing, Peizhen Han, Sanjiv Sam Gambhir, and Zhen Cheng. Radiation-luminescence-excited Quantum Dots for *in vivo* multiplexed optical imaging. *Small*, 6(10) :1087–1091, 2010.

[59] Colin M.Carpenter, Conroy Sun, Guillem Pratx, Hongguang Liu, Zhen Cheng, and Lei Xing. Radioluminescent nanophosphors enable multiplexed small-animal imaging. *Opt. Express*, 20(11) :11598–11604, May 2012.

[60] Daniel LJ. Thorek, Anuja Ogirala, Bradley J. Beattie, and Jan Grimm. Quantitative imaging of disease signatures through radioactive decay signal conversion. *Nat. Med.*, 19(10) :1345–1350, October 2013.

[61] Rongxiao Zhang, Alisha V. D'souza, Jason R. Gunn, Tatiana V. Esipova, Sergei A. Vinogradov, Adam K. Glaser, Lesley A. Jarvis, David J. Gladstone, and Brian W. Pogue. Cherenkov-Excited Luminescence Scanned Imaging. *Opt. Lett.*, 40(5) :827–830, March 2015.

[62] Brian W. Pogue, Rongxiao Zhang, Jason R. Gunn, David J. Gladstone, Lesley A. Jarvis, and Sergei Vinogradov. High resolution molecular imaging with Cerenkov-Excited Luminscence Scanned Imaging (CELSI). *Optics in the Life Sciences*, 2015.

[63] Zhenhua Hu, Jimin Liang, Weidong Yang, Weiwei Fan, Congye Li, Xiaowei Ma, Xueli Chen, Xiaopeng Ma, Xiangsi Li, Xiaochao Qu, Jing Wang, Feng Cao, and Jie Tian. Experimental Cerenkov Luminescence Tomography of the mouse model with SPECT imaging validation. *Opt. Express*, 18(24) :24441–24450, November 2010.

[64] Jianghong Zhong, Chenghu Qin, Xin Yang, Zhe Chen, Xiang Yang, and Jie Tian. Fast-specific tomography imaging via cerenkov emission. *Mol. Imaging Biol.*, July 2011.

[65] Jianghong Zhong, Jie Tian, Xin Yang, and Chenghu Qin. Whole-body Cerenkov Luminescence Tomography with the finite element SP_3 method. *Ann. Biomed. Eng.*, 39(6) :1728–1735, June 2011.

[66] Zhenhua Hu, Weidong Yang, Xiaowei Ma, Wenhui Ma, Xiaochao Qu, Jimin Liang, Jing Wang, and Jie Tian. Cerenkov luminescence tomography of aminopeptidase n (apn/cd13) expression in mice bearing ht1080 tumors. *Mol. Imaging*, 0(0) :1–9, 2012.

[67] Chenghu Qin, Jianghong Zhong, Zhenhua Hu, Xin Yang, and Jie Tian. Recent advances in Cerenkov Luminescence and Tomography Imaging. *Institute of Electrical and Electronics Engineers*, 18(3) :1084–1093, May 2012.

[68] R. Han, J. Liang, X. Qu, Y. Hou, N. Ren, and J. Tian. A source reconstruction algrithm based on adaptative hp-FEM for BioLuminescenceuminescence Tomography. *Optics Express*, 17(17) :14481–14494, 2009.

[69] A. Ishimaru. *Wave Propagation and Scattering in Random Media.* 1978.

[70] Wai-Fung Cheong, Scott A. Prahl, and ashley J. Welch. A review of the optical properties of biological tissues. *IEEE Journal of Quantum Electronics*, 26 :2166–2185, 1990.

[71] Alexey N. Bashkatov, Elina A. Genina, and Valery V. Tuchin. Optical properties of skin, subcutaneous, and muscle tissues : a review. *Journal of Innovative Optical Health Sciences*, 4(1) :9–38, 2011.

[72] Steven L. Jacques. Optical properties of biological tissues : a review. *Phys. Med. Biol.*, 58 :R37–R61, 2013.

[73] George Alexandrakis, Fernando R. Rannou, and Arion F. Chatziioannou. Tomographic bioluminescence imaging by use of a combined optical-pet (opet) system : a computer simulation feasibility study. *Phys. Med. Biol.*, 50(17) :4225–4241, September 2005.

[74] A. Kienle, F. K. Forster, and R. Hibst. Influence of the phase function on determination of the optical properties of biological tissue by spatially resolved reflectance. *Optics Letters*, 26(20) :1571–1573, 2001.

[75] S.K. Sharma and S. Banerjee. Role of approximate phase functions in Monte Carlo simulation of light propagation in tissues. *J. Opt.A 5*, pages 294–302, 2003.

[76] Alexander D. Klose and Edward W. Larsen. Light transport in biological tissue based on the simplified spherical harmonics equations. *J. Comput. Phys.*, 220 :441–470, September 2006.

[77] Hang Si Klaus Gärtner. Meshing piecewise linear complexes by constrained Delaunay tetrahedralizations. *International Meshing Roundtable*, pages 147–163, 2005.

[78] Manoj Aggarwal and Narendra Ahuja. A Pupil-Centric Model of Image Formation. *International Journal of Computer Vision*, 48(3) :195–214, November 2002.

[79] Xueli Chen, Xinbo Gao, Xiaochao Qu, Jimin Liang, Lin Wang, Da'an Yang, Anikitos Garofalakis, Jorge Ripoll, and Jie Tian. A study of photon propagation in free-space based on hybrid radiosity-radiance theorem. *Opt. Express*, 17(18) :16266–16280, August 2009.

[80] Xueli Chen, Xinbo Gao, Duofang Chen, Xiaopeng Ma, Xiaohui Zhao, Man Shen, Xiangsi Li, Xiaochao Qu, Jimin Liang, Jorge Ripoll, and Jie Tian. 3D reconstruction of light flux distribution on arbitrary surfaces from 2D multi-photographic images. *Opt. Express*, 18(19) :19876–19893, September 2010.

[81] A Molecular Optical Simulation Environment MOSE. http ://www.mosetm.net/introduction.htm.

[82] H. Li, J. Tian, F. Zhu, W. Cong, L. Wang, E. Hoffman, and G. Wang. A Mouse Optical Simulation Environment (MOSE) to investigate bioluminescent phenomena in the living mouse with the Monte Carlo method. *Acad. Radiol.*, 11(9) :1029–1038, 2004.

[83] Kuan Peng, Xinbo Gao, Jimin Liang, Xiaochao Qu, Nunu Ren, Xueli Chen, Bin Ma, and Jie Tian. Study on photon transport problem based on the platform of Molecular Optical Simulation Environment. *International Journal of Biomedical Imaging*, 2010 :1–9, 2009.

[84] Richard C. Haskell, Lars O. Svaasand, Tsong-Tseh Tsay, Ti-Chen Feng, Matthew S. McAdams, and Bruce J. Tromberg. Boundary conditions for the diffusion equation in radiative transfer. *Journal of Optical Society of America*, 11(10) :2727–2741, October 1994.

[85] M. Schweiger, S.R. Arridge, M. Hiraoka, and D.T. Delpy. The Finite Element Method for the propagation of light in scattering media : boundary and source conditions. *Med. Phys.*, 22(11) :1779–1792, November 1995.

[86] Margaret J. Eppstein, David E. Dougherty, Tamara L. Troy, and Eva M. Sevick-Muraca. Biomedical optical tomography using dynamic parameterization and Bayesian conditioning on photon migration measurements. *Applied Optics*, 38(10) :2138–2150, April 1999.

[87] B. W. Rice, M. D. Cable, and M. B. Nelson. *In vivo* imaging of light-emitting probes. *Journal of Biomedical Optics*, 6(4) :432–440, October 2001.

[88] Wenxiang Cong, Lihong V. Wang, and Ge Wang. Formulation of photon diffusion from spherical bioluminescent sources in an infinite homogeneous medium. *Biomed. Eng.*, 3, May 2004.

[89] Arnold D. Kim. Transport theory for light propagation in biological tissue. *Journal of Optical Society of America*, 21(5) :820–827, May 2004.

[90] Vadim Y. Soloviev. Tomographic bioluminescence imaging with varying boundary conditions. *Applied Optics*, 46(14) :2778–2784, May 2007.

[91] Michael Chu, Karthik Vishwanath, Alexander D. Klose, and Hamid Deghani. Light transport in biological tissue using three-dimensional frequency-domain simplified spherical harmonics equations. *Phys. Med. Biol.*, 54 :2493–2509, April 2009.

[92] Jinchao Feng, Kebin Jia, Chenghu Qin, Guorui Yan, Shouping Zhu, Xing Zhang, Junting Liu, and Jie Tian. Three-dimensional bioluminescence tomography based on bayesian approach. *Opt. Express*, 17(19), September 2009.

[93] W. Cong and G. Wang. Bioluminescence tomography based on the phase approximation model. *Journal of Optical Society of America*, 27(2) :174–179, February 2010.

[94] K.M. Case and P.F. Zweifel. *Linear Transport Theory.* Addison-Wesley, 1967.

[95] J. E. Morel, J. M. McGhee, and E. W. Larsen. A three-dimensional time-dependent unstructured tetrahedral-mesh SP_n method. *Nucl. Sci. Eng.*, 123 :319–327, 1996.

[96] M.L. Adams and E.W. Larsen. Fast iterative methods for discrete-ordinates particle transport calculations. *Prog. Nucl. Energy*, 40(1) :3–159, 2002.

[97] J.K. Fletcher. The solution of the multigroup neutron transport equation using spherical harmonics. *Nucl. Sci. Eng.*, 84 :33–46, 1983.

[98] K. Kobayashi, H. Oigawa, and H. Yamagata. The spherical harmonics method for the multigroup transport equations in x-y geometry. *Ann. Nucl. Energy*, 13(12) :663–678, 1986.

[99] C. E. Siewert. A spherical-harmonics method for multi-group or non-gray radiation transport. *Journal of Quantitative Spectroscopy and Radiative Transfer*, 49(2) :95–106, April 1993.

[100] E.M. Gelbard. *Application of spherical harmonics methods to reactor problems.* WAPD-BT-20, Bettis Atomic Power Laboratory, 1960.

[101] E.M. Gelbard, J. Davis, and J. Pearson. Iterative solutions to the P_1 and double P_1 equations. *Nucl. Sci. Eng.*, 5 :36–44, 1959.

[102] D. I. Tomasevic and E. W. Larsen. The simplified P_2 approximation. *Nucl. Sci. Eng.*, 122 :309–325, 1996.

[103] E. W. Larsen, J. E. Morel, and J. M. McGhee. Asymptotic derivation of the multigroup P_1 and simplified P_n equations with anisotropic scattering. *Nucl. Sci. Eng.*, 123 :328, 1996.

[104] R. T. Ackroyd, C. R. E. de Oliveira, A. Zolfaghari, and A. J. H. Goddard. On a rigorous resolution of the transport equation into a system of diffusion-like equations. *Prog. Nucl. Eng.*, 35(1) :1–64, 1999.

[105] P. S. Brantley and E. W. Larsen. The simplified P3 approximation. *Nucl. Sci. Eng.*, 134 :1–21, 2000.

[106] Ge Wang, Wenxiang Cong, Kumar Durairaj, Xin Qian, Haiou Shen, Patrick Sinn, Eric A. Hoffman, Geoffrey McLennan, and Michael Henry. *In vivo* mouse studies with BioLuminescence Tomography. *Opt. Express*, 14(17) :7801–7809, August 2006.

[107] Huabei Jiang. Optical image reconstruction based on the third-order diffusion equations. *Optics Express*, 4(8) :241–246, April 1999.

[108] Wenxiang Cong, Ge Wang, Durairaj Kumar, Yi Liu, Ming Jiang, Lihong V. Wang, Eric A. Hoffman, Geoffrey McLennan, Paul B. McCray, Joseph Zabner, and Alexander Cong. Practical reconstruction method for bioluminescence tomography. *Opt. Express*, 13(18) :6756–6771, September 2005.

[109] Alexander D. Klose, Vasilis Ntziachristos, and Andreas H. Hielscher. The inverse source problem based on the radiative transfer equation in optical molecular imaging. *J. Comput. Phys.*, 202 :323–345, September 2005.

[110] Alexander D. Klose, Bradley J. Beattie, Hamid Deghani, Lena Vider, Carl Le, Vladimir Ponomarev, and Ronald Blasberg. *In vivo* BioLuminescence Tomography with a blocking-off finite-difference SP_3 method and MRI/CT coregistration. *Med. Phys.*, 37(1) :329–338, January 2009.

[111] Yujie Lu, Ali Douraghy, Hidevaldo B. Machado, David Stout, Jie Tian, Harvey Herschman, and Arion F. Chatziioannou. Spectrally resolved bioluminescence tomography with the third-order simplified spherical harmonics approximation. *Phys. Med. Biol.*, 54 :6477–6493, 2009.

[112] Zhen Yuan, Xin-Hua Hu, and Huabei Jiang. A higher order diffusion model for three-dimensional photon migration and image reconstruction in optical tomography. *Phys. Med. Biol.*, 54 :65–88, December 2009.

[113] Kai Liu, Yujie Lv, Jie Tian, Chenghu Qin, Xin Yang, Shouping Zhu, Xiang Yang, Quansheng Gao, and Dong Han. Evaluation of the simplified spherical harmonics approximation in bioluminescence tomography through heterogeneous mouse models. *Opt. Express*, 18(20) :20988–21002, September 2010.

[114] Wei Li, HuangJian Yi, Qitan Zhang, Duofang Chen, and Jiming Liang. Extended Finite Element Method with simplified spherical harmonics approximation for the forward model of optical molecular imaging. *Computational and Mathematical Methods in Medecine*, 2012.

[115] S. C. Brenner and L. C. Scott. *The mathematical theory of Finite Element Methods.* Springer-Verlag, New York, 1994.

[116] Yujie Lv, Jie Tian, Wenxiang Cong, Ge Wang, Jie Luo, Wei Yang, and Hui Li. A multilevel adaptive finite element algorithm for bioluminescence tomography. *Opt. Express*, 14(18) :8211–8223, September 2006.

[117] Nikolai V. Slavine, Matthew A. Lewis, Edmond Richer, and Peter P. Antich. Iterative reconstruction method for light emitting sources based on the diffusion equation. *Medical Physics*, 33(1) :61–68, January 2006.

[118] Yujie Lu, Hidevaldo B. Machado, Ali Douraghy, David Stout, Harvey Herschman, and Arion F. Chatziioannou. Experimental bioluminescence tomography with fully paral-

lel radiative-transfer-based reconstruction framework. *Opt. Express*, 17(19) :16681–16695, September 2009.

[119] Xiaowei He, Jimin Liang, Xiaorui Wang, Jingjing Yu, Xiaochao Qu, Xiaodong Wang, Yanbin Hou, Duofang Chen, Fang Liu, and Jie Tian. Sparse reconstruction for quantitative bioluminescence tomography based on the incomplete variables truncated conjugate gradient method. *Opt. Express*, 18(24) :24825–24841, November 2010.

[120] Kai Liu, Jie Tian, Yujie Lv, Xin Yang, Shouping Zhu, and Xing Zhang. A fast bioluminescent source localization method based on generalized graph cuts with mouse model validations. *Opt. Express*, 18(4) :3732–3745, February 2010.

[121] Chaincy Kuo, Olivier Coquoz, Tamara L. Troy, Heng Xu, and Brad W. Rice. Three-dimensional reconstruction of *in vivo* bioluminescent sources based on multispectral imaging. *J. Biomed. Opt.*, 12(2), April 2007.

[122] Yujie Lu, Jie Tian, Wenxiang Cong, Ge Wang, Wei Yang, Chenghu Qin, and Min Xu. Spectrally resolved bioluminescence tomography with adaptive finite element analysis : methodology and simulation. *Phys. Med. Biol.*, 52 :4497–4512, July 2007.

[123] Olivier Coquoz, Tamara L. Troy, Dragana Jekic-McMullen, and Bradley W. Rice. Determination of depth of *in vivo* bioluminescent signals using spectral imaging techniques. *SPIE*, 2003.

[124] Hamid Dehghani, Scott C. Davis, Shudong Jiang, Brian W. Pogue, and Keith D. Paulsen. Spectrally resolved bioluminescence optical tomography. *Optics Letters*, 31(3) :365–367, February 2006.

[125] Yujie Lu, Xiaoqun Zhang, Ali Douraghy, David Stout, Jie Tian, Tony F. Chan, and Arion F. Chatziioannou. Source reconstruction for spectrally-resolved bioluminescence tomography with sparse a priori information. *Opt. Express*, 17(10) :8062–8080, May 2009.

[126] Ruby K. Gill, Gregory S. Mitchell, and Simon R. Cherry. Computed cerenkov luminescence yields for radionuclides used in biology and medicine. *Phys. Med. Biol.*, 60 :4263–4280, 2015.

[127] Di Dong, Hui Hui, Caiyun Yang, Jin Guo, Yujie Yanga, Liangliang Shi, Wei Mu, and Jie Tian. Preliminary design of a multimodality molecular imaging system. *IEEE*, 2014.

Annexe A

Annexe : implémentation

A.1 Maillage

A.1.1 Script général pour le maillage

Listing A.1 – `maillage_export_msh_mesh.m`

```
close all;
%clear all;
%clc;

%% file data

Path = '/tmp/';
file_name = '20150408_sub23_lot172_rotcrop_pyr'; % fichiers .
   ima et .dim dans "Path"
seuil = [0.01,1.4]; % Segmentation du volume TDM
file_MOSE = '/tmp/20150408_172_23_coarse'; % noms des fichiers
   créés (.msh)

%% lecture phantom voxelisé

[phantom, volsize, voxel_size] = read_ct_volume(Path, file_name
   , seuil);

analize_phantom(phantom)

%% suppression points isolés

im_tmp = phantom;
im_tmp(im_tmp == 2) = 1;
```

```
clear a;
size_ = 4;
strel = ones(size_,size_,size_);
im_tmp = imdilate(imerode(im_tmp,strel),strel);
im_tmp(phantom == 2) = 2;
phantom = im_tmp;
clear im_tmp;
analize_phantom(phantom)

% comblage trous

clear a;

phantom = fillholes3d_hetero(phantom,3);
analize_phantom(phantom)

%% crop phantom

phantom(1:85,:,295:600) = 0;
% phantom(172:266,:,:) = 0;
% phantom(:,173:178,:) = 0;
% phantom(:,:,420:600) = 0;
analize_phantom(phantom)

%% maillage du phantom

clear option;

option.homogene = 0;

option.radbound = 25;
option.angbound = 15;
option.distbound = 25;
%option.maxnode = 500;
option.reratio = 2;
option.volmax = 12

[node, face, elem] = mesh_phantom(phantom, volsize, voxel_size,
    option);

%% correction des valeurs des points en fonction du milieu (
   hétérogène)

for i=min(elem(:,5)):max(elem(:,5))
```

```
    pts = unique(elem(elem(:,5) == i,1:4));
    node(pts,4) = i;
end

print_results_msh(file_MOSE, node, face, elem, node(:,4), -2);
   % Enregistre le maillage segmenté en .msh
disp([file_MOSE '.msh']);
```

A.1.2 Fonction maillage d'un volume CT segmenté ou homogène

Listing A.2 – mesh_phantom.m

```
function [node, face, elem] = mesh_phantom(phantom, volsize,
   voxel_size, varargin)

% varargin : structure avec paramètres du maillage
%     les différents champs sont visibles plus bas

correction = 1;

%% maillage du phantom

clear opt;

disp('--------------------------');
% Valeurs par défaut
opt.radbound = 45;
opt.angbound = 7;
opt.distbound = 2;
opt(1).maxnode = 700;
opt.reratio = 2;
opt.volmax_elem = 9;
homogene = 1;


if nargin == 4
    if isfield(varargin{1},'radbound')
        opt.radbound = varargin{1}.radbound;
    end
    if isfield(varargin{1},'angbound')
        opt.angbound = varargin{1}.angbound;
    end
    if isfield(varargin{1},'distbound')
        opt.distbound = varargin{1}.distbound;
    end
```

```
    if isfield(varargin{1},'maxnode')
        opt.maxnode = varargin{1}.maxnode;
    end
    if isfield(varargin{1},'reratio')
        opt.reratio = varargin{1}.reratio;
    end
    if isfield(varargin{1},'volmax')
        opt.volmax_elem = varargin{1}.volmax;
    end
    if isfield(varargin{1},'homogene')
        homogene = varargin{1}.homogene;
    end
elseif nargin > 4
    disp('Too much input argument, function aborted !');
    return;
end

opt

%% Maillage à partir des fonctions de iso2mesh (http://iso2mesh
   .sourceforge.net/cgi-bin/index.cgi)
if homogene
    [node,elem,face]=vol2mesh(phantom, 1:volsize(1), 1:volsize
       (2), 1:volsize(3), opt, opt.volmax_elem, 1, 'cgalpoly');
else
    [node,elem,face]=vol2mesh(uint8(phantom), 1:volsize(1), 1:
       volsize(2), 1:volsize(3), opt, opt.volmax_elem, 1, '
       cgalmesh', [0:max(max(max(phantom)))]);
end

nb_points = size(node,1);

face = unique(face,'rows');
elem = unique(elem,'rows');
nb_tetra = size(elem,1);
nb_faces = size(face,1);

%% comptage points inclus dans face
surface = unique(face(:,1:3));
pts_surf = size(surface,1)
nb_points/pts_surf

clear pts_surf;
```

```
%% correction maillage homogène

count=0;
if homogene
    for i=1:nb_points
        if node(i,4) ~= 1
            node(i,4) = 1;
            count = count+1;
        end
    end
    count

% correction maillage hétérogène
else
    for i=1:nb_points
        if node(i,4) < 1
            node(i,4) = 1;
            count = count+1;
        end
    end
    count
end

%% node

node(:,1) = voxel_size(1)*node(:,1);
node(:,2) = voxel_size(2)*node(:,2);
node(:,3) = voxel_size(3)*node(:,3);

%% correction des points supplémentaires avec renumérotation
disp('correction␣maillage');
if correction
    node_bis = unique(elem(:,1:4));

    decalage = nb_points;
    pts_to_delete = [];

    for i=1:nb_points
        if isempty(find((node_bis == i),1))
            pts_to_delete = [pts_to_delete i];
        end
    end

    clear row col i j;
```

```
pts_to_delete
node(pts_to_delete,:)

node = node(node_bis,:);
nb_points = size(node,1);

disp('nombre de points réels :');
nb_points

%% delete points

size_delete = size(pts_to_delete,2);

error_count = 0;
for i=1:size_delete
    seuil_delete = pts_to_delete(i)-(i-1);

    [a,b] = find(face == seuil_delete);
    a = setdiff([1:nb_faces],a);
    face = face(a,:);
    nb_faces = size(face,1);

    for face_ind=1:nb_faces
        for j=1:3
            if face(face_ind,j) == seuil_delete
                error_count = error_count+1;
            elseif face(face_ind,j) >= seuil_delete
                face(face_ind,j) = face(face_ind,j) - 1;
            end
        end
    end


    [a,b] = find(elem == seuil_delete);
    a = setdiff([1:nb_tetra],a);
    elem = elem(a,:);
    nb_tetra = size(elem,1);

    for elem_ind=1:nb_tetra
        for j=1:4
            if elem(elem_ind,j) == seuil_delete
                error_count = error_count+1;
            elseif elem(elem_ind,j) >= seuil_delete
```

```
                    elem(elem_ind,j) = elem(elem_ind,j) - 1;
                end
            end
        end
    end
    error_count

    clear face_ind elem_ind i j seuil_delete;

end
```

A.2 Traitement des images Cerenkov 2D

A.2.1 Calibration 4 vues

A.2.1.1 Script de calibration du module 4 vues et du système optique

Listing A.3 – `calibration.m`

```
%PI_cal

close all;
%clear all;

% Informations sur l'image de calibration

Path_image = '.\Data\Manip_cerenkov_20150408\Souris\172_23\'; %
   calibration\';
image_file = '172-23_acqui_1-P32_20min_1_0s_0nm_HD_'; %
   calibration_biolum_HD_';plaque_rec_fusion
pho = 'fusion.png';

Full_path_of_file = [Path_image image_file pho];

% Ouverture images

image_pho = imread(Full_path_of_file);
info_image = imfinfo(Full_path_of_file);

size_im = [info_image.Width,info_image.Height]
image(image_pho);

center_im = [588,453];

%% calibrage centre en x
```

```
disp('centre du 4 vues, en x');
xc = impoint();
pos_D = wait(xc);
xc = pos_D(1);

x_decalage = center_im(1) - xc;
clear pos_D xc;
nb_points = 3;

%% detector parameters

angle = [-51.2,51]; % en degré, données constructeur des
   miroirs latéraux
% Longueurs en mm
niveau_support = 80;
size_pixel = 14.4/1000;
hauteur = 440;
v_prime = 25.5;
f_prime = 24;
u_prime = v_prime*f_prime/(v_prime+f_prime);
G = u_prime/v_prime; % en valeur absolue
efficacite = 28.6/100;

[det_para,coeff] = detector_param_PI(hauteur, niveau_support,
   x_decalage, angle);

det_para{:}
save det_info det_para coeff angle size_pixel hauteur v_prime
   f_prime u_prime G efficacite niveau_support;

%% Extraction des points de référence en optique pour le
   recalage
% Selection centre tritium 'trit' ou pointage souris 'point' (
   methode = 0 ou 1)
load det_info;
methode = 1;

% Sélection nombre de points à localiser et avec quelles vues.
   Tous les
% points doivent être visibles sur toutes les images choisies
   !!!

%% recalage manuel pour pts opt
```

```
plans_rec = cell(nb_points,2);
for i=1:nb_points
    plans_rec{i,1} = [0,0,1];
    plans_rec{i,2} = niveau_support+(4-i)*4;
end

disp('points recalage');
images_used = [1,2]; %HBGD
[P_opt_tmp] = fn_recalage(det_para, center_im, coeff, image_pho
   , images_used, nb_points, plans_rec, methode);

P_opt_tmp{:,:}

%%
i=1;
P_opt_ = cell(3,1);
for j=1:3
    P_opt_{j} = P_opt_tmp{j,i};
end

CT_pts_20150408_souris_172_23_tumeur_ATP32; % Script contenant
   les informations des points sur le volume TDM
save pts_ref_20150408_souris_172_23_tumeur_ATP32_new P P_opt_
   center_im;
```

A.2.1.2 Fonction modélisant le systèmes de 4 détecteurs à partir de la position du module 4 vues

Listing A.4 – detector_param_PI.m

```
function [detector_parameters,coeff_p_mm] = detector_param_PI(
   hauteur, niveau_support, x_decalage,angle_GD)

%% function details

% input :
% - position_support_x, [x1,x2,center]: position in pixel of
   the top view of the support, used to calibrate the detector
   image
% - hauteur : physical distance (mm) between the support level
   and the detector
% - position_miroir_GD [x_G,x_D] : distance in pixel of the
   mirror at support level
% - angle_GD [angle_G,angle_D] : angle par rapport à l'
```

```
   horizontale des
% miroirs
%
% output :
%
% - detector_parameters : cell of 4 detectors containing [
   x_center,y_center ; x_vec, y_vec], position of the center
   and orientation of the detector
%
% REMARQUE :
    % le repère 3D est celui des détecteurs, le plan x,y
       contient les 4
    % centres des 4 détecteurs
    % le x est horizontal et le y vertical

coeff_p_mm = 4; % Valeur expérimentale, à 440 mm, 4 pixels font
    1 mm
x_decalage_mm = x_decalage/coeff_p_mm;

detector_parameters = cell(4,1);
detector_parameters{1}=[x_decalage_mm,hauteur;0,-1];
detector_parameters{2}=[-x_decalage_mm,-hauteur+10;0,1];

%% detectors G and D

% valeurs en mm !
xG = (-73.65);
yG = niveau_support;
xD = (71.9);
yD = niveau_support;

mirror_G = [xG; yG ; angle_GD(1)]; % [x_position ; theta_mirror
   ]
mirror_D = [xD; yD ; angle_GD(2)];

detector_parameters{3} = det_lat_PI(detector_parameters{1}(1,:)
   ,mirror_G);
detector_parameters{4} = det_lat_PI(detector_parameters{1}(1,:)
   ,mirror_D);

function [detector_param] = det_lat_PI(centre_det_1,mirror_data
   )

a = tand(mirror_data(3)); % pente
```

```
b = mirror_data(2) - mirror_data(1)*a; % ordonnee a l'origine

vec_n = [(-2*b*a/(a^2+1)) , (-b*(2*a^2/(a^2+1) - 1))];
vec_n = vec_n/norm(vec_n);

D = abs([-a,1]*centre_det_1' - b)/norm([-a,1]);

vec_p = [1,-1/a]; % vec_p orienté vers le haut !
if a > 0
    vec_p = -vec_p;
end
vec_p = vec_p/norm(vec_p);

center = centre_det_1 - 2*D*vec_p;

detector_param = [center ; -vec_n];
```

A.2.1.3 Fonction pour déterminer la position de points dans le module 4 vues à partir des images 2D

Listing A.5 – `fn_recalage.m`

```
function [pt_located] = fn_recalage(det_para, center_im, coeff,
    image_fusion, images_used, nb_points, plans, methode)

% Pour une image de biolum ou photo, localise "nb_points"
   points dans le repère absolu du 4 vues.
% Une image photo seulement est nécessaire pour localiser le
   barycentre.
% Tout type d'image (n'importe quel filtre) est utilisable pour
    la localisation par pointage.
%
% Input :
%     Paramètres propres au détecteur,
%     - det_para
%     - center_im
%     - coeff
%
%     Paramètres des images
%     - image_fusion
%
%     Paramètres pour localisation des points
%     - images_used
%     - nb_points
%     - methode
```

```
%
% Output :
%      - points
%      - dist

x3 = center_im(1);
x4 = center_im(2);

point_source = cell(4,1);

for i=images_used
    if i == 1
        disp('image␣top');
    end
    if i == 2
        disp('image␣bottom');
    end
    if i == 3
        disp('image␣left');
    end
    if i == 4
        disp('image␣right');
    end
    if methode
        point_source{i} = pointCentersMan(image_fusion,
            nb_points);
    else
        point_source{i} = getCentersMan(image_fusion, nb_points
            );
    end
end

nb_im = size(images_used,2);
droites = cell(nb_points,nb_im);

i=1;
for im_used = images_used
    for j=1:nb_points
        [point,delta] = droite(det_para,im_used,coeff,
            point_source{im_used}(j,1)-x3,x4-point_source{
            im_used}(j,2));

        % passage dans le repère absolu !!!
        tmp = point(2);
```

```
        point(2) = point(3);
        point(3) = tmp;

        tmp = delta(2);
        delta(2) = delta(3);
        delta(3) = tmp;

        % stockage
        droites{j,i} = [point',delta'];
    end
    i = i+1;
end

%droites{1,:}

clear point delta;

pt_located = cell(size(droites));

for i = 1:nb_im
    for j=1:nb_points
        [pt_located{j,i}] = droite_intersect_plan(droites{j,i
           }(:,1), droites{j,i}(:,2), plans{j,1}, plans{j,2});
    end
end
```

A.2.2 Lissage et zonage

A.2.2.1 Script effectuant le traitement d'image et le zonage des différentes vues

Listing A.6 – `script_save_Im_final.m`

```
clear;
%%

traitement_images_PI;
filtre_used = [1,2,3,4,5,6,7];

%% charger image avec source tritium
% fichier image

nb_images = 7;

filtre = cell(nb_images,1);
```

```
filtre{1} = '0nm';
filtre{2} = '615nm';
filtre{3} = '665nm';
filtre{4} = '700nm';
filtre{5} = '770nm';
filtre{6} = '755nm';
filtre{7} = '790nm';

images_biolum = cell(nb_images,1);

for i=filtre_used
    Path_image = '.\Data\Manip_cerenkov_20150211\viande\';
    image_file = ['viande_1_0s_' filtre{i} '_HD_'];
    pho = 'photo.png';
    fus = 'fusion.png';
    bvr = 'biolum.txt';

    Full_path_of_file = [Path_image image_file fus];

    % ouverture images

    image_fus = imread(Full_path_of_file);
    info_image = imfinfo(Full_path_of_file);

    size_im = [info_image.Height,info_image.Width];
    %size_im = [759,972];

    image(image_fus);

    Full_path_of_file = [Path_image image_file bvr];

    images_biolum{i} = read_txt(Full_path_of_file, size_im);
end

%% zones de signal à délimiter, le reste sera utilisé pour
   définir le bruit par image

imshow(images_biolum{filtre_used(1)},[min(min(images_biolum{
   filtre_used(1)})),max(max(images_biolum{filtre_used(1)}))
   /3]);

zones = cell(4,1);

for i=1:4
```

```
    if i == 1
        disp('image␣top');
    end
    if i == 2
        disp('image␣bottom');
    end
    if i == 3
        disp('image␣left');
    end
    if i == 4
        disp('image␣right');
    end
    zones{i} = imrect_();
end

nb_pixel_S = 0;
for i=1:4
    nb_pixel_S = nb_pixel_S + zones{i}(3)*zones{i}(4);
end
nb_pixel_B = prod(size_im) - nb_pixel_S;
clear i;

save Im_chex;

%% Calcul et soustraction bruit pour chaque image
load Im_chex;

Bruit = cell(nb_images,1);
new_images_biolum = cell(nb_images,1);

for i=filtre_used
    new_images_biolum{i} = images_biolum{i};
    % Smoothing
    new_images_biolum{i} = imfilter(new_images_biolum{i},
       fspecial('gaussian',[smooth_value,smooth_value],
       smooth_sigma));
end

im_final = cell(6,1);

for i=1:5
    if i > 3
        im_final{i} = new_images_biolum{i+2};
    elseif max(size(intersect(filtre_used,[i+1,i+2]))) == 2
```

```
        im_final{i} = new_images_biolum{i+1};
    end
end

im_final{6} = new_images_biolum{1};

save im_final_20150408_souris_172_23_tumeur_ATP32_ss20-2
   im_final zones;

%% affiche images finales

for i=1:5
    figure;
    imshow(im_final{i},[min(min(im_final{i})),max(max(im_final{
       i}))]);
end
```

A.2.2.2 Fonction du traitement d'image effectué sur un blanc de référence

Listing A.7 – PI_TI.m

```
function [] = PI_TI(filtre, filtre_used, path_file, files,
   name_save, smooth_value, smooth_sigma)

nb_images = max(size(filtre));

%% load images

images_biolum_raw = cell(nb_images,1);

for i=filtre_used
    Path_image = path_file;
    image_file = [files filtre{i} '_HD_'];
    fus = 'fusion.png';
    bvr = 'biolum.txt';

    Full_path_of_file = [Path_image image_file fus];

    % ouverture images

    image_fus = imread(Full_path_of_file);
    info_image = imfinfo(Full_path_of_file);
    size_im = [info_image.Width,info_image.Height];

    Full_path_of_file = [Path_image image_file bvr];
```

```
    images_biolum_raw{i} = read_txt(Full_path_of_file, size_im)
        ;
end

%% smooth images

images_biolum_smooth = cell(nb_images,1);

for i=filtre_used
    images_biolum_smooth{i} = images_biolum_raw{i};
    images_biolum_smooth{i} = imfilter(images_biolum_smooth{i},
        fspecial('gaussian',[smooth_value,smooth_value],
        smooth_sigma));
end

%% substract images

im_final = cell(6,1);

for i=1:5
    if i > 3
        im_final{i} = images_biolum_smooth{i+2};
    elseif max(size(intersect(filtre_used,[i+1,i+2]))) == 2
        im_final{i} = images_biolum_smooth{i+2}-
            images_biolum_smooth{i+1};
    end
end

im_final{6} = images_biolum_smooth{1};

%% threshold images

%% save

save(name_save, 'images_biolum_raw', 'images_biolum_smooth', '
    im_final');
```

A.3 Mise en place modèle direct

A.3.1 Coefficients optiques

A.3.1.1 Paramètres généraux

Listing A.8 – parameters.h

```
#ifndef PARAMETERS_H
#define PARAMETERS_H

#include <stdio.h>
#include <iostream>
#include <string>
#include <fstream>
#include <stdlib.h>
#include <math.h>
#include "../../Path/pathData.h"

#define          PI                3.1415926535
#define          CELERITE          299792458
#define          degree_to_rad     .0174532888
#define          N_lambda          5

#define          n_air             1
#define          n_eau             1.33
#define          n_tissu           1.34
#define          n_cerveau         1.37
#define          n_os              1.40
#define          n_crane           1.40

#define          Reff_cerveau_air          0.449877
#define          Reff_cerveau_crane        0.006767
#define          Reff_crane_air            0.493476

#endif
```

A.3.1.2 Fonctions utilisées dans les programmes c++

Listing A.9 – fonctions.h

```
#ifndef FONCTIONS_H
#define FONCTIONS_H

#include "parameters.h"
#include "./fonctions_calculs_constantes.c"

float indice_opt(int indice);

int read_nb_milieux();
int remplissage_mu(int indice, float** table_mu_complete, int
   nombre_lambda, int* tableau_lambda, float g);
```

```
int read_indicage(int &nombre_milieux, int* indice_milieux_tmp)
    ;
void mesh2msh_f(FILE* f_mesh, FILE* f_msh);

int valeurs_mu(float *mu_crane, float *mu_cerveau, int N, int
    lambda_min, int lambda_max, float g);
float mu_fn_lambda(int value, int lambda, int milieu);

float mu_a_Hb(int lambda);
float mu_a_HbO2(int lambda);
float mu_a_W(int lambda);

int read_header(int *size_x, int *size_y, int *size_z, float *
    dx, float *dy, float *dz, FILE *file);

float norme_carre(float vecteur[], int taille);

float pente(int x, int y, int z, char image, char orientation,
    int xyz_i, float referentiel[12], float *offset);
float integrale_frontiere(float mu_eff, float D, float
    n_incident, float n_out, float d_j, float p_ij);

float R_eff_precalculated(float n_incident, float n_out);
#endif
```

Listing A.10 – fonctions.cpp

```
#include "fonctions.h"


int read_nb_milieux()
{
  FILE *reader;

  reader = fopen(PathParam PathExpe "imagesCT/indice_milieux.
     para" , "rt");
  if(reader == NULL){printf("échec␣d'ouverture␣du␣fichier␣
     indice_milieux.para␣pour␣nombre␣de␣milieux\n\n"); fclose(
     reader); return(-1);}

  int i;
  float valeur;
  char trash[10];

  int value = 0;
```

```
  while(!feof(reader))
  {
        i=0;
    fscanf(reader, "%s␣%i␣␣␣␣␣␣␣%f", trash, &i, &valeur);
    if(i > 0)
    {
      value++;
    }
  }
  fclose(reader);
  return(value);
}


int read_indicage(int &nombre_milieux, int* indice_milieux_tmp)
{
  FILE *reader;

  reader = fopen(PathParam PathExpe "imagesCT/indice_milieux.
     para" , "rt");
  if(reader == NULL){printf("échec␣d'ouverture␣du␣fichier␣
     indice_milieux.para␣pour␣indice␣des␣milieux\n\n"); fclose(
     reader); return(-1);}

  char trash[10];
  float valeur;
  rewind(reader);
  int j=1;
  int i;


  while(!feof(reader))
  {
    fscanf(reader, "%s␣␣%i␣␣␣␣␣␣%f", trash, &i, &valeur);
    if (i>0)
    {
      indice_milieux_tmp[i-1]=j;
      nombre_milieux++;
    }
    j++;
  }
  fclose(reader);
  return(0);
}
```

```
float indice_opt(int indice)
{
  int i;

  FILE *reader;

  reader = fopen(PathParam PathExpe "imagesCT/indice_milieux.
     para" , "rt");
  if(reader == NULL){printf("échec␣d'ouverture␣du␣fichier␣
     indice_milieux.para␣pour␣indice␣optique\n\n"); fclose(
     reader); return(-1);}

  char trash[10];
  float valeur;
  rewind(reader);
  while(!feof(reader))
  {
    fscanf(reader, "%s␣␣%i␣␣␣␣␣␣␣%f", trash, &i, &valeur);
    if (i == indice)
    {
      fclose(reader);
      return(valeur);
    }
  }
  fclose(reader);
}

int remplissage_mu(int indice, float** table_mu_complete, int
   nombre_lambda, int tableau_lambda[], float g)
{
  FILE *reader;

  reader = fopen(PathParam PathExpe "imagesCT/indice_milieux.
     para" , "rt");
  if(reader == NULL){printf("échec␣d'ouverture␣du␣fichier␣
     indice_milieux.para␣pour␣remplissage␣mu\n\n"); fclose(
     reader); return(-1);}

  char trash[10];
  float valeur;
  rewind(reader);
  int j=1;
  int i;
```

```
  while(!feof(reader))
  {
    fscanf(reader, "%s␣␣%i␣␣␣␣␣␣␣%f", trash, &i, &valeur);
    if (i == indice)
    {
      break;
    }
    j++;
  }

  fclose(reader);

  int k;
  for(i=0; i<nombre_lambda; i++)
  {
    table_mu_complete[i][0]=mu_fn_lambda(2, tableau_lambda[i],
       j);
    table_mu_complete[i][1]=mu_fn_lambda(1, tableau_lambda[i],
       j);
    for(k=1; k<8; k++)
    {
      table_mu_complete[i][(k+1)]=table_mu_complete[i][1] +
         table_mu_complete[i][0]*(1-pow(g,k));
    }
  }
}

  /** calcul des mu en mm^(-1)
  ***************/

float mu_fn_lambda(int value, int lambda, int milieu)
{
  float g=0.9;

  FILE *reader;

  reader = fopen(PathCode "tool/Parameters/Optic_param/
     milieux_coeff.para" , "rt");
  if(reader == NULL){printf("échec␣d'ouverture␣du␣fichier␣
     milieux_coeff.para␣nyah␣!!!!\n\n"); fclose(reader); return
     (-1);}

  int i;
```

```
  char buffer[150];
  int len = 150;

  if(milieu > 1)
  {
        for(i=0; i<milieu-1; i++)
        {
          fgets(buffer, len, reader);
        }
  }

  char trash[10];
  float a, b, Sb, x, Sw;

  fscanf(reader, "%s␣␣␣␣␣␣␣␣␣␣␣%f␣␣␣␣␣␣%f␣␣␣␣␣␣%f␣␣␣␣␣␣%f
     ␣␣␣␣␣␣%f", trash, &a, &b, &Sb, &x, &Sw);
  fclose(reader);

  float mu_s;
  mu_s= a*pow(lambda,-b)/(1-g);

  float mu_a;
  mu_a = Sb*(x*mu_a_Hb(lambda) + (1-x)*mu_a_HbO2(lambda)) + Sw*
     mu_a_W(lambda);

  if(value == 1)
  {
        return(mu_a);
  }
  else if(value == 2)
  {
        return(mu_s);
  }
}

float mu_a_Hb(int lambda)
{
  int line;
  int impair;

  line = (lambda-250)/2;
  impair = (lambda-250)%2;

  FILE *reader;
```

```
  reader = fopen(PathCode "tool/Parameters/Optic_param/
     Hb_HbO2_coeff.dat" , "rt");
  if(reader == NULL){printf("échec␣d'ouverture␣du␣fichier␣
     Hb_HbO2_coeff.dat\n\n"); fclose(reader); return(-1);}

  int i;
  char buffer[150];
  int len = 150;

  for(i=0; i<line+2; i++)
  {
        fgets(buffer, len, reader);
  }

  float Hb, HbO2;

  fscanf(reader, "%i␣␣␣␣%f␣␣␣␣␣␣%f", &i, &HbO2, &Hb);
  //check
  if(i != (lambda-impair)){printf("\n----␣erreur␣lecture␣----\n
     \n");}
  if(impair == 1)
  {
        float HbO2_2, Hb_2;
        fscanf(reader, "%i␣␣␣␣␣␣%f␣␣␣␣␣␣%f", &i, &HbO2_2, &Hb_2
           );
        Hb = (Hb + Hb_2)/2;
  }
  fclose(reader);
  return(2.303*Hb/10/64.5);
}

float mu_a_HbO2(int lambda)
{
  int line;
  int impair;

  line = (lambda-250)/2;
  impair = (lambda-250)%2;

  FILE *reader;
  reader = fopen(PathCode "tool/Parameters/Optic_param/
     Hb_HbO2_coeff.dat" , "rt");
  if(reader == NULL){printf("échec␣d'ouverture␣du␣fichier␣
     Hb_HbO2_coeff.dat\n\n"); fclose(reader); return(-1);}
```

```
  int i;
  char buffer[150];
  int len = 150;

  for(i=0; i<line+2; i++)
  {
        fgets(buffer, len, reader);
  }

  float Hb, HbO2;

  fscanf(reader, "%i␣␣␣␣%f␣␣␣␣␣␣%f", &i, &HbO2, &Hb);
  //check
  if(i != (lambda-impair)){printf("\n----␣erreur␣lecture␣----\n
     \n");}
  if(impair == 1)
  {
        float HbO2_2, Hb_2;
        fscanf(reader, "%i␣␣␣␣␣␣%f␣␣␣␣␣␣%f", &i, &HbO2_2, &Hb_2
           );
        HbO2 = (HbO2 + HbO2_2)/2;
  }
  fclose(reader);
  return(2.303*HbO2/10/64.5);
}

float mu_a_W(int lambda)
{
  int line;
  int impair;
  int pas = 10;

  line = (lambda-200)/pas;
  impair = (lambda-200)%pas;

  FILE *reader;
  reader = fopen(PathCode "tool/Parameters/Optic_param/
     Water_coeff.dat" , "rt");
  if(reader == NULL){printf("échec␣d'ouverture␣du␣fichier␣
     Water_coeff.dat\n\n"); fclose(reader); return(-1);}

  int i;
  char buffer[150];
```

```
  int len = 150;

  for(i=0; i<line+8; i++)
  {
        fgets(buffer, len, reader);
  }

  float W;

  fscanf(reader, "%i␣␣␣␣%f", &i, &W);
  //check
  if(i != (lambda-impair)){printf("\n----␣erreur␣lecture␣----\n
     \n");}
  if(impair > 0)
  {
        float W_2;
        fscanf(reader, "%i␣␣␣␣␣␣%f", &i, &W_2);
        W = (W*(pas-impair) + W_2*impair)/pas;
  }
  fclose(reader);
  return(W/10);
}

/** ************ **/

void mesh2msh_f(FILE* f_mesh, FILE* f_msh)
{
  int datasize=0;


  fprintf(f_msh, "$MeshFormat\n");
  fprintf(f_msh, "2.4␣%i␣%i\n", 0,  datasize);
  fprintf(f_msh, "$EndMeshFormat\n\n");


  /** nodes **/

  fprintf(f_msh, "$Nodes\n");
  int node_number;

  fscanf(f_mesh, "%i", &node_number);
  fprintf(f_msh, "%i\n", node_number);

  int i;
```

```
  for(i=0; i<node_number; i++)
  {
    float x_tmp, y_tmp, z_tmp;
    fscanf(f_mesh, "%f␣%f␣%f", &x_tmp, &y_tmp, &z_tmp);
    fprintf(f_msh, "%i␣%f␣%f␣%f\n", i+1, x_tmp, y_tmp, z_tmp);
  }
  fprintf(f_msh, "$EndNodes\n\n");
  /** elements : tetrahedra **/

  fprintf(f_msh, "$Elements\n");
  int elements_number;

  fscanf(f_mesh, "%i", &elements_number);
  fprintf(f_msh, "%i\n", elements_number);

  for(i=0; i<elements_number; i++)
  {
    int node1, node2, node3, node4, type_milieu;
    fscanf(f_mesh, "%i␣%i␣%i␣%i␣%i", &type_milieu, &node1, &
       node2, &node3, &node4);
    fprintf(f_msh, "%i␣%i␣1␣%i␣%i␣%i␣%i␣%i\n", i+1, 4,
       type_milieu, node1, node2, node3, node4);
  }
  fprintf(f_msh, "$EndElements\n\n");

  fclose(f_mesh);
  fclose(f_msh);
}

/** ******************** **/

int read_header(int *size_x, int *size_y, int *size_z, float *
   dx, float *dy, float *dz, FILE *file)
{
  char trash[10];
  rewind(file);
  fscanf(file, "%i␣%i␣%i", size_x, size_y, size_z);
  fscanf(file, "%s␣%s", trash, trash);
  fscanf(file, "%s␣%f␣%s␣%f␣%s␣%f", trash, dx, trash, dy, trash
     , dz);
}

/** ******************** **/
```

```
float pente(int x, int y, int z, char image, char orientation,
   int xyz_i, float referentiel[12], float *offset)
{
  float a;

  int size_x, size_y, size_z;
  size_x = (int)referentiel[7]-(int)referentiel[6];
  size_y = (int)referentiel[9]-(int)referentiel[8];
  size_z = (int)referentiel[11]-(int)referentiel[10];

  if(orientation=='T')
  {
    if(image=='T') //y constant, x et z à gérer
    {
      a = (float)(xyz_i-x)/(float)(size_z-(z+1));
      *offset = (float)((x+1)-a*(z+1));
    }
    if(image=='B') //y constant, x et z à gérer
    {
      a = (float)(xyz_i-x)/(float)(z+1);
      *offset = (float)((x+1)-a*(z+1));
    }
    if(image=='L') //y constant, x et z à gérer
    {
      a = (float)(xyz_i-(z+1))/(float)(x+1);
      *offset = (float)((z+1)-a*(x+1));
    }
    if(image=='R') //y constant, x et z à gérer
    {
      a = (float)(xyz_i-(z+1))/(float)(size_x-(x+1));
      *offset = (float)((z+1)-a*(x+1));
    }
  }
  return(a);
}


/** ******************** **/

float integrale_frontiere(float mu_eff, float D, float
   n_incident, float n_out, float d_j, float p_ij)
{
  int i;
```

```
//paramètres suivants définis de façon empirique de sorte à
   trouver un pas adapté
//marche très bien mais peut être optimisé davantage !
int n=200;
float k=0.8;
float j=10;
//***********\\

float a; /// sert d'indicateur pour la convergence de l'
   intégrale

float Reff;
Reff = R_eff_precalculated(n_incident, n_out);

float zb;
zb = 2*D*((1+Reff)/(1-Reff));

float pas=0;
float r_inte=0;
float l_i=0;

float save=0, save_1=0;
float integrale=0;

r_inte = sqrt(pow(d_j,2)+pow(p_ij,2));
//printf("\n r_inte %d = %f \n",i, r_inte);
save = exp(-mu_eff*r_inte)/r_inte;

for(i=0; i<n; i++)
{
  pas = exp((i-n*k)/j);

  a=integrale; /// ici !!!

  l_i = l_i+pas;
  r_inte = sqrt(pow(d_j+l_i,2)+pow(p_ij,2));
  //printf("\n r_inte %d = %f \n",i+1, r_inte);

  save_1=exp(-l_i/zb)*exp(-mu_eff*r_inte)/r_inte;
  integrale = integrale+(save_1 + save)/2*pas;
  save = save_1;

  a=integrale/a; /// et là !!!
```

```
    //printf("convergence: rapport = %f\n", a);
  }
  //printf("integrale frontière = %f\n", integrale);
  return(integrale/zb);
}


/** ******************** **/

float R_eff_precalculated(float n_incident, float n_out)
{
  if((n_incident == (float)n_cerveau) && (n_out == (float)
     n_crane)) return(Reff_cerveau_crane);

  if((n_incident == (float)n_crane) && (n_out == (float)n_air))
      return(Reff_crane_air);

  if((n_incident == (float)n_cerveau) && (n_out == (float)n_air
     )) return(Reff_cerveau_air);

  return(-1);
}

/** ******************** **/

float norme_carre(float vecteur[], int taille)
{
  float a=0;
  int i;
  for(i=0;i<taille;i++)
  {
    a = vecteur[i]*vecteur[i]+a;
  }
  return(a);
}

float norme_carre_mat(float matrice[], int taille_x, int
   taille_y)
{
  float a=0;
  int i;
  for(i=0;i<taille_x;i++)
  {
```

```
    a += norme_carre(&(matrice[i*taille_x]), taille_y);
  }
  return(a);
}


/** ******************** **/
```

Listing A.11 – fonctions_calculs_constantes.c

```
#include "parameters.h"

/*
#define PI 3.1415926535
#define CELERITE 299792458

#define          Reff_cerveau_air         0.449877
#define          Reff_cerveau_crane       0.006767
#define          Reff_crane_air           0.493476
*/

/**
filtres_to_lambda

classe fonction type de filtre (H "passe haut", L "passe bas"
   ou B "passe bande"), puis fonction lambda croissant

convertit H/L en B avec lambda_moy et un delta lambda associé
H-H => lambda_moy = (max (lambda_i) - min (lambda_i))/2 + min
   lambda_i
      d_lambda = (max (lambda_i) - min (lambda_i))/2
L-L => lambda_moy = (max (lambda_i) - min (lambda_i))/2 + min
   lambda_i
      d_lambda = (max (lambda_i) - min (lambda_i))/2
B => rien !
L-H => faire L - H > 0 pour vérifier si faisable
      lambda_moy = (max (lambda_i) - min (lambda_i))/2 + min
         lambda_i
      d_lambda = (max (lambda_i) - min (lambda_i))/2

trier les B (anciens et nouveaux), éventuellement enlever ceux
   avec trop grand d_lambda

end
```

```
**/


/** prototypes **/

int sphericToCartesian(double point[3], double rayon, double
   theta, double phi);
int prodVectoriel(double vec_1[3], double vec_2[3], double
   vec_f[3]);
double prodScalaire3D(double vec_1[3], double vec_2[3]);

int tri_croissant(int* tableau, int nb_lig, int nb_col, int
   colonne_triage);

int mat_transpose(float* mat, int l, int c, float* mat_t);
int prodmat(float* A, float* B, int l_a, int c_a, int c_b,
   float* P); // P=A*B
int MethodeGauss(float* mat, int N, float* mat_inverse);

float R_Fresnel(float theta, float n_incident, float n_out);
float R_eff(float n_incident, float n_out); // calcul de r_eff
   avec la relation tirée de 1994_Haskell
float R_n(float n_incident, float n_out, int ordre); // calcul
   des coefficients utilisés pour la méthode SPn

int table_Reff(); //calcule les coefficients de passage entre
   les différentes interfaces définies par l'utilisateur
int table_Rn(int ordre);
int filtres_2_lambda(int critere); //convertit les différents
   filtres en filtres passe-bande pour avoir les valeurs de
   lambda "monochromatique"

void affiche_mat(float* mat, int N, int M); //affiche matrice
   de float
void affiche_mat_int(float* mat, int N, int M, int coeff); //
   affiche une matrice de float en int pour gain de place

/** end **/

int sphericToCartesian(double point[3], double rayon, double
   theta, double phi)
{
  theta = theta*PI/180;
  phi = phi*PI/180;
```

```
  point[0] = rayon*sin(theta)*cos(phi);
  point[1] = rayon*sin(theta)*sin(phi);
  point[2] = rayon*cos(theta);
}

int prodVectoriel3D(double vec_1[3], double vec_2[3], double
   vec_f[3])
{
  vec_f[0] = vec_1[1]*vec_2[2]-vec_1[2]*vec_2[1];
  vec_f[1] = vec_1[2]*vec_2[0]-vec_1[0]*vec_2[2];
  vec_f[2] = vec_1[0]*vec_2[1]-vec_1[1]*vec_2[0];
}

double prodScalaire3D(double vec_1[3], double vec_2[3])
{
  double value = 0;

  for(int i=0;i<3;i++)
  {
        value += vec_1[i]*vec_2[i];
  }
  return(value);
}

int tri_croissant(int* tableau, int nb_lig, int nb_col, int
   colonne_triage)
{
  int copie[nb_lig][nb_col+1];
  int i,j, scan, position;

  // copie du tableau pour éviter de perdre les données pendant
      le tri
  for(j=0; j<nb_lig; j++)
  {
    for(i=0; i<nb_col; i++)
    {
      copie[j][i]=tableau[j*nb_col+i];

      if(i==(colonne_triage-1)) // si on est à la bonne ligne,
         on regarde la position
      {
        position=0;
```

```
        for(scan=0; scan<nb_lig; scan++) // regarde combien d'
           élément sont plus faible que l'élément actuel,
           détermine la position pour rangement
        {
          if(tableau[scan*nb_col+i]<tableau[j*nb_col+i]){
            position++;}
        }
        copie[j][nb_col]=position;
      }
    }
  }

  // remet les éléments dans le tableau d'origine dans l'ordre
  for(j=0; j<nb_lig; j++)
  {
    for(i=0; i<nb_col; i++)
    {
      tableau[(copie[j][nb_col])*nb_col+i]=copie[j][i];
    }
  }
  return(0);
}


int table_Reff()
{
  FILE *milieux;
  FILE *Reff_interfaces;

  milieux = fopen("../Parameters/Interfaces/
     milieux_et_interfaces.para" , "rt");
  if(milieux == NULL){printf("échec␣d'ouverture␣du␣fichier␣
     milieux_et_interfaces.para\n\n"); return(-1);}

  int n_milieu,i;
  int i1,i2;
  float reff_value;

  printf("debut␣lecture␣0\n");

  fscanf(milieux, "␣␣␣␣␣%i", &n_milieu);

  printf("n_milieu␣=␣%i\n", n_milieu);
```

```
  char nom[n_milieu][10];
  float indice[n_milieu];

  printf("debut␣lecture\n");

  for(i=0; i<n_milieu; i++)
  {
    fscanf(milieux, "%s␣%f", &(nom[i][0]), &(indice[i]));
  }

  printf("lecture␣ok\n");

  Reff_interfaces = fopen("../Parameters/Interfaces/
     reff_interfaces.dat" , "wt");
  if(Reff_interfaces == NULL){printf("échec␣d'ouverture␣du␣
     fichier␣reff_interfaces.dat\n\n"); return(-1);}

  while(!feof(milieux))
  {
    fscanf(milieux, "%i␣%i", &i1, &i2);
    i1--;
    i2--;

    reff_value = R_eff(indice[i1], indice[i2]);

    fprintf(Reff_interfaces, "%s␣%s\n%f\n\n", &(nom[i1][0]), &(
       nom[i2][0]), reff_value);

  }
  fprintf(Reff_interfaces, "#␣table␣Reff␣suivant␣interface\n#␣
     milieu␣de␣propagation␣->␣milieu␣de␣sortie");
  fclose(milieux);
  fclose(Reff_interfaces);
  return(0);
}

int table_Rn(int ordre)
{
  FILE *milieux;
  FILE *Rn_interfaces;

  milieux = fopen("../Parameters/Interfaces/
     milieux_et_interfaces.para" , "rt");
  if(milieux == NULL){printf("échec␣d'ouverture␣du␣fichier␣
```

```
    milieux_et_interfaces.para\n\n"); return(-1);}

int n_milieu,i;
int i1,i2;
float rn_value;

printf("debut␣lecture␣0\n");

fscanf(milieux, "␣␣␣␣␣%i", &n_milieu);

printf("n_milieu␣=␣%i\n", n_milieu);

char nom[n_milieu][15];
float indice[n_milieu];

printf("debut␣lecture\n");

for(i=0; i<n_milieu; i++)
{
  fscanf(milieux, "%s␣%f", &(nom[i][0]), &(indice[i]));
  //printf("%s %f\n", &(nom[i][0]), indice[i]);
}

printf("lecture␣ok\n");

Rn_interfaces = fopen("../Parameters/Interfaces/rn_interfaces
   .dat" , "wt");
if(Rn_interfaces == NULL){printf("échec␣d'ouverture␣du␣
   fichier␣rn_interfaces.dat\n\n"); return(-1);}

while(!feof(milieux))
{
  fscanf(milieux, "%i␣%i", &i1, &i2);
  i1--;
  i2--;
  /// interface directe
  fprintf(Rn_interfaces, "%s␣%s\n", &(nom[i1][0]), &(nom[i2
     ][0]));

  for(i=1; i<ordre+1; i++)
  {
    rn_value = R_n(indice[i1], indice[i2], i);

    fprintf(Rn_interfaces, "R␣%i␣␣␣␣␣␣%f\n", i, rn_value);
```

```
    }

    fprintf(Rn_interfaces, "\n");
    /// interface inverse
    fprintf(Rn_interfaces, "%s␣%s\n", &(nom[i2][0]), &(nom[i1
       ][0]));

    for(i=1; i<ordre+1; i++)
    {
      rn_value = R_n(indice[i2], indice[i1], i);

      fprintf(Rn_interfaces, "R␣%i␣␣␣␣␣␣␣%f\n", i, rn_value);
    }

    fprintf(Rn_interfaces, "\n");
  }
  fprintf(Rn_interfaces, "#␣table␣Rn␣suivant␣interface\n#␣
     milieu␣de␣propagation␣->␣milieu␣de␣sortie");
  fclose(milieux);
  fclose(Rn_interfaces);
  return(0);
}

int filtres_2_lambda(int critere)
{
        /** trouve toute les filtres passe-bande récupérables *
           */
  FILE *file_filtres;
  FILE *file_all_lambda;

  /*ouverture/création fichiers*/
  file_filtres = fopen("../Parameters/Filters/filtres.para" , "
     rt+");
  if(file_filtres == NULL){printf("échec␣d'ouverture␣du␣fichier
     ␣filtres.dat\n\n"); return(-1);}


  char trash[10];
  int modification;

  fscanf(file_filtres, "%i", &modification);
  if(modification)
  {
    int end = 1;
```

```
char type[10];
int lambda_filtre[10];
int number = 0;
int tot=0;

int h=0; //nb de filtre passe-haut
int l=0; //nb de filtre passe-bas
int b=0; //nb de filtre passe-bande

while(end)
{
  if(number==0)fscanf(file_filtres, "%s␣␣␣␣␣%s␣␣␣␣␣␣%s",
     trash, trash, trash);
  fscanf(file_filtres, "%i␣␣%c␣␣␣␣␣␣%i", &end, &(type[
     number]), &(lambda_filtre[number]));

  if(type[number]=='H')
  {
    h++;
  }
  else if(type[number]=='L')
  {
    l++;
  }
  else if(type[number]=='B')
  {
    b++;
  }

  printf("\n␣%d␣%c␣%d\n", number, type[number],
     lambda_filtre[number]);

  if(number==0){tot=end;}
  number++;
  if(number==tot){break;}
}
/* trie les filtres suivant le type */
int H[h]; //passe-haut
int L[l]; //passe-bas
int B[b][2]; //passe-bande avec delta_lambda associé

int h1=0;
int l1=0;
int b1=0;
```

```
for(end=0; end<tot; end++)
{
  if(type[end]=='H')
  {
    H[h1]=lambda_filtre[end];
    h1++;
  }
  else if(type[end]=='L')
  {
    L[l1]=lambda_filtre[end];
    l1++;
  }
  else if(type[end]=='B')
  {
    B[b1][0]=lambda_filtre[end];
    B[b1][1]=25;
    b1++;
  }
}

/* trie les filtres H et L par ordre croissant */
tri_croissant(H, h, 1, 1);
tri_croissant(L, l, 1, 1);
 //test si ça marche correctement !
 int i,j;
 printf("Passe␣haut\n");
 for(i=0; i<h; i++){printf("%d\n",H[i]);}
 printf("Passe␣bas\n");
 for(i=0; i<l; i++){printf("%d\n",L[i]);}
 printf("Passe␣bande\n");
 for(i=0; i<b; i++){printf("%d\n",B[i][0]);}


int comb_HL=0;
if((h>0) && (l>0))
{
  // test les combinaisons H-L
  // do bazar test combinaison H-L et le programmer -_-'
}

int comb_tot=b+comb_HL;
if(h>1){comb_tot=comb_tot+h-1;}
if(l>1){comb_tot=comb_tot+l-1;}
```

```
int B_f[comb_tot][6];

for(i=0; i<b; i++)
{
  B_f[i][0]=B[i][0];
  B_f[i][1]=B[i][1];
  B_f[i][2]=0;
  B_f[i][3]=0;
  B_f[i][4]=0;
  B_f[i][5]=0;
}
for(j=0; j<h-1; j++, i++)
{
  B_f[i][0]=H[j]+(H[j+1]-H[j])/2;
  B_f[i][1]=(H[j+1]-H[j])/2;
  B_f[i][2]=1;
  B_f[i][3]=H[j];
  B_f[i][4]=1;
  B_f[i][5]=H[j+1];
}
for(j=0; j<l-1; j++, i++)
{
  B_f[i][0]=L[j]+(L[j+1]-L[j])/2;
  B_f[i][1]=(L[j+1]-L[j])/2;
  B_f[i][2]=-1;
  B_f[i][3]=L[j+1];
  B_f[i][4]=-1;
  B_f[i][5]=L[j];
}
for(j=0; j<comb_HL; j++, i++)
{
  /*B_f[i][0]=H[j]+(H[j+1]-H[j])/2;
  B_f[i][1]=(H[j+1]-H[j])/2;
  B_f[i][2]=-1;
  B_f[i][3]=L[j+1];
  B_f[i][4]=1;
  B_f[i][5]=H[j];*/
}

int* B_f_ = &(B_f[0][0]);
tri_croissant(B_f_, comb_tot, 6, 1);

printf("all␣Passe␣bande\n");
```

```
    for(j=0; j<i; j++){printf("%d␣%d␣%d␣%d␣%d␣%d\n",B_f[j][0],
       B_f[j][1],B_f[j][2],B_f[j][3],B_f[j][4],B_f[j][5]);}

    // enregistre les valeurs de lambda assez monochromatique (
       delta_lambda<limite) dans un fichier

    file_all_lambda = fopen("../Parameters/Filters/lambda.dat"
       , "wt");
    if(file_all_lambda == NULL){printf("échec␣d'ouverture␣du␣
       fichier␣all_lambda.dat\n\n"); return(-1);}

    fprintf(file_all_lambda, "#␣lambda␣du␣passe-bande␣obtenu␣
       par␣soustraction␣du␣filtre␣à␣lambda1␣par␣le␣filtre␣à␣
       lambda2\n");
    fprintf(file_all_lambda, "lambda␣␣␣␣delta␣␣␣type1␣␣␣lambda1
       ␣type2␣␣␣lambda2\n");

    for(j=0; j<comb_tot; j++)
    {
      if(B_f[j][1]<critere)
      {
        fprintf(file_all_lambda,"%i␣␣␣␣␣%i", B_f[j][0], B_f[j
           ][1]);
        for(i=0; i<2; i++)
        {
          if(B_f[j][(i+1)*2]==1)fprintf(file_all_lambda, "
             ␣␣␣␣␣␣H");
          if(B_f[j][(i+1)*2]==0)fprintf(file_all_lambda, "
             ␣␣␣␣␣␣B");
          if(B_f[j][(i+1)*2]==-1)fprintf(file_all_lambda, "
             ␣␣␣␣␣L");
          fprintf(file_all_lambda, "␣␣␣␣%i", B_f[j][(i+1)*2+1])
             ;
        }
        fprintf(file_all_lambda, "\n");
      }
    }
    fclose(file_all_lambda);
  }
  rewind(file_filtres);
  fprintf(file_filtres, "0");

  fclose(file_filtres);
  return(0);
```

```
}


float R_Fresnel(float cos_theta, float n_incident, float n_out)
{
  float theta=acos(cos_theta);
  float theta_c=PI/2;

  if(n_out<n_incident)
  {
    theta_c=asin(n_out/n_incident);
  }

  if(theta>theta_c)
  {
    return(1);
  }
  else
  {
    float theta_prim = asin((n_incident/n_out)*sin(theta));
    float n_t = n_incident*cos_theta;
    float n_prim_t = n_out*cos_theta;
    float n_t_prim = n_incident*cos(theta_prim);
    float n_prim_t_prim = n_out*cos(theta_prim);
    float R_fresn;

    R_fresn = (pow((n_t_prim-n_prim_t)/(n_t_prim+n_prim_t),2)+
       pow((n_t-n_prim_t_prim)/(n_t+n_prim_t_prim),2))/2;

    return(R_fresn);
  }
}

float R_eff(float n_incident, float n_out) // calcul de r_eff
   avec la relation tirée de 1994_Haskell
{
  int i, n=10000;
  float theta[n];

  for(i=0; i<n; i++)
  {
    theta[i]=i*PI/2/(n-1);
    //printf("theta = %f\n", theta[i]);
  }
```

```
  /** méthode des trapèzes pour calcul des intégrales R_phi et
     R_j **/

  float R_phi_theta[n],R_j_theta[n];
  float R_phi=0,R_j=0;
  float R_Fresn;

  for(i=0; i<n; i++)
  {
    R_Fresn = R_Fresnel(cos(theta[i]), n_incident, n_out);
    //printf("theta= %f ;R_Fresn %d = %f\n", theta[i], i,
       R_Fresn);
    R_phi_theta[i] = 2*sin(theta[i])*cos(theta[i])*R_Fresn;
    R_j_theta[i] = 3*sin(theta[i])*pow(cos(theta[i]),2)*R_Fresn
       ;
  }

  for(i=0; i<(n-1); i++)
  {
    R_phi+=(R_phi_theta[i+1]+R_phi_theta[i])/2*(theta[i+1]-
       theta[i]);
    R_j+=(R_j_theta[i+1]+R_j_theta[i])/2*(theta[i+1]-theta[i]);
  }

  //printf("\nR_phi = %f R_j = %f \n", R_phi, R_j);

  /** fin méthode des trapèzes **/

  float Reff;

  Reff = (R_phi+R_j)/(2-R_phi+R_j);
  //printf("Reff = %f\n",Reff);
  return(Reff);
}

float R_n(float n_incident, float n_out, int ordre)
{
  int i, n=10000;
  float theta[n];
  float cos_theta[n];

  for(i=0; i<n; i++)
  {
```

```
    theta[i]=i*PI/2/(n-1);
    cos_theta[i]=cos(theta[i]);
    //printf("theta = %f\n", theta[i]);
  }

  /** méthode des trapèzes pour calcul des intégrales R_phi et
     R_j **/

  float Rn_theta[n];
  float R_Fresn;
  float Rn=0;

  for(i=0; i<n; i++)
  {
    R_Fresn = R_Fresnel(cos_theta[i], n_incident, n_out);
    //printf("theta= %f ;R_Fresn %d = %f\n", theta[i], i,
       R_Fresn);
    Rn_theta[i] = pow(cos_theta[i],ordre)*R_Fresn;
  }

  for(i=0; i<(n-1); i++)
  {
    Rn += (Rn_theta[i+1]+Rn_theta[i])/2*(theta[i+1]-theta[i]);
  }

  //printf("\nR_phi = %f R_j = %f \n", R_phi, R_j);

  /** fin méthode des trapèzes **/

  //printf("R_n = %f\n",Rn);
  return(Rn);
}

int prodmat(float* A, float* B, int l_a, int c_a, int c_b,
   float* P) // P=A*B
{
  int i, j, z;
  for (i = 0; i < l_a; i++)
  {
    for (j = 0; j < c_b; j++)
    {
      P[i*c_b+j] = 0;
      for (z = 0; z < c_a; z++)
      {
```

```
        P[i*c_b+j] += A[i*c_a+z] * B[z*c_b+j];
      }
    }
  }
  return(0);
}

int mat_transpose(float* mat, int l, int c, float* mat_t)
{
  int i,j;
  for (i = 0; i < l; i++)
  {
    for (j = 0; j < c; j++)
    {
      mat_t[j*l+i]=mat[i*c+j];
    }
  }
  return(0);
}

int MethodeGauss(float* mat, int N, float* mat_inverse)
{
  int inversible = 0;
  int k,i,colonne,colonnebis;
  float var,var1;

  for(i=0;i<N;i++)
  {
    for(k=0;k<N;k++)
    {
      if(i==k) mat_inverse[i*N+k]=1;
      else mat_inverse[i*N+k]=0;
    }
  }
  k=0;

  while((inversible == 0)&&(k<N))
  {
    if (mat[k*N+k] != 0)
    {
      var = mat[k*N+k];
      for (colonne=0;colonne<N;colonne++)
      {
          mat[k*N+colonne] = mat[k*N+colonne]/var;
```

```
            mat_inverse[k*N+colonne] = mat_inverse[k*N+colonne]/
               var;//Normalisation de la ligne contenant l'
               élément diagonal
        }
        for (i=0;i<N;i++)
        {
            if (i != k)
            {
              var1=mat[i*N+k];
              for (colonnebis=0;colonnebis<N;colonnebis++)
                {
                  mat[i*N+colonnebis] = mat[i*N+colonnebis] - mat
                     [k*N+colonnebis]*var1;
                  mat_inverse[i*N+colonnebis] = mat_inverse[i*N+
                     colonnebis] - mat_inverse[k*N+colonnebis]*
                     var1;
                }
            }
        }
        k++;
    }
    else
    {
      inversible = k+1;
    }
  }
  return inversible;
}


void affiche_mat(float* mat, int N, int M)
{
  printf("\n");
  int i,j;
  for (i = 0; i < M; i++)
  {
    printf("%d␣␣␣",i);
    for (j = 0; j < N; j++)
    {
      printf("%f␣",mat[j*M+i]);
    }printf("\n");
  }printf("\n");
}
```

```
void affiche_mat_int(float* mat, int N, int M, int coeff)
{
  printf("\n");
  int i,j,value;
  for (i = 0; i < M; i++)
  {
    printf("%d␣␣␣",i);
    for (j = 0; j < N; j++)
    {
      value=(int)(mat[j*M+i]*coeff);
      printf("%d␣",value);
    }printf("\n");
  }printf("\n");
}
```

A.3.1.3 Classe de définition d'un milieu et de ses propriétés optiques

Listing A.12 – `Milieu.h`

```
#ifndef MILIEU_H
#define MILIEU_H

#include "../Functions/fonctions.h"

class Milieu
{
public:
  /** méthodes **/
  Milieu();
  Milieu(int indice, int tableau_lambda[], int nombre_lambda_p)
     ;
  // indice fait référence à celui donné dans le phantom qui
     est retranscrit dans le fichier indices_milieux.para
  // tableau_lambda est un tableau avec les valeurs des
     longueurs d'onde possibles, nombre_lambda_p est la taille
     du tableau
  Milieu(const Milieu &A); // constructeur de recopie
  ~Milieu();

  Milieu& operator= (const Milieu& A);
  void Affich_Data_Milieu();

  /** attributs **/
  int reference_milieu; // indice du phantom pour ce milieu
  int nombre_lambda;
```

```
  float indice_optique;
  float g;
  float** table_mu_complete; // table des coefficients d'
     absorption et de diffusion en fonction de la longueur d'
     onde
};

#endif
```

Listing A.13 – Milieu.cpp

```
#include "Milieu.h"


/** constructeurs et destructeur **/
Milieu::Milieu()
{
}

Milieu::Milieu(const Milieu &A)
{
this->reference_milieu = A.reference_milieu;
this->nombre_lambda = A.nombre_lambda;
this->indice_optique = A.indice_optique;
this->g = A.g;

this->table_mu_complete = new float*[this->nombre_lambda];
for(int i = 0; i < nombre_lambda; i++)
{
  this->table_mu_complete[i] = new float[9];
}

if(this->table_mu_complete == NULL){printf("----␣allocation␣
   problem␣in␣Milieu␣constructor");}

  for(int i=0; i<A.nombre_lambda; i++)
  {
    for(int j=0; j<9; j++)
    {
      this->table_mu_complete[i][j] = A.table_mu_complete[i][j
         ];
    }
  }
}
```

```
Milieu::Milieu(int indice, int tableau_lambda[], int
    nombre_lambda_p)
{
this->reference_milieu = indice;
this->nombre_lambda = nombre_lambda_p;
this->indice_optique = indice_opt(indice);
this->g = 0.9;

this->table_mu_complete = new float*[this->nombre_lambda];
for(int i = 0; i < this->nombre_lambda; i++)
{
  this->table_mu_complete[i] = new float[9];
}
/// pour chaque lambda, mu_s, mu_a, mu_a1, mu_a2, mu_a3, mu_a4,
    mu_a5, mu_a6, mu_a7

if(this->table_mu_complete == NULL){printf("----␣allocation␣
    problem␣in␣Milieu␣constructor");}

remplissage_mu(this->reference_milieu, this->table_mu_complete,
    this->nombre_lambda, tableau_lambda, this->g);

}

Milieu::~Milieu(void)
{
  for(int i = 0; i < this->nombre_lambda; i++)
  {
    delete[] this->table_mu_complete[i];
  }
delete[] this->table_mu_complete;
}


/** méthodes **/

Milieu& Milieu::operator= (const Milieu& A)
{
this->reference_milieu = A.reference_milieu;
this->nombre_lambda = A.nombre_lambda;
this->indice_optique = A.indice_optique;
this->g = A.g;
```

```
this->table_mu_complete = new float*[this->nombre_lambda];
for(int i = 0; i < this->nombre_lambda; i++)
{
  this->table_mu_complete[i] = new float[9];
}

if(this->table_mu_complete == NULL){printf("----␣allocation␣
   problem␣in␣Milieu␣constructor");}

  for(int i=0; i<A.nombre_lambda; i++)
  {
    for(int j=0; j<9; j++)
    {
      this->table_mu_complete[i][j] = A.table_mu_complete[i][j
         ];
    }
  }
return(*this);
}

void Milieu::Affich_Data_Milieu()
{
using namespace std;

cout << "milieu␣indice␣" << this->reference_milieu << endl;
cout << "indice␣optique␣=␣" << this->indice_optique << ",␣g␣=␣"
    << this->g << endl;
  for(int i=0; i<this->nombre_lambda; i++)
  {
    for(int j=0; j<9; j++)
    {
      cout << this->table_mu_complete[i][j] << "␣␣";
    }
    cout << endl;
  }
}
```

A.3.2 Modèle SP_3

A.3.2.1 Classe de définition d'une interface entre 2 objets de classe "Milieu"

Listing A.14 – `Interface.h`

```
#ifndef INTERFACE_H
#define INTERFACE_H
```

```
#include "../Functions/fonctions.h"
#include "./Milieu.h"

class Interface
{
public:

  /** méthodes **/

  Interface();
  Interface(Milieu* mil_int, Milieu* mil_ext);
  Interface(const Interface &A);
  ~Interface();

  void Affich_Data_Interface(); // affichage des données pour
     une interface
  void calcul_Rn();
  void calcul_Jn();
  void calcul_Xi();
  void calcul_Eij();
  void calcul_Jcoeff();
  void calcul_complet(); // reprend les 3 fonctions précédentes
      en 1 seul appel
  float acces_value(int sens, char* value, int lambda); //
     value est un string qui définit la variable à afficher ('
     A2' , 'R4' , 'E12')
  // sens indique dans quel sens on traverse l'interface

  /** attributs **/

  Milieu* mil_interne; // pointe vers un élément de la classe
     Milieu au lieu de faire une copie
  Milieu* mil_externe; // pointe vers un élément de la classe
     Milieu au lieu de faire une copie

  /// différentes tables de données pour l'interface
  // premier indice pour le sens
  // 0 = direct, 1 = indirect
  float table_Rn_Inter[2][12]; // calcul des Rn jusque ordre 12
  float table_Xi_Inter[2][6][3]; // calcul des Xi (max = F3 =
     [5][2])
  float table_Eij_Inter[2][2][2]; // détermine les coeff à la
     frontière, indépendant de lambda !!!!
```

```
  float table_Jn_Inter[2][6]; // calcul des Jn jusque ordre 6
  float table_Jcoeff_Inter[2][2];
                          /// lambda fixé à 5 mais pas forcément
                              vrai fonction mil_interne
};

#endif
```

Listing A.15 – Interface.cpp

```
#include "Interface.h"

/** constructeurs et destructeur **/
Interface::Interface()
{
}

Interface::Interface(const Interface &A)
{
for(int sens=0; sens<2; sens++)
{
  for(int j=0; j<12; j++)
  {
    this->table_Rn_Inter[sens][j] = A.table_Rn_Inter[sens][j];
  }

  for(int j=0; j<6; j++)
  {
    for(int i=0; i<3; i++)
    {
      this->table_Xi_Inter[sens][j][i] = A.table_Xi_Inter[sens
         ][j][i];
    }
  }

  for(int j=0; j<2; j++)
  {
        for(int i=0; i<2; i++)
        {
  this->table_Eij_Inter[sens][j][i] = A.table_Eij_Inter[sens][j
     ][i];
        }
  }
}
}
```

```
Interface::Interface(Milieu* mil_int, Milieu* mil_ext)
{
this->mil_interne = mil_int;
this->mil_externe = mil_ext;

this->calcul_complet();
}

Interface::~Interface(void)
{
}


/** méthodes **/
void Interface::calcul_Rn()
{
for(int sens=0; sens<2; sens++)
{
  float n_incid;
  float n_out;
  if(sens==0)
  {
    n_incid = this->mil_interne->indice_optique;
    n_out = this->mil_externe->indice_optique;
  }
  else
  {
    n_out = this->mil_interne->indice_optique;
    n_incid = this->mil_externe->indice_optique;
  }

  for(int ordre=0; ordre<12; ordre++)
  {
    this->table_Rn_Inter[sens][ordre] = R_n(n_incid, n_out,
       ordre+1);
  }
}
}

void Interface::calcul_Jn()
{
for(int sens=0; sens<2; sens++)
{
```

```
  this->table_Jn_Inter[sens][0] = -0.5*this->table_Rn_Inter[
     sens][0];
  this->table_Jn_Inter[sens][1] = -3*this->table_Rn_Inter[sens
     ][1]/2;
  this->table_Jn_Inter[sens][2] = (5*this->table_Rn_Inter[sens
     ][0] - 15*this->table_Rn_Inter[sens][2])/4;
  this->table_Jn_Inter[sens][3] = (21*this->table_Rn_Inter[sens
     ][1] - 35*this->table_Rn_Inter[sens][3])/4;
  this->table_Jn_Inter[sens][4] = (270*this->table_Rn_Inter[
     sens][2] - 27*this->table_Rn_Inter[sens][0] - 315*this->
     table_Rn_Inter[sens][4])/16;
  this->table_Jn_Inter[sens][5] = (770*this->table_Rn_Inter[
     sens][3] - 165*this->table_Rn_Inter[sens][1] - 693*this->
     table_Rn_Inter[sens][5])/16;
}
}

void Interface::calcul_Xi()
{
for(int sens=0; sens<2; sens++)
{
  /// Ai
  this->table_Xi_Inter[sens][0][0] = -this->table_Rn_Inter[sens
     ][0];
  this->table_Xi_Inter[sens][0][1] = (-9*this->table_Rn_Inter[
     sens][0] + 30*this->table_Rn_Inter[sens][2] - 25*this->
     table_Rn_Inter[sens][4])/4;
  this->table_Xi_Inter[sens][0][2] = (-225*this->table_Rn_Inter
     [sens][0] + 2100*this->table_Rn_Inter[sens][2] - 6790*this
     ->table_Rn_Inter[sens][4] + 8820*this->table_Rn_Inter[sens
     ][6] - 3969*this->table_Rn_Inter[sens][8])/64;
  /// Bi
  this->table_Xi_Inter[sens][1][0] = 3*this->table_Rn_Inter[
     sens][1];
  this->table_Xi_Inter[sens][1][1] = (63*this->table_Rn_Inter[
     sens][1] - 210*this->table_Rn_Inter[sens][3] + 175*this->
     table_Rn_Inter[sens][5])/4;
  this->table_Xi_Inter[sens][1][2] = (2475*this->table_Rn_Inter
     [sens][1] - 23100*this->table_Rn_Inter[sens][3] + 74690*
     this->table_Rn_Inter[sens][5] - 97020*this->table_Rn_Inter
     [sens][7] + 43659*this->table_Rn_Inter[sens][9])/64;
  /// Ci
  this->table_Xi_Inter[sens][2][0] = (-3*this->table_Rn_Inter[
     sens][0] + 5*this->table_Rn_Inter[sens][2])/2;
```

```
  this->table_Xi_Inter[sens][2][1] = this->table_Xi_Inter[sens
     ][2][0];
  this->table_Xi_Inter[sens][2][2] = (15*this->table_Rn_Inter[
     sens][0] - 70*this->table_Rn_Inter[sens][2] + 63*this->
     table_Rn_Inter[sens][4])/8;
  /// Di
  this->table_Xi_Inter[sens][3][0] = (3*this->table_Rn_Inter[
     sens][1] - 5*this->table_Rn_Inter[sens][3])/2;
  this->table_Xi_Inter[sens][3][1] = this->table_Xi_Inter[sens
     ][3][0];
  this->table_Xi_Inter[sens][3][2] = (-15*this->table_Rn_Inter[
     sens][1] + 70*this->table_Rn_Inter[sens][3] - 63*this->
     table_Rn_Inter[sens][5])/8;
  /// Ei
  this->table_Xi_Inter[sens][4][0] = (15*this->table_Rn_Inter[
     sens][0] - 70*this->table_Rn_Inter[sens][2] + 63*this->
     table_Rn_Inter[sens][4])/8;
  this->table_Xi_Inter[sens][4][1] = (-45*this->table_Rn_Inter[
     sens][0] + 285*this->table_Rn_Inter[sens][2] - 539*this->
     table_Rn_Inter[sens][4] + 315*this->table_Rn_Inter[sens
     ][6])/16;
  this->table_Xi_Inter[sens][4][2] = this->table_Xi_Inter[sens
     ][4][1];
  /// Fi
  this->table_Xi_Inter[sens][5][0] = this->table_Xi_Inter[sens
     ][3][2];
  this->table_Xi_Inter[sens][5][1] = (45*this->table_Rn_Inter[
     sens][1] - 285*this->table_Rn_Inter[sens][3] + 539*this->
     table_Rn_Inter[sens][5] - 315*this->table_Rn_Inter[sens
     ][7])/16;
  this->table_Xi_Inter[sens][5][2] = this->table_Xi_Inter[sens
     ][5][1];
}
}

void Interface::calcul_Eij() /// Eij
{
for(int sens=0; sens<2; sens++)
{
  /// sélection du milieu d'origine
  Milieu* mil_inter;
  if(sens==0) mil_inter=this->mil_interne;
  else mil_inter=this->mil_externe;
```

```
  int nombre_longueur_donde = mil_inter->nombre_lambda;

  if(nombre_longueur_donde>5)
  {
    printf("problème : spectre du milieu plus large que celui
       prévu pour les Eij\n--calcul interrompu\n");
    break;
  }

  float coeff = ((1+this->table_Xi_Inter[sens][1][0])*(1+this->
     table_Xi_Inter[sens][1][1])-(21*this->table_Xi_Inter[sens
     ][3][0]*this->table_Xi_Inter[sens][3][1]))/21;;
  this->table_Eij_Inter[sens][0][0] = ((0.125+this->
     table_Xi_Inter[sens][2][1])*7*this->table_Xi_Inter[sens
     ][3][0] - (0.5+this->table_Xi_Inter[sens][0][0])*(1+this->
     table_Xi_Inter[sens][1][1]))/(7*coeff);
  this->table_Eij_Inter[sens][0][1] = ((0.125+this->
     table_Xi_Inter[sens][2][0])*(1+this->table_Xi_Inter[sens
     ][1][1]) - (7.0/24+this->table_Xi_Inter[sens][0][1])*7*
     this->table_Xi_Inter[sens][3][0])/(7*coeff);
  this->table_Eij_Inter[sens][1][0] = ((0.125+this->
     table_Xi_Inter[sens][2][1])*(1+this->table_Xi_Inter[sens
     ][1][0]) - (0.5+this->table_Xi_Inter[sens][0][0])*3*this->
     table_Xi_Inter[sens][3][1])/(3*coeff);
  this->table_Eij_Inter[sens][1][1] = ((0.125+this->
     table_Xi_Inter[sens][2][0])*3*this->table_Xi_Inter[sens
     ][3][1] - (7.0/24+this->table_Xi_Inter[sens][0][1])*(1+
     this->table_Xi_Inter[sens][1][0]))/(3*coeff);
}
}

void Interface::calcul_Jcoeff() /// Jcoeff
{
for(int sens=0; sens<2; sens++)
{
  /// sélection du milieu d'origine
  Milieu* mil_inter;
  if(sens==0) mil_inter=this->mil_interne;
  else mil_inter=this->mil_externe;

  int nombre_longueur_donde = mil_inter->nombre_lambda;

  if(nombre_longueur_donde>5)
  {
```

```
    printf("problème : spectre du milieu plus large que celui
       prévu pour les Eij\n--calcul interrompu\n");
    break;
  }

  this->table_Jcoeff_Inter[sens][0] = 0.25 + this->
     table_Jn_Inter[sens][0] - (0.5+this->table_Jn_Inter[sens
     ][1])*this->table_Eij_Inter[sens][0][0]/3 - this->
     table_Jn_Inter[sens][3]*this->table_Eij_Inter[sens
     ][1][0]/7;
  this->table_Jcoeff_Inter[sens][1] = -1.0/16 - 2*this->
     table_Jn_Inter[sens][0]/3 + this->table_Jn_Inter[sens
     ][2]/3 - (0.5+this->table_Jn_Inter[sens][1])*this->
     table_Eij_Inter[sens][0][1]/3 - this->table_Jn_Inter[sens
     ][3]*this->table_Eij_Inter[sens][1][1]/7;

}
}

void Interface::calcul_complet()
{
this->calcul_Rn();
this->calcul_Jn();
this->calcul_Xi();
this->calcul_Eij();
this->calcul_Jcoeff();
}

float Interface::acces_value(int sens, char* value, int lambda)
{
float resultat=0;
char lettre = value[0];
int size = strlen(value)-1;
int indice;

if(size==1) indice = int(value[2])-49;
else indice = (int(value[1])-49)*10 + int(value[2])-49;

if((size==1) || (lettre == 'R'))
{
  if(lettre == 'R')
  {
    resultat = this->table_Rn_Inter[sens][indice];
  }
```

```
  else
  {
    resultat = this->table_Xi_Inter[sens][int(lettre)-65][
       indice];
  }
}
else if((size==2) && (lettre == 'E'))
{
  resultat = this->table_Eij_Inter[sens][int(value[1])-49][int(
     value[2])-49];
}
return(resultat);
}

void Interface::Affich_Data_Interface()
{
using namespace std;

  for(int sens=0; sens<2; sens++)
  {
    if(sens==0) cout << "---␣sens␣direct␣---" << endl;
    else cout << "---␣sens␣indirect␣---" << endl;

    cout << "Rn␣:" << endl;
    for(int j=0; j<12; j++)
    {
      cout << this->table_Rn_Inter[sens][j] << "␣␣";
    }

    cout << endl << endl;
        cout << "Jn␣:" << endl;
    for(int j=0; j<6; j++)
    {
      cout << this->table_Jn_Inter[sens][j] << "␣␣";
    }

    cout << endl << endl;
        cout << "Jcoeff␣:" << endl;
        for(int j=0; j<2; j++)
        {
          cout << this->table_Jcoeff_Inter[sens][j] << "␣␣";
        }
        cout << endl << endl;
    cout << "Xi␣:" << endl;
```

```
    for(int j=0; j<6; j++)
    {
      for(int i=0; i<3; i++)
      {
        cout << this->table_Xi_Inter[sens][j][i] << "␣␣";
      }
      cout << endl;
    }

    cout << endl << endl;
    cout << "Eij␣:" << endl;

        for(int j=0; j<2; j++)
        {
          for(int i=0; i<2; i++)
          {
                cout << this->table_Eij_Inter[sens][j][i] << "␣
                   ␣";
          }
          cout << endl;
        }

  }
}
```

A.3.2.2 Calcul des matrices du systèmes SP_3 pour un maillage donné

Listing A.16 – `loadNodeDataGmsh.cpp`

```
#include "parameters.h"

int loadNodeDataGmsh(double nodeData[][2])
{
  /*FILE *reader;

  reader = fopen(PathResults PathExpe "imagesCT/" file , "rt");
  if(reader == NULL){printf("échec d'ouverture du fichier msh
     pour chargement node data\n\n"); fclose(reader); return
     (-1);}
  */
  std::ifstream dataFile(PathResults PathExpe FileMsh);
  std::string reader;

  if(dataFile.is_open())
```

```
{
        while(!dataFile.eof())
        {
          std::getline(dataFile,reader);

          if(reader.compare("$NodeData") == 0)
          {
                int number=0;
                std::string trash;
                int number_data=0;
                int number_data_int=0;
                float tra=0;

                dataFile >> number; //string tags
                for(int i=0;i<number+1;i++)
                {
                  getline(dataFile, trash);
                }
                dataFile >> number; //float tags
                for(int i=0;i<number+1;i++)
                {
                  getline(dataFile, trash);
                }
                dataFile >> number; //int tags
                for(int i=0;i<number-1;i++)
                {
                  getline(dataFile, trash);
                }
                dataFile >> number_data >> number_data_int; //
                   data node
                int id;
                std::cout << number_data_int << std::endl;
                for(int i=0;i<number_data_int;i++)
                {
                  dataFile >> id;
                  //std::cout << id << std::endl;
                  dataFile >> nodeData[id-1][1];
                  nodeData[id-1][0] = 1;

                  if(number_data>1)
                  {getline(dataFile, trash);
                  }
                }
                dataFile.close();
```

```
                return(0);
            }
        }
  dataFile.close();
  }
  else
  {std::cout << "problème ouverture fichier pour node data" <<
     std::endl;}
}
```

Listing A.17 – SPn_hetero.cpp

```
// -*- coding: utf-8; mode: c++; tab-width: 4; indent-tabs-mode
   : nil; c-basic-offset: 4 -*- vim:fenc=utf-8:ft=tcl:et:sw=4:
   ts=4:sts=4
#include <feel/feel.hpp>

#include "../../../tool/Functions/parameters.h"
#include "../../../tool/Functions/fonctions.cpp"
#include "../../../tool/Functions/loadNodeDataGmsh.cpp"
#include "../../../tool/Class/Milieu.cpp"
#include "../../../tool/Class/Interface.cpp"

int main(int argc, char**argv )
{
    //# marker1 #
    using namespace Feel;
        Environment env( _argc=argc, _argv=argv,
                     _desc=feel_options(),
                     _about=about(_name="SPn_hetero",
                                  _author="Bertrand Arnaud",
                                  _email="Arnaud.Bertrand@iphc.
                                     cnrs.fr"));
    //# endmarker1 #

    /** ****************** **/
    /** partie parametrage **/
    /** ****************** **/


    /** choix du maillage **/

        /// Informations du maillage, le fichier est désigné
           dans le .cfg du nom du programme mais il faut le
           définir dans pathData.h aussi !
```

```
    int Dim = 3;
auto mesh = loadMesh(_mesh = new Mesh<Simplex<3>>);

/** selection interpolation maillage **/
auto Vh = Pch<1>( mesh ); // ligne de choix de la methode d
   'interpolation du maillage
auto u = Vh->element(); // declaration
auto v = Vh->element(); // declaration

    /// Prépare les vecteurs associés à chaque coefficient
    auto mu_a = Vh->element();
    auto mu_a1 = Vh->element();
    auto mu_a2 = Vh->element();
    auto mu_a3 = Vh->element();

    /// Prépare le chargement des infos sur l'inhomogénéité
    int number_node = int(u.size());
    std::cout << number_node << std::endl;
    double nodedata[number_node][2];

    for(int i=0;i<number_node;i++)
    {
      for(int j=0;j<2;j++)
      {
            nodedata[i][j] = -1;
      }
    }

    /// Charge les infos sur l'inhomogénéité à partir du
       fichier, celui-ci est défini dans pathData.h !!!
    loadNodeDataGmsh(nodedata);
Vh->dof()->pointIdToDofRelation("feelpp2msh.m");

    for ( auto const& elt : elements(mesh) )
{
    for ( uint16_type p = 0; p< elt.nPoints();++p )
    {
        auto const& thept = elt.point(p);
        size_type idxNode = thept.id();

                    ///recherche si il y a une valeur
                       associée, 0 par défaut
                    double val=1;
                    if(nodedata[idxNode-1][0] >= 0)
```

```
                        {
                          val = nodedata[idxNode-1][1];
                        }

            size_type i = Vh->dof()->localToGlobal( elt,p,0 ).
               index();

            // check relation between dof and mesh node
            for ( int d=0; d<Dim;++d )
                CHECK( std::abs( Vh->dof()->dofPoint(i).get
                   <0>()[d]-thept[d] )< 1e-9 ) << "invalid␣
                   relation␣dof/node";
            u.set( i, val );
                        v.set( i, val );
        }
    }

    printf("base␣du␣maillage␣:␣done\n");

    /** ***************************** **/
    /** systeme d'equation methode SPn **/
    /** ***************************** **/

    /// pour un point donne
          /// constantes du milieu
          /// constantes du systeme

    int nombre_milieux = read_nb_milieux(); // air compris

    // lecture dans fichier indice_milieux.para
    // le fichier informe des milieux presents dans le phantom
       afin d'obtenir un indice qui permet de calculer les
       coeff mu ensuite

    int nombre_interfaces = 1; /// a determiner avec le phantom
        par la suite !!!
    // chaque interface peut être prise dans les 2 sens
    // seules les combinaisons possibles dans le phantom sont
       prises !!!

    /// Information sur les longueurs d'onde à utiliser, ne pas
        dépasser 5 !
    int nombre_longueur_donde = 5;
    int table_longueur_donde[5] = {637,680,735,780,815};
```

```
    /// Remplissage automatique des données utiles
Milieu table_milieux[nombre_milieux];

for(int i=0; i<nombre_milieux; i++)
{
  table_milieux[i] = Milieu(i+1, table_longueur_donde,
     nombre_longueur_donde);
}

for(int i=0; i<nombre_milieux; i++)
{
  table_milieux[i].Affich_Data_Milieu();
}

/** prise en compte de l'interface majeure, celle de
   contact avec l'air **/

int milieu_externe = 1;

Interface mouse_air(&table_milieux[milieu_externe], &
   table_milieux[0]);
//mouse_air.Affich_Data_Interface();

//int table_interfaces[nombre_interfaces][2];
/// prendre une convention pour le sens des interfaces,
   peut être pas nécessaire mais utile pour la
   compréhension
/// pour l'instant, plus faible indice en 1er
/// par la suite, il se peut que les milieux soient
   ordonnés de l'extérieur vers l'intérieur (indice
   croissant)

//Interface* table_interfaces = new Interface[
   nombre_interfaces];

/** mise en place du système matriciel multispectral **/

float E11, E12, E21, E22;

// nom des matrices pour le multi-spectral
    std::string Mat_B (PathResults PathExpe "data_optic/
       opt_volume/matrix_B_");
```

```
        std::string Mat_11 (PathResults PathExpe "data_optic/
           opt_volume/matrix_11_");
        std::string Mat_12 (PathResults PathExpe "data_optic/
           opt_volume/matrix_12_");
        std::string Mat_21 (PathResults PathExpe "data_optic/
           opt_volume/matrix_21_");
        std::string Mat_22 (PathResults PathExpe "data_optic/
           opt_volume/matrix_22_");

    for(int lambda=0; lambda<nombre_longueur_donde; lambda++)
    {
      E11 = mouse_air.table_Eij_Inter[0][0][0];
      E12 = mouse_air.table_Eij_Inter[0][0][1];
      E21 = mouse_air.table_Eij_Inter[0][1][0];
      E22 = mouse_air.table_Eij_Inter[0][1][1];

          for ( auto const& elt : elements(mesh) )
    {
        for ( uint16_type p = 0; p< elt.nPoints();++p )
        {
            auto const& thept = elt.point(p);
            size_type idxNode = thept.id();

                        ///recherche si il y a une valeur
                           associée qui correspond au milieu, 1
                            par défaut
                        double val=1;
                        if(nodedata[idxNode-1][0] >= 0)
                        {
                          val = nodedata[idxNode-1][1];
                        }

            size_type i = Vh->dof()->localToGlobal( elt,p,0 ).
               index();

            // check relation between dof and mesh node
            for ( int d=0; d<Dim;++d )
                CHECK( std::abs( Vh->dof()->dofPoint(i).get
                   <0>()[d]-thept[d] )< 1e-9 ) << "invalid␣
                   relation␣dof/node";

                        /// Affecte les valeurs du milieu pour
                           le point associé, milieu 1 par
                           défaut
```

```
                        mu_a.set( i, table_milieux[int(val)].
                           table_mu_complete[lambda][1] );
                        mu_a1.set( i, table_milieux[int(val)].
                           table_mu_complete[lambda][2] );
                        mu_a2.set( i, table_milieux[int(val)].
                           table_mu_complete[lambda][3] );
                        mu_a3.set( i, table_milieux[int(val)].
                           table_mu_complete[lambda][4] );
    }
}

  /** fin initialisation **/

  //printf("début calcul des matrices\n");

  /** membre de droite **/
  auto B = form2( _trial=Vh, _test=Vh, _init=true ) =
     integrate(_range=elements(mesh),
                 _expr=idt(u)*id(v));
  /** fin membre de droite **/

  int ind = 1;

  /// M11
  auto M11 = form2( _trial=Vh, _test=Vh, _init=true ) =
     integrate(_range=elements(mesh),
                  _expr = gradt(u)*trans(grad(v))/(3*idv(
                     mu_a1))
                  + idt(u)*id(v)*(idv(mu_a)) ); //
                     expression ici
  M11 += integrate(_range=boundaryfaces(mesh), _expr = -idt
     (u)*id(v)*E11/3 );

  /// M12
  auto M12 = form2( _trial=Vh, _test=Vh, _init=true ) =
     integrate(_range=elements(mesh),
                  _expr = -idt(u)*id(v)*2*idv(mu_a)/3 ); //
                      expression ici

  M12 += integrate(_range=boundaryfaces(mesh), _expr = -idt
     (u)*id(v)*E12/3 );

  /// M21
  auto M21 = form2( _trial=Vh, _test=Vh, _init=true ) =
```

```
            integrate(_range=elements(mesh),
                      _expr = -idt(u)*id(v)*2*idv(mu_a)/3 ); //
                          expression ici

        M21 += integrate(_range=boundaryfaces(mesh), _expr = -idt
            (u)*id(v)*E21/7 );

        /// M22
        auto M22 = form2( _trial=Vh, _test=Vh, _init=true ) =
            integrate(_range=elements(mesh),
                      _expr = gradt(u)*trans(grad(v))/(7*idv(
                          mu_a3))
                      + idt(u)*id(v)*(4*idv(mu_a) + 5*idv(mu_a2
                          ))/9 ); // expression ici

        M22 += integrate(_range=boundaryfaces(mesh), _expr=-idt(u
            )*id(v)*E22/7 );

        //printf("fin calcul\n");

            std::string lamm (std::to_string(lambda+1) + ".m");

        B.matrix().printMatlab(Mat_B + lamm); //mat_b_name);
        M11.matrix().printMatlab(Mat_11 + lamm);
        M12.matrix().printMatlab(Mat_12 + lamm);
        M21.matrix().printMatlab(Mat_21 + lamm);
        M22.matrix().printMatlab(Mat_22 + lamm);




          }

    printf("fin␣export␣matlab\n");

        auto e = exporter( _mesh=mesh, _name="mesh_feel" );
    e->add( "u", u);
    e->save();
    return 0;
}
```

A.4 Recalage et Reconstruction flux surfacique

A.4.0.3 Script effectuant le recalage et la reconstruction du flux optique en surface du maillage recalé

Listing A.18 – `proj_det_mesh_multi.m`

```
clear;

%%
load im_final_20150408_souris_172_23_tumeur_ATP32_ss20-2;
load pts_ref_20150408_souris_172_23_tumeur_ATP32;

file_save = '20150408_souris_172_23_coarse_abdomen_CT_tsup_ss20
   -2_';

lambda_used_ = [6];
images_used = [3]; %GHBD

%% charge les données du maillage

[node, face, elem, pts_surface, doftopid] = read_msh('.\Data\
   Manip_cerenkov_20150408\Souris\172_23\mesh\coarse\','
   mesh_feel-1_0','feelpp2msh_doftopid');

face = unique(face,'rows');
elem = unique(elem,'rows');
number_nodes = size(node,1);
number_face = size(face,1);
number_elem = size(elem,1);
nb_points_surf = size(pts_surface,1);
%% Recalage
node(:,1:3) = transform_mesh(node(:,1:3), P, P_opt_); %
   Recalage du maillage à partir des données TDM et optique

minode = min(node)

%% Reconstruction flux surfacique
nb_images = max(size(im_final));

b_y = [zones{3}(2) , zones{3}(2)+zones{3}(4) , zones{1}(2) ,
   zones{1}(2)+zones{1}(4) ,zones{2}(2) , zones{2}(2)+zones
   {2}(4) ,zones{4}(2) , zones{4}(2)+zones{4}(4)];
b_x = [zones{3}(1) , zones{3}(1)+zones{3}(3) ,zones{1}(1) ,
   zones{1}(1)+zones{1}(3) ,zones{2}(1) , zones{2}(1)+zones
```

```
   {2}(3) ,zones{4}(1) , zones{4}(1)+zones{4}(3)];


for im_=lambda_used_

    center_im = [588,453];
    tic;
    [p2f, list_pixel] = pixel2face(im_final{im_},center_im,node
       ,face, 0, images_used, b_x, b_y);
    toc;

    if isempty(list_pixel)
        disp('no data reconstructed');
        continue;
    end

    vec_value = zeros(size(list_pixel,1),1);
    for i=1:size(list_pixel,1)
        vec_value(i) = im_final{im_}(list_pixel(i,2),list_pixel
           (i,1));
    end

    face_value = p2f*vec_value;

    for i=1:number_face
        if face_value > 0
            face_value(i) = face_value(i)/size(find(p2f(i,:))
               ,2);
        end
    end


    clear vec_value i;
    save_J_plus_msh([file_save num2str(images_used) '_' num2str
       (im_)], node1, face, elem, pts_surface, zeros(
       number_nodes,1), face_value);

end
```

A.4.0.4 Fonction calculant la matrice de recalage

Listing A.19 – `transform_mesh.m`

```
function [node_f, transform, transform_1] = transform_mesh(
```

```
   node_i, pts_i, pts_f)

% input :
%     - pts_i : liste de points du repère initial
%     - pts_f : liste de pts du repère final correspondant aux
   mêmes points
%     - node_i : matrice de points du mailage initial
% output :
%     - node_f : matrice de points du maillage après
   transformation
%     - transform : matrice de transformation pour changer de
   référentiel
%     - transform_1 : inverse de transform

%% points initiaux et vecteurs corespondants

P1_opt = pts_f{1};
P2_opt = pts_f{2};
P3_opt = pts_f{3};

vec_opt = P2_opt-P1_opt;
vec_opt_2 = P3_opt-P1_opt;

clear P1_opt P2_opt P3_opt i;

%% chargement des données CT

vec_CT = pts_i{2}-pts_i{1};
vec_CT_2 = pts_i{3}-pts_i{1};

% norm(vec_opt)/norm(vec_CT)
% norm(vec_opt_2)/norm(vec_CT_2)

%% calcul de la transformation

[k_homo, transform_, transform_2] = recalage_3D(vec_CT, vec_opt
   , vec_CT_2, vec_opt_2);
transform = k_homo*transform_2*transform_;

clear vec_opt vec_opt_2 P1 P2 P3 vec_CT vec_CT_2;

number_nodes = size(node_i,1);

% recalage good !
```

```
node_i = node_i';
for i=1:number_nodes
    node_i(:,i) = transform_2*transform_*(node_i(:,i) - pts_i
       {1});
end

%node = transform*node;

for i=1:number_nodes
    node_i(:,i) = node_i(:,i) + pts_f{1};
end
node_f = node_i';
clear i;
```

A.4.0.5 Fonction calculant la matrice de recalage à partir de 3 points

Listing A.20 – recalage_3D.m

```
function [k_homo, mat_recalage, mat_recalage_2]=recalage_3D(
   vec_1, vec_1_ref, vec_2, vec_2_ref)

%% 1er recalage
k_homo = norm(vec_1_ref)/norm(vec_1);

N_vec = cross(vec_1,vec_1_ref);

sin_phi_vec = norm(cross(vec_1_ref,vec_1))/(norm(vec_1_ref)*
   norm(vec_1));
cos_phi_vec = sign(dot(vec_1,vec_1_ref))*sqrt(1-sin_phi_vec*
   sin_phi_vec);

N_vec = N_vec/norm(N_vec);

mat_cross = [0 , -N_vec(3) , N_vec(2) ;...
             N_vec(3) , 0 , -N_vec(1) ;...
             -N_vec(2) , N_vec(1) , 0];

rota = cos_phi_vec*eye(3) + (1-cos_phi_vec)*(N_vec'*N_vec) +
   sin_phi_vec*mat_cross;

mat_recalage = rota;

clearvars -except k_homo mat_recalage vec_1_ref vec_2 vec_2_ref
   ;
```

```
%% 2nd recalage

mat_recalage_2 = eye(3);

if (exist('vec_2', 'var') && exist('vec_2_ref', 'var'))
    vec_2 = (k_homo*mat_recalage*vec_2')';

    k_homo_tmp = norm(vec_2_ref)/norm(vec_2);

    N_vec = vec_1_ref/norm(vec_1_ref);

    vec_tmp_2 = vec_2 - dot(vec_2,N_vec).*N_vec;
    vec_tmp_2_ref = vec_2_ref - dot(vec_2_ref,N_vec).*N_vec;

    N_vec = cross(vec_tmp_2,vec_tmp_2_ref);
    N_vec = N_vec/norm(N_vec);

    sin_phi_vec = norm(cross(vec_tmp_2_ref,vec_tmp_2))/(norm(
       vec_tmp_2_ref)*norm(vec_tmp_2));
    cos_phi_vec = sign(dot(vec_tmp_2,vec_tmp_2_ref))*sqrt(1-
       sin_phi_vec*sin_phi_vec);

    mat_cross = [0 , -N_vec(3) , N_vec(2) ;...
                 N_vec(3) , 0 , -N_vec(1) ;...
                 -N_vec(2) , N_vec(1) , 0];

    rota = cos_phi_vec*eye(3) + (1-cos_phi_vec)*(N_vec'*N_vec)
       + sin_phi_vec*mat_cross;

    mat_recalage_2 = rota;

    k_homo = k_homo*(1+k_homo_tmp)/2;
end

clearvars -except k_homo mat_recalage mat_recalage_2;
```

A.4.0.6 Fonctions permettant la correspondance entre une face et les pixels qui observent cette face

Listing A.21 – pixel2face.m

```
function [matrix,conv_p_matrix] = pixel2face(image, center_im,
   node, face, seuil, varargin)
```

```
lim_min = min(node)-10;
lim_max = max(node)+10;

lim_min_z = lim_min(3);
lim_max_z = lim_max(3);
lim_min_x = lim_min(1);
lim_max_x = lim_max(1);

nb_face = size(face,1);

center = cell(nb_face,1);
voisins = zeros(nb_face,4);

for face_treated = 1:nb_face
    points_mat = [node(face(face_treated,1),1:3)',node(face(
       face_treated,2),1:3)',node(face(face_treated,3),1:3)'];
    center{face_treated} = find_center_N_points(points_mat);
    voisins(face_treated,1:4) = find_faces_voisines(face,
       face_treated);
end

clear points_mat face_treated;

load det_info;

% input :
%     - image : image de référence pour mettre en place le
   système, matrice
%     contenant les valeurs en ph/s/cm^2/sr
%     - det_param : paramètres des détecteurs pour visibilité
%     - node,face : info du maillage surfacique
%     - bornes : limite des pixels pris en compte pour limiter
   le nombre de points
%
% output :
%     - matrix : matrice de conversion énergétique entre pixel
   et face
%     - conv_p_matrix : organisation des pixel pour passage en
   argument

% REMARQUE :
    % Dans cette fonction, le repère 3D choisit est le repère
       absolu de
    % référence pour le P-I
```

```
xc = center_im(1);
yc = center_im(2);

nb_arg = size(varargin,2);
if nb_arg
    im_used = varargin{1};
end

if nb_arg > 1 && nb_arg < 4
    bornes_x = varargin{2};
    if nb_arg == 3
        bornes_y = varargin{3};
    else
        bornes_y = [1,752,1,752,1,752,1,752];
    end
else
    bornes_x = [1,200,201,500,501,800,801,1000];
    bornes_y = [1,752,1,752,1,752,1,752];
end

order = [3,1,2,4];

clear nb_p_max;

nb_p_max = 0;
for detector=1:4
    x1 = floor(bornes_x(2*detector-1));
    x2 = floor(bornes_x(2*detector));

    y1 = floor(bornes_y(2*detector-1));
    y2 = floor(bornes_y(2*detector));

    nb_p_max = nb_p_max + (y2-y1)*(x2-x1);
end

%% choose only pixel with signal

% care to im_used !!!
% par défaut, HBGD
% ici, il faut GHBD étant donné la conversion via 'order'
%
```

```
%im_used = [im_used(3),im_used(1:2),im_used(4)];

num_pixel = 0;
nb_pix_det = zeros(4,1);
for detector=im_used
    x1 = floor(bornes_x(2*detector-1));
    x2 = floor(bornes_x(2*detector));

    y1 = floor(bornes_y(2*detector-1));
    y2 = floor(bornes_y(2*detector));

    for x = x1:x2
        for y = y1:y2
            if image(y,x) > seuil
                num_pixel = num_pixel + 1;
                indice_pixel(num_pixel,:) = [x,y];
            end
        end
    end
    nb_pix_det(detector) = num_pixel;
end
num_pixel

if ~exist('indice_pixel','var')
    matrix = [];
    conv_p_matrix = [];
    return;
end
%% boucle de calcul

matrix = sparse(nb_face,num_pixel);

surf_pix = 0.0625e6; % dépend de la hauteur à laquelle on se
   place, en m^2 avec un facteur 1e6 pour avoir un résultat en
   ph/s/mm^2

low = 0;
for detector=im_used
    disp('detect␣begin␣!');
    count = 0;
    center_det = [det_para{order(detector)}(1,1) , 0 , det_para
       {order(detector)}(1,2)];
    vec_det = [det_para{order(detector)}(2,1) , 0 , det_para{
       order(detector)}(2,2)];
```

```
high = nb_pix_det(detector);
face_vis = cell(5,1);
face_vis{2} = -1;

for indice_pix = (low+1):high
    x = indice_pixel(indice_pix,1);
    y = indice_pixel(indice_pix,2);

    [pt_det, delta] = droite(det_para, order(detector),
       coeff, x-xc, yc-y);
    pt_det = [pt_det(1), pt_det(3), pt_det(2)];
    delta = [delta(1), delta(3), delta(2)];

    %% test quick
    v_test = 0;
    if detector == 1
        inter_test = droite_intersect_plan(pt_det, delta,
           [1,0,0], lim_max_x);
        for i=2:3
            if inter_test(i) < lim_min(i) || inter_test(i)
               > lim_max(i)
                v_test = 1;
            end
        end
    end

    if detector == 2
        inter_test = droite_intersect_plan(pt_det, delta,
           [0,0,1], lim_min_z);
        for i=1:2
            if inter_test(i) < lim_min(i) || inter_test(i)
               > lim_max(i)
                v_test = 1;
            end
        end
    end

    if detector == 3
        inter_test = droite_intersect_plan(pt_det, delta,
           [0,0,1], lim_max_z);
        for i=1:2
            if inter_test(i) < lim_min(i) || inter_test(i)
               > lim_max(i)
```

```
            v_test = 1;
        end
    end
end

if detector == 4
    inter_test = droite_intersect_plan(pt_det, delta,
       [1,0,0], lim_min_x);
    for i=2:3
        if inter_test(i) < lim_min(i) || inter_test(i)
           > lim_max(i)
            v_test = 1;
        end
    end
end

if v_test
    continue;
end
%% end test quick

% find face visible, index
%pause;

if isempty(face_vis{2}) || face_vis{2} <= 0
    prior = [1];
else
    prior = voisins(face_vis{2},:);
end

face_vis = pixel_origin(pt_det, delta, node, face,
   center, prior);

if face_vis{1}<0 || face_vis{5} < 0.1
    count = count+1;
    continue;
end

% cos_theta_det à déterminer aussi

sin_theta_det = cross(delta,vec_det);
sin_theta_det = norm(sin_theta_det)/(norm(delta)*norm(
   vec_det));
cos_theta_det = sqrt(1-sin_theta_det^2);
```

```
        % find parameters

        dist = face_vis{1};
        index = face_vis{2};
        cos_theta_face = face_vis{5}; % theta angle par rapport
            à la normale de la face
        surf_face = aire_triangle(node(face(index,1),1:3),node(
           face(index,2),1:3),node(face(index,3),1:3))*1e-6;

        % calcul coeff

        value = pi*(0.001*dist)^2/(surf_pix*surf_face*
           cos_theta_det*cos_theta_face);

        matrix(index,indice_pix) = value;

        clear table_all_data num_face data_pix;
        %disp('pix done');
    end
    low = high;
    count
end

conv_p_matrix = indice_pixel;
```

Listing A.22 – pixel_origin.m

```
function [all_data_face] = pixel_origin(point_det, vec_pixel,
   node, face, center, faces_prioritaires)

% point_det
% vec_pixel
% point_det + vec_pixel*360

vec_pixel = vec_pixel/max(abs(vec_pixel));

% input :
%   - point_det : le centre du détecteur pour tracer le rayon
   jusque la
%   face
%   - vec_pixel : direction du rayon arrivant sur le pixel
%   - node : liste des points du maillage
%   - face : liste des faces du maillage
%   - center : cellule contenant le centre pour chaque face
```

```
%
% output :
%     - all_data : cellule contenant :
%             - la distance de la face par rapport au det si
   visible
%             - l'indice de la face
%             - le centre de la face
%             - les limites de la visibilité de la face
%             - le sinus de l'angle entre la face et le rayon
   incident

% REMARQUE :
    % Dans cette fonction, le repère 3D choisit est le repère
       absolu de
    % référence pour le P-I

nb_faces = size(face,1);

all_data_face = cell(1,5);
all_data_face{1} = -1;

points = cell(3,1);



if all_data_face{1} < 0
    for face_treated = 1:nb_faces

        v_test = dist_pt_droite(point_det,vec_pixel,center{
           face_treated}');
        if v_test > 1
            continue;
        end

        points_mat = [node(face(face_treated,1),1:3)',node(face
           (face_treated,2),1:3)',node(face(face_treated,3)
           ,1:3)'];
        visible = droite_intersect_triangle(point_det,
           vec_pixel, points_mat);

        %% travail préliminaire

         vec_center_norm = norm(center{face_treated}' -
            point_det);
```

```
        if visible == 1
            if (all_data_face{1} < 0) || (all_data_face{1} >
               vec_center_norm)

                points{1} = node(face(face_treated,1),1:3);
                points{2} = node(face(face_treated,2),1:3);
                points{3} = node(face(face_treated,3),1:3);

                all_data_face{1} = vec_center_norm;

                %% how much visible

                v_1 = points{2}-points{1};
                v_2 = points{3}-points{1};

                vec_n = cross(v_1,v_2);
                vec_n = vec_n/norm(vec_n);
                visible = sqrt(1-(norm(cross(vec_n,vec_pixel))/
                   norm(vec_pixel))^2);
                all_data_face{5} = visible;

                if visible < 0.1
                    continue;
                end

                all_data_face{2} = face_treated;
                all_data_face{3} = center{face_treated};
                bornes = 'useless';
                all_data_face{4} = bornes;
            end
        end
    end
end
```

A.5 Reconstruction source

A.5.1 Algorithme de reconstruction

A.5.1.1 Script de reconstruction

Listing A.23 – `recons_script.m`

```
clear;
delete('/tmp/matschur*.mat');
```

```
disp('
   ------------------------------------------------------------------------
   ');
load recons_option;

opt.quiet = 1;
opt.tolerance = 1;

name_mesh = 'mesh_feel-1_0';

path = './data/results_for_recons/example/'; % manip_27
   -05-2014/Souris18_crane/';
k_list = [5,10,30,50,100,200];

file_ = 'J_08_04_souris/20150408
   _souris_172_23_coarse_abdomen_tsup_ss20-2_12_'; % num=HGBD,
   finish automatically by "n.msh"
%_symz_top_bot

lambda_used = [1];
File_out = '20150408_souris_172_23_abdomen_g_0-9_mua_0-01
   _mus_1_tol_1_GH_1';
for i=k_list
    opt.k = i;
    reconstruction_Cerenkov(path, name_mesh, file_, lambda_used
       , opt, File_out);
end

lambda_used = [2];
File_out = '20150408_souris_172_23_abdomen_g_0-9_mua_0-01
   _mus_1_tol_1_GH_2';
for i=k_list
    opt.k = i;
    reconstruction_Cerenkov(path, name_mesh, file_, lambda_used
       , opt, File_out);
end

lambda_used = [3];
File_out = '20150408_souris_172_23_abdomen_g_0-9_mua_0-01
   _mus_1_tol_1_GH_3';
for i=k_list
    opt.k = i;
    reconstruction_Cerenkov(path, name_mesh, file_, lambda_used
       , opt, File_out);
```

```
end

lambda_used = [4];
File_out = '20150408_souris_172_23_abdomen_g_0-9_mua_0-01
    _mus_1_tol_1_GH_4';
for i=k_list
    opt.k = i;
    reconstruction_Cerenkov(path, name_mesh, file_, lambda_used
        , opt, File_out);
end

lambda_used = [5];
File_out = '20150408_souris_172_23_abdomen_g_0-9_mua_0-01
    _mus_1_tol_1_GH_5';
for i=k_list
    opt.k = i;
    reconstruction_Cerenkov(path, name_mesh, file_, lambda_used
        , opt, File_out);
end

lambda_used = [1,2];
File_out = '20150408_souris_172_23_abdomen_g_0-9_mua_0-01
    _mus_1_tol_1_GH_12';
for i=k_list
    opt.k = i;
    reconstruction_Cerenkov(path, name_mesh, file_, lambda_used
        , opt, File_out);
end

lambda_used = [1,2,3];
File_out = '20150408_souris_172_23_abdomen_g_0-9_mua_0-01
    _mus_1_tol_1_GH_123';
for i=k_list
    opt.k = i;
    reconstruction_Cerenkov(path, name_mesh, file_, lambda_used
        , opt, File_out);
end

lambda_used = [1,3];
File_out = '20150408_souris_172_23_abdomen_g_0-9_mua_0-01
    _mus_1_tol_1_GH_13';
for i=k_list
    opt.k = i;
    reconstruction_Cerenkov(path, name_mesh, file_, lambda_used
```

```
        , opt, File_out);
end

lambda_used = [2,3];
File_out = '20150408_souris_172_23_abdomen_g_0-9_mua_0-01
    _mus_1_tol_1_GH_23';
for i=k_list
    opt.k = i;
    reconstruction_Cerenkov(path, name_mesh, file_, lambda_used
        , opt, File_out);
end

lambda_used = [1,2,3,4];
File_out = '20150408_souris_172_23_abdomen_g_0-9_mua_0-01
    _mus_1_tol_1_GH_1234';
for i=k_list
    opt.k = i;
    reconstruction_Cerenkov(path, name_mesh, file_, lambda_used
        , opt, File_out);
end

lambda_used = [1,2,3,4,5];
File_out = '20150408_souris_172_23_abdomen_g_0-9_mua_0-01
    _mus_1_tol_1_GH_12345';
for i=k_list
    opt.k = i;
    reconstruction_Cerenkov(path, name_mesh, file_, lambda_used
        , opt, File_out);
end

delete('/tmp/matschur*.mat');
```

A.5.1.2 Fonction de reconstruction

Listing A.24 – reconstruction_Cerenkov.m

```
function [] = reconstruction_Cerenkov(Path_data, name_mesh,
    file_name_results_J_msh, lambda_used, opt, File_out,
    varargin)

number_lambda = max(lambda_used);
load_Mat_xx_(Path_data,number_lambda);

n_lambda = max(size(lambda_used));
number_lambda = n_lambda;
```

```
load_mesh_info(Path_data, name_mesh, 0); % nom des fichiers
   écrits en dur mais variable !!!

load_cste_milieu(Path_data,lambda_used); % To improve when
   possible

J_plus = load_J_msh(Path_data, file_name_results_J_msh,
   lambda_used);
%J_plus(J_plus > max(J_plus)/2) = J_plus(J_plus > max(J_plus)
   /2)*5;

%save plop_ J_plus

% init système matrix
Mat_sys = [];
for lambda=lambda_used
    Mat_sys = [Mat_sys ; forward_model_f(lambda)];
end

%% ls-nonneg

coeff_regular = (max(max(Mat_sys)))^2;

if isfield(opt,'k')
    if opt.k > 0
        coeff_regular = (max(max(Mat_sys))/opt.k)^2;
    elseif opt.k == 0
        coeff_regular = 0;
    end
end

tolerance = 1e0;
quiet = 0;

if isfield(opt,'tolerance')
    tolerance = opt.tolerance;
end
if isfield(opt,'quiet')
    quiet = opt.quiet;
end


tic;
```

```
vecteur_source = ones(size(Mat_sys,2),1);
for i=1:5
    [vecteur_source, status_ls] = l1_ls_nonneg(Mat_sys, J_plus,
        coeff_regular*(6-i), tolerance, quiet,1e-3,5000,
       vecteur_source);
end

toc;

if strcmp(status_ls,'Solved')
    disp('--- Reconstruction terminée ---');
else
    disp('------ Echec !!! ------');
end

%% save reconstruction

seuil = 0;

load mesh_info;
print_results_msh([Path_data 'results_recons/' File_out '_'
   num2str(opt.k) '_recons'],node,face,elem,vecteur_source,
   seuil);
```

Listing A.25 – load_cste_milieu.m

```
function [] = load_cste_milieu(Path,lambda_used)

n_lambda = max(size(lambda_used));
n_lambda = 5;

%% prise en compte des informations du maillage

load mesh_info;

%% dï¿½clarations des constantes pour initialisation

disp('--calcul des constantes');
tic;

% constantes systï¿½me d'ï¿½quations SP3

equations = 2;
```

```
% ici, on ne considï¿½re que les paramï¿½tres indï¿½pendants
   entre eux :
% ï¿½a, ï¿½s, g en mm-1, mm-1 et sans unitï¿½ respectivement

milieu_souris = zeros(n_lambda,3);
%% coeff test
%milieu_souris(1,:) = [1 3 0.9];
%% coeff milieu
milieu_souris(1,:) = [0.93 4.9 0.9];
milieu_souris(2,:) = [0.50 4.1 0.9];
milieu_souris(3,:) = [0.24 3.3 0.9];
milieu_souris(4,:) = [0.25 2.8 0.9];
milieu_souris(5,:) = [0.18 2.5 0.9];
%% coeff high
milieu_souris(1,:) = [0.61 4.3 0.9];
milieu_souris(2,:) = [0.37 3.8 0.9];
milieu_souris(3,:) = [0.29 2.9 0.9];
milieu_souris(4,:) = [0.19 2.6 0.9];
milieu_souris(5,:) = [0.19 2.3 0.9];
%% coeff low
milieu_souris(1,:) = [3.01 5.8 0.9];
milieu_souris(2,:) = [0.61 4.4 0.9];
milieu_souris(3,:) = [0.24 3.4 0.9];

%%

table_mua_i = zeros(n_lambda,4);

for a=1:n_lambda
    table_mua_i(a,:) = [milieu_souris(a,1) (milieu_souris(a,1)+
       milieu_souris(a,2)*(1-milieu_souris(a,3))) ...
    (milieu_souris(a,1)+milieu_souris(a,2)*(1-milieu_souris(a
       ,3)^2)) (milieu_souris(a,1)+milieu_souris(a,2)*(1-
       milieu_souris(a,3)^3))];
end

clear a;

        %% constantes pour frontiï¿½re

% Rn dï¿½finis pour l'interface souris -> air, Delta n = 0.35
Rn = [0.294336 0.147677 0.0845207 0.052846 0.0355117
   0.0254459];
```

```
Xi = zeros(2*equations,equations);
% table des Ai, Bi, Ci, Di (Xi)
% pour l'instant, seulement 4 paramï¿½tres nï¿½cessaires
% pour l'interface finale (sortie vers l'air), on a les coeff
   Ji

Xi = [-Rn(1) , -9/4*Rn(1)+15/2*Rn(3)-25/4*Rn(5) ; ...
      3*Rn(2) , 63/4*Rn(2)-105/2*Rn(4)+175/4*Rn(6) ; ...
      -3/2*Rn(1)+5/2*Rn(3) , -3/2*Rn(1)+5/2*Rn(3) ; ...
      3/2*Rn(2)-5/2*Rn(4) , 3/2*Rn(2)-5/2*Rn(4) ];


        %% Constantes pour systï¿½me linï¿½aire fonction
           milieux/interfaces

%2010 Kai Liu OE SPn approximation

E=zeros(2,2);

% Attention !!!
% les c et a sont dï¿½finis arbitrairement
% le c est un indice sur les diffï¿½rentes interfaces  en
   suivant l'ordre de
% la convention

    E(1,1) = ((1/8+Xi(3,2))*Xi(4,1) - (1/2+Xi(1,1))*(1+Xi(2,2))
       /7);
    E(1,2) = ((1/8+Xi(3,1))*(1+Xi(2,2))/7 - (7/24+Xi(1,2))*Xi
       (4,1));
    E(2,1) = ((1/8+Xi(3,2))*(1+Xi(2,1))/3 - (1/2+Xi(1,1))*Xi
       (4,2));
    E(2,2) = ((1/8+Xi(3,1))*Xi(4,2) - (7/24+Xi(1,2))*(1+Xi(2,1)
       )/3);
    factor = (1+Xi(2,1))/3*(1+Xi(2,2))/7 - Xi(4,2)*Xi(4,1);
    E(:,:)=E(:,:)/factor;

clear factor;

%% dï¿½termination J+ fonction de phi_i puis fonction de
   vecteur_source

% dï¿½termination des points en surfaces
% ï¿½ventuellement dï¿½jï¿½ connue
```

```
disp('--mise␣en␣place␣relation␣pour␣dï¿½terminer␣J+␣:');
tic;

% calcul des constantes Jn puis J_coeff
% attention !!!
% suivre la convention pour choix des Rn

Jn = [-1/2*Rn(1) , -3/2*Rn(2) , 5/4*Rn(1) - 15/4*Rn(3) , 21/4*
   Rn(2) - 35/4*Rn(4)];

J_coeff = zeros(1,2); %coeff fonction phi

    J_coeff(1) = 1/4 + Jn(1) - (0.5+Jn(2))/3*E(1,1) - Jn(4)/7*E
       (2,1);
    J_coeff(2) = -1/16 - 2/3*Jn(1) + Jn(3)/3 - (0.5+Jn(2))/3*E
       (1,2) - Jn(4)/7*E(2,2);

% dï¿½termination J+ fonction phi_i
% J+ de taille ï¿½gale au nombre de points en surface

J_mat=sparse(nb_points_surf(1),number_nodes);

for i=1:nb_points_surf
    pt_i = pts_surface(i);

    J_mat(i,pt_i) = 1;

end

save J J_coeff J_mat;
```

A.5.1.3 Fonctions permettant la mise en place du modèle direct

Listing A.26 – `forward_model_f.m`

```
function [Mat_schur]=forward_model(lambda)

%cste_milieu(); % save J pour la suite, à faire en dehors de
   préférence...

%% changement de base des matrices (feelpp -> mesh)

if exist(['/tmp/matschur' num2str(lambda) '.mat'],'file')
    disp('forward_model␣loaded');
    load(['/tmp/matschur' num2str(lambda) '.mat']);
```

```
else

load mesh_info; % besoin de l'info sur le changement de base :
   msh_num2feelpp_dof
clearvars -except msh_num2feelpp_dof lambda;
tic;

load(['Mat_xx_' int2str(lambda)]); % charge les 5 matrices sous
    le format Mat_B, Mat_11, Mat_12, Mat_21, Mat_22 !!!

% changement de base

disp('changement de base');

Mat_B = msh_num2feelpp_dof\Mat_B*msh_num2feelpp_dof;
Mat_11 = msh_num2feelpp_dof\Mat_11*msh_num2feelpp_dof;
Mat_12 = msh_num2feelpp_dof\Mat_12*msh_num2feelpp_dof;
Mat_21 = msh_num2feelpp_dof\Mat_21*msh_num2feelpp_dof;
Mat_22 = msh_num2feelpp_dof\Mat_22*msh_num2feelpp_dof;

clear msh_num2feelpp_dof;

%% inversion de la matrice SP3 en utilisant inversion par bloc
   avec
% complément de Schur
% supprime la matrice initiale !

disp('inversion par bloc par complément de Schur');

P11_tmp = full(inv(Mat_11));
clear Mat_11;

schur_compl = Mat_22-Mat_21*P11_tmp*Mat_12;
clear Mat_22;

P22 = full(inv(schur_compl));
clear schur_compl;

P12 = -P11_tmp*Mat_12*P22;

P21 = -P22*Mat_21*P11_tmp;

P11 = P11_tmp + P11_tmp*Mat_12*P22*Mat_21*P11_tmp;
```

```
clear Mat_12 Mat_21 P11_tmp;

%% modèle direct mis en place

mat_1 = P11 -2/3*P12;
clear P11 P12;

mat_2 = P21-2/3*P22;
clear P21 P22;

% Somme de phi 1 et phi 2
load J; % load J pour obtenir J+ en fonction de source

Mat_schur = J_coeff(1)*J_mat*mat_1*Mat_B + J_coeff(2)*J_mat*
   mat_2*Mat_B;

clear J_coeff mat_1 mat_2 Mat_B J_mat;

save(['/tmp/matschur' num2str(lambda)],'Mat_schur');

toc;

end
```

Listing A.27 – `changement_base.m`

```
% changement de base

disp('changement␣de␣base');

Mat_B = msh_num2feelpp_dof\Mat_B*msh_num2feelpp_dof;
Mat_11 = msh_num2feelpp_dof\Mat_11*msh_num2feelpp_dof;
Mat_12 = msh_num2feelpp_dof\Mat_12*msh_num2feelpp_dof;
Mat_21 = msh_num2feelpp_dof\Mat_21*msh_num2feelpp_dof;
Mat_22 = msh_num2feelpp_dof\Mat_22*msh_num2feelpp_dof;

%clear msh_num2feelpp_dof;
```

Listing A.28 – `calc_coeff_cer.m`

```
function k = calc_coeff_cer(lambda_used)

k = zeros(6,1);
% High-pass
```

```
H = [615 660 700 770];

% Band-pass

B = [[755;805] , [790;840]];

a = size(H,2)-1;
b = size(B,2);

for i=1:a
    k(i) = 1/H(i) - 1/850;
end

for i=1:b
    k(a+i) = 1/B(1,i)-1/B(2,i);
end

k(6) = 1/400-1/850;
k = k/k(6);
```

A.5.1.4 Algorithme de reconstruction des moindres carrées à valeurs positives

Listing A.29 – l1_ls_nonneg.m

```
function [x,status,history] = l1_ls_nonneg(A,varargin)
%
% l1-Regularized Least Squares Problem Solver
%
%   l1_ls solves problems of the following form:
%
%       minimize   ||A*x-y||^2 + lambda*sum(x_i),
%       subject to x_i >= 0, i=1,...,n
%
%   where A and y are problem data and x is variable (described
    below).
%
% CALLING SEQUENCES
%   [x,status,history] = l1_ls_nonneg(A,y,lambda [,tar_gap[,
   quiet]])
%   [x,status,history] = l1_ls_nonneg(A,At,m,n,y,lambda, [,
   tar_gap,[,quiet]]))
%
%   if A is a matrix, either sequence can be used.
%   if A is an object (with overloaded operators), At, m, n
   must be
```

```
%   provided.
%
% INPUT
%   A       : mxn matrix; input data. columns correspond to
   features.
%
%   At      : nxm matrix; transpose of A.
%   m       : number of examples (rows) of A
%   n       : number of features (column)s of A
%
%   y       : m vector; outcome.
%   lambda  : positive scalar; regularization parameter
%
%   tar_gap : relative target duality gap (default: 1e-3)
%   quiet   : boolean; suppress printing message when true (
   default: false)
%
%   (advanced arguments)
%       eta     : scalar; parameter for PCG termination (
   default: 1e-3)
%       pcgmaxi : scalar; number of maximum PCG iterations (
   default: 5000)
%
% OUTPUT
%   x       : n vector; classifier
%   status  : string; 'Solved' or 'Failed'
%
%   history : matrix of history data. columns represent (
   truncated) Newton
%             iterations; rows represent the following:
%           - 1st row) gap
%           - 2nd row) primal objective
%           - 3rd row) dual objective
%           - 4th row) step size
%           - 5th row) pcg iterations
%           - 6th row) pcg status flag
%
% USAGE EXAMPLES
%   [x,status] = l1_ls_nonneg(A,y,lambda);
%   [x,status] = l1_ls_nonneg(A,At,m,n,y,lambda,0.001);
%

% AUTHOR    Kwangmoo Koh <deneb1@stanford.edu>
% UPDATE    Apr 10 2008
```

```
%
% COPYRIGHT 2008 Kwangmoo Koh, Seung-Jean Kim, and Stephen Boyd

%-----------------------------------------------------------
%       INITIALIZE
%-----------------------------------------------------------

% IPM PARAMETERS
MU              = 2;        % updating parameter of t
MAX_NT_ITER     = 400;      % maximum IPM (Newton) iteration

% LINE SEARCH PARAMETERS
ALPHA           = 0.01;     % minimum fraction of decrease in
   the objective
BETA            = 0.5;      % stepsize decrease factor
MAX_LS_ITER     = 100;      % maximum backtracking line search
   iteration

% VARIABLE ARGUMENT HANDLING
% if the second argument is a matrix or an operator, the
   calling sequence is
%   l1_ls(A,At,y,lambda,m,n [,tar_gap,[,quiet]]))
% if the second argument is a vector, the calling sequence is
%   l1_ls(A,y,lambda [,tar_gap[,quiet]])
if ( (isobject(varargin{1}) || ~isvector(varargin{1})) &&
   nargin >= 6)
    At = varargin{1};
    m  = varargin{2};
    n  = varargin{3};
    y  = varargin{4};
    lambda = varargin{5};
    varargin = varargin(6:end);

elseif (nargin >= 3)
    At = A';
    [m,n] = size(A);
    y  = varargin{1};
    lambda = varargin{2};
    varargin = varargin(3:end);
else
    if (~quiet) disp('Insufficient input arguments'); end
    x = []; status = 'Failed'; history = [];
    return;
end
```

```
% VARIABLE ARGUMENT HANDLING
t0          = min(max(1,1/lambda),n/1e-3);
defaults    = {1e-3,false,1e-3,5000,ones(n,1),t0}; % valeur de x
    initial définie en 4ème argument
given_args = ~cellfun('isempty',varargin);
defaults(given_args) = varargin(given_args);
[reltol,quiet,eta,pcgmaxi,x,t] = deal(defaults{:}); % x
   initialiser ici !

f = -x;

% RESULT/HISTORY VARIABLES
pobjs = [] ; dobjs = [] ; sts = [] ; pitrs = []; pflgs = [];
pobj  = Inf; dobj  =-Inf; s   = Inf; pitr  = 0 ; pflg  = 0 ;

ntiter  = 0; lsiter  = 0; zntiter = 0; zlsiter = 0;
normg   = 0; prelres = 0; dx =  zeros(n,1);

% diagxtx = diag(At*A);
diagxtx = 2*ones(n,1);

if (~quiet) disp(sprintf('\nSolving␣a␣problem␣of␣size␣(m=%d,␣n
   =%d),␣with␣lambda=%.5e',...
            m,n,lambda)); end
if (~quiet) disp('
   ------------------------------------------------------------------------
   ');end
if (~quiet) disp(sprintf('%5s␣%9s␣%15s␣%15s␣%13s␣%11s',...
            'iter','gap','primobj','dualobj','step␣len','pcg␣
               iters')); end

%-------------------------------------------------------------
%                MAIN LOOP
%-------------------------------------------------------------

for ntiter = 0:MAX_NT_ITER

    z = A*x-y;

    %
       ------------------------------------------------------------

    %       CALCULATE DUALITY GAP
```

```
%
    ------------------------------------------------------------


nu = 2*z;

minAnu = min(At*nu);
if (minAnu < -lambda)
     nu = nu*lambda/(-minAnu);
end
pobj   =  z'*z+lambda*sum(x,1);
dobj   =  max(-0.25*nu'*nu-nu'*y,dobj);
gap    =  pobj - dobj;

pobjs = [pobjs pobj]; dobjs = [dobjs dobj]; sts = [sts s];
pflgs = [pflgs pflg]; pitrs = [pitrs pitr];

%
    ------------------------------------------------------------

%    STOPPING CRITERION
%
    ------------------------------------------------------------

if (~quiet) disp(sprintf('%4d␣%12.2e␣%15.5e␣%15.5e␣%11.1e␣
   %8d',...
     ntiter, gap, pobj, dobj, s, pitr)); end

if (gap/abs(dobj) < reltol)
     status  = 'Solved';
     history = [pobjs-dobjs; pobjs; dobjs; sts; pitrs; pflgs
        ];
     if (~quiet) disp('Absolute␣tolerance␣reached.'); end
     %disp(sprintf('total pcg iters = %d\n',sum(pitrs)));
     return;
end
%
    ------------------------------------------------------------

%        UPDATE t
%
    ------------------------------------------------------------

if (s >= 0.5)
```

```
        t = max(min(n*MU/gap, MU*t), t);
    end

    %
       -------------------------------------------------------------

    %        CALCULATE NEWTON STEP
    %
       -------------------------------------------------------------


    d1 = (1/t)./(x.^2);

    % calculate gradient
    gradphi = [At*(z*2)+lambda-(1/t)./x];

    % calculate vectors to be used in the preconditioner
    prb     = diagxtx+d1;

    % set pcg tolerance (relative)
    normg   = norm(gradphi);
    pcgtol  = min(1e-1,eta*gap/min(1,normg));

    if (ntiter ~= 0 && pitr == 0) pcgtol = pcgtol*0.1; end

if 1
    [dx,pflg,prelres,pitr,presvec] = ...
        pcg(@AXfunc_l1_ls,-gradphi,pcgtol,pcgmaxi,@Mfunc_l1_ls
           ,...
            [],dx,A,At,d1,1./prb);
end
    %dx = (2*A'*A+diag(d1))\(-gradphi);

    if (pflg == 1) pitr = pcgmaxi; end

    %
       -------------------------------------------------------------

    %   BACKTRACKING LINE SEARCH
    %
       -------------------------------------------------------------

    phi = z'*z+lambda*sum(x)-sum(log(-f))/t;
    s = 1.0;
```

```
    gdx = gradphi'*dx;
    for lsiter = 1:MAX_LS_ITER
        newx = x+s*dx;
        newf = -newx;
        if (max(newf) < 0)
            newz   =  A*newx-y;
            newphi =  newz'*newz+lambda*sum(newx)-sum(log(-newf
               ))/t;
            if (newphi-phi <= ALPHA*s*gdx)
                break;
            end
        end
        s = BETA*s;
    end
    if (lsiter == MAX_LS_ITER) break; end % exit by BLS

    x = newx; f = newf; % nouveau x poser
end


%-------------------------------------------------------------
%       ABNORMAL TERMINATION (FALL THROUGH)
%-------------------------------------------------------------
if (lsiter == MAX_LS_ITER)
    % failed in backtracking linesearch.
    if (~quiet) disp('MAX_LS_ITER␣exceeded␣in␣BLS'); end
    status = 'Failed';
elseif (ntiter == MAX_NT_ITER)
    % fail to find the solution within MAX_NT_ITER
    if (~quiet) disp('MAX_NT_ITER␣exceeded.'); end
    status = 'Failed';
end
history = [pobjs-dobjs; pobjs; dobjs; sts; pitrs; pflgs];

return;

%-------------------------------------------------------------
%       COMPUTE AX (PCG)
%-------------------------------------------------------------
function [y] = AXfunc_l1_ls(x,A,At,d1,p1)

y = (At*((A*x)*2))+d1.*x;

%-------------------------------------------------------------
```

```
%        COMPUTE P^{-1}X (PCG)
%-------------------------------------------------------------
function [y] = Mfunc_l1_ls(x,A,At,d1,p1)

y = p1.*x;
```

Arnaud BERTRAND

Logo partenaire

Mise en place de l'imagerie Cerenkov 3D

Résumé

L'imagerie moléculaire vise à étudier les processus biologiques *in vivo*. L'imagerie Cerenkov est une technique d'imagerie moléculaire qui se développe depuis 2009. Le principe est d'injecter un radiotraceur, molécule marquée par un isotope radioactif, puis à enregistrer le signal optique émis par effet Cerenkov. L'imagerie Cerenkov permet d'imager des radiotraceurs émettant des rayonnements β+ (positon) et β- (électron).

L'effet Cerenkov se produit lorsqu'une particule chargée se déplace dans un milieu avec une vitesse supérieure à celle de la lumière dans ce même milieu. Si ce seuil est dépassé, on observe alors une émission de photons optiques appelée rayonnement Cerenkov. Le spectre de cette émission s'étend de l'UV à l'IR de manière continue et le nombre de photons émis en fonction de la longueur d'onde varie en $1/\lambda^2$.

Mon thèse consiste à développer l'imagerie Cerenkov 3D pour reconstruire la distribution du radiotraceur *in vivo*. Nous disposons d'une plateforme d'imagerie nommée AMISSA (A Multimodality Imaging System for Small Animal) dont le but est de développer et de mettre à disposition des outils d'imagerie moléculaire pour du petit animal.

Résumé en anglais

Molecular imaging aims to study biological processes *in vivo*. Cerenkov imaging is a molecular imaging technology that has developed since 2009. The principle is to inject a radioactive tracer molecule labeled with a radioactive isotope, then recording the optical signal emitted by the Cerenkov effect. The Cerenkov imaging allows imaging radiotracers emitting β+ radiation (positron) and β- (electron).

The Cerenkov effect occurs when a charged particle moves through a medium with a speed greater than that of light in this same medium. If this threshold is exceeded, we observed an emission of optical photons called Cerenkov radiation. The emission spectrum of this extends from UV to IR continuously and the number of photons emitted as a function of the wavelength varies by $1/\lambda^2$.

My PhD is to develop 3D imaging Cerenkov to reconstruct the distribution of the radiotracer in vivo. We have an imaging platform named Amissa (A Multimodality Imaging System for Small Animal) whose purpose is to develop and make available tools for molecular imaging of small animals.