Méthode de reconstruction adaptative en tomographie par rayons X - Optimisation sur architectures parallèles de type GPU

Michele Arcangelo Quinto

UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Signal, Image, Parole, Télécoms**

Arrêté ministériel : 7 août 2006

Présentée par

Michele Arcangelo Quinto

Thèse dirigée par **Dominique Houzet**

préparée au sein du
CEA LIST, Laboratoire Imagerie Traitement et Tomographie
et de l'**École Doctorale E.E.A.T.S.**

Méthode de reconstruction adaptative en tomographie par rayons X - Optimisation sur architectures parallèles de type GPU

Thèse soutenue publiquement le **5 Avril 2013**,
devant le jury composé de :

M. Julien BERT
LaTIM-INSERM U650, Co-Rapporteur
Mme Fanny BUYENS
CEA LIST, Encadrante de thèse
Mme Christine CHAPPARD
B2OA UMR7052 INSERM, Examinatrice
M. Jean-Philippe DOMENGER
Université Bordeaux 1 LaBRI, Rapporteur
M. Bertrand GRANADO
Université Pierre et Marie Curie LIP6 UMR7606, Examinateur
M. Dominique HOUZET
Gipsa-lab Grenoble-INP, Directeur de thèse
M. Dimitris VISVIKIS
LaTIM-INSERM U650, Rapporteur

Remerciements

Je tiens à remercie Julien Bert, Jean-Philippe Domenger et Dimitris Visvikis, rapporteurs, pour l'intérêt qu'ils ont porté à mon travail. Leurs remarques et corrections ont largement contribué à la version finale de ce manuscrit.

Je voudrais également remercier Christine Chappard et Bertrand Granado pour avoir accepté de participer au jury de soutenance en tant qu'examinateurs.

Je tiens à adresser les plus vifs remerciement à Fanny Buyens, qui m'a soutenu pendant ces trois années de thèse, et à son encadrament avisé, quotidian et exigeante de auquelle cette thèse doit beaucoup.

Je remercie mon directeur de thèse Dominique Houzet pour son aide et sa disponibilité à m'accueillir au sein du laboratoire Gipsa-lab de Grenoble pour travailler et discuter sur les aspects de la parallélisation.
Je remercie Tomas Toczek pour son aide concerant les mises en œuvre sur GPU.

J'adresse mes plus sincères remerciements à collègues du laboratoire LITT : Ismail Ben-Tekaya, Caifang Cai, Anthony Cazasnoves, Marius Costin, Carole Force, Caroline Vienne et Han Wang pour tous les échanges scientifiques et pour leur sympathie et bons moments passés au cours de ces trois ans de thèse.
Un remerciement particulier à Caifang avec qui j'ai partagé le bureau pendant la période de la thèse.

Je remercie également Domique Chambelland, Philipe Reverchon et Bernard Rattoni pour la leur sympathie et les bons moments passés au cours de repas.

Je témoigne toute ma reconnaissance amicale à Juan-Carlos Garcia-Hernandez pour son soutien dans les moments difficiles de la thèse, son aide et ses conseils, et «toujours» là pour aller à prendre un verre.

Je remercie mes parents, mes frères et ma sœur pour leur compréhension et leur aide morale.

Résumé

La reconstruction tomographique à partir de données de projections est un problème inverse largement utilisé en imagerie médicale et de façon plus modeste pour le contrôle non destructif. Avec un nombre suffisant de projections, les algorithmes analytiques permettent des reconstructions rapides et précises. Toutefois, dans le cas d'un faible nombre de vues (imagerie faible dose) et/ou d'angle limité (contraintes spécifiques liées à l'installation), les données disponibles pour l'inversion ne sont pas complètes, le mauvais conditionnement du problème s'accentue, et les résultats montrent des artefacts importants. Pour aborder ces situations, une approche alternative consiste à discrétiser le problème de reconstruction, et à utiliser des algorithmes itératifs ou une formulation statistique du problème afin de calculer une estimation de l'objet inconnu. Ces méthodes sont classiquement basées sur une discrétisation du volume en un ensemble de voxels, et fournissent des cartes 3D de la densité de l'objet étudié. Les temps de calcul et la ressource mémoire de ces méthodes itératives sont leurs principaux points faibles. Par ailleurs, quelle que soit l'application, les volumes sont ensuite segmentés pour une analyse quantitative. Devant le large éventail d'outils de segmentation existant, basés sur différentes interprétations des contours et de fonctionnelles à minimiser, les choix sont multiples et les résultats en dépendent.

Ce travail de thèse présente une nouvelle approche de reconstruction simultanée à la segmentation des différents matériaux qui composent le volume. Le processus de reconstruction n'est plus basé sur une grille régulière de pixels (resp. voxels), mais sur un maillage composé de triangles (resp. tétraèdres) non réguliers qui s'adaptent à la forme de l'objet. Après une phase d'initialisation, la méthode se décompose en trois étapes principales que sont la reconstruction, la segmentation et l'adaptation du maillage, qui alternent de façon itérative jusqu'à convergence. Des algorithmes itératifs de reconstruction communément utilisés avec une représentation conventionnelle de l'image ont été adaptés et optimisés pour être exécutés sur des grilles irrégulières composées d'éléments triangulaires ou tétraédriques. Pour l'étape de segmentation, deux méthodes basées sur une approche paramétrique (*snake*) et l'autre sur une approche géométrique (*level set*) ont été mises en oeuvre afin de considérer des objets de différentes natures (mono- et multi- matériaux). L'adaptation du maillage au contenu de l'image estimée est basée sur les contours segmentés précédemment, pour affiner la maille au niveau des détails de l'objet et la rendre plus grossière dans les zones contenant peu d'information. En fin de processus, le résultat est une image classique de reconstruction tomographique en

niveaux de gris, mais dont la représentation par un maillage adapté au contenu propose directement une segmentation associée. Les résultats montrent que la partie adaptative de la méthode permet de représenter efficacement les objets et conduit à diminuer drastiquement la mémoire nécessaire au stockage. Dans ce contexte, une version 2D du calcul des opérateurs de reconstruction sur une architecture parallèle type GPU montre la faisabilité du processus dans son ensemble. Une version optimisée des opérateurs 3D permet des calculs encore plus efficaces.

Mots clés : Reconstruction tomagraphique, Segmentation, Méthode de *Levet Set*, Maillage adapatatif, Calcul parallèle, Processeur graphique GPU

Abstract

Tomography reconstruction from projections data is an inverse problem widely used in the medical imaging field. With sufficiently large number of projections over the required angle, the FBP (filtered backprojection) algorithms allow fast and accurate reconstructions. However in the cases of limited views (lose dose imaging) and/or limited angle (specific constrains of the setup), the data available for inversion are not complete, the problem becomes more ill-conditioned, and the results show significant artifacts. In these situations, an alternative approach of reconstruction, based on a discrete model of the problem, consists in using an iterative algorithm or a statistical modelisation of the problem to compute an estimate of the unknown object. These methods are classicaly based on a volume discretization into a set of voxels and provide 3D maps of densities. Computation time and memory storage are their main disadvantages. Moreover, whatever the application, the volumes are segmented for a quantitative analysis. Numerous methods of segmentation with different interpretations of the contours and various minimized energy functional are offered, and the results can depend on their use.

This thesis presents a novel approach of tomographic reconstruction simultaneously to segmentation of the different materials of the object. The process of reconstruction is no more based on a regular grid of pixels (resp. voxel) but on a mesh composed of non regular triangles (resp. tetraedra) adapted to the shape of the studied object. After an initialization step, the method runs into three main steps: reconstruction, segmentation and adaptation of the mesh, that iteratively alternate until convergence. Iterative algorithms of reconstruction used in a conventionnal way have been adapted and optimized to be performed on irregular grids of triangular or tetraedric elements. For segmentation, two methods, one based on a parametric approach (snake) and the other on a geometric approach (level set) have been implemented to consider mono and multi materials objects. The adaptation of the mesh to the content of the estimated image is based on the previous segmented contours that makes the mesh progressively coarse from the edges to the limits of the domain of reconstruction. At the end of the process, the result is a classical tomographic image in gray levels, but whose representation by an adaptive mesh to its content provide a correspoonding segmentation. The results show that the method provides reliable reconstruction and leads to drastically decrease the memory storage. In this context, the operators of projection have been implemented on parallel archituecture called GPU. A first 2D version shows the feasability of the full process, and an optimized version of the 3D operators provides more efficient computations.

Keywords : Tomographic Reconstruction, Segmentation, Levet Set Method, Adaptive Mesh, Parallel Computing, GPU Graphic Processor

Table des matières

Table des figures

Liste des tableaux

Introduction

La tomographie est une technique d'imagerie couramment employée en médecine, qui a prouvé son intérêt dans l'industrie. Plus qu'une simple méthode de contrôle non destructif, la tomographie offre aujourd'hui un large panel d'applications industrielles, de la caractérisation de défauts à l'analyse dimensionnelle, en passant par la numérisation 3D et la rétro-ingénierie.

La tomographie est un problème inverse pour lequel on cherche à reconstruire un objet à partir de ses projections, celles-ci étant acquises à l'aide d'un scanner. Dans la plupart des applications de tomographie par rayon X, les méthodes de reconstruction utilisées sont les méthodes analytiques. Dans le cas d'un nombre suffisant de projections, ces méthodes permettent en effet des reconstructions rapides et fiables. Cependant, dans le cas d'un nombre limité de projections (applications médicales faible dose), ou d'un domaine angulaire réduit (contraintes expérimentales spécifiques), les données disponibles pour l'inversion sont insuffisantes ; le mauvais conditionnement du problème s'accentue, et les résultats sont entâchés d'artéfacts. Dans ces situations, une approche alternative consiste à discrétiser le problème de reconstruction, et à utiliser des algorithmes itératifs ou une formulation statistique du problème afin de calculer une estimation de l'objet inconnu. Le manque d'information est alors compensé par l'introduction d'*a priori* ou de contraintes qui permettent de converger vers une solution acceptable. Ces méthodes sont classiquement basées sur une discrétisation du volume en un ensemble régulier de voxels, et fournissent des cartes 3D de la densité de l'objet étudié. A la différence des méthodes analytiques, les méthodes itératives sont coûteuses en termes de temps de calcul et de ressources mémoire, ce qui rend leur utilisation peu usuelle. Ceci est par ailleurs accentué par le fait que les détecteurs, toujours plus résolus, génèrent des jeux de données de plus en plus volumineux. Cependant, avec l'émergence des architectures parallèles de type GPU l'implantation de codes optimisés a permis d'accélérer de façon significative les deux principaux opérateurs de ces algorithmes (que sont la projection et la rétroprojection), ce qui augmente dorénavant leur potentiel.

L'analyse des images reconstruites consiste ensuite à utiliser les outils de traitement d'image afin de segmenter la ou les région(s) d'intérêt. Cette étape essentielle, préliminaire aux traitements de haut niveau, consiste à décomposer une image en régions homogènes. Cependant, beaucoup d'études ont été menées depuis deux décennies aboutissant à de nombreuses méthodes de segmentation, différentes interprétations des contours et diverses fonctionnelles d'énergie à minimiser.

L'objectif de ce travail est de proposer une méthode de reconstruction permettant d'accéder à une image conventionnelle en niveaux de gris, directement associée à une segmentation des matériaux de l'objet étudié, afin de s'affranchir de tout post traitement dans ce domaine. Pour cela, nous avons choisi de ne plus discrétiser le volume en une grille régulière de voxels mais de représenter l'image par un maillage composé de triangles (tétrahèdres) non réguliers s'adapatant aux contours de l'objet. L'idée étant de décrire finement les détails de la scène, et plus grossièrement les régions contenant peu d'information. Après une phase d'initialisation, le processus se décompose en trois étapes principales que sont la reconstruction, la segmentation et l'adaptation du maillage, qui alternent de façon itérative jusqu'à convergence. Une structure de données simple pour laquelle la valeur des niveaux de gris est stockée sur la face des éléments du maillage a été choisie, puis des algorithmes itératifs de reconstruction communément utilisés (EM et gradient conjugué) ont été adaptés pour effectuer les calculs sur des grilles irrégulières de triangles. L'implantation de la projection et de la rétroprojection a été parallélisée et portée sur des cartes graphiques (GPU) afin de diminuer le temps de calcul de ces opérateurs dont les éléments sont calculés en ligne pour chaque modification/adaptation du maillage. Pour l'étape de segmentation, deux méthodes ont été étudiées : la première, basée sur un modèle déformable paramétrique ou *snake*, ne permet de considérer que les objets composés d'un seul matériau ; la deuxième, basée sur un modèle déformable géométrique par lignes de niveau ou *level sets*, élargit le champ d'application aux objets multi matériaux. La segmentation obtenue sert ensuite de base à l'adaptation du maillage, dont la finalité est de réduire le nombre de paramètres à estimer au cours de la reconstruction, et par conséquent d'accélérer progressivement les calculs. En fin de processus, le résultat est une image classique de reconstruction tomographique en niveaux de gris, dont la représentation par un maillage adapté au contenu associe directement une segmentation. L'irrégularité du maillage permet de diminuer considérablement la quantité d'éléments à stocker. Les jeux de données sont alors nettement moins volumineux que ceux générés par les reconstructions sur grilles fixes, dont le nombre d'éléments doit être assez élevé afin d'être en adéquation avec les détecteurs toujours plus résolus.

Les trois premiers chapitres de ce document sont consacrés à la justification de nos choix tant au point de vue de la représentation que des méthodes de segmentation ou encore de la stratégie de parallélisation. Les trois chapitres suivants concernent la mise en oeuvre de la méthode dans son ensemble, sa validation, son accélération et son application à des jeux de données simulées et expérimentales.

Le premier chapitre est un rappel du principe de la tomographie, et des méthodes mathématiques permettant la reconstruction des volumes observés. Les algorithmes de reconstruction utilisés dans la suite du travail y sont présentés dans leur formulation conventionnelle (grilles régulières de pixels). Différentes méthodes de représentation des images autres que les grilles régulières de voxels sont présentées, et notre choix d'éléments triangulaires est justifié.

Le deuxième chapitre introduit la segmentation, et se focalise plus particulièrement sur les méthodes de segmentation existantes dites approches contours exploitées dans

notre méthode simultanément à la reconstruction. Le modèle déformable paramétrique appelé *snake* et le modèle déformable géométrique par *level sets* sont exposés. Leurs limites, avantages et inconvénients sont discutés.

Le troisième chapitre présente les stratégies possibles pour la parallélisation des algorithmes de reconstruction itératifs en tomographie par rayons X. Un état de l'art des différentes architectures parallèles et des travaux réalisés dans ce domaine est proposé permettant de situer le travail dans ce contexte.

Le quatrième chapitre est consacré à la présentation de la méthode dans son ensemble : reconstruction, segmentation et adaptation du maillage à l'objet estimé. Concernant la reconstruction ou l'estimation de l'objet étudié, nous décrivons l'implantation des opérateurs de projection et de rétroprojection sur des grilles irrégulières composées de triangles (tétraèdres) et leur validation. Nous expliquons également l'implantation de deux algorithmes de reconstruction usuellement utilisés dans le cadre conventionnel (grilles pixélisées (voxélisées), régulières et fixes) adaptés à la représentation adaptative sur grilles irrégulières composées de triangles (tétraèdres).

Nous présentons ensuite les éléments développés pour l'introduction d'une étape de segmentation au processus de reconstruction. Ici, deux méthodes de détection de contours sont considérées : une première méthode utilisant un modèle paramétrique ou *snakes* restreinte aux seuls cas d'objets monomatériaux, et une seconde basée sur les lignes de niveaux ou *levels set* et le modèle de Mumford-Shah permettant de traiter des objets constitués d'un ou plusieurs matériaux. Finalement, nous considérons la dernière étape du processus dans laquelle nous avons proposé une technique pour adapter le maillage aux objets (matériaux) segmentés, c'est à dire générer un maillage dont la taille et la densité des mailles d'adaptent en fonction de l'information locale. L'ensemble de la chaîne de reconstruction proposée a été évaluée sur des données expérimentales et numériques, pour la reconstruction d'objets composés d'un ou plusieurs matériaux. Les résultats sont exposés à la fin de ce chapitre et montrent que la partie adaptative de la méthode permet de représenter efficacement les objets et conduit à réduire drastiquement la mémoire nécessaire au stockage. Bien que le temps de calcul diminue avec les itérations et donc la simplification du maillage, celui-ci reste trop important notamment pour les étapes de projection et rétroprojection.

Dans ce contexte, le cinquième chapitre est consacré à la mise en place des calculs de ces deux principaux opérateurs sur une architecture parallèle de type GPU. Différentes solutions algorithmiques sont étudiées en 2D et en 3D, en particulier en évaluant l'impact de la précision des calculs en virgule flottante simple précision et double précision, la régularité des accès mémoire et le taux de parallélisme. Différentes dimensions du problème sont analysées, comme le nombre de tétraèdres du volume et le temps d'accès aux données. Les performances obtenues sont détaillées montrant une grande efficacité de l'implémentation parallèle sur GPU. Des solutions alternatives sont proposées pour améliorer notamment l'initialisation des calculs et les conflits d'accès en mémoire utilisant des opérations atomiques.

Chapitre 1

Tomographie par rayons X

Née au début des années 70 à des fins d'applications médicales, la tomographie par rayons X a beaucoup évoluée depuis, tant dans le domaine médical qu'industriel. En effet, cette technique d'imagerie prometteuse a aujourd'hui adapté ses paramètres au domaine industriel dont tous les secteurs peuvent bénéficier (aéronautique, secteur automobile, fonderie, industrie minière ou pétrolière, secteur agro-alimentaire). Voir l'intérieur d'un objet pour le reconstruire en 3D et localiser toute hétérogénéité, singularité, vide ou inclusion présents dans un objet semble être un atout pour la mise au point, la fabrication et le contrôle des matériaux.

La tomographie par rayons X ou tomographie par transmission, est une technique non invasive (non destructive) qui permet la reconstruction d'images « en coupe » ou de volumes tridimensionnels d'un objet. Son principe repose sur l'analyse multidirectionnelle de l'interaction d'un faisceau de rayons X avec la matière, par enregistrement par des détecteurs du rayonnement transmis après traversée de l'objet. Les données acquises sont collectées suivant des orientations multiples dont le nombre et le pas varient selon l'application, le type d'appareil et la résolution. À l'aide de ces données, une image numérique en niveaux de gris est reconstruite mathématiquement, traduisant point par point le coefficient d'atténuation local du faisceau incident. La tomographie par rayons X permet donc d'accéder au coeur de la matière pour en apprécier les variations d'absorptions radiologiques et les différences de composition (voir FIGURE 1.1).

Usuellement, les algorithmes utilisés pour la reconstruction s'appuient sur une représentation du volume de reconstruction par une grille régulière composée de pixels (resp. voxels en 3D). Les détecteurs imageurs étant de plus en plus résolus, le nombre d'éléments composant cette grille augmente également de façon à reconstruire l'objet étudié en conservant la résolution spatiale. Le nombre d'inconnues à estimer peut donc être très important, et les temps de calculs des algorithmes algébriques et statistiques trop élevés pour être couramment utilisés.

Dans ce chapitre, nous rappelons dans un premier temps les principes physiques de la tomographie par rayons X, les principes mathématiques de la reconstruction [88, 89], ainsi que les différents types d'algorithmes existants. Dans un second temps, nous

discutons des différentes manières de représenter une image, et nous nous attardons plus particulièremnet sur les méthodes qui s'affranchissent des grilles conventionnelles de pixels.

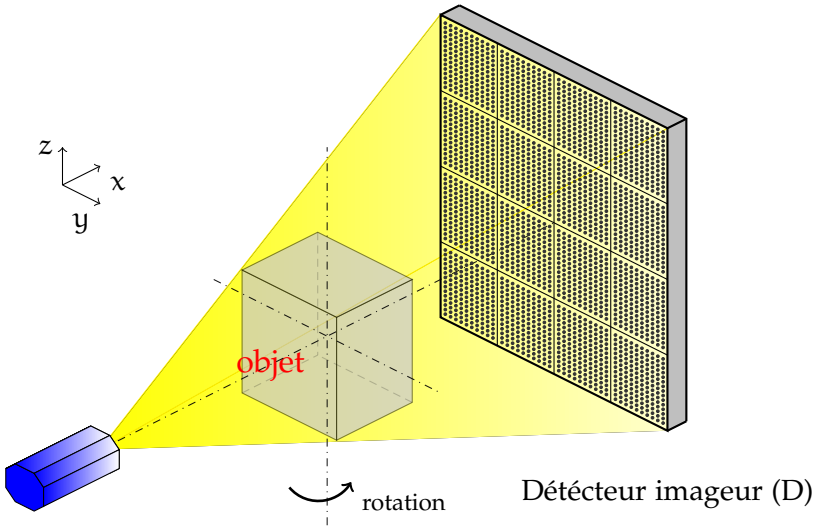


FIGURE 1.1 – Schéma d'un système de tomographie industrielle par rayons X comprenant un générateur à rayons X (S) et un détecteur imageur (D) fixes. L'objet à imager est posé entre ces deux éléments sur un système mécanique permettant sa rotation autour de l'axe vertical z.

1.1 Principe physique

Lorsqu'un faisceau de photons traverse un milieu, chaque matière traversée atténue le flux de photons en fonction de sa nature et de son épaisseur. Ainsi, mesure-t-on une différence d'intensité du flux de photons entre l'entrée et la sortie de la matrière traversée. Pour un mileu homogène et un faiseau monochromatique, cette différence s'exprime par la loi de Beer-Lambert :

$$I(l) = I(0) \cdot e^{-\mu(x)l} \tag{1.1}$$

où μ est le coefficient linéaire d'atténuation caractéristique du milieu (matériau) traversé, l est l'épaisseur de matière traversée, $I(0)$ et $I(l)$ sont respectivement l'intensité du flux entrant et sortant.

Cette équation peut être généralisée pour un milieu non homogène, en considérant une succession de milieux homogènes de longueur infinitésimale. Ainsi, l'équation (1.1) devient :

$$I(l) = I(0) \cdot \exp\left(-\int_0^l \mu(x)dx\right) \tag{1.2}$$

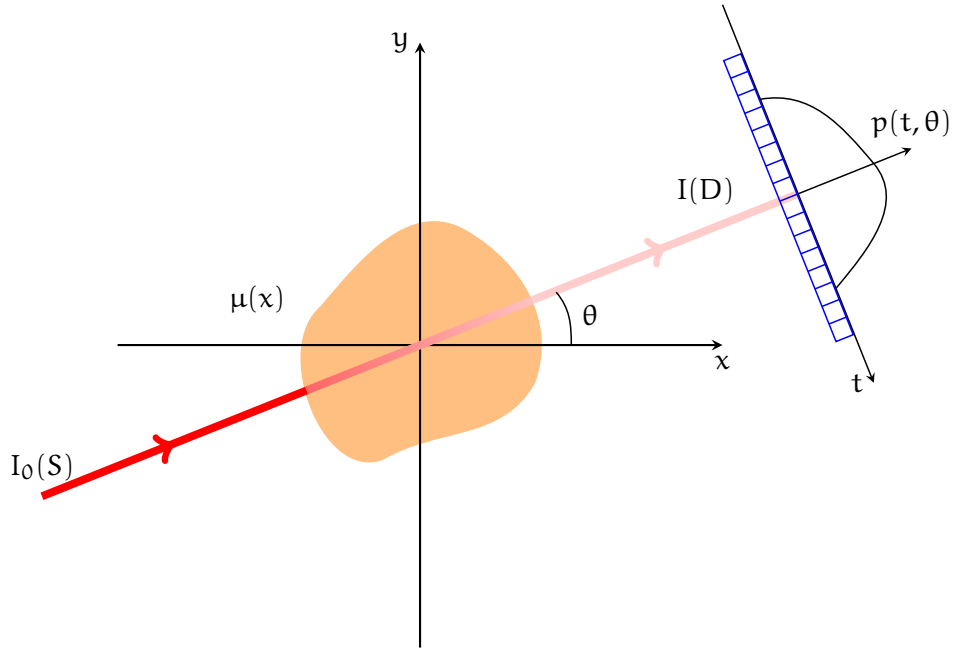


FIGURE 1.2 – Le rayon en traversant l'objet est atténué et on mesure son intensité à l'aide d'un détecteur. On parle de projection définie à l'angle donné θ.

et donc

$$\int_S^D \mu(x)dx = \ln\left(\frac{I_0(S)}{I(D)}\right) \tag{1.3}$$

où S est la position de la source de rayonnement et D l'emplacement du point de détection. $I_0(S)$ désigne la valeur d'intensité incidente au point source S et $I(D)$ celle transmise au point détecteur (voir FIGURE 1.2). Ainsi, la mesure en positon t et à l'angle θ peut être approchée par

$$p(t,\theta) \approx \int_S^D \mu(x)dx \tag{1.4}$$

L'ensemble des mesures selon un angle donné en fonction de la positon t constitue une projection. On notera par ailleurs que l'ensemble des mesures pour une coupe donnée et selon tous les angles θ forme un sinogramme. La reconstruction d'une image ou d'un volume consiste à rechercher une estimation $\hat{\mu}(x)$ de $\mu(x)$ en tout point de coordonnées (x, y, z) de l'objet étudié.

1.2 Acquisition des projections

Depuis le début de l'utilisation de l'imagerie par rayons X dans le domaine médical puis pour les applications industrielles, les systèmes d'acquisition ont beaucoup évolué. Selon les différentes générations de détecteurs, on peut distinguer trois principales géométries d'acquisition (voir FIGURE 1.3) :

- La géométrie parallèle où le système est formé d'une source et d'un détecteur, et pour laquelle les rayons sont émis suivant des droites parallèles. L'exploration de l'objet est conduite par la succession de translations et de rotations du couple source - détecteur.
- La géométrie en éventail, ou *fan beam*, où le système est formé d'une source et d'un détecteur linéaire ou en arc de cercle, et pour laquelle les rayons émis divergent suivant l'angle d'ouverture du générateur à rayons X. Les projections sont acquises par la succession de rotations du couple source-détecteur autour de l'objet.
- La géométrie conique, ou *cone beam*, où le système est constitué d'une source et d'un détecteur plan (détecteur imageur), et pour laquelle les rayons sont émis de manière conique. Les projections sont acquises par la succession de rotations du couple source-détecteur autour de l'objet. Ces systèmes ont été conçus pour permettre une reconstruction 3D de l'objet étudié.

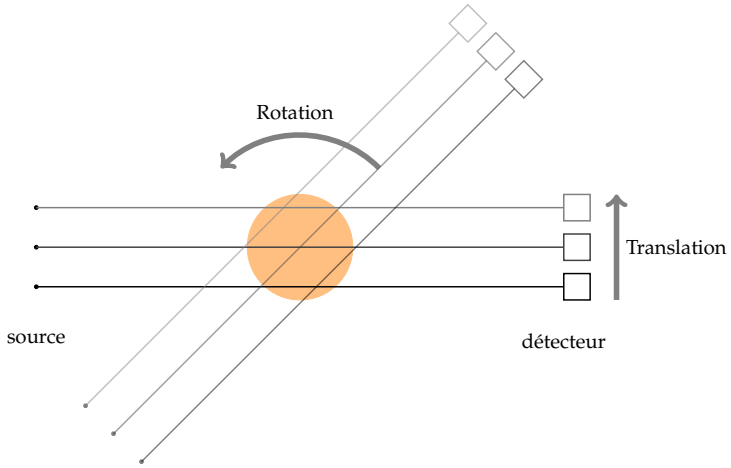
Les contraintes d'acquisition imposées par l'application médicale d'une part et industrielle d'autre part peuvent engendrer des configurations d'acquisition non conventionnelles. On entend par conventionnelle une configuration permettant l'acquisition des projections sur l'ensemble du domaine angulaire (2π radians) en nombre suffisant.

En effet, dans le domaine médical, on cherche à minimiser la dose délivrée au patient. Celle-ci dépendant de l'intensité du faisceau de rayons X et du temps d'exposition, il convient de jouer sur ces paramètres pour réduire la dose au patient. La diminution du flux induit une augmentation du bruit dans les données acquises, et donc le développement d'algorithmes robustes. La réduction du temps d'exposition peut se traduire par la décroissance du nombre de projections (nombre de vues limité).

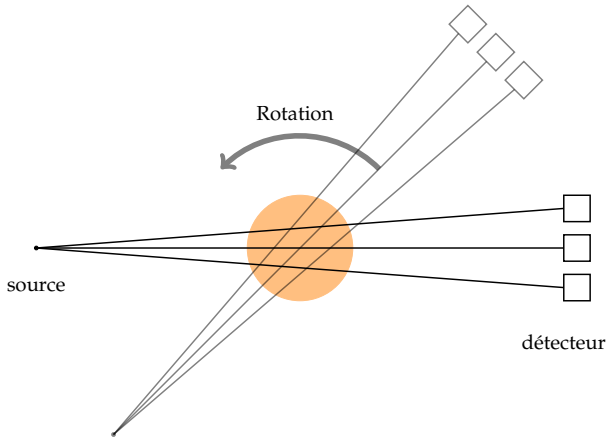
Pour les applications industrielles, les contraintes sont plutôt liées à la grande diversité des matériaux analysés et à la géométrie des pièces étudiées, dont les dimensions peuvent s'étendre de l'échelle nanométrique à l'échelle métrique. Compte tenu de ces observations, une résolution spatiale des volumes reconstruits plus fine que pour les applications médicales (0.5mm) est nécessaire. Dans ce domaine, des détecteurs à haute résolution spatiale sont utilisés, ce qui induit des jeux de données très volumineux à traiter. En outre, les conditions expérimentales de contrôle non destructif (CND) sont variées, et certaines géométries peuvent contraindre le domaine angulaire. L'imagerie de phénomènes dynamiques peut également contraindre le temps d'exposition et/ou le nombre de vues.

Ainsi, peut-on résumer différentes configurations d'acquisition possibles de la manière suivante :

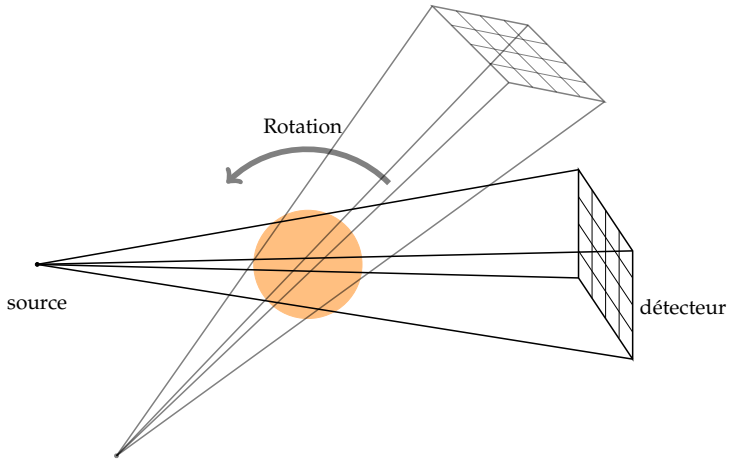
- nombre suffisant de projections sur un domaine angulaire complet (FIGURE 1.4a)
- faible nombre de projections sur un domaine angulaire complet (nombre de vues limité) (FIGURE 1.4b)
- acquisition sur un domaine angulaire réduit (angle de vue restreint) (FIGURE 1.4c)



(a) Géométrie parallèle : configuration typique des systèmes de première génération où l'acquisition est faite par translations et rotations du couple source-détecteur.

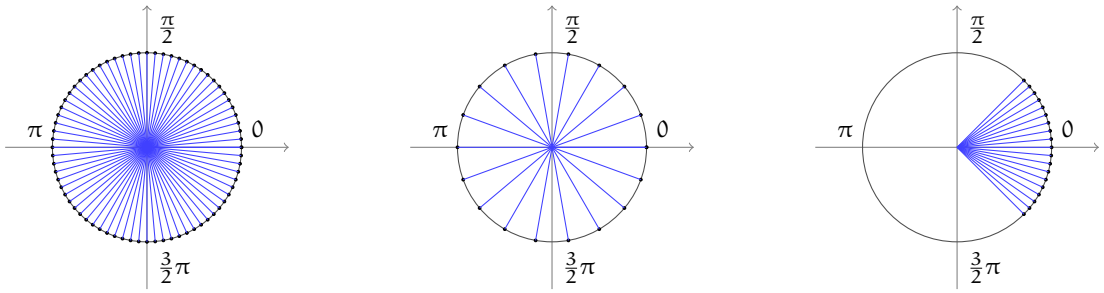


(b) Géométrie en éventail : configuration des systèmes de deuxième et troisième générations ; le détecteur est linéaire ou en arc de cercle. L'acquisition de projections 1D est effectuée par rotation du couple source-détecteur.



(c) Géométrie conique : configuration de dernière génération dans laquelle le détecteur est plan (détecteur imageur). L'acquisition de projections 2D est faite par rotation de l'ensemble source-détecteur autour de l'objet.

FIGURE 1.3 – Exemple de géométries d'acquisition.



(a) Domaine angulaire complet et grand nombre de projection.

(b) Domaine angulaire complet et faible nombre de projections.

(c) Domaine angulaire réduit.

FIGURE 1.4 – Exemple de configurations d'acquisition.

1.3 Méthodes de reconstruction

Afin de reconstruire l'image, il faut modéliser mathématiquement les mesures réalisées par le détecteur. Cette modélisation a été proposée en 1917 par Radon. La transformée de Radon 2D décrit la projection d'une fonction $f(x,y) \in \mathbb{R}^+$ sur une droite L paramétrée par (t,θ), telle que $L(t,\theta) = \{x \mid \langle \theta, x \rangle = t\}$ définit le passage de l'espace cartésien à l'espace des projections. Alors les projections qui correspondent aux mesures en un point p du détecteur à un angle de projection donné θ peuvent s'écrire par la transformée de Radon de la manière suivante :

$$p(t,\theta) = \mathcal{R}f(t,\theta) = \int_{L_\theta} f_p(r,\phi)\, dl \tag{1.5}$$

$$= \int_{L_\theta} f(t\cos\theta - u\sin\theta, t\sin\theta + u\cos\theta)\, du \tag{1.6}$$

Les algorithmes de reconstruction permettent de retrouver les valeurs de la fonction objet f en tout point de l'espace $\{f(x,y),\ (x,y) \in \mathbb{R}^2\}$ à partir de l'ensemble des mesures de projection $\{p(t,\theta),\ \theta \in [0,\pi[,\ t \in \mathbb{R}\}$ (avec $p(t,\theta) = (-t, \theta + \pi)$).

Pour cela, il faut inverser l'opérateur de Radon $\mathcal{R}$. Il existe différentes approches que l'on peut classer selon deux familles :

- les algorithmes analytiques, très rapides en temps de calcul et de haute qualité de reconstruction, mais aussi très sensibles au bruit. Ce type d'algorithme est condictionné par Shannon [88] ne peut fournir de reconstructions acceptables dans le cas d'un faible nombre de projections ou celui de projections acquises sur un domaine angulaire réduit.
- les algorithmes itératifs, plus lents en temps de calcul mais plus robustes. L'avantage incontestable de ces méthodes est qu'elles offrent la possibilité d'intégrer des informations *a priori* sur l'image, et permettent la reconstruction de volume à partir de données incomplètes (nombre de vues limité ou domaine angulaire restreint). Ces

méthodes utilisent une approche discrète du problème. On distingue deux types d'algorithmes : les algorithmes algébriques qui considèrent les données comme des valeurs déterministes, et les algorithmes statistiques, qui considèrent les données comme une variable aléatoire qui peut être décrite par une loi.

1.3.1 Approche analytique

L'algorithme analytique est une modélisation continu-continu, c'est-à-dire que les données sont interprétées comme les échantillons d'une fonction à variable continue. Les algorithmes analytiques permettent de trouver l'opérateur inverse de Radon de façon analytique en utilisant le théorème de la coupe centrale (*Fourier Slice theorem*) qui crée une correspondance entre l'espace image et l'espace de projection dans le domaine de Fourier :

$$\mathcal{F}\{\mathcal{R}\{f(\cdot,\theta)\}\}(q) = \int_{\mathbb{R}} p(t,\theta)e^{-2\pi i q t}dt \tag{1.7}$$

Le théorème de la coupe centrale établit que la transformée de Fourier d'une projection correspond à une ligne de la transformée de Fourier de l'image qui passe par l'origine et fait un angle θ avec l'axe des abscisses (voir FIGURE 1.5).

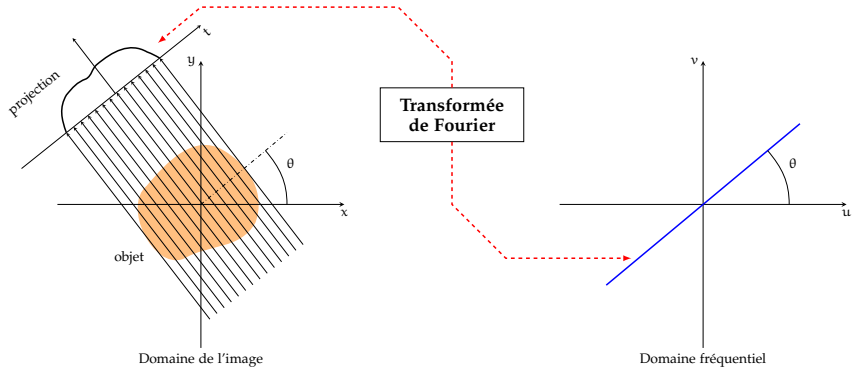


FIGURE 1.5 – Théorème de la coupe centrale : la transformée de Fourier d'une projection correspond à une ligne de la transformée de Fourier de l'image qui passe par l'origine et fait un angle θ avec l'axe des abscisses [65].

À partir de l'équation (1.7), on peut trouver une approximation de la fonction f, notée $\hat{f}$ en appliquant la transformée de Fourier inverse

$$\hat{f} = \mathcal{F}^{-1}\{\mathcal{F}\{p(t,\theta)\}\} \tag{1.8}$$

Le théorème de la coupe centrale permet de reconstruire une coupe tomographique de manière directe à l'aide de la transformée de Fourier inverse. Ainsi, lors une reconstruction, utilisant le théorème de la coupe centrale, on calcule d'abord la transformée de Fourier 1D de chaque projection.

Puis le calcul de la transformée de Fourier inverse 2D aboutit à l'estimation d'une coupe tomographique. Cette méthode, appelée inversion directe, nécessite donc autant

de transformées de Fourier 1D que d'angles de projection, et une transformée de Fourier inverse 2D. La rétroprojection de l'ensemble des projections fournit cependant une version floue de f car elle suppose implicitement qu'un certain nombre d'hypothèses soient vérifiées, qui ne sont en réalité pas toujours vraies (par exemple, faisceau monochromatique infiniment fin).

La méthode analytique la plus couramment utilisée est la rétroprojection filtrée, ou FBP (*Filtered Back-Projection*), qui est la reconstruction par rétroprojection des projections filtrées [15]. Elle repose sur le résultat suivant :

$$f = \mathcal{B}(\hat{p}) \tag{1.9}$$

avec $\hat{p}(t,\theta) = \mathcal{F}^{-1}\{\mathcal{F}p(t,\theta)(U).|U|\}$, et qui exprime f comme la rétroprojection des projections filtrées par le filtre rampe $|U|$. Le filtrage nécessite des opérations 1D, et la rétroprojection est une sommation des contributions de chaque projection.

Les conditions réelles d'acquisition ne permettant de mesurer des projections p_θ que pour un nombre limité d'angles θ_k, et le nombre fini d'éléments du détecteur impliquant des mesures échantillonnées, la fonction f ne sera reconstruite que sur une grille discrète, pour un nombre fini de points. On cherche alors à retrouver les valeurs de f en tout point de la grille discrète $\{f(i,j),\ 0 \leqslant i < N,\ 0 \leqslant j < N\}$ à partir de l'ensemble des mesures de projection $\{p(t_d,\theta_k),\ 0 \leqslant t_d < N_{det},\ 0 \leqslant \theta_k < N_{proj}\}$.

Pour cela, on peut utiliser les méthodes analytiques qui définissent les équivalents discrets des opérateurs (transformée de Radon, rétroprojection, transformée de Fourier) puis discrétisent les formules d'inversion découlant des théorèmes ci-dessus, ou bien des méthodes algébriques qui utilisent une approche différente en discrétisant l'équation de projection fournissant ainsi un système d'équations linéaires résolues par des méthodes algébriques.

Dans ce document, nous développerons davantage les méthodes itératives qui nous permettent de discrétiser l'espace de reconstruction sur une grille non conventionnelle, et laissent la possibilité de traiter des données incomplètes ce que les méthodes analytiques ne permettent pas car seule une partie limitée de l'espace de Fourier est remplie par les données.

1.3.2 Approche itérative

Les méthodes itératives modélisent le problème inverse par une approche discrète-discrète : les projections sont interprétées comme des échantillons d'une fonction à variable discrète, et l'objet est représenté par une fonction inconnue $f(x,y)$, qui est défini comme une combinaison linéaire de fonctions de base dont la plus couramment utilisée est celle des fonctions atomiques de type pixel (voir FIGURE 1.6). Alors :

$$p_j = \sum_i f_i \int \varphi_i(t_d \cos\theta_k - u \sin\theta_k, t_d \sin\theta_k + u \cos\theta_k)\, du \tag{1.10}$$

Sous forme matricielle, le système s'écrit :

$$p = H f \tag{1.11}$$

où le vecteur p contient les mesures des projections. Il est de taille $[N_D \times N_p]$, où N_D est le nombre de pixels détecteur imageant chaque projection et N_p le nombre de projections. Le vecteur f, de taille $[N \times N]$, est l'objet à reconstruire. La matrice H, de taille $[N_D \times N_p, N \times N]$ est la matrice de projection. Elle ne dépend que de la géométrie d'acquisition. Les éléments h_{ij} représentent la contribution de chaque pixel j au rayon i. Dans le cas le plus simple, $h_{ij} = 1$ si le rayon j traverse le pixel i (0 sinon), mais il existe d'autres choix plus coûteux en terme de stockage mémoire : par exemple, le coefficient h_{ij} peut être proportionnel à la longueur du rayon j interceptée dans le pixel i, ou à la surface de recouvrement du rayon i avec le pixel j. La matrice H est très creuse : la plupart de ses éléments sont nuls.

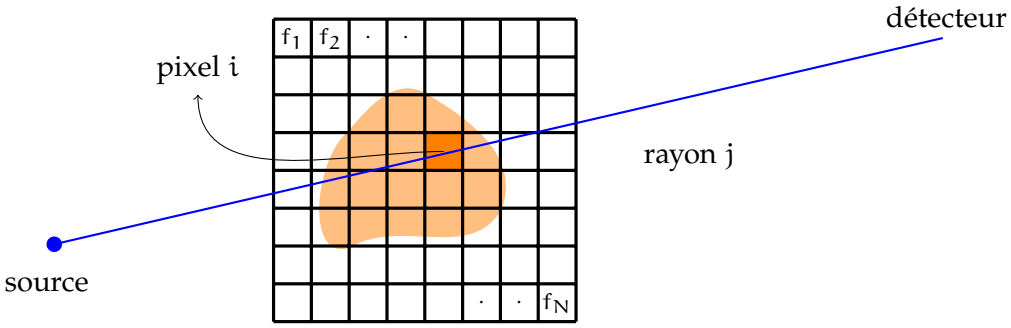


FIGURE 1.6 – Discrétisation 2D sur grille pixélisée.

La matrice H étant très creuse, la résolution du problème (1.11) par inversion directe de la matrice H semble très intéressante. Cependant, en pratique, cette solution n'est pas réalisable car H est souvent de très grande taille, pas nécessairement carrée et surtout très mal conditionnée [15]. Le bruit qui entache les données rend le problème mal posé au sens de Hadamard [54] dans la mesure où au moins l'une des trois conditions suivantes n'est pas vérifiée : existence, unicité et stabilité de la solution.

Du fait de la grande taille de H, la résolution ne peut s'effectuer que par itérations successives : on initialise l'image de façon arbitraire $\hat{f}^0$ (par exemple une image de valeur constante en tout point ou un bruit uniforme gaussien), puis on procède selon un principe d'essai et d'erreur : à chaque itération n, l'estimation de l'image $\hat{f}^n$ est projetée suivant un opérateur de projection, et le résultat $\hat{p}^n$ est comparé aux projections mesurées. On corrige alors l'estimation suivante $\hat{f}^{n+1}$ à partir de l'erreur en appliquant un terme correctif c^n. Dans ce cadre, on distingue les méthodes additives des méthodes multiplicatives, les premières proposant une correction additive de la forme $\hat{f}^n = \hat{f}^{n+1} + c^n$, les secondes une correction multiplicative du type $\hat{f}^n = \hat{f}^{n+1} \cdot c^n$. Ainsi, l'algorithme itératif converge-t-il vers une solution, reconstruisant progressivement les basses puis les hautes fréquences. Afin de maîtriser l'influence du bruit et donc d'empêcher le processus de diverger, celui-ci

doit être contraint. Cette contrainte ou régularisation est traduite par exemple par l'arrêt du processus au bout d'un certain nombre d'itérations.

Opérateurs de projection

Différents opérateurs de projection et de rétroprojection définissant une modélisation discrète du processus d'obtention des mesures ont été proposés. Dans le cas conventionnel d'un objet représenté par des pixels (resp. voxels), on distingue principalement trois types de modèles usuels : *ray-driven*, *pixel-driven*, et *distance-driven*.

Le projecteur *ray-driven* est défini en reliant un rayon du point source au centre du pixel détecteur considéré. La mesure de projection est alors la somme pondérée de tous les pixels traversés par le rayon. Une approche usuelle consiste à pondérer la contribution de chaque pixel par la longueur d'intersection du rayon dans ce pixel [55, 111, 129]. Une interprétation alternative consiste à effectuer une interpolation linéaire entre deux valeurs de pixels pour chaque ligne ou colonne interceptée par le rayon [60]. La rétroprojection est la transposée de cette opération, où la valeur de pixels est mise à jour avec des contributions pondérées provenant de chaque rayon qui le traverse. Les méthodes *ray-driven* modélisent généralement bien la projection mais ont tendance à introduire des artéfacts de type moiré dans le calcul de la rétroprojection.

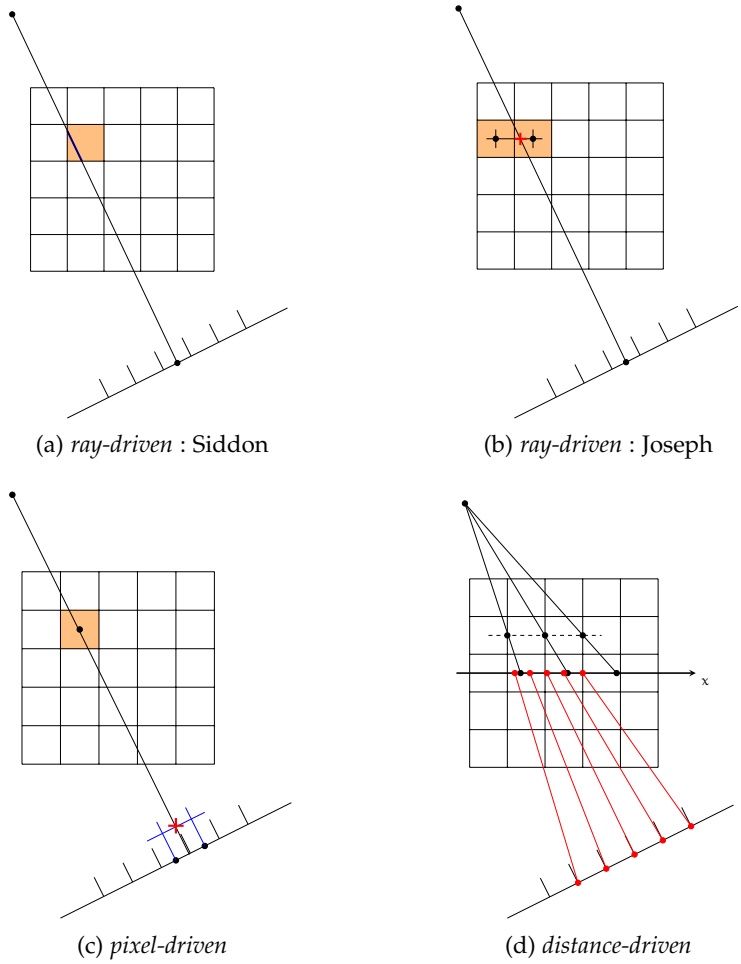
Le projecteur *pixel-driven* est défini par le rayon reliant le point source et le centre du pixel image considéré. La valeur de projection obtenue en un pixel détecteur est calculée par interpolation (linéaire ou des plus proches voisins) des valeurs obtenues aux coordonnées d'intersection des rayons sur la ligne détecteur. Les formes simples de ces projecteurs sont peu utilisées car elles introduisent des artefacts hautes fréquences [129, 80]. Ces artéfacts peuvent être réduits en utilisant des interpolations plus complexes mais au détriment d'une complexité de calcul accrue.

Le projecteur *distance-driven* quant à lui est basé sur la conversion du problème de projection-rétroprojection en un problème rééchantillonné en 1D. Le détecteur et l'image à projeter sont mis en correspondance l'un avec l'autre le long des directions des lignes de projection. Chaque point d'une ligne de l'image est imagé de manière unique sur un point situé sur le détecteur, et vice-versa. Cela permet de définir une longueur de chevauchement entre chaque pixel de l'image, et chaque pixel du détecteur. Cette longueur de recouvrement est calculée en traçant les frontières des pixels d'une ligne d'image d'intérêt sur le détecteur, et toutes les limites de pixels détecteur sur la ligne médiane de l'image. En pratique une seule ligne commune est utilisée [33].

La FIGURE 1.7 illustre les trois types de modèles de projection pour une fonction atomique de type pixel.

Algorithmes algébriques

Dans la littérature, il existe un grand nombre d'algorithmes mettant en oeuvre différentes méthodes mathématiques de résolution de systèmes linéaires pour la reconstruction tomographique. On peut citer par exemple les algorithmes ART (*Algebraic*



(a) *ray-driven* : Siddon (b) *ray-driven* : Joseph

(c) *pixel-driven* (d) *distance-driven*

FIGURE 1.7 – Exemple de projecteurs - (a) et (b) Projecteurs *ray-driven* : un rayon relie le point source à travers l'image au centre d'un pixel détecteur. La mesure du pixel détecteur correspond à la somme pondérée des valeurs des pixels traversés par le rayon- (c) Projecteur *pixel-driven* : un rayon relie la source au centre d'un pixel image ; la valeur d'un pixel détecteur est calculée par interpolation des valeurs obtenues aux intersections des rayons et de la ligne détecteur - (d) *distance-driven* : mise en correspondance de pixels détecteur et des pixels images sur un axe de l'image ; la longueur de chevauchement est utilisée comme pondération [33].

Reconstruction Technique), SIRT (*Simultaneous Iterative Reconstruction Technique*), et leurs variantes (MART, SART, ...), ILST (*Iterative Least Square Technique*), etc. Ces méthodes sont basées sur la méthode de blocs de Kaczmarz [64] qui consiste à diviser le problème en sous-systèmes (contenant plusieurs équations linéaires du système initial) et à appliquer à chacun de ces blocs un algorithme de descente du gradient.

À l'étape $n+1$, l'algorithme de reconstruction peut s'écrire :

$$\hat{f_k}^{n+1} = \hat{f_k}^{n} + \lambda_k c_k^n \tag{1.12}$$

où $k = \{1, \cdots, K\}$ désigne le bloc considéré (K le nombre total de blocs), λ_k et c_k sont respectivement un paramètre réel pouvant varier et un terme de correction.

L'algorithme ART utilise un rayon de projection ($K = 1$) pour corriger un coefficient f_i de f. Il est très utilisé en tomographie mais sa convergence est lente. Par ailleurs, la non négativité de la solution n'est pas toujours assurée, ce qui a suscité le développement de variantes telle que la version multiplicative ART (ou MART) dans laquelle la correction est multiplicative.

L'algorithme SART (*Simultaneous* ART) permet d'accélérer la reconstruction en découpant le problème de telle sorte que chaque bloc contient les équations d'une projection $K = N_D$, c'est-à-dire un ensemble de rayons pour une position θ du système d'acquisition.

Enfin, l'algorithme SIRT affecte toutes les projections à un bloc ($K = N_D \times N_p$). La correction d'un pixel tient compte de l'ensemble des rayons qui le traversent. Ces méthodes convergent moins vite que les méthodes ART mais elles améliorent la qualité de l'image reconstruite notamment dans le cas de données bruitées.

Une autre approche pour résoudre le problème inverse repose sur la recherche d'une solution approchée au sens des moindres carrés :

$$\hat{f} = \arg\min_f J(f) \tag{1.13a}$$

$$J(f) = \|p - Hf\|_2^2 \tag{1.13b}$$

La solution est celle qui minimise l'écart aux données, au sens de la norme L^2. La solution des moindres carrés s'écrit

$$\hat{f} = (H^t H)^{-1} Hf \tag{1.14}$$

La recherche de la solution (1.14) se pose comme un problème d'optimisation dont la minimisation peut être conduite itérativement par des algorithmes tels que la descente de gradient, le gradient conjugué ou l'algorithme de Newton [67].

Cependant, l'instabilité de la solution due au mauvais conditionnement de la matrice H induit l'introduction d'un terme de régularisation dans l'expression (1.13b) traduisant des informations *a priori* sur les propriétés de la solution (par exemple, caractéristique sur l'homogénéité des régions constituant l'image, sur leurs contours, leurs textures). La solution est alors un compromis entre fidélité aux mesures et fidélité aux informations *a priori*.

Algorithmes statistiques

Les méthodes statistiques utilisent une formulation probabiliste du problème : l'incertitude des données mesurées (les projections) est représentée par une loi de probabilité.

On cherche l'image f la plus probable connaissant les projections p observées. En d'autres termes, on cherche à maximiser la probabilité d'avoir f quand les projections valent p par exemple maximiser $P(f|p)$. Et, d'après le théorème de Bayes, on a :

$$P(f|p) = \frac{P(p|f)P(f)}{P(p)} \tag{1.15}$$

où $P(p) = 1$ car les projections p sont connues (mesures), $P(f)$ est la probabilité *a priori* de l'image et vaut donc 1 s'il n'y a pas d'hypothèse *a priori* (pas de régularisation), et $P(p|f)$ est la vraisemblance des projections p. Ainsi, dans le cas où l'on n'introduit pas de régularisation :

$$P(f|p) = P(p|f) \tag{1.16}$$

et maximiser $P(f|p)$ revient à maximiser la vraisemblance $P(p|f)$ (minimiser l'écart entre les projections calculées et observées).

Selon la nature du bruit considérée, gaussienne ou poissonienne, différents types d'algorithmes sont proposés pour résoudre le problème. Dans le cas où la statistique prise en compte pour la détection des photons est la loi poissonienne, les données de projection p_i recueillies sur les différents pixels détecteur sont considérées comme des variables aléatoires, indépendantes entre elles et distribuées selon la loi de Poisson. En regroupant ces mesures dans un vecteur p de moyenne $\bar{p}$, on écrit alors :

$$P(p|\bar{p}) = \frac{(\bar{p})^p \cdot e^{-\bar{p}}}{p!} \tag{1.17}$$

avec $\bar{p} = Hf$ ou $\bar{p_i} = \exp\left(-\sum_j h_{ij} \cdot f_j\right)$.

La probabilité de mesurer p_i dépend uniquement de son espérance mathématique $\bar{p}$ basée sur la distribution f. Alors :

$$P(p_i|\bar{p}) = P(p_i|f) \tag{1.18}$$

D'après (1.17) et (1.18), la vraisemblance $P(p|f)$ s'écrit :

$$P(p|f) = \prod_{i=1}^{M} \frac{(\bar{p_i})^{p_i} \cdot e^{-\bar{p_i}}}{p_i!} \tag{1.19}$$

Les méthodes de maximum de vraisemblance consistent à estimer la distribution de f qui maximise $P(p|f)$, ou d'une façon équivalente son logarithme (log-vraisemblance de p), soit :

$$\ln P(p|f) = \sum_{i=1}^{M} (-\ln(p_i!) + \bar{p_i} \ln(\bar{p_i}) - \bar{p_i}) \tag{1.20}$$

On cherche à résoudre :

$$\hat{f} = \arg\max_f P(p|f) \tag{1.21}$$

Pour cela, on utilise l'algorithme MLEM (*Maximum Likelihood Expectation-Maximization*) (ou une variante) qui, pour chaque itération, consiste en deux étapes [108, 74, 73] :

– une étape d'estimation qui calcule l'espérance de la log-vraisemblance compte tenu des projections p_i mesurées et de l'estimation courante des f_k de l'image :

$$E[\ln(P(z|f)|(p, f^n))] \tag{1.22}$$

z étant une variable auxiliaire non observée.

– une étape de maximisation résoud les équations de la vraisemblance étant donnée l'estimation courante de l'objet, ce qui revient à maximiser l'espérance en annulant les dérivées partielles par rapport à f_k :

$$\frac{\partial}{\partial f_k} E[\ln(P(z|f)|(p, f^n))] = 0, \forall k \tag{1.23}$$

Le schéma itératif peut s'exprimer sous la forme suivante :

$$f_j^{n+1} = f_j^n . \frac{\sum_{i \in J_j} \exp\left(-\sum_{k \in I_i} h_{ik} f_k^n\right) . h_{ij}}{\sum_{i \in J_j} p_i h_{ij}} \tag{1.24}$$

où J_j désigne l'ensemble des projections passant par le pixel image j et I_i l'ensemble des pixels image contribuant au rayon de projection i.

Cet algorithme converge lentement (il peut exiger entre 50 et 200 itérations), et les solutions sont de plus en plus bruitées au fur et à mesure des itérations. Le manque de régularisation rend la solution instable. Le seul moyen de régulariser est de stopper l'algorithme au bout d'un nombre donné d'itérations, ce qui en soi n'est pas très performant et dépend des données traitées. La version OSEM, ou *Ordered Subset EM*, permet d'accélérer sensiblement l'algorithme MLEM en triant et regroupant l'ensemble des projections p en sous-ensembles ordonnés. Le facteur d'accélération est alors directement proportionnel au nombre de sous-ensembles utilisés.

Une façon de généraliser cette approche est d'introduire des informations *a priori* sur les paramètres inconnus f et sur les données p à travers des quantités probabilistes. La distribution *a posteriori* de f est donnée par la relation de Bayes (1.15). Le problème principal réside dans la conversion de l'information *a priori* en une loi de probabilité. Ce problème difficile est largement étudié, et nous n'aborderons pas ici le sujet.

Dans cette première partie, nous avons vu que les méthodes de reconstruction itératives nécessitent une modélisation discrète du processus d'obtention des mesures. La représentation de l'objet d'intérêt est le point de départ à l'élaboration d'un modèle de projection tomographique sur le détecteur. La représentation conventionnelle de l'espace de reconstruction en tomographique est la représentation pixélisée (voxélisée), c'est-à-dire une grille régulière constituée de carrés (cubes). Nous abordons dans la prochaine section d'autres façons de discrétiser une image, pour ensuite utiliser une autre base de fonctions atomiques pour la représentation de l'objet à reconstruire. La finalité d'un changement de base est de diminuer le nombre d'inconnues pour réduire le temps de reconstruction des méthodes itératives, et se donner les moyens d'introduire une de ces méthodes dans un processus global de reconstruction simultanée à la segmentation de l'objet estimé. De façon plus générale, il s'agit de rendre les méthodes itératives plus usuelles.

1.4 Représentation d'une image

Une image f est un signal multidimensionnel réel et continu qui est souvent approximé par un nombre fini de fonctions atomiques, $\varphi(x)$ afin de pouvoir être traité numériquement.

Ainsi, une fonction continue peut être approchée par une somme finie de fonctions :

$$f(x) \approx \sum_k f_k \varphi_k(x - x_k) \tag{1.25}$$

$k = \{k_1, k_2, ..., k_n\} \in \mathbb{Z}^n$ correspond aux indices des points de la grille discrète dans l'espace de dimension n, $x_k = \{x_{k_1}, x_{k_2}, ..., x_{k_n}\} \in \mathbb{R}^n$ sont les coordonnées sur la grille discrète, et $x = \{x_1, x_2, ..., x_n\} \in \mathbb{R}^n$ celles des points continus de f.

Le choix de la base de fonctions atomiques $\varphi(x)$ agit sur la modélisation de la fonction continue mais aussi sur la cohérence du projecteur avec les mesures. Ce choix influe aussi sur la complexité des algorithmes et la taille mémoire nécessaire. Les prochains paragraphes exposent différents types de fonctions atomiques proposés dans la littérature.

1.4.1 Fonction constante

Une première représentation consiste à prendre comme fonction atomique $\varphi(x)$ une fonction constante par morceaux telle que :

$$\varphi(x) = \begin{cases} 1 & \text{si } |x| < \delta \\ 0 & \text{sinon} \end{cases} \tag{1.26}$$

Cette formulation est une discrétisation avec un pas d'échantillonnage δ. Cette représentation a un comportement anisotropique, ce qui peut amener vers une modélisation grossière de la fonction continue si le pas d'échantillonnage n'est pas assez fin. Cependant prendre un pas d'échantillonnage δ petit augmente considérablement le coût de calcul et la taille mémoire. Par contre, le lien direct entre $(x, y) \rightarrow (i, j)$ [†] permet de réduire la complexité de la mise en œuvre des algorithmes de traitement d'images.

La plupart des algorithmes de reconstruction en tomographie utilisent cette discrétisation du volume de reconstruction. Ceci permet de simplifier le coût de calcul de l'opérateur de projection (SECTION 1.3.2). Lorsque δ est petit, la taille importante du volume de reconstruction et de la matrice de projection H peut engendrer un problème de limitation en taille mémoire sur un ordinateur commun.

1.4.2 Base d'ondelettes

La représentation en base d'ondelettes permet une approche multirésolution de l'objet. Les ondelettes permettent une représentation parcimonieuse et locale d'une image. C'est-à-dire que l'on peut décomposer une image dans la base d'ondelettes en utilisant un nombre d'atomes mineurs qui permettent de tenir compte des variations locales. Un exemple d'ondelette est donné par la fonction de Haar :

$$\varphi(x) = \begin{cases} 1 & 0 \leqslant x < 1/2 \\ -1 & 1/2 \leqslant x < 1 \\ 0 & \text{sinon} \end{cases} \tag{1.27}$$

†. (i, j) sont les indices pour accéder au tableau 2D qui définit l'image

En imagerie, on trouve également les bases telles que les *ridgelets* [16, 37], les *wedgelets* [36] ou encore les *contourlet*[35].

La mise en œuvre des bases d'ondelettes en reconstruction tomographique permet d'obtenir des volumes de reconstruction en haute résolution mais impliquent des algorithmes de reconstruction plus complexes dans lesquels intervient la transformée en ondelettes [30].

1.4.3 Fonction radiale

Une autre représentation de l'image a été proposée par Lewitt [76] en utilisant une fonction radiale, aussi appelée *blob*. La fonction *blob* est définie par une généralisation de la fonction de Kaiser-Bessel. Cette fonction a la caractéristique d'être symétrique avec une forme semblable à une cloche qui décroît doucement vers zéro. La fonction *blob* est définie par :

$$\varphi(x) = \begin{cases} \dfrac{I_m\left(\alpha\sqrt{1-\left(\frac{x}{a}\right)^2}\right)}{I_m(\alpha)}\left(\sqrt{1-\left(\frac{x}{a}\right)^2}\right)^m & \text{si} \quad 0 \leqslant x \leqslant a \\ 0 & \text{sinon} \end{cases} \tag{1.28}$$

où α contrôle la largeur de la forme, a son amplitude et I_m est la fonction de Bessel modifiée d'ordre m. Une image représentée par des *blobs* apparaît lisse, homogène et douce par rapport à une image représentée par une fonction constante. Cette base de fonctions atomiques a tendance à adoucir les contours des objets ; ils apparaissent donc moins nets. Par ailleurs, afin de couvrir tout le domaine image les *blobs* doivent se superposer. En général, les *blob* sont placés sur des grilles cartésiennes ou hexagonales. Même si le *blob* permet une meilleure approximation de l'image que les fonctions continues, ils engendrent une complexité de calcul plus importante.

Utilisée dans la reconstruction tomographique 3D par Lewitt [77, 119], la base de fonctions de type *blob* permet de représenter l'image avec moins de fonctions atomiques, ce qui réduit la taille du problème. En revanche, l'opérateur de projection devient complexe et donc coûteux en temps de calcul.

1.4.4 Fonction polynomiale

Une autre base de fonctions atomiques est donnée par les B-splines reconnues pour leur reproduction fidèle de la constante et la conservation des bords francs. Les B-splines sont définies par le degré h des fonctions polynomiales qui la composent. La base de fonctions proposée par [83] s'exprime par :

$$b(x) = \prod_{q=0}^{h} \beta_q^0(x) \tag{1.29}$$

où $b(x)$ représente la B-spline de ordre h, obtenu par convolution de un nombre h de β^0 †.

†. β^0 est un function polynomiale d'ordre zéro

Pour h grand, les B-splines sont une bonne approximation de la fonction *blob*. C'est pourquoi les B-splines peuvent remplacer les fonctions de type *blob* pour représenter le volume de reconstruction en tomographie [83, 39], d'autant plus que l'approche polynomiale permet d'obtenir une approximation de l'image avec un coût de calcul plus modeste.

1.4.5 Fonction linéaire nœudale

Dans le contexte du codage vidéo, l'utilisation de maillages déformables a permis de compresser les données à transmettre [120]. À partir d'un photogramme, on génère un maillage constitué d'éléments polygonaux, tels que le triangle ou le quadriangle, qui couvrent le domaine de l'image sans se superposer. Ce maillage est adapté au contenu du photogramme par déplacement des nœuds (sommets des éléments). L'image est obtenue par interpolation linéaire de la position et de la valeur des nœuds. Cette fonction s'inspire des éléments finis de Lagrange du premier ordre [32, 7] :

$$\varphi(x) = \begin{cases} 2x & 0 \leqslant x < 1/2 \\ 2(1-x) & 1/2 \leqslant x \leqslant 1 \\ 0 & \text{sinon} \end{cases} \tag{1.30}$$

À partir des conclusions de [120], Brankov *et al.* utilisent la représentation par maillage adaptatif dans le cadre de la reconstruction tomographique par émission (SPECT) [12]. Cette représentation est un compromis entre la fonction constante et la fonction polynomiale.

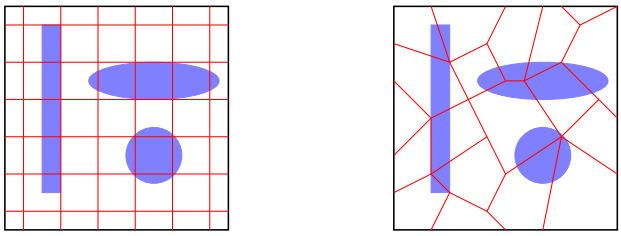
1.5 Pavages

Dans la section précédente, nous avons décrit en substance les fonctions atomiques les plus usitées en traitement d'images pour approcher une fonction continue. Ainsi, l'échantillonnage de l'espace continu $\mathbb{R}^n$ dans l'espace discret $\mathbb{Z}^n$ simplifie la représentation de l'image. Pour cela, on réalise un pavage du plan (l'image), c'est-à-dire une partition de l'espace continu $\mathbb{R}^n$ en cellules (ou pavés) élémentaires contiguës remplissant tout l'espace.

Un pavage peut être réalisé par un ensemble de points distribués dans l'espace régulièrement ou manière aléatoire (voir FIGURE 1.8).

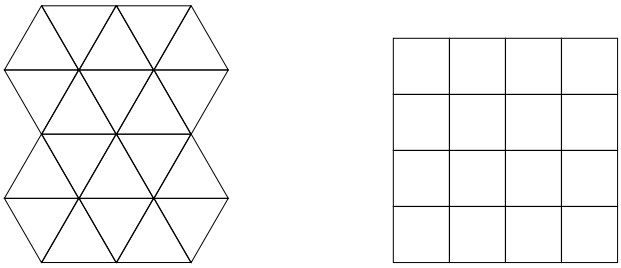
1.5.1 Pavages réguliers

Les pavages réguliers sont constitués de cellules de base de type polygone régulier qui, par translation à partir de l'origine, remplissent tout l'espace. La FIGURE 1.9 montre les pavages réguliers de type triangulaire, carré, hexagonal et octogonal. On peut remarquer que les pavages triangulaire, carré et hexagonal recouvrent entièrement l'espace, contrairement au pavage octogonal. Afin de compléter les ouvertures, il faut ajouter des carrés au

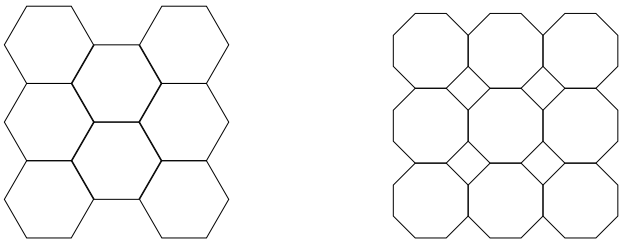


(a) Pavage régulier (b) Pavage irrégulier

FIGURE 1.8 – Exemple de pavages de l'espace.



(a) Pavage triangulaire (b) Pavage carré



(c) Pavage hexagonal (d) Pavage octogonal

FIGURE 1.9 – Exemple de pavages réguliers utilisés pour discrétiser un espace dans $\mathbb{R}^2$.

centre des octogones. Ainsi, les seules possibilités de pavage régulier dans $\mathbb{R}^2$ sont les pavages triangulaire, carré et hexagonal.

Une propriété importante caractérise le pavage régulier : la récursivité. Cette propriété permet de décomposer de manière récursive un pavage en cellules de propriétés géométriques identiques mais plus petites. Cette propriété est à la base des représentations hiérarchiques de type *quadtrees* ou les pyramides morphologiques [71], qui sont particulièrement importantes pour modifier la résolution spatiale des images localement. Cette propriété ne concerne que les pavages carrés et triangulaires.

1.5.2 Pavages irréguliers

Les pavages irréguliers sont aussi appelées diagrammes de Voronoi [38]. Celui-ci est défini par un ensemble donné de points, appelés germes, contenus dans l'espace

euclidien, et consiste en la décomposition de l'espace en régions appelées cellules. Les régions sont des polytopes convexes en tant qu'intersection de demi espaces. En 2D, la partition est relativement simple : le diagramme se construit en déterminant la médiatrice de chaque couple de germes, celle-ci définissant la frontière entre les cellules de deux germes distincts.

Si $\mathcal{P} = \{p_1 ..., p_n\}$ $p \in \mathbb{R}^n$ définit l'ensemble des points de l'espace, alors on associe à chaque point p_i, la région de Voronoi $V(p_i)$ telle que :

$$V(p_i) = \left\{ x \in \mathbb{R}^n \ : \ \|x - p_i\| \leqslant \|x - p_j\|, \forall j \leqslant n \right\} \tag{1.31}$$

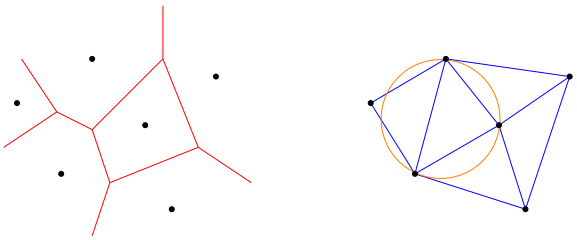


FIGURE 1.10 – Diagramme de Voronoi et triangulation de Delanauy.

Le diagramme de Voronoi définit de manière naturelle le domaine d'influence autour de chaque site. Alors, les relations d'adjacence entre les cellules fournissent une construction naturelle d'un graphe de voisinage appelé triangulation de Delaunay [8, 10]. La FIGURE 1.10 présente un diagramme de Voronoi et la triangulation de Delaunay associée.

Soit $\mathcal{P} = \{p_1, ..., p_n\}$ $p_i \in \mathbb{R}^2$, $i = \{1, ..., n\}$, un ensemble de points. La triangulation de Delaunay de l'ensemble $\mathcal{P}$ est une triangulation telle qu'aucun point de l'ensemble $\mathcal{P}$ n'est contenu dans le cercle circonscrit d'un des triangles de cette triangulation. La triangulation de Delaunay maximise le plus petit angle de l'ensemble des angles des triangles, évitant ainsi les triangles « allongés ».

Une triangulation de Delaunay contrainte est une triangulation contrainte qui tente d'être autant que possible de Delaunay. On entend par triangulation contrainte une triangulation d'un ensemble de points $\mathcal{P}$ devant prendre en compte parmi ses arêtes un ensemble donné de segments reliant des points entre eux ; ces arêtes sont dites contraintes. Dans ce cas, les triangles d'une triangulation de Delaunay contrainte ne répondent pas forcément à la propriété de cercle circonscrit vide mais à une propriété plus faible. Cette propriété est que le cercle circonscrit de n'importe quelle facette n'enferme aucun sommet visible depuis l'intérieur de cette facette (voir FIGURE 1.11).

1.6 Conclusions

Dans ce chapitre, nous avons décrit le principe physique de la tomographie par rayons X. Nous avons également présenté les différents types d'algorithmes de reconstruction qui permettent d'obtenir la cartographie des coefficients d'atténuation d'un objet à partir de ses projections Nous nous sommes intéressés aux algorithmes itératifs qui permettent la

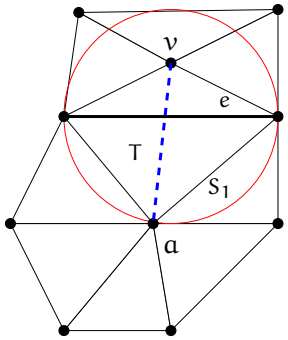


FIGURE 1.11 – Exemple de triangulation de Delaunay contrainte.

reconstruction d'un objet à partir de données incomplètes par l'introduction d'un terme de régularisation ou d'information *a priori*, au détriment d'un coût de calcul plus élevé que les méthodes analytiques. Dans la suite du travail, nous proposons d'exploiter l'algorithme statistique de maximum vraisemblance MLEM et celui du gradient conjugué dans le processus de reconstruction associé à la segmentation de l'objet. Ce choix est arbitré sur les bases de la simplicité de mise en oeuvre de ces algorithmes, dans le but de les adapter à une grille de discrétisation autre que celle usuellement utilisée dans la reconstruction tomographique.

Cette nouvelle représentation du volume de reconstruction sera basée sur un pavage de type irrégulier, de façon à adapter le maillage au contenu de l'image estimée. La maille élémentaire sera le triangle, et l'ensemble des points de l'espace constituant le maillage sera en partie défini par les contours de l'objet étudié. Pour cela, nous avons mis en oeuvre et intégré au processus de reconstruction global une méthode de segmentation. Le chapitre suivant est consacré à la présentation des deux méthodes de segmentation que nous avons retenues dans ce cadre-là.

Chapitre 2

Détection des contours

Dans le chapitre précédent, nous avons décrit le principe de la tomographie par rayons X ainsi que les méthodes de reconstruction qui permettent d'obtenir des images de l'objet étudié. Dans la plupart des applications (en imagerie industrielle et médicale), le but est ensuite d'isoler une ou plusieurs régions d'intérêt afin de procéder à une analyse et d'en tirer les caractéristiques souhaitées. On peut citer, par exemple, l'orientation et la taille d'une fissure ou les dimensions d'un défaut dans le domaine du Contrôle Non Destructif (CND), ou encore la grosseur et la forme d'une tumeur ou les contours d'un organe ou d'un os en imagerie médicale. L'étape de segmentation est donc une étape essentielle à l'analyse d'une image. Cependant le problème est évidemment mal posé car on se sait jamais dire quelle est la segmentation idéale. Il existe donc de nombreuses techniques et différentes interprétations des contours.

Dans ce contexte, afin de rendre la segmentation des images traitées indépendante de tout opérateur et continue d'un examen à un autre, nous proposons d'intégrer cette étape dans le processus de reconstruction tomographique. Dans le même temps, afin de réduire le nombre d'inconnues à estimer au cours des itérations (et par conséquent le temps de calcul), nous simplifierons la grille de reconstruction en adaptant les éléments de volume aux contours des objets (matériaux) segmentés.

Après une classification non exhaustive des méthodes existantes, nous consacrons ce chapitre aux méthodes de segmentation qui utilisent des modèles géométriques par contours actifs paramétriques et géodésiques, méthodes que nous avons intégrées et testées dans le processus global de reconstruction développé dans ce travail de thèse.

2.1 Généralités

La segmentation d'image est l'étape majeure conduisant à l'analyse des données. C'est une étape critique qui conditionne la qualité des mesures effectuées ultérieurement. Il s'agit principalement d'isoler dans l'image les objets sur lesquels va porter l'analyse, de séparer les structures d'intérêt du fond. Il existe de nombreuses méthodes de segmentation d'images pour lesquelles les principales difficultés sont dues à la complexité des images et l'ambiguïté des données correspondantes, ainsi qu'à l'incertitude et au bruit associés à

ces données.

Bien que la nomenclature utilisée dans la littérature est souvent ample, et que la classification des méthodes varie (certaines techniques peuvent être affectées à différents groupes), nous proposons de suivre celle de Erdt *et al.* [41] qui distinguent les méthodes statistiques (histogrammes) et géométriques (*Region growing, Split and Merge*) qui utilisent la valeur des pixels, des méthodes par optimisation (contours actifs, champs de Markov) qui définissent une fonctionnelle de coût ou une stratégie de parcours de l'image.

Les méthodes par histogrammes sont en général rapides à calculer et peu sensibles au bruit mais elles n'intègrent pas (ou peu) d'information géométrique ou topologique sur les régions. Ce sont des méthodes globales au sens où la décision d'appartenance d'un pixel dépend toujours de l'image entière. A l'autre extrême, on peut considérer les méthodes locales pour lesquelles l'étiquettage d'un pixel dépend uniquement, ou essentiellement, de son voisinage (par exemple la segmentation par croissance de régions).

Dans les méthodes par optimisation, le problème de la segmentation est formalisé par l'estimation d'une fonction bidimensionnelle qui doit avoir certaines propriétés (régulière, constante par morceaux, aux bords irréguliers, etc) tout en étant proche de l'image analysée. On recherche un compromis entre ces différentes propriétés antagonistes en minimisant une fonctionnelle de coût qui va dépendre de l'image analysée, de la répartition calculée, des contours associés à la segmentation et de la fonction recherchée qui est représentée par ses restrictions sur chaque région.

Étant donnée une image, il existe toujours plusieurs segmentations possibles. Une bonne méthode de segmentation est donc celle qui permet d'arriver à une bonne interprétation. Elle doit donc avoir simplifié l'image sans pour autant en réduire trop le contenu.

La FIGURE 2.1 propose un schéma de classification hiérarchique des méthodes de segmentation, séparant les méthodes basées sur l'image des méthodes basées sur un modèle géométrique. Cette première division est elle-même subdivisée en sous-classes l'une considérant une information locale et l'autre une information globale.

2.1.1 Méthodes basées sur l'image

Mise en œuvre d'une information locale

Une approche classique de segmentation d'une image est le seuillage. Cette méthode consiste à fixer un seuil au-dessus duquel le pixel appartient à la région sinon il est considéré comme appartenir au fond. La valeur du seuil peut être choisie de façon empirique, à partir de l'histogramme de l'image ou encore en considérant les propriétés locales de l'image par exemple la valeur moyenne des pixels voisins.

Les approches par filtrage linéaire définissent ce qu'en littérature on appelle la détection de contours. Les détecteurs de contours s'appuient sur la localisation d'un maximum ou minimum de la dérivée première ou d'un passage par zéro d'une dérivée seconde. L'estimation de la dérivée est obtenue par convolution d'un masque avec l'image. Il existe différents types de filtres ; on peut citer les filtres de Roberts, de Prewitt, de Sobel ou

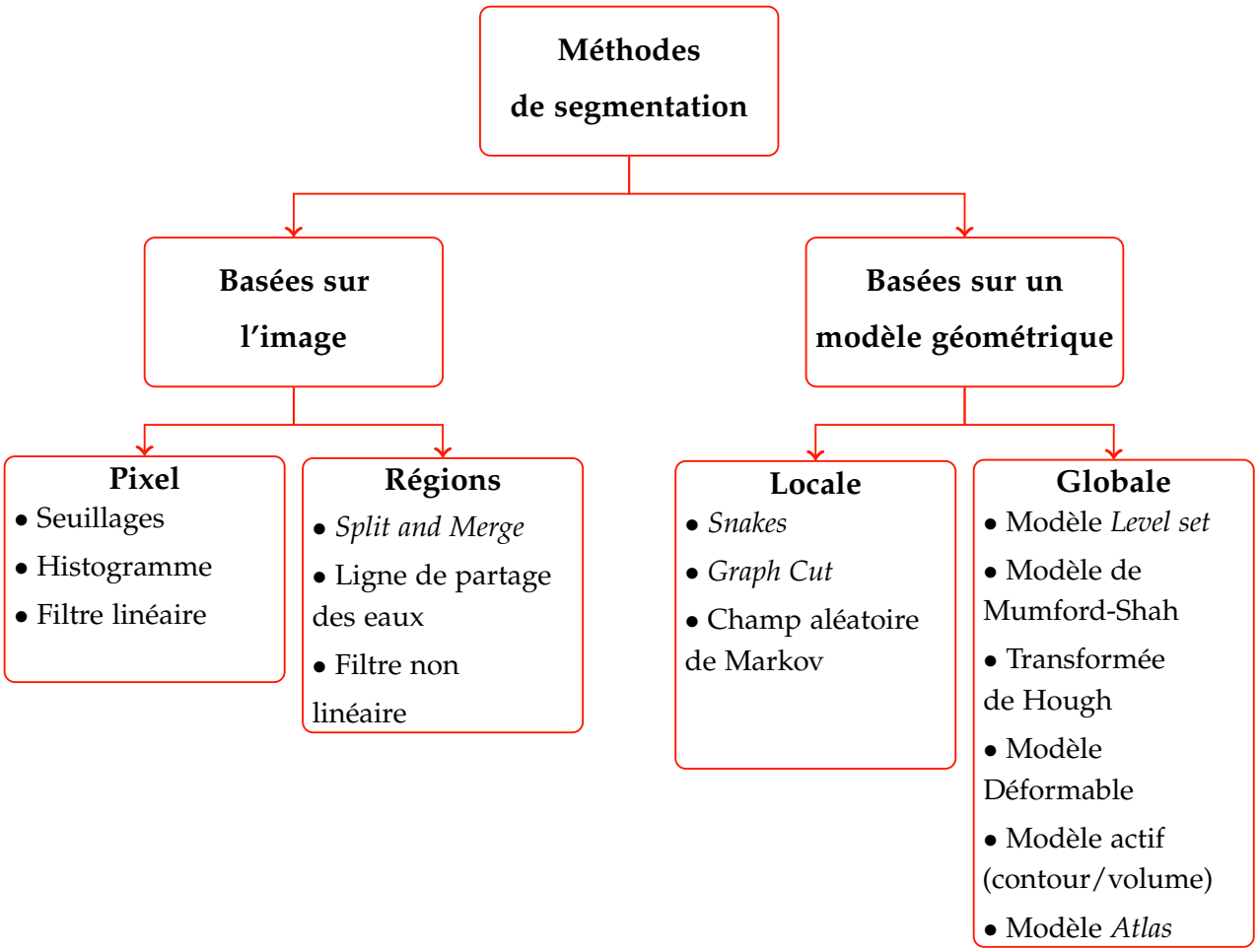


FIGURE 2.1 – Exemple de classification des méthodes de segmentation.

de Frei-Chen [101, 52], dont les tailles des masques peuvent varier. Dans le détecteur de passage par zéro du laplacien, on trouve l'opérateur de Marr et Hildreth qui est considéré comme le prototype des détecteurs de contours [81]. Cet opérateur est très sensible au bruit, à cause de l'estimation de la dérivée seconde ; souvent l'image est donc filtrée par un filtre gaussien. Cette opération donne lieu à un nouvel opérateur : la différence de gaussiennes. Depuis, un travail original sur les opérateurs de détection de contours a été proposé par Canny [17] qui définit le critère d'un filtre linéaire optimal par la qualité de détection, la localisation et l'unicité de la détection.

Ces méthodes sont très rapides pour calculer une segmentation, mais elles n'intègrent pas d'information géométrique ou topologique sur les régions : elles sont très fortement dépendantes du contenu de l'image entière.

Mise en œuvre d'une information globale

Les méthodes de segmentation utilisant une infomation globale sont basées sur les informations ou les relations qui sont communes à un groupe de pixels. Celles-ci peuvent concerner la valeur moyenne ou bien des critères géométriques d'un pixel par rapport à ses voisins, en considérant par exemple une fenêtre de taille donnée. Contrairement aux méthodes précédemment citées, celles-ci permettent de déterminer les aspects topologiques des régions, celles-ci étant souvent déterminées à partir d'un germe (pixel) initialement choisi par l'opérateur. Dans le cas d'une image très bruitée, le résultat d'une segmentation correcte n'est pas garanti [4].

Dans ce groupe de méthodes, on peut évoquer la méthode *split-and-merge*. L'idée de cette approche consiste dans un premier temps à diviser (*split*) l'image en blocs contenant exclusivement des pixels répondant à un critère donné, puis à regrouper (*merge*) les blocs voisins s'ils sont similaires. Le résultat final est un ensemble jointif de régions homogènes de différentes tailles recouvrant l'image entière. L'implantation de cette méthode s'effectue à travers la mise en œuvre d'une structure appelée *quadtree*.

On peut également évoquer l'algorithme de la ligne de partage des eaux (ou *watershed*) [9] qui considère une image en niveaux de gris (NG) comme un relief. En innondant ce relief, des lacs se forment et les barrages les délimitant constituent la segmentation finale. L'inconvénient majeur de cette technique est la sursegmentation. Diverses méthodes de fusion de régions ont été proposées pour palier ce problème, par exemple la fusion itérative de paires de régions voisines [94, 95]. Les contours des régions segmentées sont caractérisés par l'effet de carré (pixel). Ainsi on perd l'information des régions contenant des contours lisses.

2.1.2 Méthodes basées sur un modèle géométrique

Mise en œuvre d'un *a priori* local

Les modèles déformables basés sur un *a priori* local s'appuient sur la modélisation d'une courbe ou d'une surface paramétrique pour segmenter les régions d'intérêt. La courbe est déformée sous l'action de forces interne et externe. La force interne liée à la courbe agit sur l'élasticité ou la rigidité de la courbe elle-même. La force externe, elle, attire la courbe vers le contour des objets.

A la différence des méthodes précédentes, les modèles déformables sont caractérisés par une équation de mouvement dont la solution est obtenue par minimisation d'une fonctionnelle. Les contours déformables ont trouvé une large implantation dans la segmentation des images médicales, car la forme connue des organes permet d'incorporer un *a priori* dans les contours déformables afin d'améliorer leur robustesse en présence du bruit texturé ou d'occultations partielles [82, 104, 87, 5].

Une des principales difficultés de ces modèles réside dans le choix de paramètres appropriés permettant d'initialiser la courbe. En effet, la courbe doit être placée à proximité de la région à segmenter afin d'obtenir une segmentation correcte.

Les méthodes fondées sur les coupes de graphes (*Graph Cut*) [11, 49] permettent d'optimiser des fonctions de coût de manière globale, et ainsi d'éviter les minima locaux. Par ailleurs, dans les applications où la forme à segmenter est connue *a priori*, le processus de segmentation peut être guidé par un modèle de forme ou des contraintes sur celle-ci [31, 53]. L'avantage de cette méthode des coupes de graphes est sa capacité à donner efficacement une solution optimale pour l'utilisation conjointe de différentes informations sur l'image. Peu de travaux utilisent ce type de méthodes notamment à cause de la modélisation de la forme de l'objet et de son intégration dans l'algorithme.

Mise en œuvre d'un *a priori* global

Les modèles déformables basés sur un *a priori* global s'appuient sur la modélisation en utilisant l'évolution de la courbe ou la surface non paramétrique. On peut citer par exemple les *level set* [46, 3, 24]. Ces méthodes permettent de s'affranchir des problèmes issus de l'initialisation ou de la paramétrisation des courbes. L'évolution de la courbe est conduite par l'optimisation d'un critère, ce qui implique la mise en œuvre d'algorithmes de minimisation souvent complexes et lourds en temps de calcul.

Dans ce type d'algorithme, on peut également mentionner les méthodes de segmentation par recalage d'atlas. L'atlas est un modèle de référence de la région à segmenter. Il s'agit ensuite de recaler ce modèle dans l'image par un modèle déformable. Par exemple, ce type d'approche est utilisé en oncologie, pour mesurer la taille d'une tumeur ou caractériser sa forme. Les régions d'intérêt sont recalées sur une image issue d'une reconstruction de tomodensitométrie ou de résonance magnétique [122, 58, 28]. Concernant ces approches, la qualité de la segmentation dépend de la méthode de recalage et de la manière de créer le modèle de l'atlas.

Dans ce travail de thèse, nous nous sommes intéressés à la segmentation d'images au cours du processus de reconstruction tomographique (RX). Il s'agit d'images bruitées, et ce d'autant plus que la segmentation intervient précocement dans le processus itératif de reconstruction. Par ailleurs, notons que l'estimation de la valeur du niveau de gris des pixels évolue au cours du processus global de reconstruction. Ces considérations ne nous permettent pas d'envisager l'exploitation de méthodes basées sur l'image.

Nous avons choisi de nous tourner vers la deuxième classe de méthodes basées sur des modèles géométriques. Dans un premier temps, nous avons considéré un modèle déformable paramétrique (ou *snake*), puis dans un second temps nous avons mis en œuvre une méthode d'ensembles de niveaux (ou *level set*). Dans la suite, nous présentons ces méthodes plus en détail.

2.2 Modèle déformable

Les modèles déformables sont exprimés sous la forme d'un problème variationnel, où l'on utilise l'équation d'Euler-Lagrange pour décrire l'évolution du modèle. Dans ce travail de thèse, nous considérons deux types de modèle déformable : le premier utilise une représentation du contour par une courbe paramétrique [66], et le deuxième une représentation du contour par fonction implicite [18].

2.2.1 Modèle déformable paramétrique : *snake*

Le modèle paramétrique classique de contour actif déformable ou *snake* a été introduit par Kass *et al.* [66]. Le contour est défini par une courbe (en général une spline)

$$\gamma(s,t) = (x(s), y(s)), \quad s \in [0,1] \tag{2.1}$$

qui se déplace à l'intérieur du domaine de l'image en fonction du temps (itérations) (voir FIGURE 2.2). Le *snake* est influencé par une force interne, F_{int} liée à la flexion de la courbe, et une force externe F_{ext} qui attire la courbe vers les discontinuités, par exemple vers les contours de l'objet.

Ainsi, le *snake* est caractérisé par l'énergie fonctionnelle $\mathcal{E}$, qui peut être écrite comme

$$\mathcal{E} = \int_0^1 \left(E_{int}(\gamma(s,t)) + E_{ext}(\gamma(s,t))\right)\, ds \tag{2.2}$$

Énergie interne

L'énergie interne de la courbe est décrite comme la somme deux termes (2.3) :

- la dérivée première tient compte des variations de la longueur de la courbe ; elle est contrôlée par l'élasticité α du contour
- la dérivée seconde décrit les variations de la courbure ; elle est contrôlée par la raideur β du contour

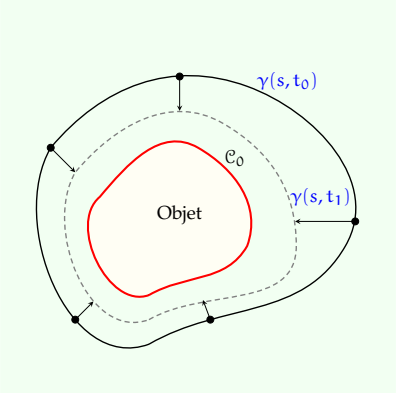


FIGURE 2.2 – Exemple d'évolution d'un *snake* (en noir) d'une itération à l'autre dans le but d'atteindre le contour de l'objet C_0 (en rouge).

$$E_{int} = \frac{1}{2}\left(\alpha\left|\gamma'(s)\right|^2 + \beta\left|\gamma''(s)\right|^2\right) \tag{2.3}$$

α et β sont des paramètres de pondération qui contrôlent la déformabilité du contour. Par exemple, si $\beta = 0$, le terme de la dérivée seconde est nulle, et la courbe ne peut pas représenter d'angles aigus.

Énergie externe

L'énergie externe, qui concerne l'image, est représentée par une fonctionnelle de potentiel qui attirant la courbe vers les bords de l'objet de l'image. Kass *et al.* [66] décrivent l'énergie externe comme une combinaison pondérée de deux énergies (2.4) fonctionnelles afin de contraindre le contour à s'approcher d'une forme donnée.

$$E_{ext} = w_{image}.E_{image} + w_{term}.E_{term} \tag{2.4}$$

La composante la plus importante de l'énergie externe est l'énergie de l'image E_{image}. Celle-ci est définie par la norme du gradient et permet d'identifier les caractéristiques de l'image telles que les discontinuités.

$$E_{image} = -|\nabla I(x, y)|^2 \tag{2.5}$$

où I est l'image à segmenter.

Ainsi, la courbe est attirée vers les zones de l'image avec des forts gradients. Cette énergie peut être considérée comme un opérateur de détection de contours. Pour améliorer la stabilité de la solution dans le cas d'une image bruitée, l'utilisation d'un filtre à noyau gaussien peut être mise en œuvre, et l'énergie de l'image E_{image} peut se réécrire ainsi :

$$E_{image} = -|\nabla\left[I(x, y) * G_\sigma(x, y)\right]|^2 \tag{2.6a}$$

avec G_σ le noyau gaussien qui est caractérisé par l'écart-type σ.

Afin d'améliorer la détection des coins (ou angles aigus), une énergie supplémentaire est considérée : l'énergie de terminaisons. Celle-ci permet d'identifier les terminaisons et les coins. Elle est représentée par la courbure κ des contours présents dans l'image. La courbure de la ligne de niveau peut s'écrire [107]

$$\kappa = \nabla \cdot \left(\frac{\nabla I}{|\nabla I|} \right) \tag{2.7}$$

En développant κ (voir ANNEXE A), l'énergie de terminaisons s'exprime :

$$E_{term} = \frac{I_x^2 I_{yy} - 2 I_x I_y I_{xy} + I_y^2 I_{xx}}{|\nabla I|^3} \tag{2.8}$$

où I est ici l'image convoluée au noyau gaussien [66].

La combinaison des énergies de bord et de terminaison donne à la courbe la caractéristique de se coller aux bords (contours) ou aux coins (angles) de l'objet à segmenter.

En accord avec le calcul variationnel, le contour $\gamma(s)$ qui minimise l'énergie (2.2) doit satisfaire l'équation d'Euler :

$$\frac{\partial}{\partial s}\left(\alpha\gamma'(s)\right) - \frac{\partial^2}{\partial s^2}\left(\beta\gamma''(s)\right) - \nabla E_{ext} = 0 \tag{2.9}$$

Celle-ci peut être vue comme un équilibre entre les forces interne F_{int} et externe F_{ext} :

$$F_{int} + F_{ext} = 0 \tag{2.10}$$

avec :

$$F_{int} = \frac{\partial}{\partial s}\left(\alpha\gamma'(s)\right) - \frac{\partial^2}{\partial s^2}\left(\beta\gamma''(s)\right) \tag{2.11}$$

et

$$F_{ext} = -\nabla E_{ext} \tag{2.12}$$

La force interne maîtrise l'allongement et le fléchissement de la courbe pendant que la force externe déplace le contour vers les caractéristiques de l'image. Pour résoudre le problème (2.9), on introduit une variable temporelle t, qui rend le contour dynamique :

$$\gamma(s,t) = (x(s,t), y(s,t)) \tag{2.13}$$

Le problème (2.9) avec la définition de contour dynamique (2.13) peut être décrit par l'équation de mouvement de Lagrange :

$$\frac{\partial}{\partial t}\gamma(s,t) = \frac{\partial}{\partial s}\left(\alpha\gamma'(s,t)\right) - \frac{\partial^2}{\partial s^2}\left(\beta\gamma''(s,t)\right) - \nabla E_{ext} \tag{2.14}$$

Lorsque la force interne équilibre la force externe, l'équation d'Euler est satisfaite [115].

2.2.2 *Snake* basé sur le flux du vecteur gradient

Les modèles déformables basés sur le contour actif sont souvent renforcés par l'utilisation d'informations *a priori* sur la forme de l'objet. Dans certains problèmes, les informations *a priori* sont bien modélisées par des modèles statistiques, et permettent une bonne segmentation des images. C'est le cas par exemple pour les formes des organes ou des structures osseuses. Dans d'autres problèmes où les formes des structures sont variables ou n'ont pas de forme cohérente, il est nécessaire de mettre en œuvre des formules de contraintes plus génériques.

En fait, le contour actif basé sur la formulation classique [66] n'arrive pas à segmenter un objet caractérisé par des concavités en forme de U (voir FIGURE 2.3a). Cette formulation traditionnelle du contour actif [66] présente également l'inconvénient de limiter le rayon d'action de la force externe, directement lié au paramètre σ du filtre gaussien (2.6a). Une forte valeur de σ peut augmenter le rayon d'action de la force externe, mais au détriment d'un lissage plus marqué des bords et par conséquent une moins bonne localisation de ceux-ci.

Xu *et. al* [123] proposent une nouvelle formulation de la force externe afin d'améliorer le contour actif classique. À partir du modèle formulé en (2.10), la force externe est remplacée par le flux du vecteur gradient (*Gradient Vector Flows*). Celle-ci est obtenue par minimisation de l'énergie fonctionnelle :

$$\mathcal{E}_{GVF} = \iint_{\mathbb{R}^2} \mu(v_x^2 + v_y^2 + w_x^2 + w_y^2) + |\nabla f|^2 |\vec{u} - \nabla f|^2 \, dx dy \tag{2.15}$$

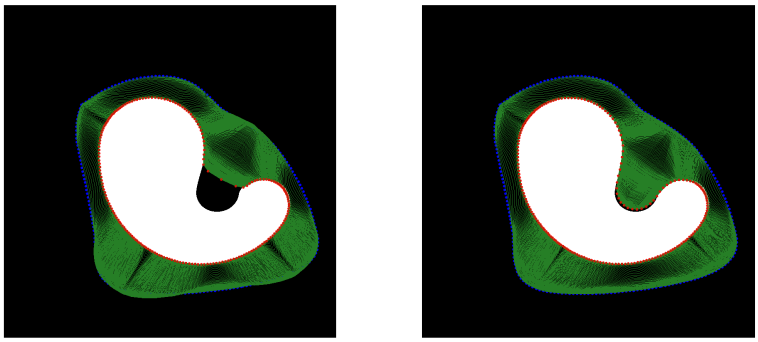
où $\vec{u}$ est le champ vectoriel défini comme $\vec{u} = (v(x,y), w(x,y))$. Avec v et w les vecteurs de force de l'image $f(x,y)$. Celle-ci représente la cartographie des caractéristiques de l'image $f(x,y)$ et elle est définie par

$$f(x,y) = -E_{ext}(x,y) \tag{2.16}$$

où $E_{ext}(x,y)$ est définie en (2.4).

La formulation variationnelle permet d'obtenir un résultat qui varie très doucement lorsque le gradient est faible. En particulier, lorsque $|\nabla f|$ est petit, l'énergie est dominée par la somme des carrés des dérivées partielles du champ vectoriel. Cette condition donne un champ qui varie très lentement dans les régions homogènes. Lorsque $|\nabla f|$ est grand, le second terme domine l'intégrale, ce qui maintient le champ vectoriel proche du gradient de cartographie des caractéristiques de l'image $f(x,y)$. μ est un paramètre de régularisation qui régit le compromis entre le premier terme et le deuxième terme de l'intégrale. Ce paramètre doit être réglé en fonction de la quantité de bruit présent dans l'image [124].

La FIGURE 2.3 montre l'évolution du contour initial (ligne bleu) vers le contour final (ligne rouge). Les lignes vertes représentent les états intermédiaires de la courbe au cours des itérations. À la différence de la formulation classique du *snake*, le contour actif avec le flux du gradient vecteur arrive à segmenter les objets caractérisés par des régions concaves.



(a) (b)

FIGURE 2.3 – Comportement du contour actif classique (a) et du contour actif basé sur le flux du vecteur gradient (b) dans le cas d'un objet présentant une concavité en forme de U.

Les figures (voir FIGURE 2.3) ont été obtenues par l'algorithme de *snake*, qui a été mis en œuvre par Kroon [72]. Cet algorithme utilise la formulation de *snake* de Kass [66], et permet en outre de pendre en compte le flux du vecteur gradient introduit par Xu [123] (SECTION 2.2.2). La mise en œuvre de cet algorithme est illustrée dans l' ALGORITHME 1, où la signification des paramètres utilisés est la suivante :
pour le contour C^0 :

- M : nombre de points composant le contour
- α : l'élasticité du contour
- β : la rigidité du contour

pour l'expression des énergies :

- w_{image} : poids pour la détection des arêtes
- w_{term} : poids pour la détection des terminaisons aigües
- σ_1 : calcul du gradient de l'image
- σ_2 : calcul du gradient de l'énergie externe
- σ_3 : calcul du laplacien de l'image prenant en compte le flux du vecteur gradient
- μ : paramètre de régularisation
- κ : poids pour le flux du vecteur gradient

Bien que le contour actif paramétrique ou *snake* soit une approche de segmentation très utilisée en traitement d'image, il présente néanmoins des inconvénients qui peuvent limiter l'application dans certains cas. Parmi les facteurs limitants : la représentation explicite nécessite une étape de re-paramétrisation de la courbe afin d'éviter l'intersection de la courbe avec elle même. De plus, la représentation explicite ne permet pas au contour une évolution qui prend en compte les changements topologiques tels que la segmentation de plusieurs objets. Enfin, en raison de l'approche de minimisation locale, le résultat de la segmentation dépend fortement de la position initiale de la courbe.

Algorithme 1 *Snake.*

1 : Initialisation : $\alpha, \beta, \mu, \kappa, M$
2 : Initialisation : $\sigma_1, \sigma_2, \sigma_3$
3 : Initialisation : w_{image}, w_{term}
4 : Initialisation : C^0
5 : $E_{ext} \leftarrow (I, w_{image}), w_{term}, \sigma_1)$ // Calcul de l'énergie externe
6 : $F_{ext} \leftarrow (E_{ext}, \sigma_2)$ // Calcul de la force externe
7 : $GVF \leftarrow (F_{ext}, \sigma_3, \mu)$ // Calcul du *Gradient Vector Flow*
8 : $F_{int} \leftarrow (C^0, \alpha, \beta)$ // Calcul de la force interne
9 : Initialisation : $n = 0$
10 : **while** $(n \leqslant N)$ **do**
11 : $C^{n+1} \leftarrow (C^n, GVF, F_{int}, \kappa)$ // Déplacement de la courbe
12 : $n \leftarrow n + 1$
13 : **end while**

2.2.3 Contour actif par modèle déformable géométrique ou contour géodésique

Dans la section précédente nous avons vu comment l'introduction d'une nouvelle force externe (flux du gradient vecteur) permet d'améliorer l'efficacité du *snake* classique. Cependant, des propriétés indésirables caractérisent ce modèle. Premièrement, le contour modélisé par une courbe paramétrique spline est facile à initialiser lorsqu'on dispose d'un modèle statistique de l'objet à segmenter, et souvent on obtient une segmentation correcte de l'objet. Mais, lorsqu'on ne dispose pas d'un *a priori* sur la forme de l'objet à segmenter, le *snake* n'arrive pas à segmenter correctement l'objet. Deuxièmement, lorsqu'on considère plusieurs objets dans une image et que le contour initial les entoure, il n'est pas possible de détecter tous les objets.

Caselles *et al.* [19] et Kichenassamy *et al.* [68] ont proposé une nouvelle énergie, basée sur le modèle de [66], qui est invariant par rapport à la paramétrisation de la courbe. La nouvelle énergie fonctionnelle est definie par :

$$\mathcal{E} = \int_0^{L(C)} g(|\nabla I(C(s)|)ds \tag{2.17}$$

Avec ds la longueur euclidienne de l'élément infinitésimal, $L(C)$ la longueur euclidienne de la courbe C définie par

$$\begin{aligned} L(C) &= \int_0^1 |C'(q)|dq \\ &= \int_0^{L(C)} ds \end{aligned} \tag{2.18}$$

Ainsi, la fonctionnelle (2.17) est une nouvelle longueur obtenue en pondérant la longueur de l'élément euclidien ds par la fonction g qui contient des informations concernant les frontières des objets. $g(|\nabla I(C(q))|)$ est la fonction de détection de contour qui a la

caractéristique d'une fonction monotone décroissante, et qui est définie telle que :

$$\begin{aligned} \lim_{x \to 0} g(x) &= 1 \\ \lim_{x \to +\infty} g(x) &= 0 \end{aligned} \tag{2.19}$$

Des exemples classiques de fonctions monotones décroissantes sont données par [18, 78, 79].

Caselles *et al.* [19] ont montré également que la minimisation de l'énergie (2.17) représente en fait une courbe géodésique dans un espace de Riemann. Cette courbe géodésique est calculée par le calcul des variations. Si on ajoute une variable temporelle t artificielle, alors on peut considérer une famille de courbes $C(t)$ telle que :

$$\mathcal{E}(t) = \int_0^1 g(|\nabla I(C(q,t)|)|C'(q,t)|dq \tag{2.20}$$

L'évolution de la courbe peut être décrite par l'équation d'Euler-Lagrange qui permet de minimiser la fonctionnelle (2.20) :

$$\frac{\partial C(t)}{\partial t} = g\,\kappa\,\mathcal{N} - \langle \nabla g, \mathcal{N} \rangle \mathcal{N} \tag{2.21}$$

où $\mathcal{N}$ est la normale unitaire à la courbe C et κ sa courbure. Le premier terme de (2.21) représente la courbure moyenne du mouvement, pondérée par la fonction de détection de bord g. Celle-ci lisse la forme de la courbe et en diminue la longueur. Le deuxième terme de (2.21) attire la courbe vers les limites des objets en créant une attraction centrée sur les bords.

Dans [18] une approche par la méthode de *level set* [93, 106] est utilisée pour résoudre l'équation de mouvement de la courbe (2.21). En fait la modélisation par la méthode de *level set* permet de décrire la courbe C en utilisant une représentation implicite.

Dans la section suivante, nous présentons la méthode de *level set* et sa mise en œuvre pour la détection de contours.

2.3 Méthode de *Level Set*

La méthode de *level set* a été introduite par Osher et Sethian [93]. Initialement, la méthode a été appliquée pour résoudre les problèmes de propagation de flammes, où la vitesse de la flamme est donnée en fonction de la courbure moyenne locale du front de flamme. Cette formulation est à la base de la méthode de *level set*, aujourd'hui utilisée dans de nombreuses applications de physiques, dynamique des fluides, graphiques, visualisation, traitement d'images, vision par ordinateur, contrôle, etc. [92, 107].

2.3.1 Généralités

À la différence de la représentation paramétrique (explicite) où le contour est représenté par l'ensemble des points qui l'établissent, la représentation implicite décrit le contour par

une fonction implicite. Celle-ci conduit à une représentation qui est indépendante de la paramétrisation du contour.

Par exemple, supposons que l'axe réel $\mathbb{R}$ soit divisé en trois parties par les points $-x_0$ et x_0 (voir FIGURE 2.4). On peut définir chaque partie par deux sous-domaines : $\Omega^- =]-x_0, x_0[$ le domaine intérieur, et $\Omega^+ =]-\infty, -x_0[\bigcup]x_0, \infty[$ le domaine extérieur. Le bord entre les deux régions est représenté par l'ensemble des points $\partial\Omega = \{-x_0, x_0\}$. Celui-ci représente l'interface Γ entre les deux régions décrites par une représentation explicite. Dans une représentation implicite, l'interface est définie par la ligne de niveau zéro d'une fonction $\phi(x)$.

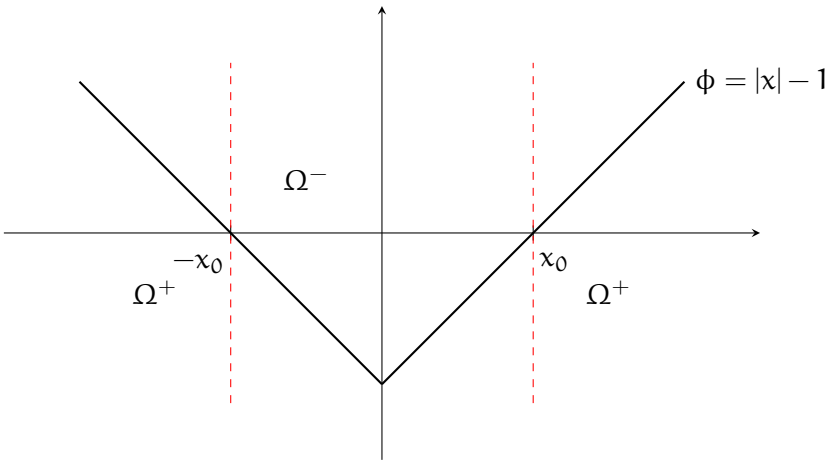


FIGURE 2.4 – Exemple d'une fonction implicite.

La fonction $\phi(x)$, appellée fonction *level set*, est définie par une fonction de distance signée

$$|\phi(x)| = d_r(x) \tag{2.22}$$

La fonction de distance signée représente la distance entre le point x et l'interface Γ, et le signe est déterminé afin qu'il soit négatif à l'intérieur et positif l'extérieur. La fonction $d_r(x)$, $x \in \mathbb{R}^n$ a les propriétés suivantes :

$$d_r(x) > 0 \; \forall \; x \; \in \; \Omega^+ \tag{2.23a}$$

$$d_r(x) < 0 \; \forall \; x \; \in \; \Omega^- \tag{2.23b}$$

$$d_r(x) = 0 \; \forall \; x \; \in \; \partial\Omega \tag{2.23c}$$

$$|\nabla d_r(x)| = 1 \tag{2.23d}$$

Alors, on peut définir la méthode de *level set* [93], telle qu'une fonction implicite $\phi(x, t)$ dynamique, qui se déplace dans l'espace $\mathbb{R}^n$ en fonction du temps t. Ainsi, l'interface peut être récupérée à n'importe quel instant de temps t par le positionnement au niveau zéro de la fonction implicite :

$$\Gamma(t) = \{x \mid \phi(x, t) = 0\} \tag{2.24}$$

Le mouvement de la fonction implicite $\phi(x,t)$ vers l'interface Γ peut être décrit par l'équation de mouvement d'Euler-Lagrange :

$$\frac{\partial \phi}{\partial t} + \nabla \phi(x,t) \cdot \vec{V}(x) = 0 \tag{2.25}$$

L'équation (2.25) est connue aussi sous le terme d'équation *level set* ou *level set equation* [92]. Le terme $\vec{V}(x) = \frac{\partial x}{\partial t}$ représente le champ de vitesse qui permet l'évolution de l'interface $\phi(x,t) = 0$ dans l'espace (voir FIGURE 2.5).

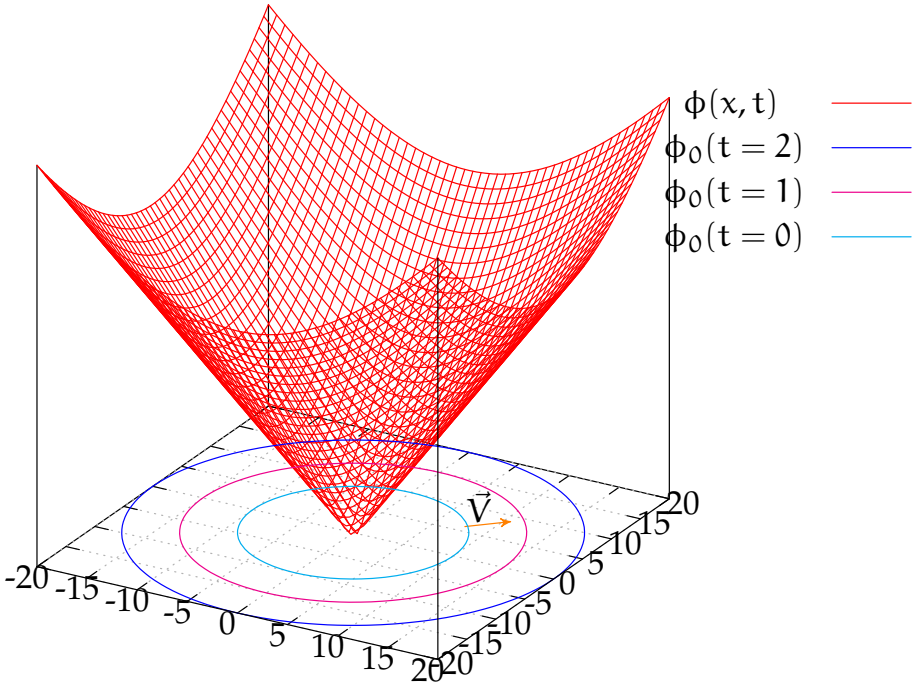


FIGURE 2.5 – Fonction *level set* représentée dans l'espace 3D.

Normalement, le champ de vitesse est constitué d'une composante tangentielle et d'une composante normale

$$\vec{V}(x,t) = (V_t \mathcal{T} + V_n \mathcal{N}) \tag{2.26}$$

mais seule la composante normale est prise en compte. En effet, la composante normale est dirigée dans la même direction que le gradient de la fonction implicite $\nabla\phi(x,t)$, et la contribution de la composante tangentielle est nulle [40] ($V_t \mathcal{T} \cdot \nabla\phi(x,t) = 0$). Ainsi l'équation de *level set* s'ecrit :

$$\frac{\partial \phi}{\partial t} + V_n \mathcal{N} \cdot \nabla \phi(x,t) = 0 \tag{2.27}$$

avec

$$\mathcal{N} \cdot \nabla\phi(x,t) = \frac{\nabla\phi(x,t)}{|\nabla\phi(x,t)|} \cdot \nabla\phi(x,t) \tag{2.28}$$

$$= |\nabla\phi(x,t)|$$

L'équation (2.25) devient :

$$\frac{\partial\phi}{\partial t} + V_n \cdot |\nabla\phi(x,t)| = 0 \tag{2.29}$$

où V_n est la composante normale. Celle-ci est caractérisée par la courbure moyenne κ, telle que $V_n = -\kappa$ [92]. Ainsi, le mouvement d'une courbe, définie dans le cadre de la formulation *level set*, et basée sur la courbure moyenne s'écrit :

$$\frac{\partial\phi}{\partial t} = \kappa\,|\nabla\phi(x,t)| \tag{2.30}$$

2.3.2 Modèle géométrique par *level set*

À partir de l'équation (2.30) Caselles *et al.* [18] ont développé un modèle géométrique de contour actif basé sur la méthode de *level set*. L'évolution de la courbe est décrite par la fonction de *level set* ϕ, $\{x \in \mathbb{R}^2 : \phi(x,t) = k\}$, $k \in \mathbb{R}$, qui varie suivant la direction normale et avec une vitesse dépendante de la courbure moyenne. En décrivant la courbure moyenne κ dans l'équation (2.7), en fonction de ϕ, l'équation de *level set* s'écrit :

$$\frac{\partial\phi}{\partial t} = |\nabla\phi|\,\nabla\cdot\left(\frac{\nabla\phi}{|\nabla\phi|}\right) \tag{2.31}$$

En outre dans [18] un terme de force constante υ est introduit dans la direction normale au *level set*, ainsi qu'un terme multiplicatif $g(x)$ afin d'arrêter le *level set* sur le contour désiré. Alors l'équation (2.31) se réécrit

$$\frac{\partial\phi}{\partial t} = g(x)\,|\nabla\phi|\,\underbrace{\left(\nabla\cdot\frac{\nabla\phi}{|\nabla\phi|} + \upsilon\right)}_{\beta} \tag{2.32}$$

où, $\upsilon \in \mathbb{R}^+$, est une valeur constante.

Le terme β fait en sorte que le déplacement du *level set* ϕ soit proportionnel à la courbure moyenne. Le terme υ, appelé force de ballon [27], permet d'augmenter significativement la vitesse de convergence vers la solution en régime permanent, et la détection des objets non convexes.

La fonction $g(x)$, qui satisfait les propriétés décrites en (2.19), est définie par :

$$g(x) = \frac{1}{1 + |\nabla G_\sigma * I|^2} \tag{2.33}$$

où $G_\sigma * I$ est la convolution de l'image avec un noyau gaussien, afin de réduire les effets de bruit dans l'image.

Le terme $g(x)$ permet de contrôler la vitesse de mouvement de ϕ. Lorsque, ϕ se rapproche des bords de l'objet, la valeur de $|\nabla G_\sigma * I|^2$ est très grande et donc $g(x)$ tend vers zéro. Ceci représente une condition stationnaire, $\frac{\partial \phi}{\partial t} = 0$, pour laquelle le *level set* s'arrête.

Cependant, la condition $g(x) = 0$ ne se vérifie que dans le cas d'une détection idéale des bords. Ainsi, l'équation (2.32) n'est pas appropriée à la détection de bords caractérisés par une haute variation du gradient. Dans le travail de Caselles *et al.* [19], un nouveau terme du gradient est pris en compte de telle sorte que l'équation (2.32) devient :

$$\frac{\partial \phi}{\partial t} = |\nabla \phi| \nabla \cdot \left(g(x) \frac{\nabla \phi}{|\nabla \phi|} \right) + \upsilon\, g(x) |\nabla \phi| \tag{2.34}$$

$$= g(x) |\nabla \phi| \underbrace{(\kappa + \upsilon)}_{\beta} + \underbrace{\nabla \phi \cdot \nabla g(x)}_{nouveau\,terme} \tag{2.35}$$

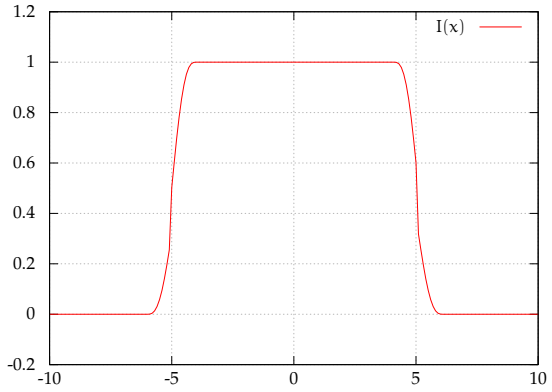
La figure (FIGURE 2.6) montre le rôle que joue le terme $\nabla g(x)$ dans le cas 1D. Comme on peut observer, les directions des gradients pointent vers l'intérieur des vallées de la fonction $g(x)$. Tout d'abord, cette force permet de propager la courbe vers le bord, mais elle maintient également la courbe en position stable.

Comme il est montré dans [19], l'équation (2.34) représente la solution au problème (2.21) dans le cadre d'une formulation par *level set*. Le contour est obtenu par le niveau zéro de la fonction *level set*, $\phi(t = 0)$. Bien que le modèle obtenu en (2.34) ne dépende pas de la paramétrisation du contour, une question importante reste encore : le choix de la fonction d'arrêt $g(x)$. Celle-ci étant basée sur le gradient, elle souffre des mêmes problèmes que les opérateurs de détection de bords. Pour améliorer la détection des bords, différentes solutions ont été proposées : par exemple les travaux de [96, 97, 45] proposent des opérateurs de détection plus performants qui utilisent des filtres anisotropes.

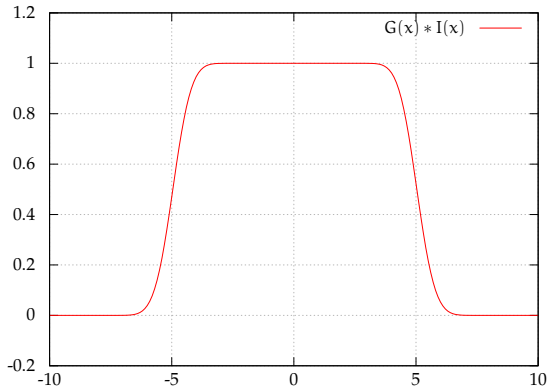
La méthode de segmentation présentée ci-dessus est basée sur l'intensité du gradient de l'image, et est donc sujette aux problèmes des régions à faible contraste. Cependant, des approches différentes ont été utilisées dans la segmentation d'image comme par exemple les méthodes basées sur les régions qui s'appuient sur la fonctionnelle de Mumford-Shah [86, 22]. Nous abordons cette méthode dans la section suivante.

2.4 Modèle de segmentation de *Mumford-Shah*

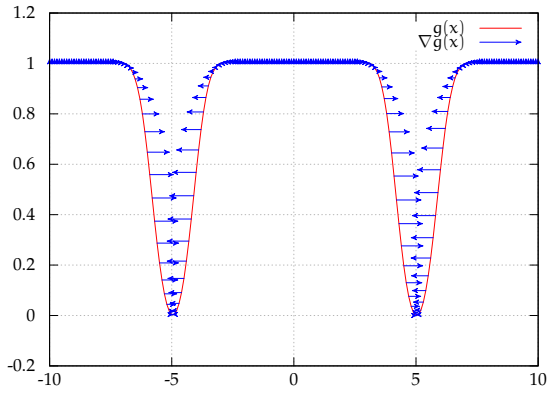
Le modèle de contour actif formulé sur le modèle *snake* ou géométrique dépend d'une fonction d'arrêt g, souvent liée au gradient de l'image I que l'on cherche à segmenter. Dans le cas d'une image très bruitée, le gradient peut avoir de fortes oscillations sur les frontières, ce qui peut amener à une fonction d'arrêt instable. Par ailleurs, l'utilisation de filtres à noyaux larges (ou très larges) peut adoucir ou faire disparaître les discontinuités qui caractérisent les bords. Chan et Vese [22] proposent un modèle de segmentation qui



(a) Signl 1D



(b) Convolution de $I * G$



(c) Fonction d'arrêt g en rouge et gradient de la fonction d'arrêt en bleu

FIGURE 2.6 – Principe géométrique de la force d'attraction en 1D. Comme illustré en (c) le gradient pointe vers les vallées de bord.

n'utilise pas de fonction d'arrêt basée sur le gradient mais sur l'approche de segmentation de Mumford-Shah [86].

Dans le modèle de Mumford-Shah [86], le problème de segmentation consiste à calculer

la décomposition du domaine R d'une image f telle que :

$$R = R_1 \bigcup \cdots \bigcup R_n \tag{2.36}$$

de façon à ce que :

- l'image f varie doucement à l'intérieur de chaque région R_i
- l'image f varie très rapidement sur les frontières entre les différentes régions R_i

Ainsi, le problème de segmentation d'une image f définie par les propriétés ci-dessus, consiste à trouver une fonction g qu'approxime f. On peut définir le problème de minimisation de la fonctionnelle ainsi :

$$\mathcal{E}(g, \Gamma) = \mu^2 \iint_R (f - g)^2 \, dx \, dy + \iint_{R-\Gamma} \|\nabla g\|^2 \, dx \, dy + \nu \int_\Gamma ds \tag{2.37}$$

où $\int_\Gamma ds$ est la longueur totale des contours.

La fonctionnelle est composée de trois termes :

- le premier terme impose que g se rapproche de f au sens des moindres carrés,
- le deuxième terme impose que g varie faiblement à l'intérieur de chaque région R_i,
- le dernier terme assure que la longueur des frontières soit la plus courte possible.

En vision par ordinateur (g, Γ) est interprété comme une *cartoon* de l'image f : g est une image où les frontières sont dessinées de façon précise et nette.

La technique de segmentation introduite par Chan et Vese [22], appelé ACWE (*Active Contour Without Edges*) est une méthode issue du modèle de Mumford-Shah dans le cas d'une approximation par une fonction constante par morceaux. À la différence du modèle Mumford-Shah où chaque région est approximée par une fonction lisse, dans le modèle de Chan et Vese les régions sont approximées par des fonctions constantes par morceaux. Par exemple, supposons qu'une image I_0 soit constituée par deux régions : une région intérieure avec une intensité $I_0^{\Omega^+}$ et une région extérieure avec une intensité $I_0^{\Omega^-}$, tel que illustré en FIGURE 2.7. C_0 représente la frontière entre les deux régions, $I_0 \approx I_0^{\Omega^+}$ à l'intérieur de C_0 et $I_0 \approx I_0^{\Omega^-}$ à l'extérieur de C_0.

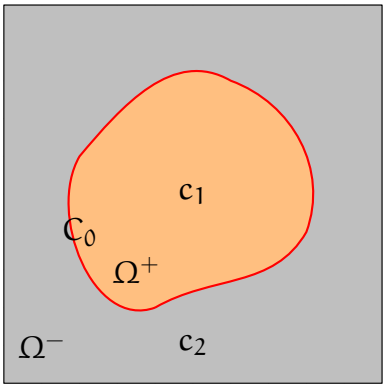


FIGURE 2.7 – Représentation du modèle de segmentation d'une image de Chan et Vese.

À partir de cette considération, on peut définir un terme d'approche :

$$F_1(C) + F_2(C) = \int_{\Omega^+} (I_0 - c_1)^2 \, dx + \int_{\Omega^-} (I_0 - c_2)^2 \, dx \tag{2.38}$$

avec Ω^+ et Ω^- les régions interne et externe respectivement, C une variable qui définit l'évolution de la courbe initiale, et c_1, c_2 deux constantes qui dépendent de la courbe C et qui représentent les valeurs moyennes de I_0 à l'intérieur et à l'extérieur de la courbe C. C_0, le vrai contour de l'objet minimise le critère (2.38) :

$$\min_C\{F_1 + F_2\} \approx 0 \approx F_1(C_0) + F_2(C_0) \tag{2.39}$$

En outre, un terme de régularisation comme la longueur du contour a été indroduit dans le critère (2.39) afin d'imposer une contrainte de douceur sur la géométrie du contour qui sépare les régions constantes. Ainsi, on definit :

$$\mathcal{E}_{CV}(c_1, c_2, C) = \underbrace{\nu \int_C ds}_{\text{longueur de } C} + \mu \left(\int_{\Omega^+} (I_0(x) - c_1)^2 \, dx + \int_{\Omega^-} (I_0(x) - c_2)^2 \, dx \right) \tag{2.40}$$

Chan et Vese [22] résolvent le problème par une formulation basée sur la méthode de *level set* (2.41). La variable C est remplacée par la fonction de *level set* $\phi(x)$. Ainsi, les régions Ω^+ et Ω^- sont définies par la fonction d'Heaviside $H(x)$, qui modélise la fonction caractéristique $\phi(x)$:

$$\begin{aligned} \mathcal{E}_{CV}(c_1, c_2, \phi) &= \nu \int_\Omega |\nabla H(\phi(x))| \, dx \\ &\quad + \mu \int_\Omega H(\phi(x))(f(x) - c_1)^2 + (1 - H(\phi(x)))(f(x) - c_2)^2 \, dx \end{aligned} \tag{2.41}$$

Ainsi, avec $\phi(x)$ fixée, on peut écrire la minimisation de l'énergie (2.41) par rapport aux constantes c_1 et c_2 comme :

$$c_1(\phi(x)) = \frac{\int_\Omega f(x) H(\phi(x)) \, dx}{\int_\Omega H(\phi(x)) \, dx} \tag{2.42a}$$

$$c_2(\phi(x)) = \frac{\int_\Omega f(x)(1 - H(\phi(x))) \, dx}{\int_\Omega (1 - H(\phi(x))) \, dx} \tag{2.42b}$$

Autrement dit, les constantes c_1 et c_2 représentent la valeur moyenne de $f(x)$ pour $\phi \geqslant 0$ et $\phi < 0$, respectivement.

Pour c_1 et c_2 fixés, la courbe optimale peut être calculée en utilisant le flux du gradient :

$$\frac{\partial \phi}{\partial t} = \delta(\phi) \left(\nabla \cdot \frac{\nabla \phi}{|\nabla \phi|} - \mu((c_1 - f(x))^2 - (c_2 - f(x))^2) \right) \tag{2.43}$$

L'équation (2.43) décrit l'équation d'Euler-Lagrange associée à la fonction ϕ, qui évolue selon la direction normale à la courbe (FIGURE 2.8).

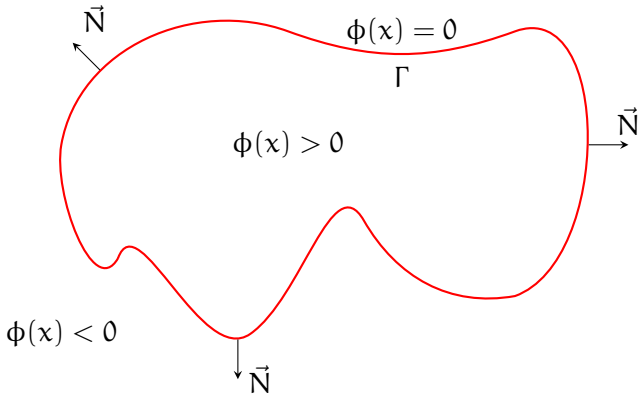


FIGURE 2.8 – Évolution de la fonction *level set* en direction de la normale à la courbe, le contour est défini par $\Gamma = \{x : \phi(x) = 0\}\, x \in \mathbb{R}^2$.

Finalement, la méthode de Chan et Vese peut être résumée par l' ALGORITHME 2 :

Algorithme 2 *Algorithme de la méthode de Chan et Vese.*

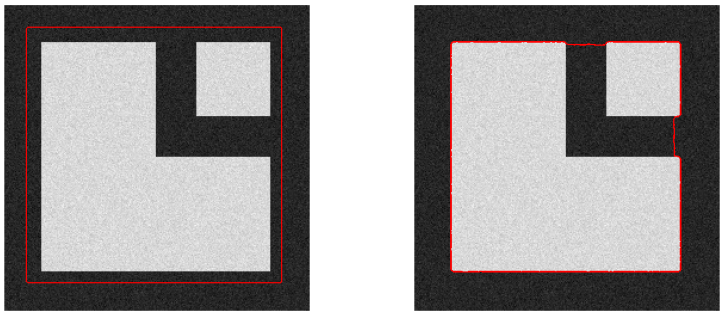
$n = 0$
Initialisation $\phi^0 = \phi_0$
while $\frac{\partial \phi}{\partial t} = 0$ **do**
  Calculer $c_1(\phi^n)$ et $c_2(\phi^n)$ par (2.42a) et (2.42b)
  Résoudre (2.43) // déplacement de la courbe
  Réinitialiser ϕ^n tel que $|\phi^n| = 1$ // fonction de distance signée
end while

Le modèle de Chan et Vese est aussi appelé *two-phase segmentation* car la minimisation est faite sur deux valeurs c_1 et c_2. La solution de l'équation aux dérivées partielles est résolue numériquement par la méthode des différences finies.

2.5 Méthode de segmentation convexe

Le modèle de Chan et Vese effectue une minimisation des fonctions qui admettent deux valeurs ; celles-ci ne forment pas un ensemble convexe. Ainsi comme le modèle de *snake*, ce modèle dépend d'une manière importante de l'initialisation du contour initial. Les minima locaux génèrent de fausses conditions stationnaires qui ne permettent pas à la courbe de se coller au contour de l'objet (voir FIGURE 2.9 [75]).

Plusieurs méthodes ont été proposées pour rendre le modèle de segmentation convexe comme par exemple l'approche utilisée par Chan *et. al* [21]. Les auteurs proposent un modèle de segmentation hybride qui combine le modele de Mumford Shah dans le cas d'une fonction constante par morceaux et le modèle de débruitage ROF (Rudin Osher Fatemi) [100]. Un autre modèle est un modèle hybride entre le modèle de contour actif géodésique et le modèle ROF. Chan *et. al* [21] mettent en place un modèle appelé GCS



(a) Image avec le contour intial (rouge)(b) Image avec le contour final (rouge)

FIGURE 2.9 – Exemple d'une segmentation en appliquant le modèle Chan and Vese.

(pour *Globally Convex Segmentation*) [21] qui supprime le problème lié au modèle non convexe ; il est plus fiable et stable numériquement. La formulation de GCS s'appuie sur le fait que la solution stationnaire du flux du gradient (2.43) correspond à la solution stationnaire de :

$$\frac{\partial \phi}{\partial t} = \left(\nabla \cdot \frac{\nabla \phi}{|\nabla \phi|} - \mu((c_1 - f(x))^2 - (c_2 - f(x))^2) \right) \tag{2.44}$$

Ainsi l'équation (2.44) représente la descente du gradient de l'énergie fonctionnelle

$$\int_\Omega |\nabla \phi| + \mu \int_\Omega \phi r(x)\, dx \tag{2.45}$$

où $r = ((c_1 - f(x))^2 - (c_2 - f(x))^2)$.

$$\mathcal{E} = |\nabla \phi|_1 + \mu \langle \phi, r \rangle \tag{2.46}$$

L'énergie (2.46) n'est pas convexe, et il n'existe pas de minimum global à moins de borner la solution dans l'intervalle $[0, 1]$. Ainsi le problème de minimisation s'écrit :

$$\min_{0 \leqslant \phi \leqslant 1} |\nabla \phi|_1 + \mu \langle \phi, r \rangle \tag{2.47}$$

À partir de la solution obtenue par minimisation de (2.50), le contour de la région segmentée est trouvé par seuillage de la fonction de *level set* :

$$\Gamma = \{x : \phi(x) > \alpha\},\ x \in \mathbb{R}^2,\ \alpha \in]0, 1[\tag{2.48}$$

Finalement, la solution (2.50) peut être vue comme une approche convexe de la méthode proposée par Chan *et al.* [22].

Cependant, Bresson *et al.* [14] ont modifié l'énergie (2.50) pour prendre en compte des informations issues d'une fonction de détection de contours (2.33). Ainsi, ils obtiennent

un modèle qui se rapproche de celui du contour actif *snake* ou géodésique. Ils remplacent la norme TV (pour *Total Variation)* dans l'énergie (2.45) par la norme TV pondérée :

$$\begin{aligned} TV_g &= \int_\Omega g|\nabla\phi| \\ &= |\nabla\phi|_{TV_g} \end{aligned} \tag{2.49}$$

En remplaçant la norme TV standard avec cette version pondérée, le modèle favorise la segmentation de régions où la fonction de détection des bords assume de valeur faible. En ce sens, le modèle GCS est un modèle hybride entre les modèles GAC et le *snake* ACWE.

$$\min_{0\leqslant\phi\leqslant 1} |\nabla\phi|_{TV_g} + \mu\langle\phi, r\rangle \tag{2.50}$$

Un algorithme optimal qui minimise le critère (2.50) a été mis en œuvre par Bresson *et al.* [13, 51],appelé *Globally Convex Segmentation*. Cet algorithme est détaillé dans l'ALGORITHME 3.

Algorithme 3 GCS (*two phase level set*).

Initialisation : $n = 0$
while $\|\phi^{n+1} - \phi^n\| \geqslant \epsilon$ **do**
  Calcul de $r^n = \left(c_1^n - I\right)^2 + \left(c_2^n - I\right)^2$ // réinitialisation du *level set*
  $\phi^{n+1} \leftarrow$ résoudre par Gauss-Seidel (r^n, d^n, b^n)
  $d^{n+1} \leftarrow shrink_g\left(\nabla\phi^{n+1} + b^n, \lambda\right)$
  $b^{n+1} \leftarrow b^n\nabla\phi^{n+1} + d^{n+1}$
  Mise à jour de c_1^{n+1} et c_2^{n+1}
end while

La réinitilisation est nécessaire pour garantir que la fonction de *level set* ϕ ait la propriété d'une fonction de distance signée (2.23a). Les contours sont obtenus par la fonction de *level set* ϕ telle que :

$$\Gamma = \{x \mid \phi(x) \geqslant \alpha\}\ x \in \mathbb{R}^2,\ \alpha \in]0, 1[\tag{2.51}$$

L'ALGORITHME 3 contient des paramètres de réglage. Le premier est le paramètre μ qui contrôle la régularisation et par conséquent le niveau de détail qui peut être segmenté ; sa valeur est choisie telle que $\mu \in [10^2, 10^4]$, $\mu \in \mathbb{R}$. Le deuxième est le paramètre λ qui pondère le terme de la fonction de pénalité quadratique, et influence la vitesse de convergence ; sa valeur est telle que $\lambda \in [0.1, 10]$, $\lambda \in \mathbb{R}$.

2.6 Conclusions

Dans ce chapitre nous avons abordé le thème de la segmentation d'image montrer qu'il existe une grande quantité de méthodes allant des plus simple aux plus sophistiquées. Nous avons décrit dans les détails les méthodes de segmentation basées sur contour actif : *snake* et géodésique. Dans le modèle du *snake*, le contour est défini par une courbe

paramétrique. Cette courbe se déplace vers les bords d'un objet sous l'action d'une force interne et une force externe. Cependant, les résultats de la segmentation dépendent de la paramétrisation de la courbe et de sa position initiale.

C'est pourquoi, les modèles de segmentation basés sur une modélisation du contour par une fonction implicite, donnent de meilleurs résultat pour la segmentation de formes complexes. Dans la suite du travail, ces méthodes sont mises en œuvre dans un processus global de reconstruction - segmentation d'une image en tomographie par rayons X. Cependant, avant d'aborder la présentation et la mise en œuvre de la méthode dans son ensemble (méthode de reconstruction itérative et méthode de segmentation), nous proposons d'aborder des notions d'architectures parallèles pour l'accélération des calculs. Cet aspect est considéré notamment pour l'optimisation du temps de calcul de l'algorithme de reconstruction itératif adapté à une grille irrégulière composée de triangles (resp. tétraèdres).

Chapitre 3

Problématique de la parallélisation

Le principe de la parallélisation est d'accélérer l'exécution d'un processus. Du point de vue conceptuel, on divise le volume du problème en sous volumes, qui peuvent être exécutés dans le même temps. Par exemple l'algorithme de réduction/sommation peut être parallélisé en découpant le vecteur en sous vecteurs, chacun avec une taille de N/M, M étant le nombre de processus utilisés. Les approches séquentielle et parallèle sont illustrée respectivement sur la FIGURE 3.1 et la FIGURE 3.2.

On classifie généralement l'accélération selon deux types *software* et *hardware*. Dans le premier cas on cherche à améliorer l'algorithme, en réduisant la complexité, les opérations arithmétiques coûteuses, les accès en mémoire, ou par exemple on utilise la programmation par *multi-threads*. Le *thread*, aussi connu comme processus léger, est l'unité de base d'exécution de un CPU (*Central Processing Unit*). Il est constitué d'un identificateur, d'un compteur de tâche, d'un ensemble de variable locales (par exemple des registres) et d'une pile d'appels [112]. Les *threads* en tant que tels sont une première approche du parallélisme qui peut aller jusqu'au vrai parallélisme sur une machine multi-cœurs. Dans le deuxième cas on utilise une architecture parallèle pour accélérer les calculs.

Il existe différentes architectures parallèles comme par exemple les processeurs multi-cœurs, les FPGA (*Field Programmable Gate Array*), le processeur *Cell*, les processeurs graphiques GPU (*Graphic Processor Unit*), et les *clusters* constitués de processeurs multi-cœurs CPU ou de processeurs multi-cœurs CPU et de GPU.

Dans la première partie de ce chapitre nous présenterons un état de l'art de solutions algorithmiques pour accélérer la reconstruction tomographique. Ensuite nous présenterons quelques classifications d'architectures parallèles. Finalement, nous présenterons différentes stratégies de mise en œuvre.

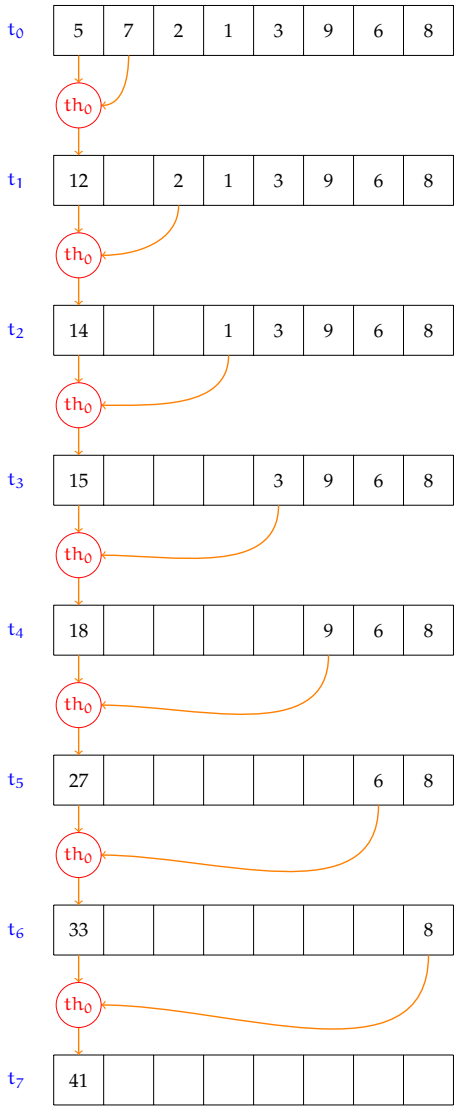


FIGURE 3.1 – Exemple d'une exécution séquentielle de l'algorithme de réduction.

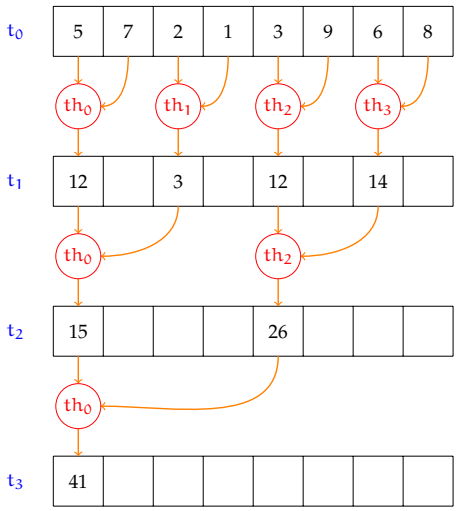


FIGURE 3.2 – Exemple d'une exécution parallèle de l'algorithme de réduction.

3.1 Accélération des algorithmes de reconstruction itératifs

Une première technique utilisée pour accélérer les algorithmes itératifs a été la mise en oeuvre de l'algorithme d'*Order Subset* (OS). Le concept de l'algorithme OS est de trier et regrouper les sinogrammes en sous-ensembles ordonnés. Afin d'améliorer la convergence, la dimension de chaque sous-ensemble ou bloc est proportionnelle à la taille d'une projection. Le facteur d'accélération introduit par l'algorithme OS est égal au nombre des blocs définis. L'algorithme OS a été utilisé pour la première fois pour l'accélération de l'algorithme EM, alors appelé OSEM [74]. Ensuite, il a été étendu à d'autres algorithmes itératifs comme SART ou SIRT [126].

Cependant, le calcul de la matrice $H = (h_{i,j})$ est l'opération la plus coûteuse en temps de calcul. La taille de la matrice H est de l'ordre de $N^2 \times M$, ($N^3 \times M$ en 3D). Où N^2 (N^3) est la taille du volume à reconstruire et M la taille des projections, qui est égale à $N_p \times D$, avec N_p le nombre d'angles de vues et D le nombre de pixels détecteur. La dimension de stockage en mémoire de la matrice H, pour diverses configurations de reconstruction en 2D, est reportée dans le tableau TABLE 3.1. On remarque aisément que le stockage de cette matrice pour des données volumineuses peut être très coûteuse.

Cependant, la matrice H a la caractéristique d'être creuse. Ainsi, une première approche est de rendre H plus compacte, par exemple en ne stockant que les entrées non nulles [56]. C'est une solution souvent adoptée en reconstruction 2D, mais qui atteint ses limites en 3D, car la taille à stocker est importante. Une solution plus efficace est de calculer les coefficients de la matrice H en ligne. Les calculs par lancer de rayons est une manière rapide d'estimer ces coefficient (*ray-casting*).

L'optimisation du calcul de la matrice H permet d'améliorer les performances de la projection et de la rétroprojection qui sont les opérateurs de base de l'algorithme de

$N \times N$	$N_p \times D$	32-bit	64-bit
256×256	256×360	22.5Go	45Go
512×512	512×360	180Go	360Go
1024×1024	1024×360	1.44To	2.44To

TABLE 3.1 – Taille de la matrice H en fonction du nombre de pixels détecteur et du nombre de projections.

reconstruction itératif. Comme introduit dans le Chapitre 1, les méthodes utilisées pour calculer la projection et la rétroprojection sont classées comme suit :

- *ray-driven* pour laquelle le rayon est centré sur le pixel du détecteur
- *pixel-driven* pour laquelle le rayon est centré sur le pixel de l'image.

Dans le cas des méthodes *ray-driven*, au cours de la projection, les *threads* traversent le volume en faisant des accès en lecture aux valeurs des pixels. Chaque *thread* écrit sa valeur de projection dans l'adresse mémoire qui lui correspond. Il n'y a donc pas de conflit d'écriture ici.

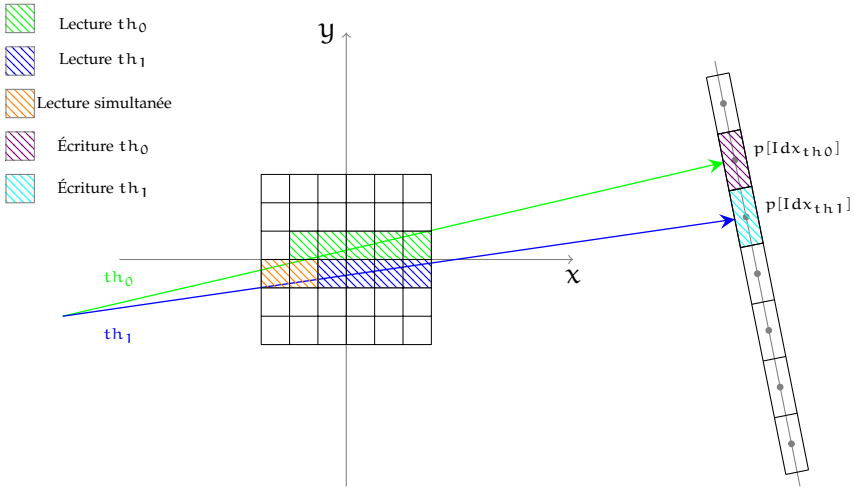
Lors de la rétroprojection, les opérations de mise à jour des valeurs des pixels deviennent des opérations d'écriture ce qui peut générer des conditions de concurrence critique lorsqu'au moins deux *threads* cherchent à modifier la même valeur d'un pixel simultanément. L'accès doit être séquentiel afin d'obtenir une valeur correcte. C'est pourquoi, la rétroprojection par *ray-driven* est moins performante que la projection (voir FIGURE 3.3).

Au contraire, dans la projection par la méthode *pixel-driven*, les *threads* écrivent leur valeur de projection en condition de concurrence critique. Et inversement dans la rétroprojection les *threads* font des lectures concurrentes mais la mise à jour de la valeur du pixel est distincte et donc sans conflit (voir FIGURE 3.4).

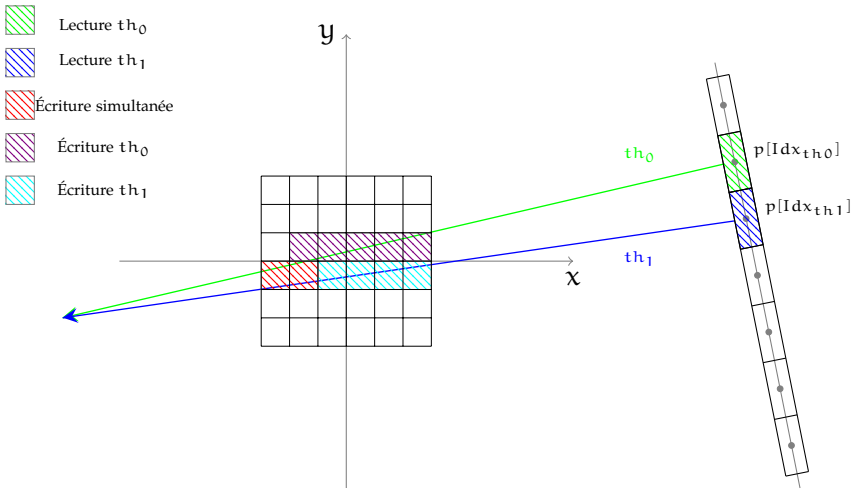
En résumé, le projecteur *ray-driven* fait des opérations *gather* : les *threads* écrivent dans leurs propres positions mémoire, par contre le rétroprojecteur fait des opérations *scatter* : les *threads* font une mise à jour des pixels communs. Dans le cas des méthodes *pixel-driven*, c'est l'inverse le projecteur fait des opérations *scatter* et le rétroprojecteur fait des opérations *gather*. Pour des raisons d'efficacité, il est préférable de toujours effectuer des opérations *gather* pour éviter les conflits en écriture. Par contre le choix de la méthode conduit à un parallélisme différent puisque dans un cas le parallélisme se fait sur les pixels détecteurs et dans l'autre sur les pixels de l'image. Le nombre réduit de détecteurs réduit le taux de parallélisme exploitable.

	ray-driven	*voxel-driven*
Projecteur	*Gather*	*Scatter*
Rétroprojecteur	*Scatter*	*Gather*

TABLE 3.2 – Opérations d'écriture en mémoire.

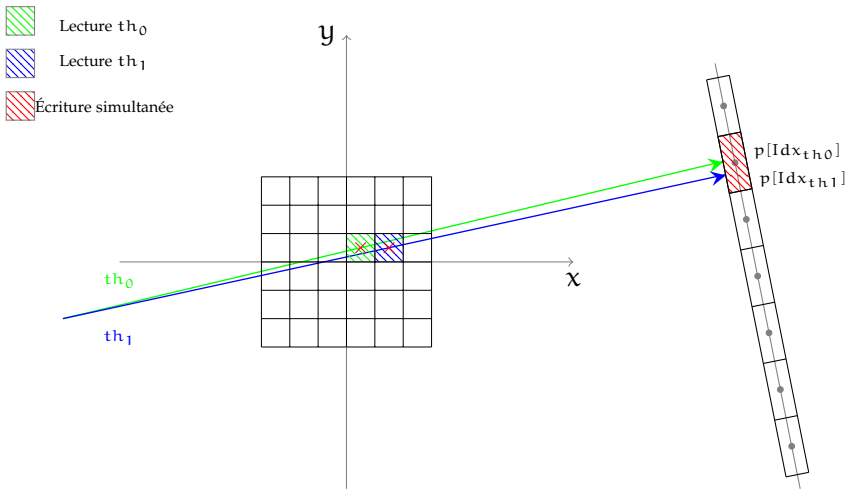


(a) Projection

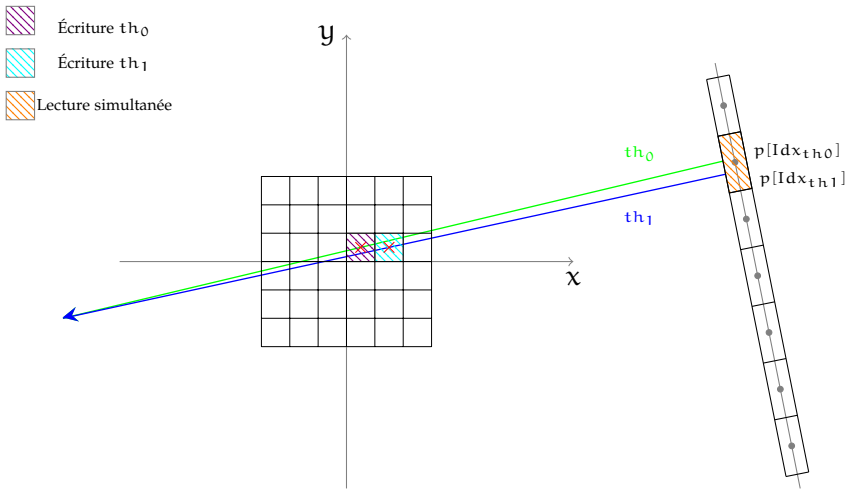


(b) Rétroprojection

FIGURE 3.3 – Accès en mémoire de la méthode *ray-driven*.



(a) projection



(b) rétroprojection

FIGURE 3.4 – Accès en mémoire de la méthode *pixel-driven*.

3.2 Architectures parallèles

Une architecture parallèle est constituée par un ensemble de processeurs qui coopèrent et communiquent. Les architectures parallèles peuvent être classifiées selon :

- le contrôle :
  - ◇ SIMD (*Single Instruction Multiple Data*), la même instruction est exécutée simultanément sur plusieurs données, on parle aussi de machine vectorielle
  - ◇ MIMD (*Multiple Instruction Multiple Data*), chaque processeur exécute ses instructions de manière indépendante l'une de l'autre
- la mémoire :
  - ◇ partagée : plusieurs processeurs partagent la mémoire du système et communiquent à travers de cette mémoire
  - ◇ distribuée : chaque processeur a sa propre mémoire

Dans une architecture à mémoire distribuée, chaque processeur élabore les données, qui sont contenues dans sa propre mémoire, et il communique avec les autres processeurs à travers un réseau. Un débit faible du réseau peut ralentir la communication entre les processeurs, qui peut avoir un impact fort sur les performances. MPI (*Message Passing Interface*) est un exemple de librairie permettant la communication entre les processeurs. C'est le cas des clusters de CPU.

3.3 Parallélisation sur architectures parallèles

Dans cette partie du chapitre nous présentons les technologies actuelles utilisées pour l'accélération des algorithmes de reconstruction : FPGA, processeur *Cell*, GPU, CPU multicoeurs et cluster de CPU. Nous donnerons une description plus détaillée de l'architecture des GPU utilisées dans ce travail de thèse.

3.3.1 FPGA

Un FPGA est un circuit logique programmable, l'unité fondamentale est la porte logique (AND, OR, XOR, etc.) et la bascule D. Il est constitué d'un tableau de portes logiques et de bascules, qui peut être configuré après la fabrication. Il a un domaine d'utilisation très large, en télécommunications, traitement d'image, automatique, etc. Il est utilisé dans les systèmes embarqués pour sa faible consommation d'énergie. En outre, il est utilisé pour la simulation des circuits intégrés avant de créer l'ASIC (*Application Specific Integrated Circuit*) correspondant. Le développement sur FPGA a un coût élevé en terme de temps de mise en œuvre des algorithmes. Malgré tout, leur emploi comme coprocesseur permet d'atteindre d'importants facteurs d'accélération.

La rétroprojection de l'algorithme FBP en *parallel-bin* a été accélérée sur FPGA [1]. Différemment à d'autres mises en œuvre où la rétroprojection est implémentée par la transformée inverse de Fourier, les auteurs ont choisi une implémentation par filtre de

type FIR *Finite Inpulse Response*. Malgré les implémentations faites sur divers FPGA, donnant des performances en cycle d'horloge et des résultats de synthèse (nombre de portes logiques utilisées), les auteurs ne fournissent pas d'informations concernant les facteurs d'accélération obtenus.

Kim *et al.* [69] prennent en compte une méthode itérative. Pour mieux bénéficier des performances du FPGA, ils utilisent une représentation des données en virgule fixe sur 16-bits. L'erreur introduite par une précision de calcul en virgule fixe est négligeable par rapport à une reconstruction en virgule flottante. L'utilisation de virgule fixe permet de réduire la complexité soit d'un point de vue de calcul soit d'un point de vue d'adressage de la mémoire. En outre, un nombre inférieur de portes logiques est nécessaires pour la synthèse de l'algorithme. De ce fait la consommation énergétique est réduite. La stratégie de parallélisation adoptée est le découpage du volume en blocs. La taille de ces blocs est optimisée pour maximiser la largeur du bus mémoire. Ils atteignent un facteur d'accélération d'environ $\times 10$ par rapport à un processeur *quad-core*.

Un travail significatif sur une utilisation combinée de FPGA et GPU a été faite par Chen *et al.* [23]. L'algorithme de reconstruction pris en compte est l'algorithme itératif EM-TV. L'algorithme EM est implémenté sur FPGA et l'algorithme TV sur GPU. La représentation en virgule fixe est adoptée pour réduire la complexité comme [69]. Les opérateurs de projection et de rétroprojection sont implémentés par la méthode *ray-driven*. Le rétroprojecteur par *ray-driven* est de type *gather* en accès de mémoire (voir TABLE 3.2). Par conséquent la rétroprojection sera moins performante par rapport à la projection. Pour réduire les accès en mémoire du rétroprojecteur les auteurs ont mis en place un masque binaire. La mise à jour du pixel est faite par rapport à la valeur du masque binaire. Le masque binaire nécessite un seul bit pour contenir l'information, donc son stockage en mémoire n'est pas coûteux. Les performances du projecteur et du rétroprojecteur ont été évaluées sur une architecture multi-FPGA et sur des cartes GPUs Tesla C1060 et Fermi GTX 480. L'architecture multi-FPGA est plus rapide d'un facteur $\times 2$ par rapport à la carte Tesla, mais elle a le même temps d'exécution que sur la carte Fermi. Comme nous l'avons évoqué auparavant, la rétroprojection sera moins performante par rapport à la projection, cela se confirme par les résultats. Le projecteur est deux fois plus rapide que le rétroprojecteur sur l'architecture Tesla. Sur l'architecture *Fermi*, le projecteur est trois fois plus rapide par rapport au rétroprojecteur. Un des principaux intérêts de l'architecture multi-FPGA est qu'elle consomme moins d'énergie par rapport aux cartes GPU.

3.3.2 Processeur *Cell*

Le processeur *Cell* est optimisé pour le calcul vectoriel, il est constitué de huit cœurs dédiés au calcul, les SPE (*Synergistic Processing Elements*), et d'un cœur principal, le PPE (*Power Processor Element*) qui prend en charge le contrôle, la planification des tâches et les transferts mémoire.

Le processeur Cell est sorti en 2005. Il a été tout de suite utilisé dans le monde scientifique comme une solution architecturale parallèle à faible coût.

La plus grande partie des travaux utilisant le processeur *Cell* en reconstruction tomographique a été faite par Kachelrieß [61, 62, 63]. Un algorithme de reconstruction en *cone-beam* par FDK a été utilisé. La stratégie de parallélisation adoptée est le découpage du volume en sous-volumes comme dans [99]. Chaque SPE reconstruit le sous-volume à partir des sous-projections correspondantes. La taille du sous-volume est choisie de façon à pouvoir être stockée dans la mémoire locale de chaque SPE. La taille de cette mémoire est de 256 Ko. Elle contient les instructions à exécuter et les données à traiter. Le chargement des données de projection de la mémoire globale vers cette mémoire locale est fait par une technique de *double buffering*, qui permet de masquer le temps de transfert par du recouvrement calculs/communications. Le PPE se charge du transfert des projections en mémoire locale et du transfert du sous volume en mémoire globale. Le facteur d'accélération obtenu ici est $\times 14$ par rapport à un processeur *Xenon*. La ressource limitée en taille de mémoire locale, la difficulté d'implémentation du code et l'évolution des processeurs modernes avec un nombre plus important de cœurs (8, 12, 16 cœurs) ont rendu le processeur *Cell* moins attractif.

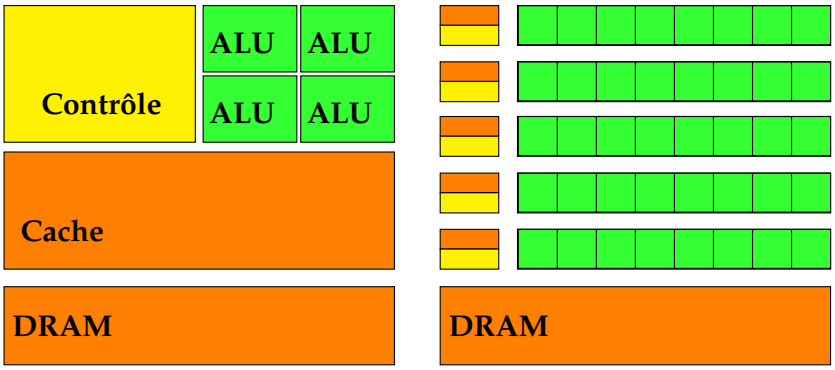
Knaup *et al.* [70] ont utilisé deux processeurs *Cell* de la carte *dual Cell Blade*, pour implémenter un algorithme itératif OSC (*Order Subset Convex*) couplé avec un algorithme FDK. Un processeur *Cell* est utilisé pour faire la reconstruction FDK, qui sert à initialiser l'algorithme itératif qui est exécuté par l'autre processeur. Le découpage du volume a été adopté comme stratégie de parallélisation : chaque SPE reconstruit un bloc du volume.

3.3.3 GPU

Les cartes graphiques sont utilisées à la base pour le calcul du rendu de scènes en 3D sur une grille 2D de pixels. L'introduction d'unités de calcul avec plus de précision et d'une programmabilité par un langage de haut niveau a rendu plus accessible l'utilisation des cartes graphiques dans le domaine du calcul scientifique. Aujourd'hui les cartes GPU sont vues comme des coprocesseurs utilisés pour accélérer une partie d'un problème. En effet avec les architectures GPUs on peut obtenir des performances, qui sont évaluées en terme d'opérations flottante par second, $\times 10$ par rapport aux valeurs réalisables avec les architectures de type CPUs. Cet écart est du principalement à une conception différente entre les deux types de processeurs (voir FIGURE 3.5). Le processeur de type CPU est optimisé pour le code séquentiel. Il se sert de la logique de contrôle pour permettre des instructions d'un *thread* d'être exécutées en parallèle. Le CPU est doté d'une grande taille de mémoire *cache*, qui réduit les temps de latence d'accès aux donnés.

Au contraire, le processeur de type GPU est optimisé pour l'exécution d'un nombre massif de *threads*. Le processeur GPU compte moins de transistors dédiés aux unités de contrôle. Ainsi, le processeur GPU est mieux adapté pour exécuter des opérations simples

sur de très gros volumes de données.



(a) CPU (b) GPU

FIGURE 3.5 – Difference entre l'architecture CPU et GPU [90]

Le développement sur carte GPU Nvidia est fait par une interface de programmation de haut niveau, CUDA (*Compute Unified Device Architecture*), qui est une extension du langage *C*. Le langage CUDA facilite la parallélisation du code sur l'architecture parallèle des GPU. L'architecture de base de la carte Fermi par exemple est constituée de 448 processeurs. Les processeurs, ou SP (*Stream Processor*), sont groupés en blocs. Le bloc, (qui contient 48 processeurs) est appelé MP (*Multi Processor*). L'architecture mémoire du GPU est hiérarchique :

- registre : utilisé pour stocker les variables d'exécution, le nombre de registres est limité par MP
- mémoire partagée : définie pour chaque MP, permet la communication entre les SP à l'intérieur du même MP
- cache L1/L2 (*prefech*) des données stockées en mémoire globale, permet la réduction du temps d'accès aux données
- mémoire constante, est un cache à accès en lecture seule
- mémoire texture, est un cache à accès en lecture seule (avec interpolation ou accès directe)
- mémoire globale, accès en lecture-écriture.

Les MPs d'un GPU ont une architecture de type SIMD, la même instruction est exécutée physiquement en parallèle par 16 *threads*. Les *threads* sont regroupés en blocs et les blocs forment une grille. On parle ainsi de *kernel grid*. Les *threads* contenus dans un bloc sont exécutés par un même MP en groupe de 2x16 *threads* (détails sur la gestion de *threads* lors d'une exécution en [90]). Depuis leur utilisation en calcul scientifique, différentes architectures se sont succédées : G71, G80, Fermi et aujourd'hui Kepler pour ce qui est des familles du constructeur Nvidia. Le nombre des SP a progressivement augmenté (l'architecture Kepler compte plus de 1500 SP). Les opérations atomiques en précision flottante sont disponibles à partir de l'architecture Fermi. De plus la précision flottante respecte les standards IEEE. D'autres langages de programmation ont été développés

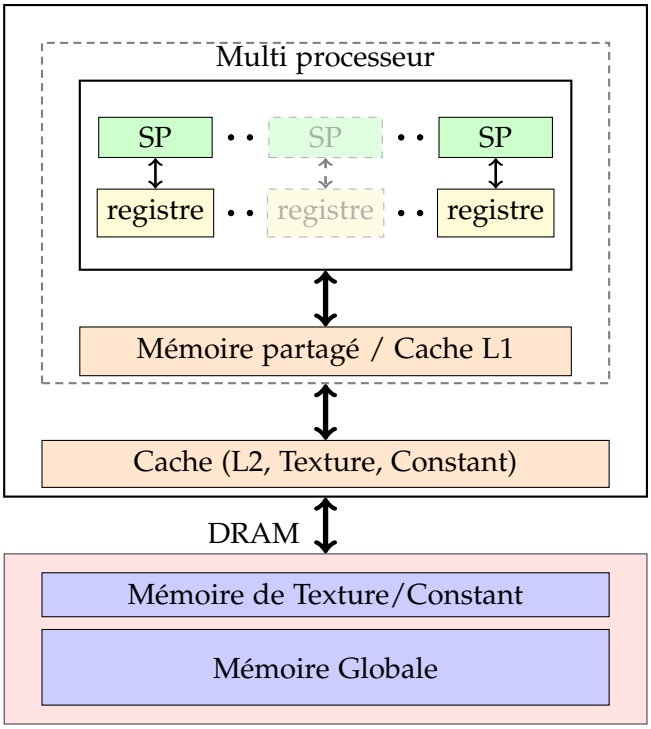


FIGURE 3.6 – Structure hiérarchique de la mémoire GPU.

comme *C++*, *OpenCL*, *Python*, etc.

Les premiers travaux d'accélération de la reconstruction en tomographie sur carte GPU ont été effectués en utilisant le pipeline graphique [85, 84]. En effet l'opérateur de projection tomographique est similaire au rendu d'une scène 3D sur un plan 2D. Mais la précision des calculs en virgule fixe sur 12 bits est insuffisante pour atteindre des images de qualité. Par la suite l'utilisation du pipeline graphique a été remplacée par le langage CUDA. Ceci donne plus de liberté au développement pour la mise en œuvre d'algorithmes parallèles.

Okitsu [91] *et al.* dans la reconstruction *cone-beam* utilisant un algorithme analytique tel que FDK, adoptent une stratégie différente par rapport à ce que nous avons analysé jusqu'à maintenant. Ils prennent en compte un volume de taille 512^3 voxel du fait de la limitation de la taille de mémoire des GPU † par rapport au jeu de données. Pour un gros volume il n'est pas possible d'exécuter entièrement la reconstruction sur GPU. Leur choix a été de stocker le volume de reconstruction entièrement sur le GPU et à la fin de la reconstruction le volume est copié dans la mémoire du CPU. Leur technique de parallélisation se déroule ainsi : les premières n projections sont transférées en mémoire GPU à partir du CPU, chaque projection est traitée comme une image, filtrée et rétro-projetée dans l'espace du volume par une méthode de *voxel-driven* (*pixel-driven* en 3D)

†. carte Nvidia GTX 8880 avec 768 Mo

et chaque *thread* est responsable de la mise à jour d'un voxel. Cette opération est faite itérativement sur la projection afin de reconstruire le volume. Lorsque il y a de la mémoire vide sur le GPU de nouvelles n projections sont envoyées sur le carte. Le transfert de mémoire entre le CPU et le GPU est entièrement masqué par les calculs. La méthode peut être représentée par l'organigramme suivant (voir FIGURE 3.7) :

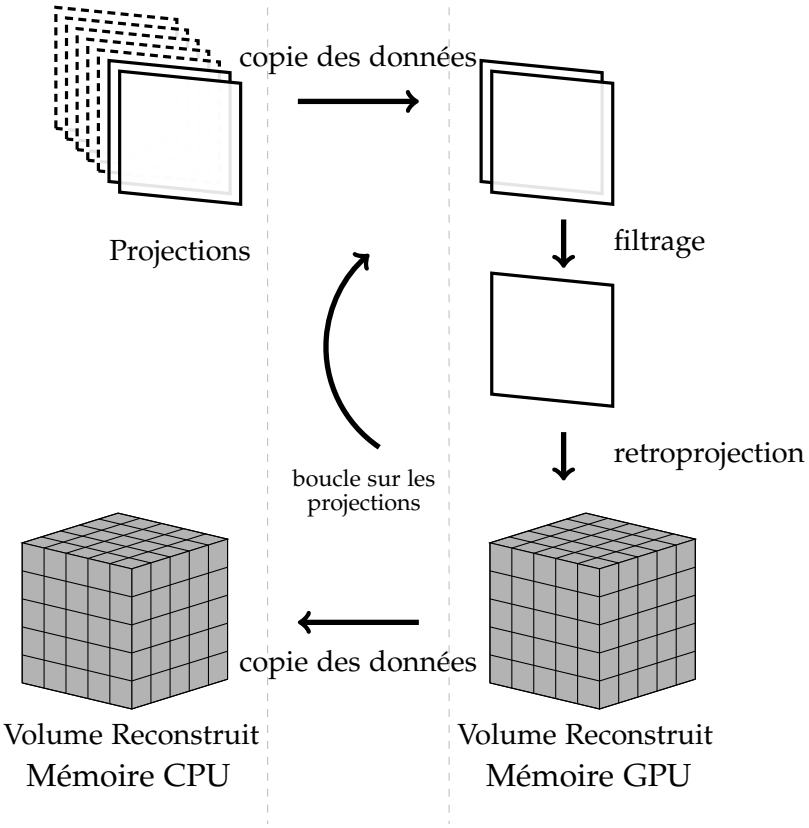


FIGURE 3.7 – Schéma de la mise en œuvre de la méthode de reconstruction sur GPU par Okitsu *et al.* [91].

D'autres travaux sur l'accélération de l'algorithme analytique ont été abordés par [125, 130, 25]. Les stratégies de parallélisation suivantes sont adoptées : découpage du volume à reconstruire et préchargement en mémoire partagée interne au GPU afin de réduire les accès en mémoire globale car les accès en mémoire globale demandent plus de cycles d'horloge pour y accéder.

Dans les travaux de Chou [25] *et al.*, les auteurs utilisent le GPU pour accélérer l'opérateur de projection. La projection est de type *ray-driven* à échantillonnage régulier. La projection avec projecteur à échantillonnage régulier se base sur la somme des valeurs obtenues par interpolation bilinéaire des quatre pixels voisins dans l'image. Ils proposent une approche *multithreads* par multi-rayons par rapport à celle classique d'un *thread* par rayon. Leur stratégie de parallélisation repose sur le découpage du volume à reconstruire en sous-volumes. Ils créent un bloc de *threads* en 3D de la même taille que le sous-volume.

Ainsi un *thread* est chargé de calculer la contribution du voxel à la projection considéré. Le voxel est caractérise par l'identificateur du *thread*. Chaque *thread* écrit cette valeur en mémoire partagé. Lorsque tous les *threads* ont fini, la valeur de chaque projection est écrite en mémoire globale par une opération de réduction/sommation.

La taille du bloc a été choisie afin d'avoir un nombre de registres par *thread* suffisant, afin d'éviter l'écriture des registres en mémoire cache. Généralement, la taille du bloc dépend de l'architecture † utilisée.

Pour bénéficier de l'interpolation matérielle offerte par les GPUs, les auteurs utilisent la mémoire de texture pour stocker le sous-volume à traiter. La mise en œuvre du lancer de rayons par le modèle *multithreads* par multi-rayons accélère d'un facteur $\times 9$ l'opérateur de projection par rapport à l'approche classique d'un *thread* par rayon.

Gac *et al.* [48] ont mis en oeuvre un algorithme de reconstruction itératif 3D avec un projecteur implémenté par la méthode de *ray-driven* à échantillonnage régulier et le rétroprojecteur est implémenté par la méthode *voxel-driven*. Lors de la projection, un bloc du sous-volume est stocké en mémoire de texture et la valeur à chaque échantillon est calculée par interpolation trilinéaire. Vice-versa, les sous-projections sont stockées en mémoire de texture où la valeur du pixel est calculée par interpolation bilinéaire. Les données sont stockées pour atteindre la localité spatiale et temporelle des accès en mémoire. Leur mise en œuvre sur une carte GPU ‡ atteint un facteur d'accélération $\times 100$ pour une itération, par rapport à une implémentation CPU en reconstruisant un volume de $256^2 \times 96$.

Puis, Gac *et al.* [47] ont développé des algorithmes de reconstruction 3D sur un serveur de calcul multi-GPU. Du point de vue de l'algorithme de reconstruction utilisé, les auteurs ont utilisé les travaux développés précédemment [48]. La reconstruction d'un volume de la taille 1024^3 a été calculée sur un serveur de calcul avec huit GPUs†. Afin de pouvoir stocker les données en mémoire sur chaque GPU un découpage du volume a été réalisé. Plus en détail, pour la projection, le volume a été découpé en deux sous volumes selon le plan horizontal. En même temps, le plan du détecteur a été divisé selon l'axe vertical et selon les angles de vue. Pour la rétroprojection, chaque GPU reconstruit une tranche horizontale du volume à partir du demi-plan de projection correspondant. Cette parallélisation a permis d'atteindre un facteur d'accélération de $\times 300$ par rapport à un code CPU non optimisé.

3.3.4 CPU et Cluster

Les CPUs ont eux-mêmes évolué avec les multi-coeurs (jusqu'à 16 actuellement) et avec les extensions de jeu d'instructions vectorielles (MMX, SSE, AVX). Ces jeux d'instructions étaient au départ destinés à l'affichage mais ils se sont généralisés au calcul parallèle en général. Par contre leur programmation reste explicite et plus lourde que celle des GPU.

†. carte Tesla C1060 avec 4 Go DRAM
‡. carte Nvidia GTX 295

Des compilateurs/paralléliseurs sont prévus pour aider au portage de code, mais sans la connaissance de l'architecture, il est difficile d'espérer obtenir toutes les performances possibles.

Le risque avec les architectures parallèles, est d'obtenir des performances même inférieures à celles d'un code séquentiel sur CPU. En fait, la solution idéale est de combiner les avantages des CPUs avec ceux des GPUs en les couplant et en connaissant bien chaque architecture. Augmenter les performances peut se faire au travers d'un cluster. Un cluster est un ensemble d'ordinateurs connectés entre eux par un réseau. On fait référence à cette topologie comme architecture de calcul distribué. Un exemple d'implémentation sur cluster est présenté dans les travaux de Reimann *et al.* [99]. La reconstruction est faite par une méthode analytique FDK [42]. L'idée de base est le découpage du volume en sous-volumes. Chaque ordinateur accède à tout l'ensemble des projections mais il est en charge de reconstruire seulement un sous-volume. À la fin des reconstructions, les sous-volumes sont rassemblés pour constituer le volume entier. Il existe par ailleurs maintenant des clusters de couples CPU/GPU qui permettent de bénéficier des avantages de chacun : les GPUs pour les calculs massivement parallèles réguliers et les CPUs pour les tâches indépendantes et le code séquentiel.

3.4 Conclusions

Nous avons analysé dans ce chapitre les aspects de la parallélisation en général. En particulier nous avons étudié les aspects de la mise en œuvre des opérateurs de projection et rétroprojection lors d'une parallélisation par rapport aux accès en mémoire.

Nous avons parcouru les diverses architectures parallèles (cluster, FPGA, processeur *Cell*, GPU et cluster) qui ont été utilisées en reconstruction tomographique. L'évolution des architectures spécifiques (FPGA et GPU) a remplacé progressivement aussi bien les clusters que les processeurs *Cell* pour leurs performances et leur facilité de programmation et d'installation (un ordinateur de bureau est équipé de cartes GPU performantes).

Le développement sur carte FPGA est plus laborieux par rapport à celui sur GPU. Mais, une carte FPGA permet de réduire le coût de la consommation énergétique sans que les performances ne soient réduites.

Étant donné le contexte de notre projet, les GPUs semblent un bon compromis en termes de performances/facilité de programmation grâce à l'interface de programmation CUDA.

Nous avons pris en compte dans cette étude une parallélisation en procédant à un découpage au niveau de la projection. En effet l'utilisation d'une grille irrégulière rend délicat l'opération de découpage du volume à reconstruire qui n'est pas géométriquement simple à découper. Par contre elle s'adapte très bien à une projection de type *ray-driven* que nous avons retenue.

Chapitre 4

Méthode de reconstruction basée sur un maillage adaptatif (ATM)

L'analyse des données est précédée d'une étape essentielle de segmentation permettant de délimiter les régions d'intérêt à considérer. Au cours des deux dernières décennies, de nombreuses études ont été menées conduisant à de multiples méthodes de segmentation proposant diverses interprétations des contours et différentes fonctionnelles d'énergies à minimiser. Dans ce contexte, l'idée de proposer une méthode de reconstruction de tomographie intégrant une étape de segmentation offre l'opportunité d'accéder à une reconstruction classique d'une image en niveaux de gris couplée à une segmentation des différents matériaux composant l'objet étudié, et permet donc de s'affranchir d'une étape de post-traitement d'images pour l'analyse des résultats. La méthode proposée dans ce travail, que nous noterons MAT (Maillage Adaptatif Triangulaire) ou ATM (*Adaptive Triangular Mesh* en anglais) par la suite, et décrite dans ce chapitre développe cette idée. Le processus mis en œuvre est une méthode de reconstruction d'images en tomographie par rayons X basée sur un algorithme itératif connu mais adapté à une représentation volumique de l'espace objet se démarquant de celle usuellement utilisée dans ce domaine (c'est-à-dire. voxélisation de l'espace de reconstruction). On ne considère plus ici un volume bidimensionnel décrit par des pixels (resp. tridimensionnel décrit par des voxels) mais un ensemble de triangles (resp. de tétraèdres) s'adaptant au contenu du volume objet. La densité des mailles élémentaires, leur position et leur forme sont optimisées par l'algorithme grâce à une étape de segmentation des différents matériaux composant l'objet étudié, simultanée à la reconstruction. En fin de processus, la méthode permet d'accéder à l'objet reconstruit dans son intégralité ainsi qu'à sa segmentation complète, la taille des données étant optimisée par le nombre d'éléments composant le maillage. Ainsi s'affranchit-on ici de toute étape supplémentaire de post-traitement, ce qui allège le traitement des données et conduit à une reproductibilité fiable du processus. Par ailleurs, la représentation de l'image par un maillage adapté à son contenu (par exemple à l'information qu'elle comporte) permet de réduire considérablement la taille mémoire de l'image reconstruite (réduction des problèmes de stockage informatique) et de traiter des données issues de détecteurs toujours plus pixélisés.

Ce chapitre s'articule autour des différentes étapes du processus, c'est-à-dire reconstruction, segmentation, adaptation du maillage, qui alternent successivement jusqu'à convergence (voir FIGURE4.1). Nous présenterons un résumé bibliographique des méthodes de reconstructions qui alternent la reconstruction à la segmentation. Puis, les discrétisations non pixélises utilisées dans la reconstruction tomographique. Ensuite, nous abordons tout d'abord l'étape d'initialisation qui consiste à construire un premier maillage à partir d'un nuage de points et d'affecter une valeur initiale à chaque maille. Nous présentons ensuite la phase de reconstruction tomographique dont la fonction est d'estimer les niveaux de gris dans l'image (coefficients d'atténuation des matériaux) par l'utilisation d'un algorithme itératif connu. Les principaux opérateurs de projection et de rétroprojection, adaptés aux mailles triangulaires (et tétraédriques), sont validés. Dans un troisième temps, nous décrivons l'étape de segmentation : la mise en œuvre de la méthode et son intégration dans le processus de reconstruction. Finalement, nous expliquons comment adapter le maillage au contenu de l'image segmentée et présentons les outils utilisés.

4.1 État de l'art des méthodes de reconstruction / segmentation simultanée

Les méthodes de reconstruction classiques sont basées sur la discrétisation du volume dans un ensemble de voxels et fournissent une carte de densité 3D. L'analyses du volume (de l'objet reconstruit) est ensuite mené grâce à l'application d'outils de traitement d'images notamment pour segmenter des régions d'intérêt. Beaucoup d'études ont été faites pendant les deux dernières décennies menant à de nombreuses méthodes de segmentation avec les différentes interprétations des contours et des diverses fonctionnelles à minimiser. Dans la dernière décennie, des auteurs ont proposé de nouvelles approches qui incluent la segmentation simultanément à la reconstruction.Ces autres approches consistent dans un établissement en modèle global des frontières.

4.1.1 Modèles géométriques

Les approches alternatives, aussi appelées méthodes basées sur la forme, modèlent la forme des objets à reconstruire en optimisant alternativement la position des frontières et les paramètres décrivant leur distribution d'intensité. Récemment, les modèles non paramétriques comme les contours actifs ont été développés et utilisés dans le domaine de la segmentation d'image. Plusieurs auteurs se sont concentrés sur la reconstruction d'une image binaire 3D composée d'un objet compact et uniforme totalement inclus dans un contexte uniforme [114]. La scène est reconstruite en utilisant des modèles superficiels paramétriques (splines) dont les paramètres sont évalués à partir des données. Les résultats obtenus avec cette approche polyédrique restent fortement liés à l'initialisation, pour laquelle les auteurs proposent d'utiliser une méthode globale des surfaces harmoniques. Senasli *et al.* [104, 105] ont aussi proposé d'utiliser un contour actif basé sur un processus

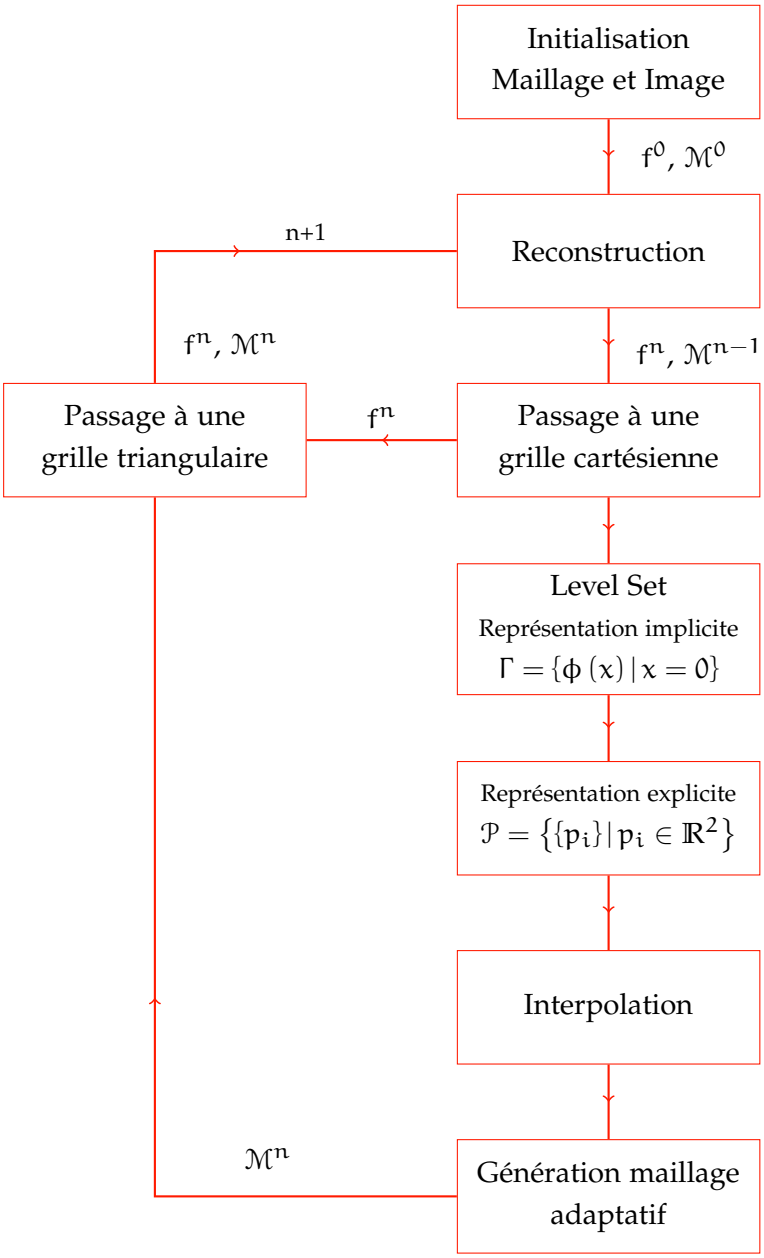


FIGURE 4.1 – Synopsis de la méthode ATM

stochastique pour la reconstruction d'une image binaire tomographique dans le contexte de faible quantité de données. La courbe définie par une fonction spline est utilisée pour décrire les contours de vaisseaux. En utilisant ces méthodes basées sur la forme d'un objet dans la reconstruction tomographique, Gaullier *et al.* [50] ont proposé de définir un *a priori* sur la forme, qui est décrite par des moments de Legendre. Cette information *a priori* établit une contrainte sur la longueur de la courbe qui doit être appropriée à l'objet reconstruit. Zheng *et al.* proposent d'utiliser un modèle superficiel statistique de l'objet et un modèle de recalage pour reconstruire un modèle superficiel 3D d'un patient à partir des radiographies 2D calibrées [131]. La technique est basée sur une méthode itérative de recalage non rigide des formes caractéristiques extraites d'un modèle superficiel 3D statistique avec les données radiographies acquises.

Les méthodes de segmentation utilisant le *level set* ont premièrement été présentées dans le domaine du traitement des images pour la segmentation d'images et pour la reconstruction des formes. Alors, ils ont été appliqués pour simultanément évaluer les valeurs dans les formes reconstruites et des résultats prometteurs ont été obtenus pour leur utilisation dans des problèmes de reconstruction tomographique [102, 103, 44, 2, 127, 98]. Schweiger *et al.* [102, 103] ont utilisés le *level set* pour reconstruire la forme des objets à partir de données bruitées et acquise sur un domaine angulaire restreint en tomographie électromagnétique. Feng *et al.* ont utilisé la méthode de *level set* avec un petit nombre de coefficients de texture pour reconstruire dans le cadre d'un nombre de vues limitées en tomographie [43]. Yoon *et al.* ont proposé un algorithme itératif pour la reconstruction tomographique qui segmente simultanément et reconstruit le domaine d'image [127, 128]. La segmentation et la reconstruction sont réalisées en alternant une étape de segmentation qui fait évoluer la fonction de *level set two-phase* et une étape de reconstruction qui calcule une estimation des valeurs d'intensité de l'objet inconnu utilisant une méthode d'optimisation du gradient conjugué comme algorithme de reconstruction. La fonction *two-phase* présente des limitations : elle peut fournir des informations suffisantes lorsqu'un seul matériau est intéressant par rapport à d'autres structures qui ont moins d'intérêt (ils se mêlent à l'image de fond). Une extension de la méthode est nécessaire lorsque l'objet est composé de plusieurs matériaux, par exemple en ajoutant une deuxième fonction de *level set*.

Différentes fonctionnelles comme l'énergie élastique liée avec le détecteur de bord, ou les fonctionnelle basées sur les régions d'une image, par exemple la fonctionnelle de Mumford-Shah, ont été considéré et les modèles géométriques ont été modélisés par les courbes paramétriques comme les *snake* ou le *level set*. En 2007, Anneau *et al.* ont proposé l'idée de segmenter simultanément et reconstruire à partir des données mesurées en tomographie X. [57]. Le travail est basé sur l'observation que les méthodes de segmentation échouent souvent lorsque l'image est très bruitée. Dans la méthode proposée, le contour de segmentation et la densité correspondante sont trouvés comme minimisation de la fonctionnelle de Mumford-Shah directement à partir des données

mesurées.

4.1.2 Représentations d'images alternatives

Toutes les méthodes de reconstruction et segmentation précédemment décrites utilisent une représentation d'image qui consiste à diviser le volume dans un tableau régulier de pixels (ou voxels dans la 3D) et la supposition que l'objet reconstruit est constant à l'intérieur de chaque élément de la grille. Quelques auteurs ont proposé d'autres sortes de représentation pour améliorer la reconstruction des bords francs (les objets biologiques sont lisses). En 1992, Lewitt a introduit la fonction de Bessel-Kaiser, aussi appelée *blob* comme une représentation d'image alternative [77]. L'auteur a décrit une famille d'éléments de volumes sphériques dont deux paramètres contrôlent le lissage d'image (ou contraire des pixels conventionnels qui sont discontinus). De plus la symétrie de ces éléments sphériques mène au calcul efficace des projections. Cette approche augmente la qualité des images reconstruites en comparaison à celles obtenue par une approche voxélisée.

Dans le cadre de tomographie SPECT, Brankov *et al.* ont proposé une méthode de reconstruction en 2D d'une image basée sur une discrétisation irrégulière triangulaire [12]. Le modèle est basé sur un échantillonnage non uniforme dont la densité des noeud est directement liés aux détails de l'objet. La génération du maillage est basée sur une image de référence obtenue en utilisant une reconstruction FDK (Feldkamp-Kress-Davis) dont les noeuds des mailles sont placés à partir d'une cartographie des caractéristiques de l'image qui est calculée sur la grille régulier de pixels de l'image de référence. Le nombre de noeuds est déterminé par le critère de la longueur de description minimale (MDL pour *Minimum Description Length* en anglais) qui code l'image de référence avec le nombre minimal de bits. Des algorithmes itératifs classiques comme EM et MAP (*Maximun A Prosteriori*) ont été adaptés à la maille triangulaire pour la reconstruction. Les auteurs démontrent qu'un modèle de maille adaptative au contenu fournit une discrétisation précise du problème avec moins d'inconnues qu'en utilisant une grille pixélisée et il peut ainsi améliorer le temps de calcul et la qualité des images reconstruites puisque la densité de maille est adaptée aux détails de l'image.

Battle *et al.* [6] ont proposé une méthode qui explore la déformation d'une maille triangulaire en évaluant directement les déplacements des sommets de triangles dans le cadre de l'approche Bayésienne. La méthode est appliquée sur des données SPECT. Les objets sont représentés par une surface triangulaire incluant une région uniforme. Cette approche s'avère être simple et efficace, mais soulève des questions complexes pour la reconstruction de plus qu'une région. Ils ont proposé une méthode itérative où la carte inconnues est modélisée comme un ensemble de régions géométriques ayant un coefficient d'atténuant uniforme. La reconstruction consiste à déformer l'espace afin d'approximer au mieux les données acquises.

Sitek *et al.* dans [113] ont proposé une méthode 3D de reconstruction tomographique en utilisant une représentation du volume par un nuage de points et représenté par une tétraèdralisation. La densité des noeuds définit la résolution locale de l'image et l'intensité des points du volume est évaluée par une interpolation linéaire entre les valeurs des quatre noeuds qui définissent chaque tétraèdre. Le calcul de la matrice du système est lourd et complexe, mais peut être optimisé grâce à l'interpolation linéaire matérielle offerte par les processeurs graphiques (GPU).

4.2 Choix d'une structure de données adaptée

L'opérateur de projection représente la contribution de chaque triangle traversé par un rayon donné. Le rayon est défini par deux points : la position de la source et celle du pixel détecteur.

Dans le cas d'une grille régulière de pixels, l'approche usuelle est de calculer les intersections d'un rayon donné avec les pixels de la grille et de les sommer. Cette approche est coûteuse en temps de calcul mais peut être stockée définitivement dans la matrice de projection, la grille étant immuable au cours de la reconstruction.

Dans le cas où nous nous plaçons, c'est-à-dire une grille irrégulière composée de mailles élémentaires dont la position et la forme changent au cours de la reconstruction, l'approche usuelle doit être optimisée afin d'augmenter l'efficacité calculatoire, la matrice de projection ne pouvant être stockée une fois pour toutes. L'idée est de créer une structure de données qui permette de stocker pour chaque maille les informations des mailles voisines (mailles ayant une arête en commun). Ainsi, lorsqu'un rayon donné traverse une première maille, nous parcourons et trouvons de façon itérative toutes les mailles traversées par le rayon. Le nombre de tests à effectuer est strictement réduit au nombre de triangles traversés, et le nombre d'intersections à calculer est égal au nombre de mailles plus un. L'approche utilise un algorithme de lancer de rayons.

4.2.1 Maille de base

Structure triangle

Dans ce travail, un algorithme de lancer de rayon sur un maillage adaptatif a été développé se basant sur une méthode arêtes/mailles (voir Algorithme 4). L'implantation de l'algorithme de lancer de rayons utilise alors trois structures de données ainsi définies en 2D dans le tableau (Table 4.1) :

- `Point` : 2 flottants pour les coordonnées `x` et `y`
- `Edge` : 2 entiers correspondants aux indices des deux `Point` de l'extrémité, et 2 entiers pour stocker les indices des `Triangle` auxquels elle appartient. La valeur -1 indique que `Edge` se situe à la limite externe du maillage.
- `Triangle` : 3 entiers pour identifier les trois `Edge` et un flottant pour stocker la valeur du niveau de gris.

Les structure arêtes/maille permettent de stocker les informations du voisinage de chaque triangle. En effet les informations stockés sont les indices des triangles voisins. Ainsi, à partir d'un triangle donné on peut savoir les triangles voisins par leurs indices.

Point	Edge	Triangle
`struct { double x,y; } Point;`	`struct { int idx, Point A; Point B; int Ntr1; int Ntr2; } Edge;`	`struct { int idx, int idxEdge1; int idxEdge2; int idxEdge3; double value; } Triangle;`

TABLE 4.1 – Structures de données de `Point`, `Edge` et `Triangle` utilisées pour l'implantation de la méthode.

Algorithme 4 *Traversée du maillage irrégulier composé de triangles par la méthode arête/triangle.*

```
1 : A partir de e_0
2 : p_old ← intersection(rayon, e_0)          // Calcule le premier point d'intersection
3 : if p ← intersection(rayon, e_1) then
4 :    l = calcule(p_old, p)                  // Calcule la longueur d'intersection
5 :    p_old ← p                              // Met à jour du prochain point de départ
6 :    next ← e_1.idx()
7 : end if
8 : if p ← intersection(rayon, e_2) then
9 :    l = calcule(p_old, I)                  // Calcule la longueur d'intesection
10 :   p_old ← p                              // Met à jour le prochain point de départ
11 :   next ← e_2.idx()
12 : end if
13 : triangle next ← E(next).triangle         // Calcule l'indice du triangle suivant
```

Structure tétraèdre

Le codage en 3D se base sur le travail de Toczek [116] qui propose un algorithme dans le cadre du rendu d'image. Les structures de données 3D sont les suivantes :

- `Sommet` : 3 flottants pour les coordonnées x, y et z
- `Tétraèdre` : 4 entiers pour identifier les `Sommet`
- `Voisin` : les indices des quatre voisins, ainsi alloué en position i correspond le voisin opposé au sommet i.
- `Densité` : les densités (flottants) de chaque tétraèdre

On note le rayon r par son point d'origine O et sa direction $\vec{d}$.

Le principe de l'algorithme de traversée d'un maillage composé de tétraèdres se base sur la méthode de comparaison de paramètres : on considère un tétraèdre défini par ses quatre `Sommet` A, B, C et D ; pour chaque face i, définie par un triplet de `Sommet` par exemple (A, B, C), on procède de la façon suivante :

- on calcule le vecteur normal $\vec{n}$ à la face, par exemple $\vec{BA} \times \vec{BC}$
- on calcule $s_1 = \vec{BD} \cdot \vec{n}$ et $s_2 = \vec{d} \cdot \vec{n}$
  si s_1 et s_2 sont de signes opposés, cela traduit le fait que le rayon r est susceptible de sortir par cette la face ; alors on calcule le paramètre $p_i = \dfrac{\vec{OB} \cdot \vec{n}}{\vec{d} \cdot \vec{n}}$

La face de sortie est celle pour laquelle $p = \min\{p_i\}$. La longueur d'intersection est obtenue par soustraction entre p actuel et le précédent.

Une variante à cette méthode consiste à stocker les équations de plans. Les équations sont précalculées ce qui rend la traversée plus rapide.

Les différentes données (coordonnées, équations de plan, etc.) sont organisées dans des tableaux comme décrit dans le tableau (TABLE 4.2).

Type de données	Structure utilisée : tableau de
`coordonnées`	flottants $[x_1,y_1,z_1,....,x_n,y_n,z_n]$, $\{x_n, y_n, z_n\} \in \mathbb{R}$ $n = 1, \cdots, N$
`plans`	flottants $[a_1,b_1,c_1,d_1,....,a_m,b_m,c_m,d_m]$, $\{a_m, b_m, c_m, d_m\} \in \mathbb{R}$, $m = 1, \cdots, M$
`sommets`	entiers $[i_1,j_1,k_1,l_1,....,i_K,j_K,k_K,l_K]$ $\{i_k, j_k, k_k, l_k\} \in \mathbb{Z}$, $k = 1, \cdots, K$
`voisins`	entiers $[i_1,j_1,k_1,l_1,....,i_k,j_k,k_k,l_k]$ $\{i_k, j_k, k_k, l_k\} \in \mathbb{Z}$, $k = 1, \cdots, K$
`tétraèdres`	entiers $[i_1,j_1,k_1,l_1,....,i_k,j_k,k_k,l_k]$ $\{i_k, j_k, k_k, l_k\} \in \mathbb{Z}$, $k = 1, \cdots, K$
`densités`	flottant $[f_1,....,f_k]$ $f \in \mathbb{R}$, $k = 1, \cdots, K$

TABLE 4.2 – Structures de données utilisées dans le cas 3D - N représente le nombre de sommets du volume tétraédrique, K le nombre de tétraèdres, M représente le nombre de plans des facettes du volume tétraédrique. Le tableau des sommets est utilisé seulement pour calculer les équations de plans.

4.2.2 Discrétisation de l'espace

Dans le Chapitre 1 nous avons décrit le modèle de représentation d'une image par une fonction nodale, où l'image est discrétisée par un pavage irrégulier (maillage). Ainsi l'image sur chaque élément du maillage (triangle) est définie par une fonction continue :

$$f_n(x) = \sum_{i=1}^{3} f(x_i)\,\psi_i(x)\ x \in D_n,\ D_n \in \mathcal{M} \tag{4.1}$$

où $\psi_i(x)$ est la fonction nœudale définie au sommet i (aussi appelée *shape function* [7]). N représente le nombre de mailles triangulaires qui constituent le maillage $\mathcal{M}$ et D_n représente le nième élément triangulaire du maillage M. Si on associe une fonction constante à chaque élément comme :

$$f_n(x) = f(x_i)\mathbb{1}_{D_n},\ D_n \in \mathcal{M} \tag{4.2}$$

Alors, l'image entière f est discrétisée par un maillage $\mathcal{M} \subset R^2$, est définie comme :

$$f = \sum_{n=1}^{N} f_n \mathbb{1}_{D_n},\ D_n \in \mathcal{M} \tag{4.3}$$

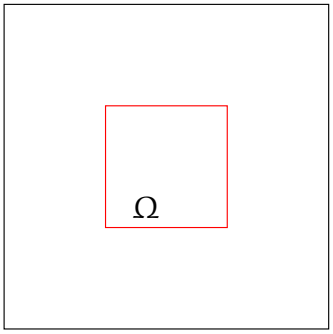
où N est le nombre de triangles dans $\mathcal{M}$, et f_n est la valeur associée à chaque triangle.

Nous avons choisi d'associer la valeur des triangles sur leur face (fonction constante) plutôt que d'associer les valeurs estimées sur les sommets des triangles (fonction continue). En effet, ce choix permet une représentation de l'objet la plus adaptée dans le contexte de la reconstruction tomographique. La FIGURE 4.2, montre la représentation d'un objet simple (un carré voir FIGURE 4.2a) avec les deux représentations citées (constante FIGURE 4.2c et continue FIGURE 4.2d) étant donné un maillage $\mathcal{M}$ dont les éléments sont adaptés aux contours de l'objet (voir FIGURE 4.2b) : on constate aisément que la représentation par une fonction constante reproduit fidèlement l'objet étudié contrairement à la représentation par une fonction continue qui ne permet pas de matérialiser correctement les contours de l'objet. La FIGURE 4.3 montre qu'il faut augmenter considérablement le nombre d'éléments triangulaires pour approcher une représentation plus fidèle à l'objet. Les coins et les contours de l'objet restent cependant mal dessinés.

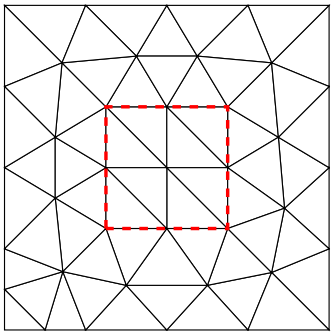
4.3 Initialisation du maillage

L'étape d'initialisation permet de construire la grille initiale sur laquelle sera ensuite estimée une première reconstruction de l'objet étudié. Dans cette étape, nous définissons donc la forme et le nombre de triangles constituant ce premier maillage. Trois types de maillages peuvent être utilisés :

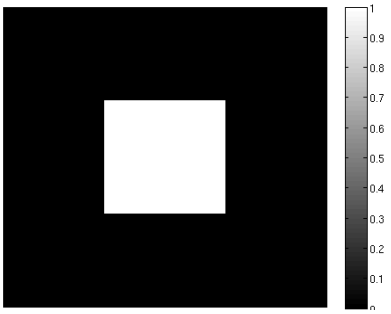
1. Un maillage irrégulier dont les caractéristiques répondent à une triangulation de Delaunay (voir FIGURE 4.4a)
2. Un maillage régulier dont les éléments sont des triangles isocèles rectangles isométriques (voir FIGURE 4.4b)



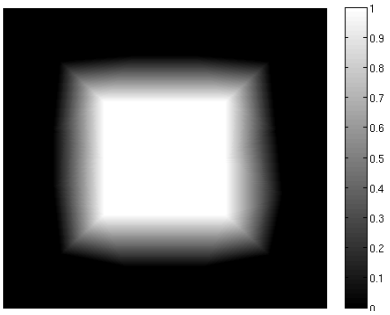
(a) Objet test



(b) Maillage $\mathcal{M}$

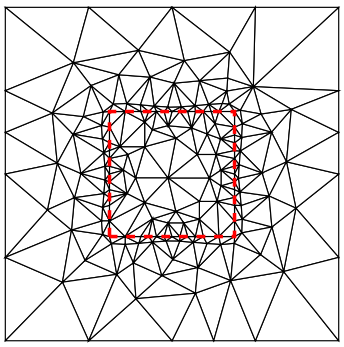


(c) Représentation par la fonction constante

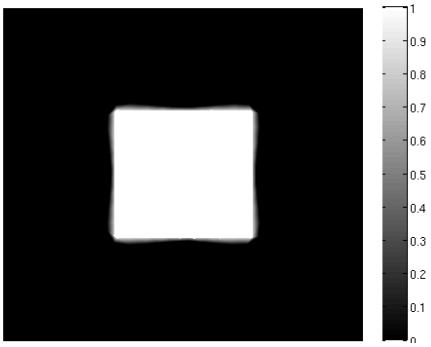


(d) Représentation par la fonction continue

FIGURE 4.2 – Représentation d'un objet test 2D carré (a) sur un maillage $\mathcal{M}$ composé de 58 mailles triangulaires (b) par une fonction constante (c) et par une fonction continue (d).



(a) Maillage $\mathcal{M}'$



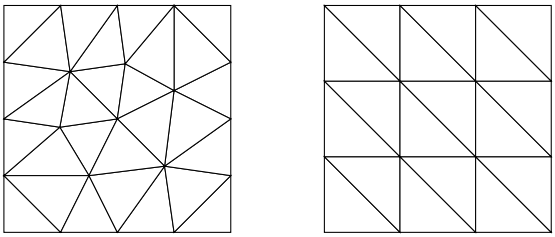
(b) Représentation par la fonction continue

FIGURE 4.3 – Représentation de l'objet test carré à l'aide du maillage $\mathcal{M}'$ composé de 177 mailles triangulaires (a).

3. Un maillage préexistant d'un objet CAO ou d'un examen antérieur du même objet avec cette méthode

Dans ce travail, seuls les deux premiers types d'initialisation ont été mis en œuvre. Le troisième type d'initialisation n'a pas été abordé mais pourrait être appliqué dans le cadre du contrôle non destructif de pièces manufacturées, de *reverse engineering* ou encore pour le suivi de pathologies ou l'évolution d'un processus lent en imagerie médiale (par exemple déformation de l'os pour le suivi de l'arthrose). Ces dernières applications nécessiteraient cependant une méthode de recalage et la définition d'un protocole d'acquisition afin de reproduire des conditions d'acquisition similaires d'un examen à un autre.

Notons que dans le cadre de ces travaux, la librairie CGAL (Computational Geometry Algorithms Library) [20] a largement été utilisée pour manipuler les maillages. C'est le cas pour cette première étape pour générer les maillages irréguliers.



(a) Maillage irrégulier (b) Maillage régulier

FIGURE 4.4 – Exemple de maillages initiaux.

Afin de réduire considérablement les coûts en terme de temps de calcul, nous proposons de diminuer dès le départ le nombre d'inconnues à estimer en générant un maillage initial fonction du nombre de pixels détecteur utilisés lors de l'expérimentation. Alors le nombre de triangles K du maillage initial est tel que :

$$\frac{1}{10} N^d \leqslant K \leqslant N^d \tag{4.4}$$

où d est la dimension de reconstruction. Et $N = \max\{N_x, N_y\}$, où N_x et N_y représentent le nombre de pixels détecteur selon les axes x et y.

Notons que dans le cas usuel d'une reconstruction basée sur une discrétisation par pixels les dimensions de la grille de reconstruction sont égales à N^d[15].

4.4 Reconstruction tomographique

L'étape de reconstruction met en œuvre un algorithme itératif de reconstruction tomographique connu, adapté à la discrétisation du volume en maille triangulaire dans le cas bidimensionnel et tétraédrique dans le cas tridimensionnel. Les modifications apportées concernent principalement les principaux opérateurs de projection et de rétroprojection.

4.4.1 Projecteur / Rétroprojecteur

Comme nous l'avons rappelé dans le premier chapitre, la matrice de projection H contient la contribution de chaque élément du volume objet à chaque rayon de projection :

$$H_{i,j} = \begin{pmatrix} h_{1,1} & h_{1,2} & \cdots & h_{1,j} \\ h_{2,1} & h_{2,2} & \cdots & h_{2,j} \\ \vdots & \vdots & \ddots & \vdots \\ h_{i,1} & h_{i,2} & \cdots & h_{i,j} \end{pmatrix} \tag{4.5}$$

Les contributions des éléments de la grille à chaque rayon de projection peuvent être calculées de diverses façons. Les rayons considérés sont définis par le projecteur utilisé. Dans ce travail, nous considérons un projecteur de type *pixel-driven* (rayons centrés sur les pixels détecteur), et calculons la contribution de chaque élément de volume comme la longueur des rayons interceptés par les triangles en 2D (resp. les tétraèdres en 3D).

L'élément $h_{i,j}$ représente la longueur l'intersection du rayon i dans le triangle j (voir schéma FIGURE 4.5). Contrairement au cas conventionnel où la grille reste fixe, les éléments h_{ij} ne pourront pas être stockés tout au long du processus de reconstruction : chaque fois que les éléments du maillage seront modifiés, la matrice H devra être recalculée.

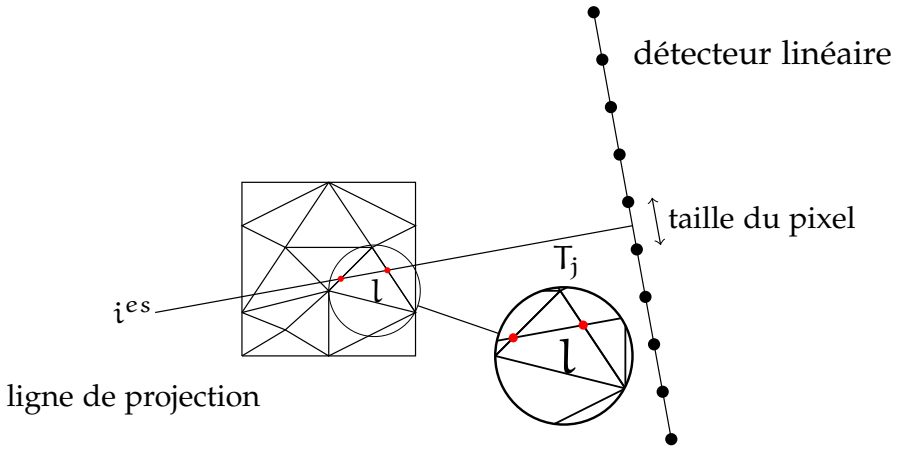


FIGURE 4.5 – Longueur du rayon i intercepté par le triangle j.

4.4.2 Validation du projecteur en 2D et en 3D

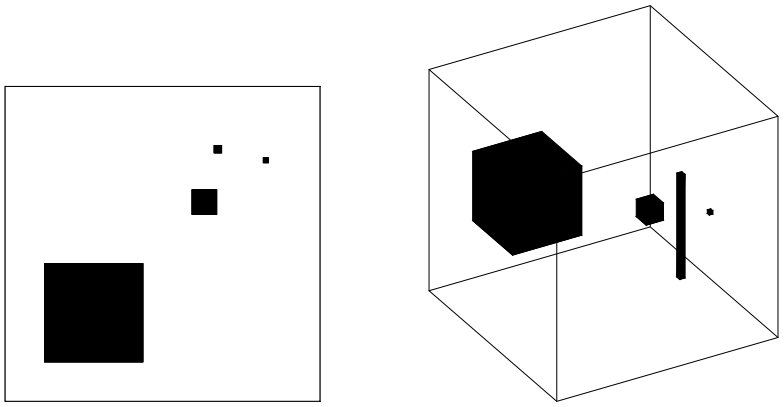
Dans cette section, nous présentons les résultats concernant la validation de l'opérateur de projection. En 2D et en 3D, la validation sera conduite sur l'objet décrit dans le tableau (voir TABLE 4.3) : l'objet est constitué de quatre parallélépipèdes de différentes tailles, le plus petit étant représenté dans le cas conventionnel par un seul pixel. Dans le cas 2D, l'opérateur de projection, adapté à une représentation par triangles à valeur constante, a

été validé en 2D en utilisant une section $z = 0$. La FIGURE 4.6 illustre l'objet 3D défini dans le tableau (TABLE 4.3), ainsi que la coupe utilisée pour le cas 2D.

Les projections calculées avec le projecteur proposé seront comparées aux projections analytiques. Nous avons utilisé une configuration d'acquisition de type *fan-beam* en 2D et de type *cone-beam* qui est résumée dans le tableau (voir TABLE 5.5).

Objet	x_{min}, x_{max}	y_{min}, y_{max}	z_{min}, z_{max}
cube externe	-0.25, 0.25	-0.25, 0.25	-0.25, 0.25
évidement 1	-0.1875, -0.03125	-0.1875, -0.03125	-0.078125, 0.078125
évidement 2	0.046875, 0.085938	0.046875, 0.085938	-0.019531, 0.019531
évidement 3	0.082031, 0.09375	0.144531, 0.15625	-0.117188, 0.117188
évidement 4	0.160153, 0.167969	0.128906, 0.136719	-0.003906, 0.003906

TABLE 4.3 – Description de l'objet test utilisé pour la validation du projecteur.



(a) Coupe 2D obtenue à partir de l'objet 3D $z = 0$

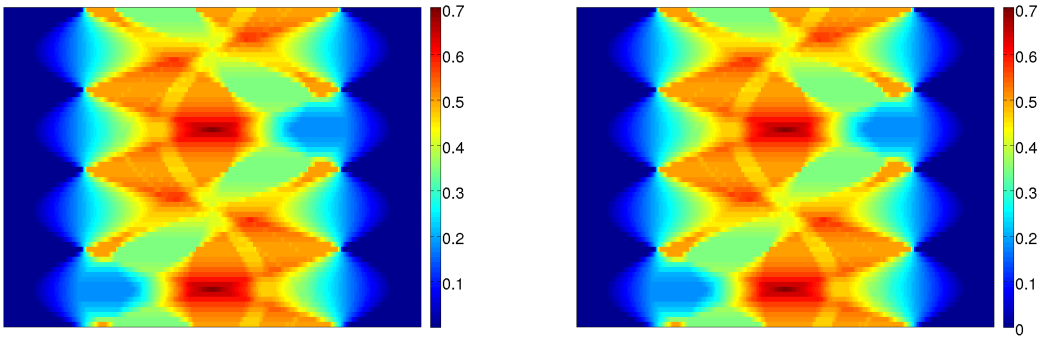
(b) Objet 3D

FIGURE 4.6 – Objet utilisé pour la validation du projecteur 2D et 3D.

La FIGURE 4.7 montre les projections calculées de façon analytique et les projections calculées avec le projecteur 2D proposé. LaFIGURE 4.8 montre les erreurs calculées entre les projections analytiques et celles obtenues avec le projecteur 2D proposé. On constate des valeurs très faibles de l'ordre de 10^{-22} (erreur quadratique) ce qui permet de valider la mise en œuvre du projecteur 2D.

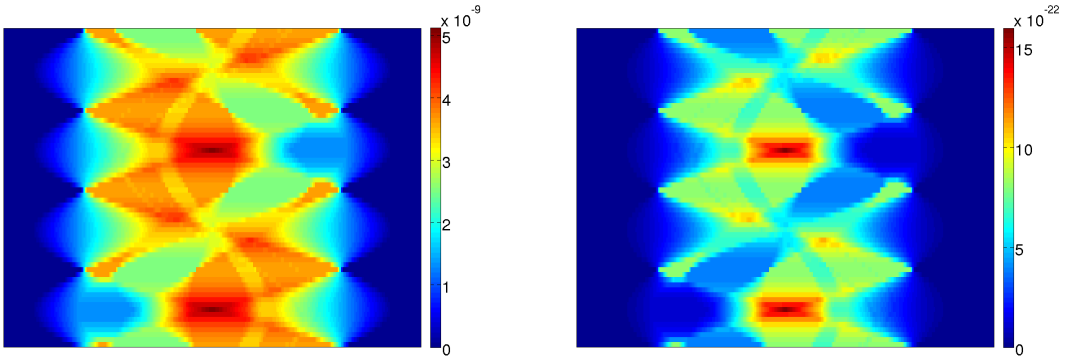
Paramètre	Valeur
distance source-objet	98mm
distance objet-détecteur	132mm
dimensions du détecteur	14.2mm $\times$ 14.2mm
taille du détecteur en pixels	256 $\times$ 256
nombre de projections	64

TABLE 4.4 – Configuration d'acquisition utilisée pour les calculs de projections.



(a) (b)

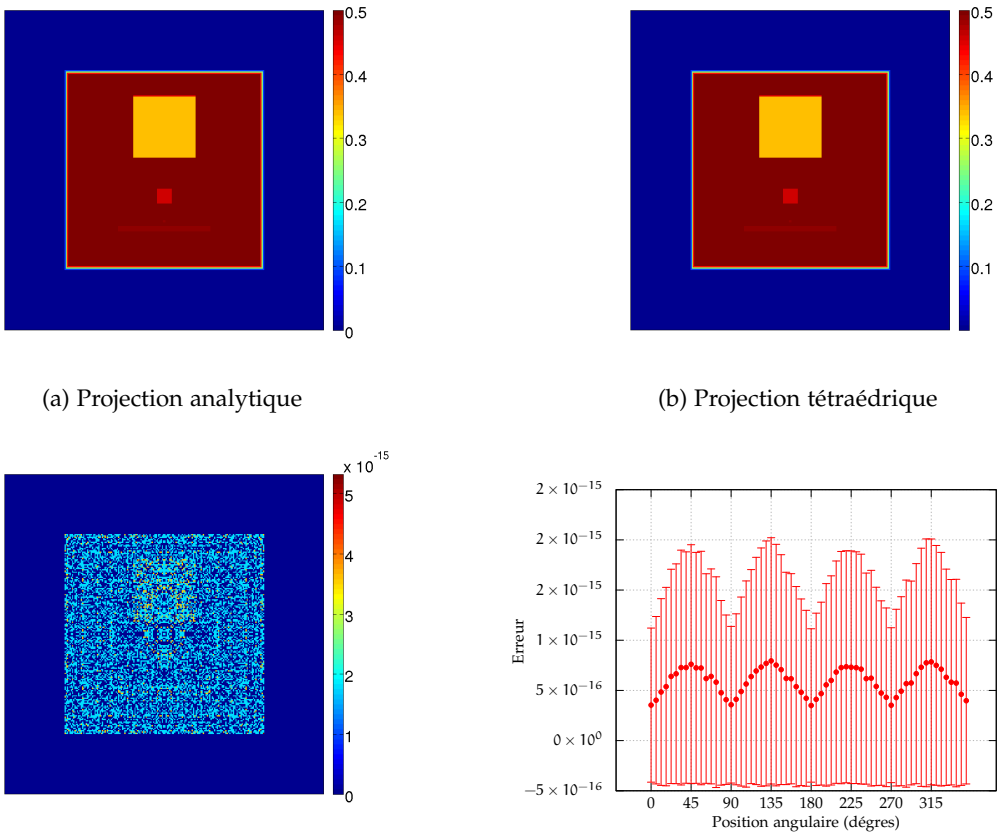
FIGURE 4.7 – Validation du projecteur en 2D - Sinogrammes de l'objet test obtenu de façon analytique (a), et avec le projecteur adapté à la représentation sur grille irrégulière (b).



(a) Erreur absolue (b) Erreur quadratique moyenne

FIGURE 4.8 – Erreurs entre les sinogrammes obtenus de façon analytique et avec le projecteur proposé.

La FIGURE 4.9 montre les projections de l'objet à l'angle de rotation $\theta = 0^\circ$ obtenues avec le projecteur analytique et avec le projecteur 3D proposé (noté Projection tétraédrique dans la figure par abus de langage). La FIGURE 4.9d illustre les résultats des erreurs calculées entre les projections analytique et tétraédrique (par abus de langage). De la même façon que dans le cas 2D, on constate des erreurs de l'ordre de 10^{-15}. Cette valeur négligeable permet de valider l'implantation du projecteur 3D proposé.



(a) Projection analytique

(b) Projection tétraédrique

(c) Erreur quadratique entre les projections (a) et (b)

(d) Erreur quadratique à chaque angle de projection

FIGURE 4.9 – Validation des projections en 3D.

4.4.3 Algorithmes de reconstruction

Pour effectuer la reconstruction sur un maillage donné, nous avons adapté et mis en œuvre deux algorithmes itératifs utilisés dans le cadre usuel de grilles régulières de pixels. Le premier algorithme est l'algorithme EM (*Expectation-Maximization*), et le second est un algorithme de minimisation du gradient conjugué. On note :

- K le nombre de triangles du maillage
- N le nombre de projections (égal au nombre de pixels détecteur multiplié par le nombre d'angles de vue)

– p les données (d'origine numérique ou expérimentale)

Algorithme EM

Dans le cadre d'une loi de Poisson (cf. Chapitre 1 SECTION 1.3.2), le schéma itératif de l'algortihme EM [34] qui maximise la vraisemblance s'exprime :

$$f_j^{n+1} = f_j^n . \frac{\sum_{i \in J_j} \exp\left(-\sum_{k \in I_i} h_{ik} f_k^n\right) . h_{ij}}{\sum_{i \in J_j} p_i h_{ij}} \tag{4.6}$$

On part d'une solution initiale f_0 non nulle et positive (correction multiplicative), par exemple une image de valeur constante en toutes les mailles, puis on opère selon l'implantation du schéma itératif mentionné par l'ALGORITHME 5 où I est le nombre d'itérations fixé par l'utilisateur pour éviter la divergence de l'algorithme (cf. Chapitre 1 SECTION 1.3.2).

Algorithme 5 *Expectation-Maximization* (EM).

```
1 : for i : 0 → I − 1 do
2 :   for k : 0 → K − 1 do
3 :     Initialisation : num = den = 0
4 :     for n : 0 → N − 1 do
5 :       h ← (ray[n], triangle[k])      // Calcule la longueur d'intersection
6 :       l_p ← (ray[n], M_k)            // Calcule la n^e projection
7 :       num ← num + h · exp(−l_p)
8 :       den ← den + h · exp(−p[n])
9 :     end for
10 :    if den ≠ 0 then
11 :      f[k] ← f[k] · num/den          // Met à jour la solution
12 :    end if
13 :  end for
14 : end for
```

Algorithme Gradient Conjugué

On utilise l'algorithme du gradient conjugué pour minimiser (par moindres carrés) la fonctionnelle (cf. Chapitre 1 SECTION 1.3.2) :

$$J(f) = \|p - Hf\|_2^2 \tag{4.7}$$

La routine du gradient conjugué peut être exprimée de la façon suivante :

1. Initialisation :

$$d_0 = r_0 = H^T\left(p - H\hat{f}_0\right) \quad \left(\hat{f}_0 \in \mathbb{R}^+\right) \tag{4.8a}$$

2. Routine itérative :

$$a_i = \frac{\langle r_i, r_i\rangle}{\langle d_i, H^T(Hd_i)\rangle} \tag{4.8b}$$

$$\hat{f}_{i+1} = \hat{f}_i + a_i d_i \tag{4.8c}$$

$$r_{i+1} = r_i - a_i H^T(Hd_i) \tag{4.8d}$$

$$b_i = \frac{\langle r_{i+1}, r_{i+1}\rangle}{\langle r_i, r_i\rangle} \tag{4.8e}$$

$$d_{i+1} = r_{i+1} + b_i d_i \tag{4.8f}$$

L'implantation de l'algorithme du gradient conjugué est décrit par l'ALGORITHME 6. On initialise le vecteur f recherché par f_0 (qui peut être une image constante), puis on itère le processus jusqu'à ce que l'erreur entre les données et l'estimation de f soit inférieure à $\epsilon << 1$, $\epsilon \in \mathbb{R}^+$ (convergence).

Algorithme 6 *Gradient Conjugué* (GC).

Initialisation : $f_0 \leftarrow 0$ // Initialisation de l'image
Initialisation : $L_{old} = b = a = 0$
$tmp_{p1} \leftarrow Projection(f_0)$ // Projection de l'image initiale
for $m : 0 \rightarrow M-1$ **do**
  $tmp_{p2}[m] = p[m] - tmp_{p1}[m]$ // Calcul de l'erreur entre la projection estimée et la projection réelle
  $L_{old} = L_{old} + tmp_{p2}[m]^2$
end for
$L_{tol} = L = L_{old}$ // Initialisation du critère à minimiser
$d = r_1 \leftarrow Rétroprojection(tmp_{p2})$ // Calcul du résidu par rétroprojection de l'erreur
while $(n = 1 \rightarrow N)$ && $(L_{tol} > \epsilon)$ **do**
  $tmp_{p1} \leftarrow Projection(d)$ // Projection du résidu
  $L = 0$
  for $m : 0 \rightarrow M-1$ **do**
    $tmp_{p2}[m] = p[m] - tmp_{p1}[m]$
    $L = L + tmp_{p2}[m]^2$ //Calcul du critère à minimiser
  end for
  $L_{tol} = 1 - \frac{L_{old}}{L}$
  $L_{old} = L$
  if $(L_{tol} < \epsilon)$ **then**
    return
  end if
  $ata \leftarrow Rétroprojection(tmp_{p2})$ // Rétroprojection $ata = H^t H d$
  $c_1 = c_2 = 0$
  for $k : 0 \rightarrow K-1$ **do**
    $c_1 = c_1 + r_1[k]^2$
    $c_1 = c_2 + d_1[k] \times ata[k]$
  end for
  $a = \frac{c_1}{c_2}$ // Calcul du pas de descente
  $c_1 = c_2 = 0$
  for $k : 0 \rightarrow K-1$ **do**
    $f[k] = f[k] + a \times d[k]$ // Mise à jour de la solution
    $r_2[k] = r_1[k] - a \times ata[k]$
    $c_1 = c_1 + (r_2[k] - r_1[k])^2$
    $c_2 = c_2 + r_1[k]^2$
  end for
  $b = \frac{c_1}{c_2}$
  for $k = 0 \rightarrow K-1$ **do**
    $d[k] = r_2[k] - b \times d[k]$ // Mise à jour de la direction du gradient
    $r_1[k] = r_2[k]$
  end for
end while

4.4.4 Performance des algorithmes de reconstruction

Dans cette partie nous analysons les performances des deux algorithmes de reconstruction implémentés et décris précédemment : EM et gradient conjugué. Afin d'évaluer les aspects en terme de qualité d'image reconstruite, vitesse de convergence et temps de calcul, les performances ont été évaluées selon les conditions suivantes :

- discrétisation par une grille de triangulaire régulière de l'objet
- même nombre d'itérations

L'objet test est formé par un carré, auquel on a associé une valeur constante égale à 1 à l'intérieur du carré et 0 à l'extérieur (voir FIGURE 4.10).

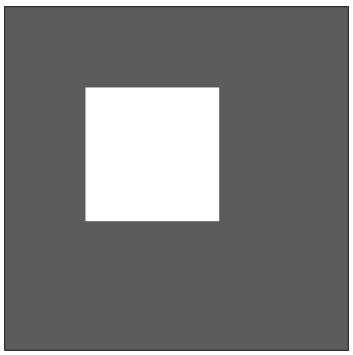
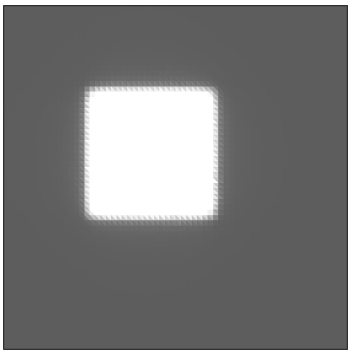


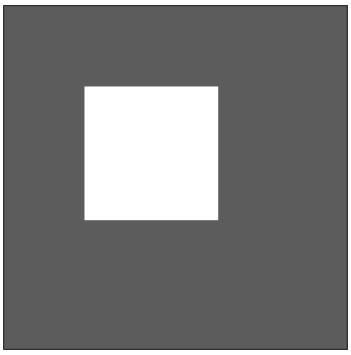
FIGURE 4.10 – Objet test : carré décentré.

L'objet a été reconstruit à partir de 50 itérations pour les deux algorithmes. À partir de l'image obtenue avec le gradient conjugué on atteint une erreur absolue de 10^{-4} par rapport à une erreur de 0.8 obtenue avec EM (sur les quatre coins du carré) (voir FIGURE 4.11). La FIGURE 4.12 illustre le profil horizontal et le profil vertical de l'objet reconstruit avec la méthode EM et le gradient conjugué.

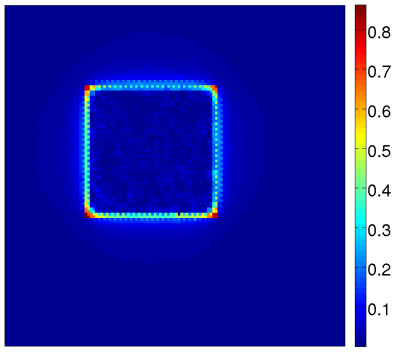
La FIGURE 4.13 montre la fonction de coût des deux algorithmes au fur et à mesure des itérations. On observe qu'après 30 itérations la fonction de coût de l'algorithme EM décroît moins rapidement au contraire de celle du gradient conjugué. De plus, le temps d'exécution pour 1 itération est égal à 220 secondes avec EM et 75 secondes avec le gradient conjugué. Ainsi, nous avons retenu l'algorithme du gradient conjugu comme algorithme de reconstruction par la suite.



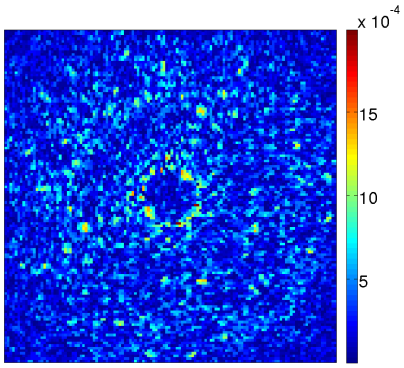
(a) Algorithme *EM*



(b) Algorithme gradient conjugué

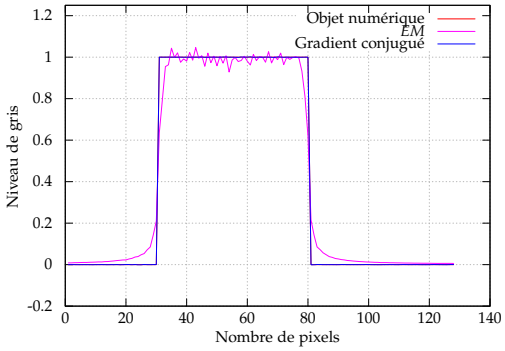


(c) Erreur absolue entre l'objet (algorithme *EM*) et l'objet test

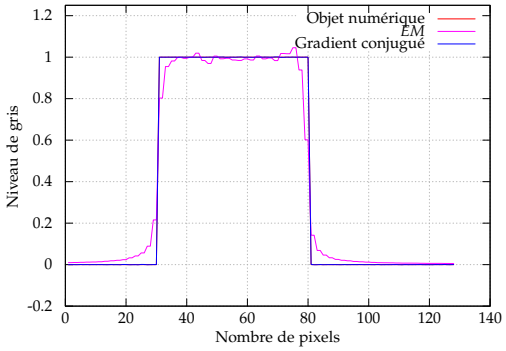


(d) Erreur absolue entre l'objet (algorithme gradient conjugué) et l'objet test

FIGURE 4.11 – Evaluation des performances des algorithmes de reconstruction adaptés aux grilles irrégulières, à partir d'une représentation de l'objet test sur une grille composée de 16384 mailles

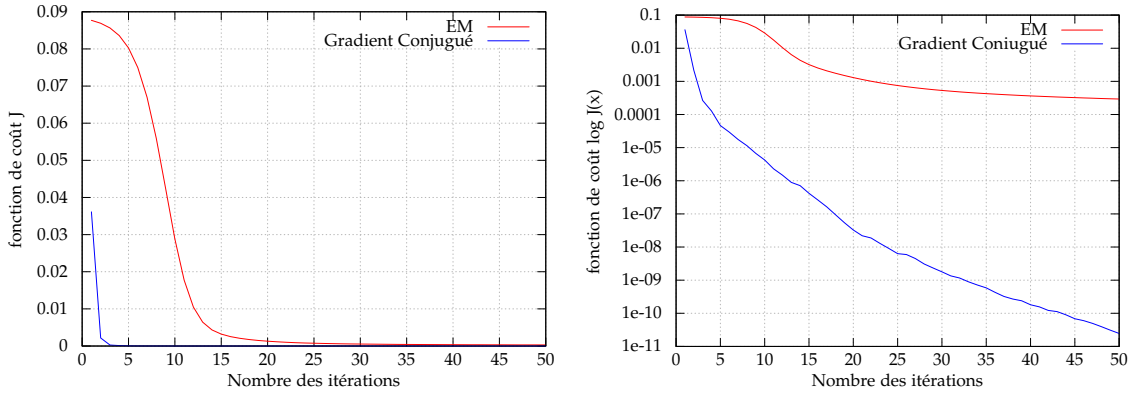


(a) Profil horizontal



(b) Profil vertical

FIGURE 4.12 – Profils vertical et horizontal des reconstructions EM et GC passent par le centre du carré.



(a) Échelle linéaire (b) Échelle logarithmique

FIGURE 4.13 – Comparaison des vitesses de convergence des deux algorithmes.

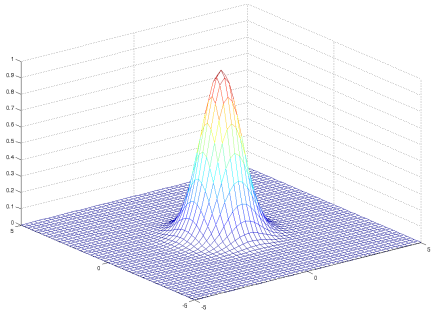
4.5 Segmentation

Dans cette section, nous décrivons l'implantation des méthodes de segmentation utilisées. Dans un premier temps, nous considérons le cas où l'objet étudié est composé d'un seul matériau (cas mono-matériau), puis dans un second, celui où l'objet est composé de plusieurs matériaux (cas multi-matériaux).

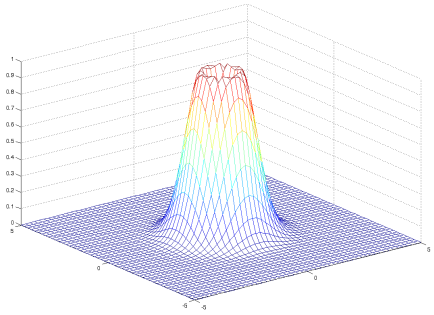
4.5.1 Passage d'une grille irrégulière à une grille régulière

Les méthodes de segmentation utilisées s'appuient sur une discrétisation de l'image basée sur une grille pixélisée. En effet, les méthodes s'appuient sur un calcul du gradient de l'image. Or les approches existantes [59] pour estimer un gradient d'une image représentée par un maillage triangulaire aboutissent à des résultats peu exploitables. La FIGURE 4.14 illustre un exemple d'une fonction gaussienne discrétisée sur une grille pixélisée (voir FIGURE 4.14a) et sur une grille irrégulière composée par des triangles (voir FIGURE 4.14c).

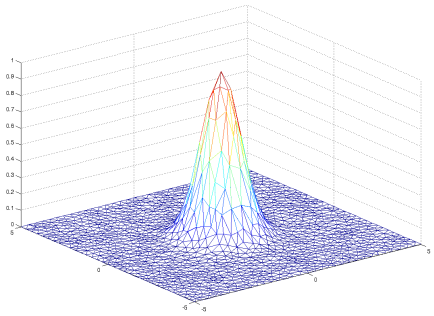
La FIGURE 4.14b montre le résultat du calcul du gradient de cette fonction sur une grille régulière de pixels, à comparer avec la FIGURE 4.14d qui elle représente le calcul du gradient de cette même fonction sur une grille irrégulière de triangles. On observe aisément le manque de douceur et de continuité dans le gradient calculé sur une grille irrégulière de triangles, en comparaison avec le gradient calculé sur une grille régulière de pixels.



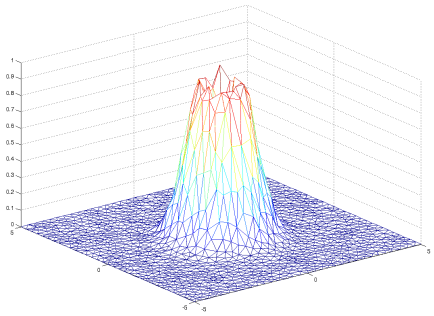
(a) Représentation par une grille pixélisée



(b) Représentation du gradient calculé sur une grille pixélisée



(c) Représentation sur une grille irrégulière de triangles



(d) Représentation du gradient calculé sur une grille irrégulière de triangles

FIGURE 4.14 – Exemple du calcul du gradient de la fonction $f(x, y) = \exp(-x^2 - y^2)$ (a) et (c) sur une grille pixélisée (b) et sur une grille irrégulière de triangles (d).

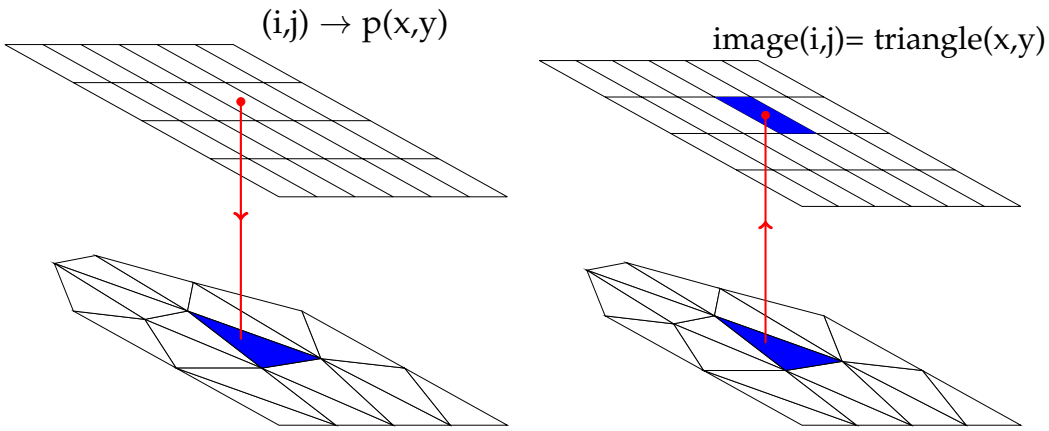
Le mauvais calcul du gradient engendrant une mauvaise segmentation, nous avons décidé à ce stade de passer à une grille régulière pour la segmentation. Ce choix est également fait dans le travail de [12], où une grille pixélisée est utilisée pour le calcul du gradient et du laplacien dans le cadre de méthodes par différences finies.

Le passage d'une grille irrégulière de triangles à une grille régulière de pixels s'effectue de la façon suivante (voir FIGURE 4.15) : pour chaque pixel de la grille régulière

- on calcule ses coordonnées barycentriques
- on localise le triangle associé
- on affecte la valeur du pixel à celle du triangle correspondant

Le passage inverse, par exemple, le passage de la grille pixélisée à la grille triangulaire, est dual (voir FIGURE 4.16) : pour chaque triangle

- on calcule les coordonnées barycentriques
- on localise le pixel associé
- on affecte la valeur du triangle à celle du pixel correspondant



(a) Localisation du triangle associé (b) Affectation de la valeur du pixel

FIGURE 4.15 – Passage d'une grille irrégulière de triangles à une grille régulière de pixels.

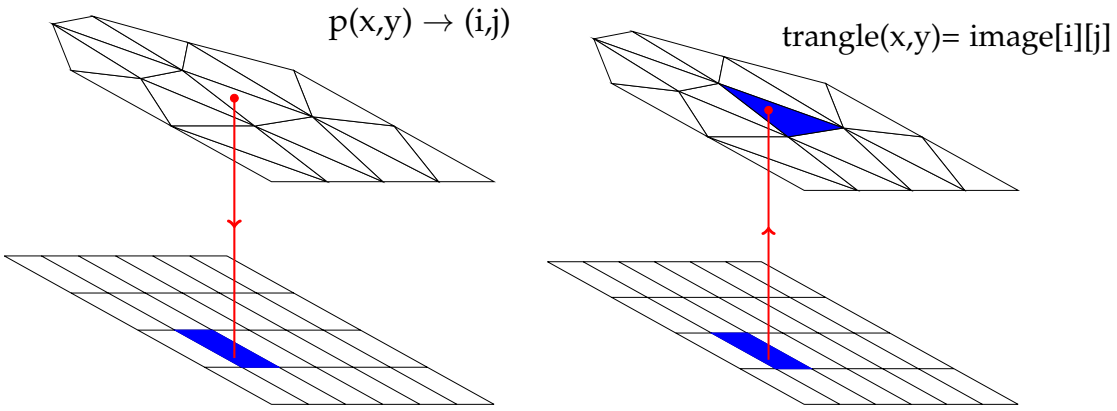
La question se pose quant à la résolution de la grille régulière de pixels. Étant donné un volume de reconstruction de taille $N_x \times N_y \times N_z$ pixels ($N_x \times N_y$ en 2D). Considérons que $N_x = N_y = N_z = N$, et que la longueur métrique du côté est L. La résolution δ de la grille pixélisée est :

$$\delta = \frac{L}{N} \tag{4.9}$$

Considérant le maillage irrégulier composé de triangles $\mathcal{M}$, on calcule la longueur moyenne des arêtes $\bar{l}_{\mathcal{M}}$:

$$\bar{l}_{\mathcal{M}} = \frac{1}{E_{\mathcal{M}}} \sum_i \sum_{i \neq j} \|v_i - v_j\| \quad v \in \mathcal{V}_{\mathcal{M}} \tag{4.10}$$

où $E_{\mathcal{M}}$ est le nombre total des arêtes du maillage $\mathcal{M}$, et $\mathcal{V}_{\mathcal{M}}$ l'ensemble des nœuds constituant le maillage $\mathcal{M}$.



(a) Localisation du pixel associé (b) Affectation de la valeur du triangle

FIGURE 4.16 – Passage d'une grille régulière de pixels à une grille irrégulière de triangles.

On définit alors le critère suivant pour calculer la résolution δ_c de la grille pixélisée considérée pour la conversion triangles $\rightarrow$ pixels :

$$\delta_c = \frac{\delta}{\bar{l}_{\mathcal{M}}} \cdot N \tag{4.11}$$

Afin, d'éviter que la résolution soit trop fine ou trop large, on contraint la valeur calculée de telle sorte qu'elle vérifie la condition $\frac{N}{2} \leqslant \delta_c \leqslant 2N$.

4.5.2 Cas mono-matériau

Segmentation par *snake*

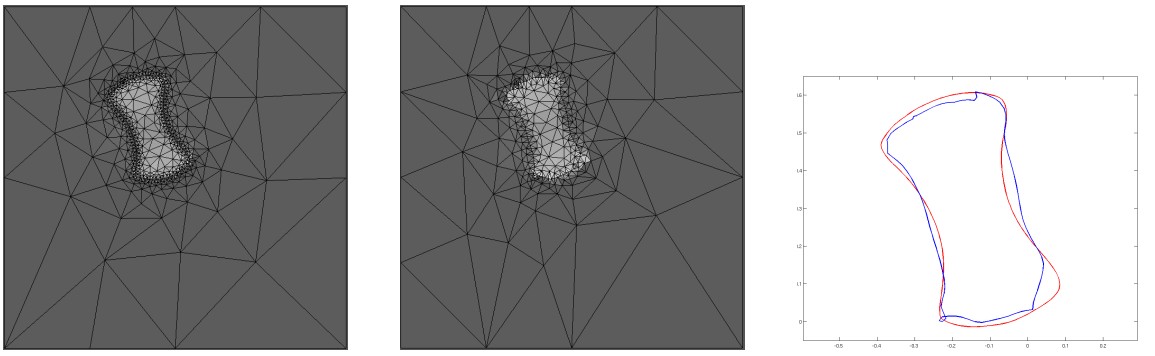
Dans cette partie, nous exploitons ATM pour la reconstruction 2D de données numériques puis expérimentales, en prenant pour algorithme de reconstruction tomographique l'algorithme EM et pour méthode de segmentation l'algorithme de *snake* décrit dans SECTION 2.2.2 du Chapitre 2. L'adaptation du maillage s'effectue à l'aide de la fonction *Mesh Generation* de la librairie CGAL [20], et s'appuie sur les points du contour issu de la segmentation explicite.

Les données expérimentales sont issues du système d'acquisition de micro-tomographie du laboratoire du CEA LIST (DISC/LITT Laboratoire Image Traitement et Tomographie). L'objet imagé est un foret de diamètre $\varnothing = 150\mu m$, dont nous proposons une reconstruction 2D de la coupe centrale.

La FIGURE 4.17 montre les résultats obtenus pour différentes valeurs des paramètres d'élasticité et de rigidité, resp. α et β. La FIGURE 4.18, quant à elle, montre les résultats obtenus pour différentes valeurs des paramètres de pondération de l'énergie externe, w_{image} et w_{term}. Sur ces figures, on constate aisément l'influence des valeurs de ces paramètres sur les résultats de la segmentation (perte de concavité, mauvaise définition des angles).

Or, la forme de l'objet n'étant pas connue *a priori*, ces valeurs doivent être fixées de façon empirique à travers différents tests. Par conséquent il n'est pas possible de garantir

la convergence vers la solution réelle.

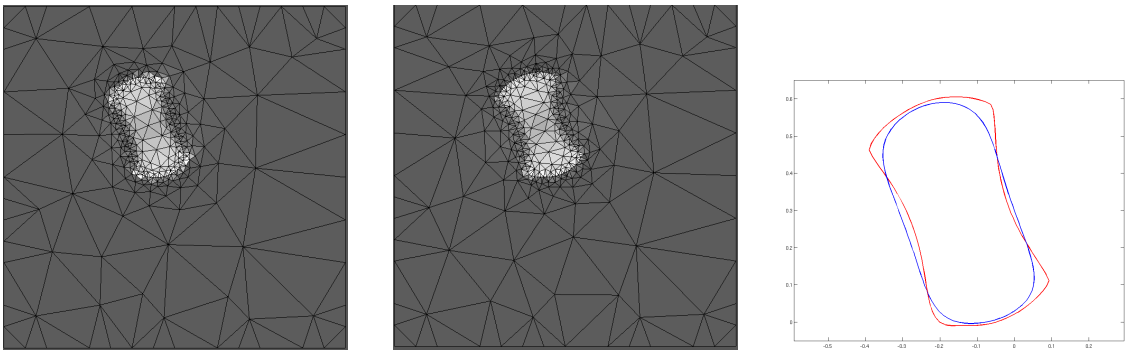


(a) Maillage final pour $\alpha = 0.2$ et $\beta = 0.2$

(b) Maillage final pour $\alpha = 0.5$ et $\beta = 0.5$

(c) Contours segmentés : cas (a) en bleu - cas (b) en rouge

FIGURE 4.17 – Influence des valeurs des paramètres d'élasticité α et de rigidité β sur la segmentation finale.



(a) Maillage final pour $w_{image} = 2.0$ et $w_{term} = 0.01$

(b) Maillage final pour $w_{image} = 3.0$ et $w_{term} = 0.04$

(c) Contours segmentés : cas (a) en rouge - cas (b) en bleu

FIGURE 4.18 – Influence des valeurs des paramètres de la force externe, w_{edge} et w_{term}, sur la segmentation finale.

Bien que la méthode de segmentation par *snake* ait l'avantage, par sa représentation explicite, de générer un maillage adapté au contenu de l'image directement à partir des points du contour, l'inconvénient majeur du choix des valeurs de ses nombreux paramètres nous incite à nous tourner vers la seconde méthode, celle des *levels set*.

Segmentation par *two-phase level set*

Dans le cas d'objets mono-matériau (voir FIGURE 4.19), seules deux phases distinctes sont présentes dans l'image (le fond et l'objet estimé) et on fixe $\alpha = \frac{\max\{\phi(x)\}}{2}$.

La mise en œuvre de la segmentation par la méthode *twophase level set* est faite suivante l'algorithme (voir ALGORITHME 3) qui a été décrit dans la SECTION 2.5 du Chapitre 2.

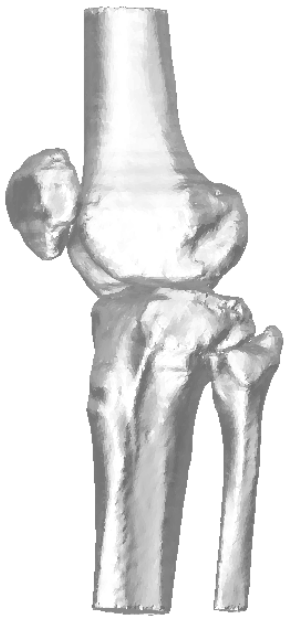


FIGURE 4.19 – Objet mono-matériau.

Dans le cas de cette représentation implicite du contour, il est nécessaire de mettre en œuvre une étape supplémentaire pour déterminer les points qui serviront à l'adaptation du maillage. Dans le prochain paragraphe, nous présentons l'utilisation des *levels set* dans le cas d'une segmentation d'un objet multi-matériaux et nous aborderons ensuite l'étape mise en œuvre pour placer les points à partir desquels un maillage adapté au contenu (à la segmentation implicite) est généré.

4.5.3 Cas multi-matériaux

Dans le cas d'un objet multi-matériaux (voir FIGURE 4.20), la segmentation des contours des différents matériaux est effectuée avec le même algorithme que précédemment (*two phase level set*) ALGORITHME 3, en utilisant plusieurs lignes de niveaux de la fonction *level set* u. Le résultat est la fusion des contours obtenus :

$$\Gamma = \bigcup_{i=1}^{r} \{x \mid \phi(x) \geqslant \alpha_i\} \; x \in \mathbb{R}^2, \; \alpha_i \in]0,1[\tag{4.12}$$

4.6 Génération du maillage adapté au contenu de l'image

Dans cette partie, nous abordons la méthode d'adaptation du maillage au contenu de l'image et les critères développés. Tout d'abord, nous rappelons ce qu'est la triangulation de Delaunay et en particulier la triangulation de Delaunay avec contraintes (en 2D), puis nous présentons les contraintes mises en place pour générer un maillage progressivement grossier des contours vers les bords du volume de reconstruction.

FIGURE 4.20 – Objet multi-matériaux.

4.6.1 Maillage 2D

Soit $\mathcal{P}$ l'ensemble de points de l'espace tels que $\mathcal{P} = \{p_1, \ldots, p_n\}$ $p_i \in \mathbb{R}^2$, $i = \{1, \ldots, n\}$ représente l'ensemble des contours segmentés Γ (Γ étant la représentation implicite des contours de l'objet obtenue à l'étape de segmentation).

Le maillage 2D que l'on cherche à générer est une triangulation de Delaunay contrainte dans laquelle les arêtes contraintes sont celles issues de la segmentation, c'est-à-dire les segments reliants les points respectifs de chaque ensemble de points décrivant les contours segmentés.

Pour cela nous utilisons la fonction *Mesh Generation* de la librairie CGAL [20] qui utilise une méthode d'arbre de Steiner pour insérer les autres points du maillage. Ces nouveaux points dits de Steiner [29] sont placés de telle sorte à réduire le coût de connexion entre les points par un problème d'optimisation combinatoire.

Le critère de forme (de qualité) des triangles pris en compte est :

$$B = \frac{r}{l} \tag{4.13}$$

où r est le rayon du cercle circonscrit au triangle considéré, et l la longueur de l'arête la plus courte du triangle. Il existe une relation entre B et l'angle le plus petit du triangle β_{min} :

$$B = \frac{1}{2 \cdot \sin \beta_{min}} \tag{4.14}$$

Et donc :

$$\beta_{min} = \arcsin \frac{1}{2B} \tag{4.15}$$

La convergence d'algorithme est assurée pour une valeur de $B \geqslant \sqrt{2}$, ce qui correspond à un angle minimal admissible $\beta_{min} = 20.7°$ [109].

Le critère de taille des triangles est contrôlé par le paramètre l_{max} qui définit la longueur maximale des arêtes d'un triangle. Le choix de l_{max} joue un rôle fondamental

dans la création du maillage. Ainsi, pour que les mailles soient générés de façon à respecter les contraintes et permettre un maillage plus grossier en s'éloignant des contours, la valeur de l_{max} doit satisfaire la condition suivante :

$$l_{max} \gg \frac{L_C}{K_C} \tag{4.16}$$

où L_C est la longueur du contour C, K_C le nombre des points décrit le contour C.

La longueur du contour est donnée par :

$$L_C = \sum_{i=0}^{K_C} \|p_{i+i} - p_i\| \quad p_i \in \mathcal{P} \tag{4.17}$$

Dans le cas où plusieurs contours sont décrits par l'ensemble $\mathcal{P}$, la valeur optimale l_{opt} est :

$$l_{opt} \gg \frac{L_{C_{min}}}{K_{C_{min}}} \tag{4.18}$$

où $L_{C_{min}}$ est la longueur du contour le plus court C_{min}, et $K_{C_{min}}$ le nombre de points qui décrivent ce plus petit contour C_{min}.

Afin d'obtenir un maillage adapté aux contours et isotropique, chaque contour C est représenté par K points qui doivent être répartis de façon homogène sur C (autrement dit les points doivent être équidistants les uns des autres). Dans le cas de plusieurs contours segmentés, Γ est un ensemble de courbes implicites, et l'ensemble des points $\mathcal{P}$ doit être réparti de façon homogène en décrirant convenablement chaque contour. Ainsi le nombre de points optimal K_{C_i} pour chaque contour C_i est donné par :

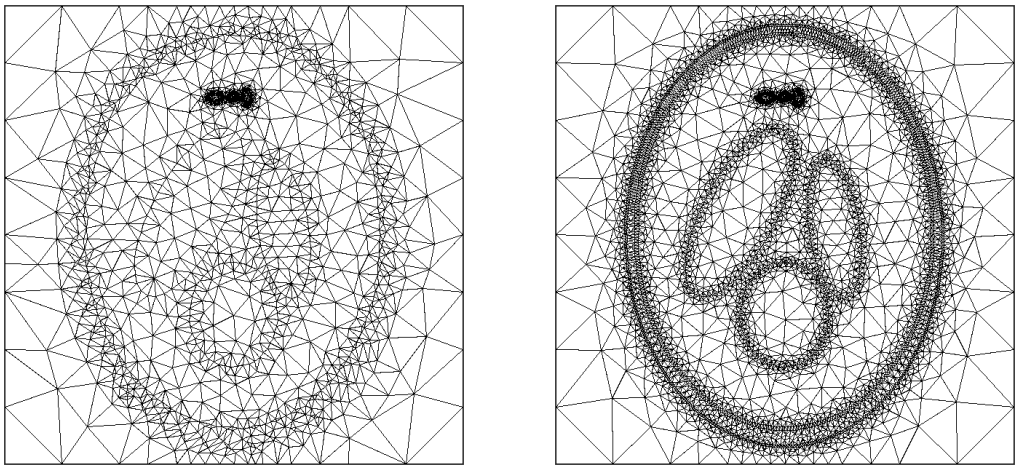
$$K_{C_i} = \frac{L_{C_i}}{L_{C_{min}}} \cdot k \quad i = \{1, \ldots, N_C\} \tag{4.19}$$

avec L_{C_i} la longueur du contour C_i, $L_{C_{min}}$ la longueur du contour plus court, et N_C le nombre de contours. La valeur k est une constante choisie par l'utilisateur.

La FIGURE 4.21 montre, pour k points fixés pour chaque contour (voir FIGURE 4.21a) et k points calculés avec le critère proposé pour chaque contour FIGURE 4.21b) dans le cas de la représentation du fantôme numérique de Shepp-Logan. La représentation en FIGURE 4.21b décrit plus finement les contours.

4.6.2 Étude du paramètre k

Une étude sur le paramètre k a été conduite pour analyser son influence sur le maillage au cours du processus de reconstruction ATM. L'objet considéré est le fantôme de Shepp-Logan. L'étape de segmentation est appliquée après dix itérations de l'algorithme de reconstruction (gradient conjugué) afin de trouver l'ensemble des contours implicites Γ. À partir des contours implicites, on extrait les contours explicites C_i à l'aide d'un fonction issue de la librairie Gnuplot [121], et l'ensemble des points nécessaires à la génération du maillage progressif est calculé selon la formule (4.19), avec k variant de 50 à 200.



(a) Nombre de points identique pour chaque contour

(b) Nombre de points calculés avec le critère proposé pour chaque contour

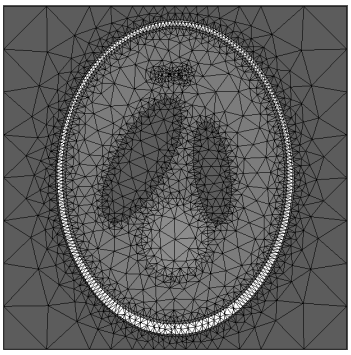
FIGURE 4.21 – Exemple de génération du maillage considérant ou non le calcul du nombre de points optimal pour la description de chaque contour du fantôme Shepp-Logan.

La FIGURE 4.22 montre les résultats obtenus pour $k = 50, 100, 150, 200$. Pour chaque valeur de k figure l'image en niveaux de gris (NG) (colonne de gauche) et le maillage correspondant (colonne de droite). On constate que la valeur de ce paramètre influence modérément la description des grosses structures mais de façon plus conséquente les structures de petite taille. Par ailleurs, on note qu'augmenter la valeur de k augmente le nombre de triangles constituant le maillage. Dans le cas d'un objet composé de structures de tailles équivalentes (longueurs des contours du même ordre de grandeur), on peut donc prendre $50 < k < 100$. Dans le cas d'un objet composé de structures de tailles variées, il est nécessaire de trouver un compromis entre une valeur de k suffisamment élevée pour permettre la description des différentes longueurs, mais pas trop élevée pour ne pas engendrer un nombre de mailles trop importante. On peut par exemple prendre $k = 150$ dans de tels cas.

4.6.3 Étude de l'influence du nombre d'itérations pour la première estimation de l'objet sur la segmentation

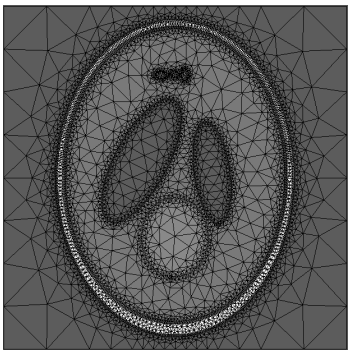
Dans cette section, nous évaluons le nombre d'itérations nécessaires (N_{ite}) à une première estimation de l'objet par l'algorithme de reconstruction tomographique suffisante pour accéder à une première segmentation correcte. Cette évaluation est menée car les premières itérations de reconstruction sur le maillage initial sont coûteuses en terme de temps de calcul car celui-ci comporte beaucoup de mailles. L'idée est donc d'optimiser le nombre d'itérations sur ce maillage pour générer rapidement un maillage adapté au contenu de l'image (celui-ci comportant moins de mailles). Ainsi nous diminuons le temps de reconstruction.

(a) Image NG pour $k = 50$



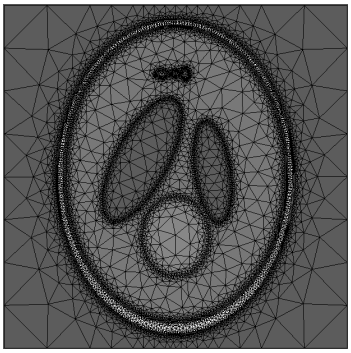
(b) Maillage correspondant (4296 triangles)

(c) Image NG pour $k = 100$



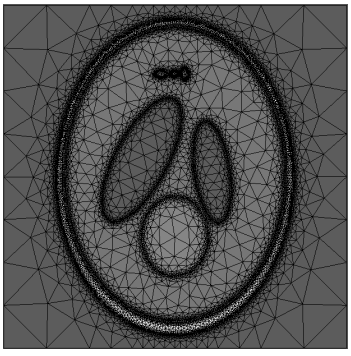
(d) Maillage correspondant (9522 triangles)

(e) Image NG pour $k = 150$



(f) Maillage correspondant (15528 triangles)

(g) Image NG pour $k = 150$

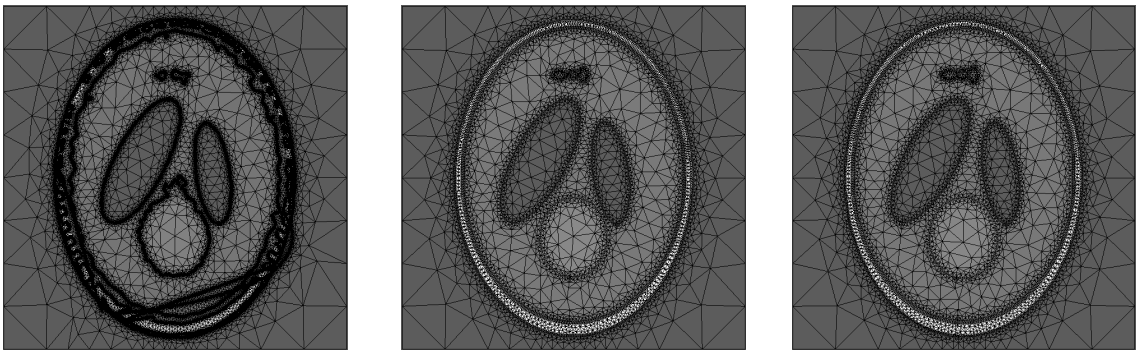


(h) Maillage avec 22199 triangles

FIGURE 4.22 – Influence du paramètre k sur l'adaptation du maillage au contenu de l'image.

Dans cette étude, l'algorithme de reconstruction est le gradient conjugué, et l'objet est le fantôme de Shepp-Logan, représentant un objet complexe composé de différents matériaux.

La FIGURE 4.23 montre les résultats obtenus après $N_{ite} = \{5, 10, 15\}$. On constate que pour un objet complexe il est nécessaire de fixer un nombre suffisant d'itérations pour garantir une première segmentation proche de la solution (ici $5 < N_{ite} < 10$). Au-delà, le temps de calcul s'allonge sans conséquences bénéfiques sur la segmentation.



(a) 5 itérations (b) 10 itérations (c) 15 itérations

FIGURE 4.23 – Influence du nombre d'itérations de reconstruction sur la segmentation puis la génération d'un maillage adapté.

Dans cette section nous avons décrit comment obtenir un maillage adaptatif à partir des contours issus de l'étape de segmentation. Un critère a été établi afin de créer un maillage prenant en compte la différence de taille des régions segmentées. L'étude sur le paramètre k nous a permis de définir une borne inférieure, et montre le compromis à faire dans le cas d'un grand rapport de longueur de contours. L'influence de la valeur du paramètre β_{min} n'a pas été étudié : nous l'avons fixée à la valeur par défaut préconisée par la fonction [20] qui évite d'obtenir des triangles allongés. Nous avons étudié l'influence du nombre d'itérations pour la première estimation de l'objet par l'algorithme de reconstruction (gradient conjugué) afin de garantir une segmentation convenable.

4.7 Résultats : reconstructions tomographiques 2D

Dans cette partie, nous présentons les résultats obtenus par ATM (méthode proposée) pour des reconstructions 2D de données numériques mono et multi matériaux (modèle statistique d'un genou et Shepp-Logan), puis de données expérimentales acquises sur un système d'acquisition de tomographie RX du laboratoire CEA LIST. Les reconstructions ont été effectuées à partir de données complètes puis incomplètes considérant les cas suivants :

– nombre de projections équidistantes sur 360° égal à 360, 90, 36 et 18 projections

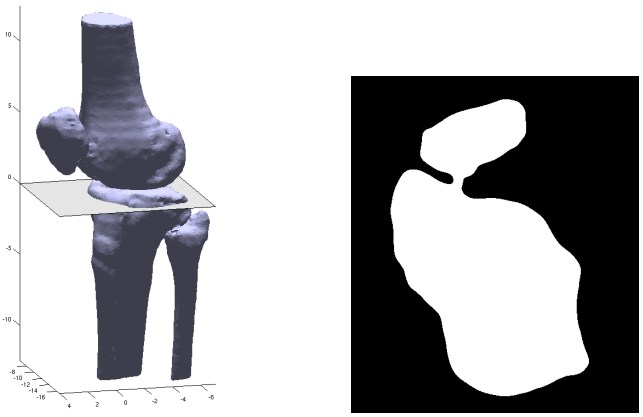
– nombre de projections équidistantes sur 180° égal à 180 projections, sur 140° égal à 140 projections et sur 100° égal à 100 projections

Les résultats ont été comparés à des reconstructions obtenues par une méthode analytique par rétroprojection filtrée (FBP) [30]. Le filtre utilisé est le filtre Hann avec une fréquence de coupure $f_c = 0.49$. Les reconstructions à partir d'un domaine angulaire restreint sont comparées à celles obtenues avec une méthode itérative OS-EM (*Order-Subset Expectation Maximization*) sur grille régulière pixélisée †.

Pour l'étape de segmentation, dans le cas de la reconstruction d'un objet mono-matériau le contour est obtenu en fixant la valeur $\alpha = \frac{\max\{\phi(x)\}}{2}$, où $\phi(x)$ représente la fonction implicite. Dans le cas d'un objet multi-matétiaux, les différentes valeurs de α sont calculées à partir de l'histogramme de la fonction implicite $\phi(x)$.

4.7.1 Données numériques : objet mono-matériau

Ici nous proposons la reconstruction d'une coupe 2D d'un modèle statistique du genou humain (voir FIGURE 4.24). Les projections ont été simulées avec le logiciel CIVA développé au CEA LIST [26]. La configuration d'acquisition est décrite dans le tableau (voir TABLE 4.5).



(a) (b)

FIGURE 4.24 – Modèle statistique du genou 3D (a) et coupe à reconstruire (b).

Le tableau (voir TABLE 4.6) résume les valeurs des paramètres utilisés dans ATM. L'initialisation du maillage est un maillage régulier composé de 100140 triangles. La FIGURE 4.25 montre les résultats obtenus. On observe sur cette figure que la méthode ATM est robuste à la diminution du nombre de projections utilisées.

Nous avons également reconstruit la coupe du genou à partir d'un domaine angulaire restreint de 180°, 140°et 100°. Les valeurs des paramètres utilisés pour la reconstruction ATM sont résumés dans le tableau (voir TABLE 4.7) . Les résultats sont montrés dans la FIGURE 4.26.

†. L'algorithme OSEM a été développé dans le laboratoire LITT du CEA LIST

Paramètre	Valeur
distance source-objet	1080 mm
distance objet-détecteur	80 mm
taille détecteur	151.5 mm $\times$ 151.5 mm
nombre de pixels détecteur	1024 $\times$ 1024

TABLE 4.5 – Configuration d'acquisition utilisée sous CIVA pour les mesures simulées de l'objet genou.

	Paramètres				
nb	nb iterations	nb itérations GC			nb points
projections	globales (N_{iter})	$N_{iter} = 1$	$N_{iter} = 2$	$N_{iter} = 3$	initiaux
360	2	10	140	-	200
90	2	10	140	-	200
36	3	15	75	100	200
18	3	40	100	100	200

TABLE 4.6 – Paramètres ATM pour la reconstruction du genou (domaine angulaire complet).

	Paramètres				
nb	nb itérations	nb itérations GC			nb points
projections	globales (N_{iter})	$N_{iter} = 1$	$N_{iter} = 2$	$N_{iter} = 3$	initiaux
180	3	15	75	100	200
140	3	15	75	100	200
100	3	20	80	100	200

TABLE 4.7 – Paramètres de reconstruction 2D ATM des données numériques du genou (domaine angulaire restreint).

La FIGURE 4.27 illustre le contour du genou obtenu à partir de différentes reconstructions.

Dans le cas d'un domaine angulaire complet, les contours obtenus avec 360, 90 et 36 projections se superposent. Sur le contour obtenu à partir de 18 projections on perd les informations de la petite zone concave. Dans le deuxième cas, le contour commence à se dégrader à partir de 140 projections.

4.7.2 Données expérimentales : objet mono-matériau

Dans un second temps, la méthode ATM a été appliquée pour la reconstruction de données expérimentales d'un foret. La configuration d'acquisition est résumée dans le tableau (TABLE 4.8). Le rapport signal sur bruit (ou RSB) des données est de l'ordre de 20dB.

Les paramètres utilisés pour la reconstruction ATM sont résumés dans le tableau (TABLE 4.9). Le maillage initial est constitué de 55241 triangles.

Paramètre	Valeur
distance source-objet	22.5 mm
distance objet-détecteur	127.5 mm
dimensions du détecteur	14 mm $\times$ 14 mm
nombre de pixsels détecteur	256 $\times$ 256

TABLE 4.8 – Configuration d'acquisition tomographique utilisée pour le foret.

	Paramètres				
nb	nb itérations	nb itérations GC			nb points
projections	globales (N_{iter})	$N_{iter} = 1$	$N_{iter} = 2$	$N_{iter} = 3$	initiaux
360	2	10	100	-	150
90	2	10	100	-	150
36	2	20	100	-	150
18	3	40	40	100	150

TABLE 4.9 – Paramètres ATM pour la reconstruction du foret (domaine angulaire complet).

Les reconstructions obtenues dans le cas d'un domaine complet par la méthode ATM sont illustrées en FIGURE 4.28.

Nous avons également, étudié le comportement de la méthode proposée dans le cas d'un domaine angulaire restreint. Les reconstructions sont illustrées en FIGURE 4.29.

nb projections	Paramètres				
	nb itérations globales (N_{iter})	nb itérations GC			nb points initiaux
		$N_{iter} = 1$	$N_{iter} = 2$	$N_{iter} = 3$	
180	2	20	100	-	150
140	2	20	100	-	150
100	2	20	100	-	150

TABLE 4.10 – Paramètres ATM pour la reconstruction du foret (domaine angulaire restreint).

La FIGURE 4.30 illustre le contour du foret obtenu à partir de différentes reconstructions. Dans le cas d'un domaine angulaire complet les contours obtenus avec 360, 90 et 36 projections se superposent. Le contour obtenu à partir de 18 projections est moins fidèle dans les zones concaves. Dans le deuxième cas, le contour commence à se dégrader à partir de 140 projections.

4.7.3 Données numériques : objet multi-matériaux

Nous avons validé la méthode proposée dans le cas d'un objet composé de plusieurs matériaux. Nous avons pris en compte le fantôme numérique Shepp-Logan, composé de différentes régions décrites par des ellipses. Le fantôme Shepp-Logan a été reconstruit à partir des données de projection calculées de façon analytique, dont la configuration tomographique d'acquisition est résumée dans le tableau (TABLE 4.9).

Paramètre	Valeur
distance source objet	98 mm
distance objet détecter	132 mm
taille détecteur	14.5 mm $\times$ 14.2 mm
résolution détecteur en pixels	256 $\times$ 256

TABLE 4.11 – Configuration tomographique utilisée pour le fantôme numérique Shepp-Logan.

Les reconstructions du fantôme Shepp-Logan obtenues par la méthode FBP et ATM sont illustrées en FIGURE 4.31.

La FIGURE 4.32 illustre les contours obtenus par la méthode ATM superposés au fantôme numérique Shepp-Logan.

4.7.4 Discussion

Les résultats montrent, dans le cas d'un nombre suffisant de projections, que les reconstructions obtenues par la méthode proposée représentent des images de reconstruction selon le sens «classique», c'est-à-dire une cartographie de coefficients d'absorption représentés sur des niveaux de gris. De plus, une segmentation de l'objet est obtenue dont la représentation est caractérisée par un maillage adaptatif.

Cependant, dans le cas d'une reconstruction à partir d'un nombre faible de projections, plusieurs itérations sont nécessaires afin d'obtenir une première segmentation correcte de l'objet.

Nous avons constaté une importante réduction du temps de calcul pour une itération d'algorithme de reconstruction (gradient conjugué). En effet lors des premières itérations sur un maillage initial de 100140 triangles le temps est 30 minutes par itération, qui se réduit à 30 secondes par itération après l'étape de segmentation, après laquelle le maillage est constitué de 1135 triangles. Ces valeurs concernent la reconstruction du genou à partir d'un nombre de projections fixé à 360.

Les images reconstruites par la méthode ATM ont la caractéristique d'occuper un faible espace de stockage en mémoire. En effet, la représentation par mailles adaptatives permet de représenter les zones homogènes par des triangles de grande taille donc peu

de coefficients. Par exemple, si on considère les images reconstruites des trois objets (genou, foret, Shepp-Logan) dans le cas d'une reconstruction à partir de 360 projections, on observe une réduction de la taille de données à stocker jusqu'à un facteur de 150 (voir TABLE 4.12).

Objet	Représentation de l'image	Type de données	Taille du fichier	Gain
Genou	1024×1024 pixels	binaire (64 bits)	8 Mo	×150
	1135 triangles	ASCII	52 Ko	
Foret	256×256 pixels	binaire (64 bits)	524 Ko	×11
	920 triangles	ASCII	44 Ko	
Shepp-Logan	256×256 pixels	binaire (64 bits)	524 Ko	×1.1
	9522 triangles	ASCII	472 Ko	

TABLE 4.12 – Taille de stockage des images reconstruites sur une une grille régulière pixélisée et sur une grille irrégulière triangulaire. Les résultats sont issus des reconstructions à partir de 360 projections.

4.8 Conclusions

Après avoir décrit les quelques méthodes existantes de reconstruction proposant une segmentation simultanées (TEP), nous avons présenté notre méthode de reconstruction en tomographie par rayons X (ATM) basée sur une représentation tétraédrique à travers ses principales étapes : initialisation, reconstruction, segmentation et adaptation du maillage au contenu de l'image.

Nous avons présenté la structure de données utilisée pour représenter un maillage en 2D et 3D. Ainsi que les choix qui ont été fait pour représenter une image en utilisant une discrétisation non régulière.

La méthode de reconstruction a été décrite dans son ensemble : validation de l'opérateur de projection et mise en œuvre des algorithmes de reconstruction.

Nous avons présenté l'étude qui nous a permis de discuter des inconvénients du contour actif, *snake*, par rapport à la méthode d'évolution d'une courbe par la méthode de *level set*. La méthode de segmentation par modèle de contour actif paramétrique *snake* souffre d'un problème d'initialisation du contour initial et d'un réglage minutieux des paramètres.

La méthode de segmentation adoptée fait le lien entre un modèle basé sur les régions et celui de contour actif basé sur une formulation par *level set*. Dans cette formulation, le contour n'est plus défini par la fonction de *level set* à niveau zéro, mais par une fonction implicite dont les valeurs sont comprises entre $]0, 1[$.

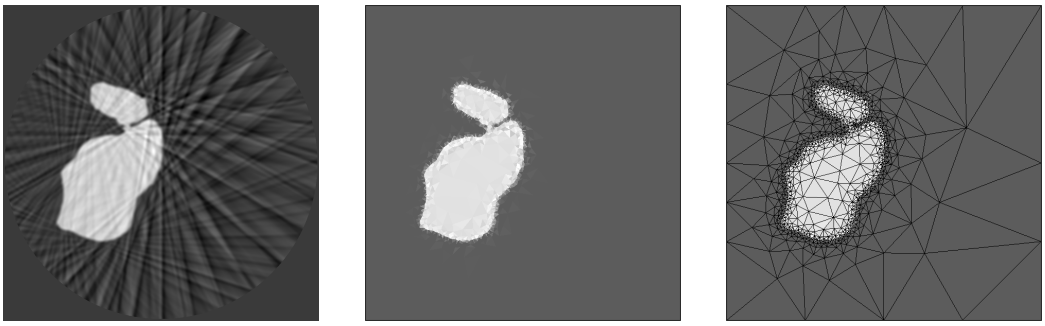
Afin d'obtenir une représentation du maillage cohérente avec à la taille des objets,

un critère de longueur de contour a été défini. Celui-ci permet de représenter un objet par un nombre de points proportionnel au rapport entre sa longueur et celle du contour plus court. Une étude a été faite sur l'influence du nombre d'itérations de l'algorithme de reconstruction avant de poursuivre avec l'étape de segmentation dont dépend l'adaptation du maillage. Nous avons vu que 10 itérations de reconstruction sont suffisantes pour obtenir une bonne segmentation. Mais lorsqu'on reconstruit à partir d'un nombre faible de projections, par exemple 36, un nombre d'itérations plus élevé est nécessaire.

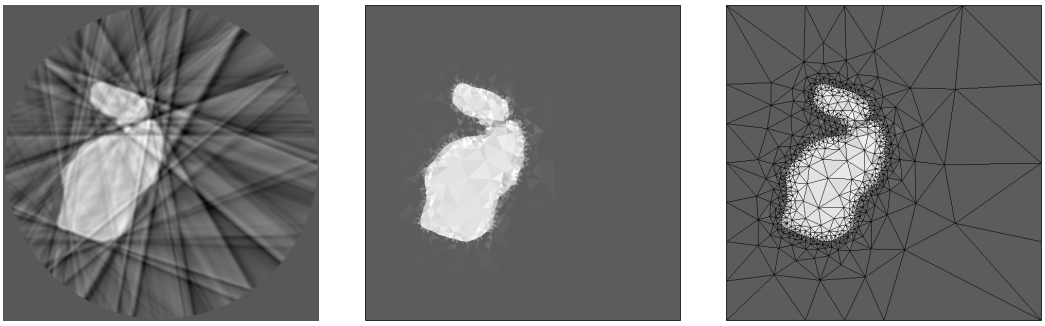
Enfin les résultats de reconstruction ont été présentés. Trois cas d'études : un objet numérique défini par un seul matériau, un objet réel et un fantôme numérique composé de plusieurs matériaux. Une étude sur le nombre de projections a été faite pour évaluer la robustesse de la méthode proposée en considérant un objet mono-matériau. Dans les deux cas on arrive à reconstruire et à segmenter correctement les objets jusqu'à 36 projections équidistantes sur 360°. Lorsqu'on reconstruit sur un domaine angulaire restreint ($\pm$ 70° et $\pm$ 50°) les images obtenues sont affectées par des artefacts dus aux projections manquantes. La minimisation d'un critère avec la pris en compte d'un terme de régularisation nous permettrait d'obtenir une reconstruction avec moins d'artefacts.

(a) Reconstruction FBP (b) Reconstruction ATM (c) Maillage ATM (1135 triangles)

(d) Reconstruction FBP (e) Reconstruction ATM (f) Maillage ATM (1141 triangles)



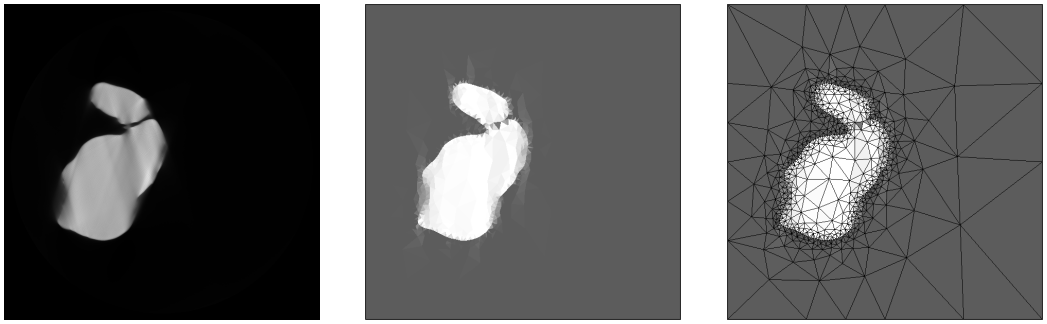
(g) Reconstruction FBP (h) Reconstruction ATM (i) Maillage ATM (1129 triangles)



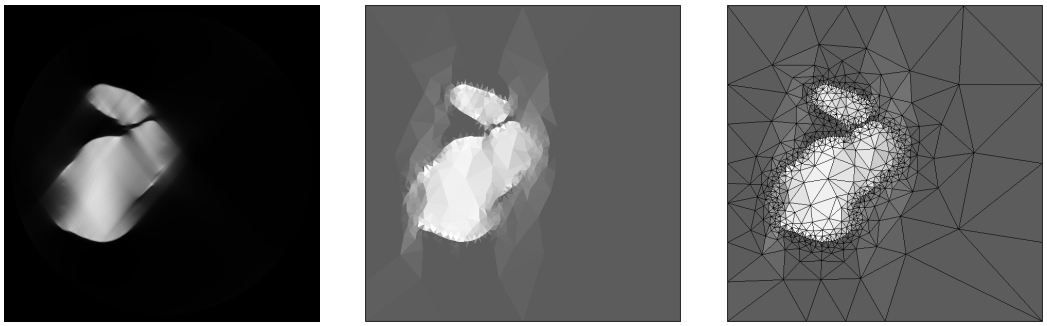
(j) Reconstruction FBP (k) Reconstruction ATM (l) Maillage ATM (1178 triangles)

FIGURE 4.25 – Reconstruction à partir de 360, 90, 36 et 18 projections équidistribuées sur 360 degrés (de haut en bas) - Colonne de gauche : reconstruction FBP (images 1024×1024 pixels), colonne centrale : reconstructions ATM avec l'algorithme GC (image NG), colonne de droite : maillages ATM correspondants.

(a) Reconstruction OSEM (b) Reconstruction ATM (c) Maillage ATM (1255 triangles)

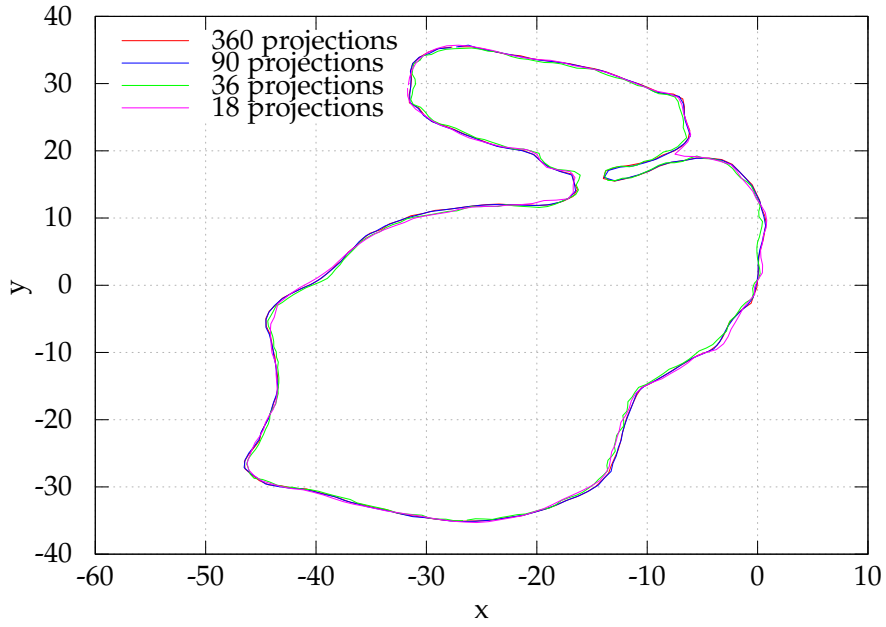


(d) Reconstruction OSEM (e) Reconstruction ATM (f) Maillage ATM (1198 triangles)

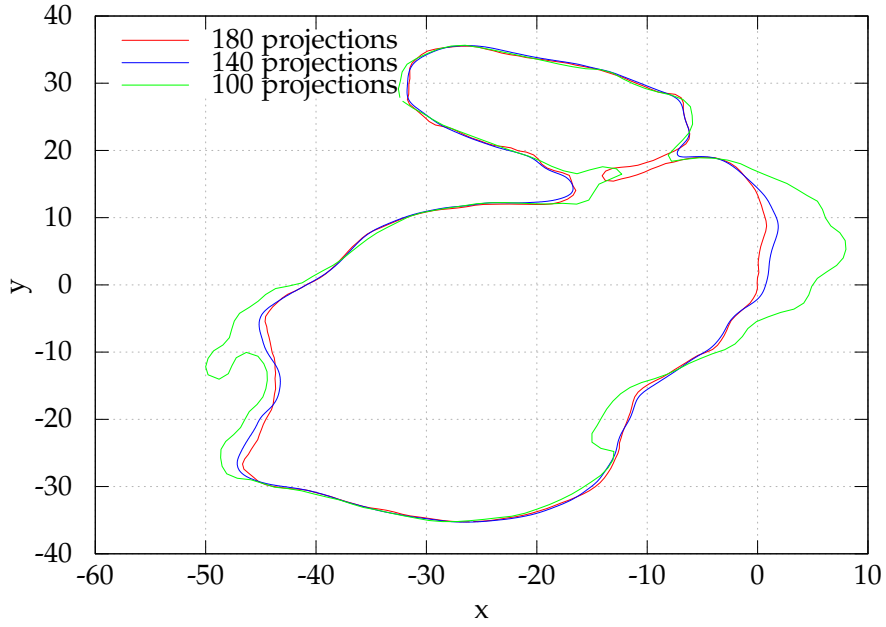


(g) Reconstruction OSEM (h) Reconstruction ATM (i) Maillage ATM (1129 triangles)

FIGURE 4.26 – Reconstructions à partir de 180, 140 et 100 projections sur un domaine angulaire restreint : $\pm$ 90°, $\pm$ 70°et $\pm$ 50°(de haut en bas) - Colonne de gauche : reconstructions OSEM 1024×1024 pixels, colonne centrale : reconstructions ATM avec l'algorithme GC (image NG), et colonne de gauche : maillages ATM correspondants.

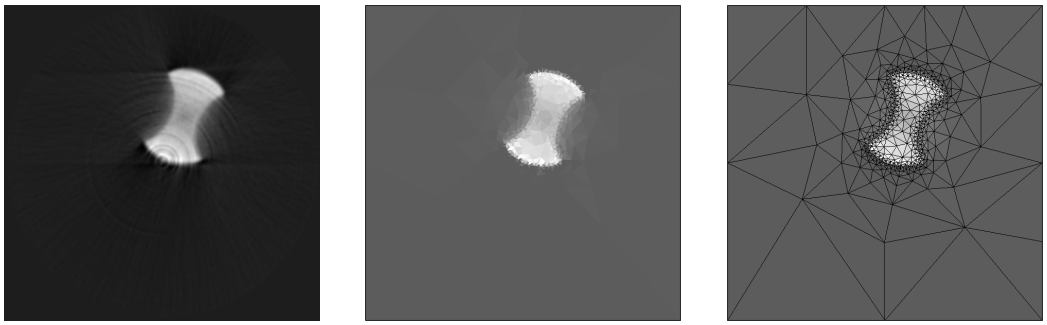


(a) Domaine angulaire complet

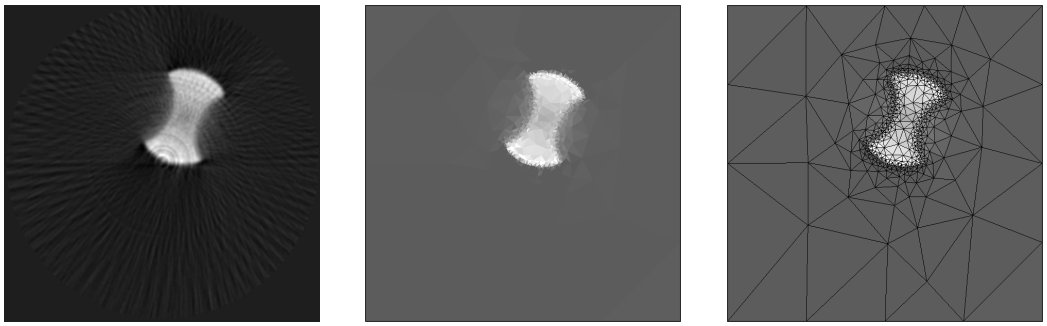


(b) Domaine angulaire réduit

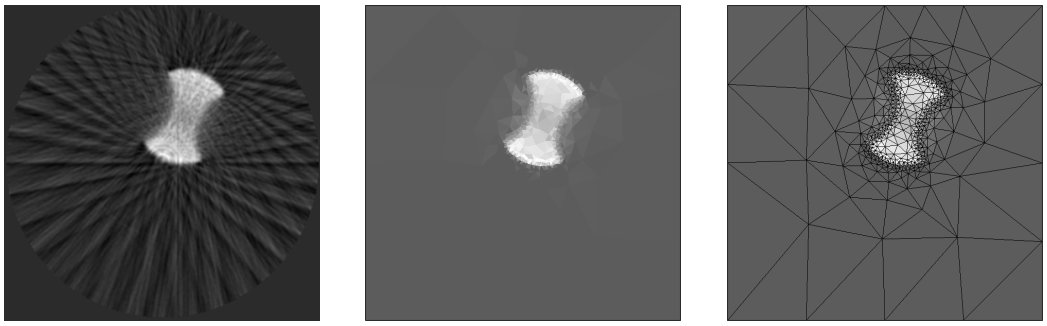
FIGURE 4.27 – Contour du genou final obtenu avec la méthode ATM : à partir de 360, 90, 36 et 18 projections équidistribuées sur 360 degrés (a) et à partir de 180, 140 et 100 projections sur un domaine angulaire restreint (b).



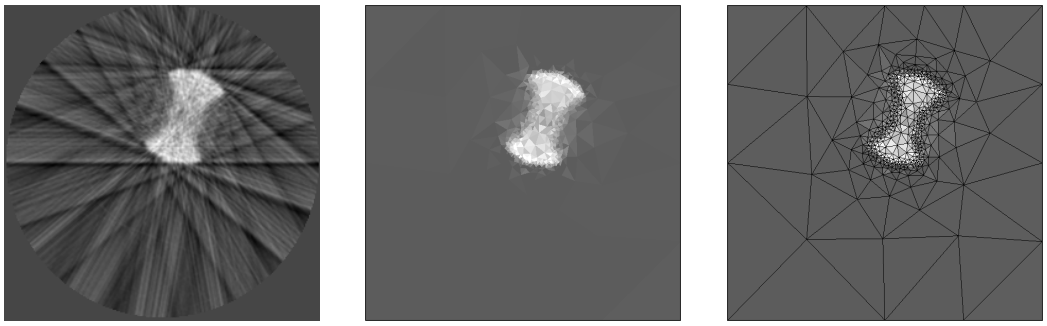
(a) Reconstruction FBP pixels (b) Reconstruction ATM (c) Maillage ATM (920 triangles)



(d) Reconstruction FBP pixels (e) Reconstruction ATM (f) Maillage ATM (946 triangle)

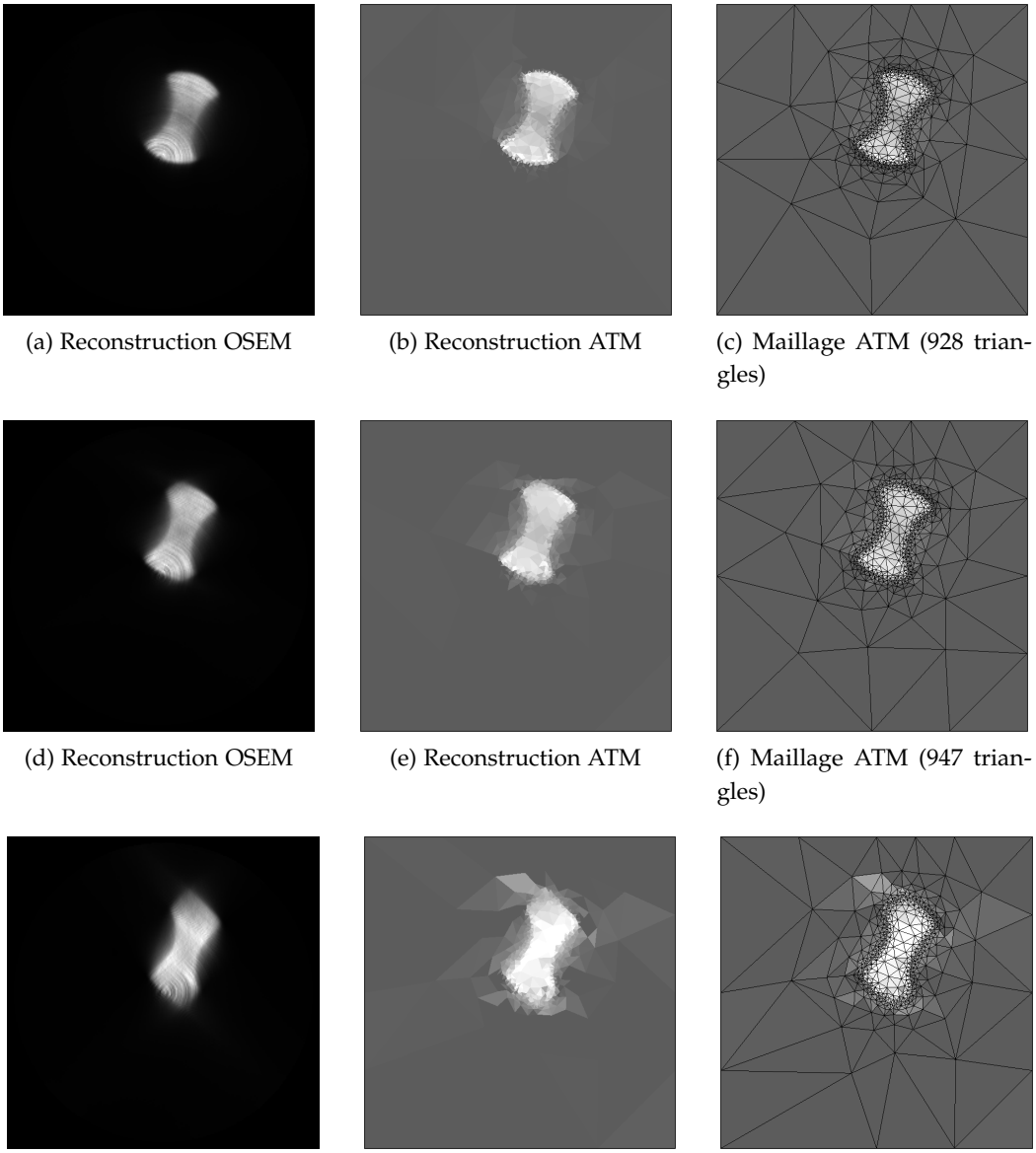


(g) Reconstruction FBP pixels (h) Reconstruction ATM (i) Maillage ATM (940 triangle)



(j) Reconstruction FBP pixels (k) Reconstruction ATM (l) Maillage ATM (941 triangle)

FIGURE 4.28 – Reconstructions à partir de 360, 90, 36 et 18 projections équidistribuée sur 360 degrés (de haut en bas) - Colonne de gauche : reconstruction FBP (images 256×256 pixels), colonne centrale : reconstructions ATM avec l'algorithme GC (image NG), colonne de droite : maillages ATM correspondants.

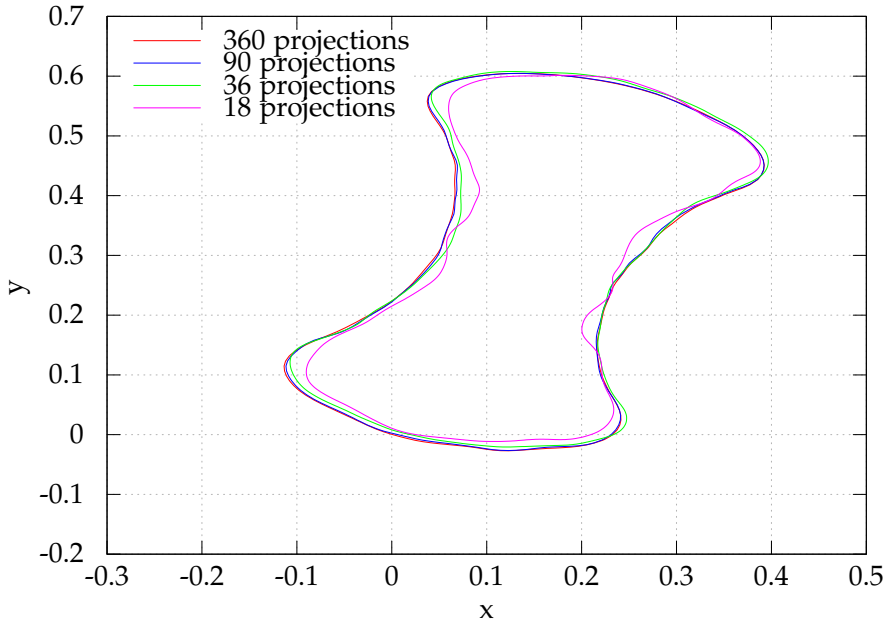


(a) Reconstruction OSEM (b) Reconstruction ATM (c) Maillage ATM (928 triangles)

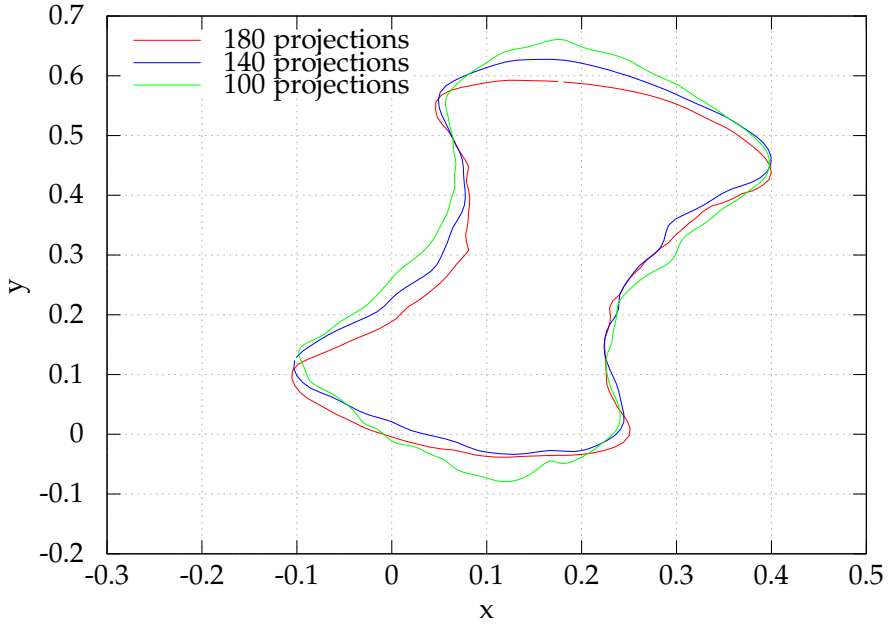
(d) Reconstruction OSEM (e) Reconstruction ATM (f) Maillage ATM (947 triangles)

(g) Reconstruction OSEM (h) Reconstruction ATM (i) Maillage ATM (934 triangles)

FIGURE 4.29 – Reconstructions à partir de 180, 140 et 100 projections sur un domaine angulaire restreint : $\pm$ 90˚, $\pm$ 70˚ et $\pm$ 50˚(de haut en bas) - Colonne de gauche : reconstructions OSEM 256×256 pixels, colonne centrale : reconstructions ATM (image NG),colonne de droite : maillages ATM correspondants.

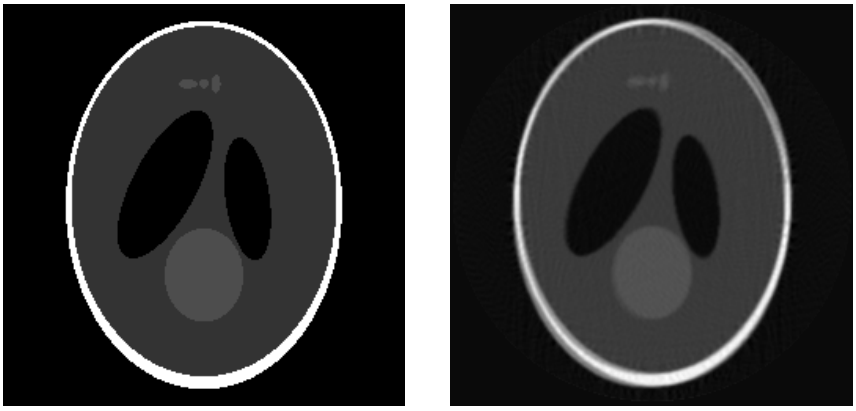


(a) Domaine angulaire complet

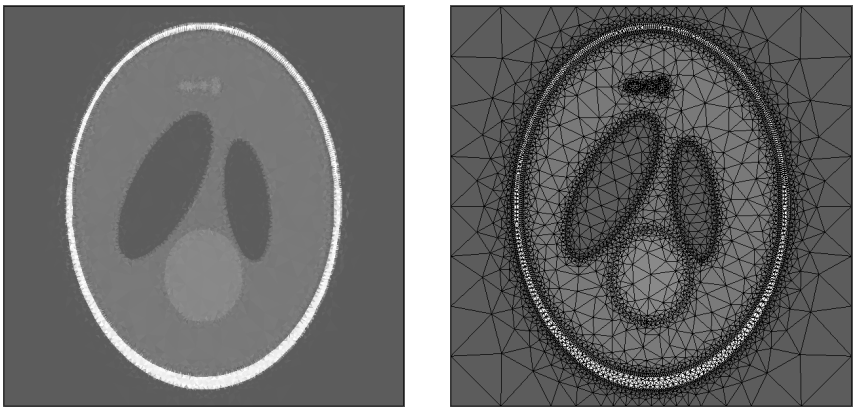


(b) Domaine angulaire réduite

FIGURE 4.30 – Contour du foret final obtenu avec la méthode ATM : à partir de 360, 90, 36 et 18 projections équidistribuées sur 360 degrés (a) et à partir de 180, 140 et 100 projections sur un domaine angulaire restreint (b).



(a) Fantôme numérique (b) Reconstruction FBP



(c) Reconstruction ATM (d) Maillage ATM (9522 triangle)

FIGURE 4.31 – Reconstruction du fantôme Shepp-Logan (a) : reconstruction FBP 256×256 (b), reconstruction ATM (c) maillages ATM correspondants (d).

FIGURE 4.32 – Contours obtenus par la méthode ATM superposés au fantôme Shepp-Logan.

Chapitre 5

Mise en œuvre sur carte graphique

Dans ce chapitre nous présentons les aspects de parallélisation sur carte graphique. Tout d'abord, nous exposons la faisabilité en 2D, cas pour lequel nous obtenons un faible taux d'accélération. Ensuite, nous détaillons plusieurs méthodes en 3D utilisées dans le rendu de volumes tétraédriques, développées par Toczek [116], et nous présentons les performances obtenues dans le cas du rendu tétraédrique en 3D sur trois scènes.

À partir de ces expériences, nous avons sélectionné une solution 3D pour la projection. Nous présentons la validation de cet opérateur de projection, avec les performances obtenues en changeant soit le nombre de tétraèdres, soit la taille (en pixel) du détecteur, qui représente aussi la taille de la projection. En outre, une comparaison de performances entre flottants en simple et double précision est traitée. Pour terminer cette partie, nous considérons la rétroprojection en 3D.

5.1 Opérateur de projection en 2D

La mise en œuvre du projecteur 2D, en utilisant la structure des données présentée dans le Chapitre 4 a été faite en associant un *thread* à chaque pixel du détecteur.

Le facteur d'accélération obtenu n'est pas satisfaisant. En effet, il est inférieur à un. L'explication de cette perte est liée aux structures de données, triangle/edge, utilisées pour l'implantation de la méthode de lancer de rayons.

Il faut noter que l'implémentation du lancer de rayons avec les structures triangle/edge nous a permis d'accélérer le projecteur d'un facteur $\times 35$, par rapport à une mise en œuvre classique.

Nous avons effectué un *profiling* (caractérisation) du code avec l'outil *Visual Profiler*, proposé par Nvidia [90], ce qui nous a permis d'identifier les facteurs limitant, parmi lesquels un taux d'occupation faible (rapport entre le nombre des *threads* exécutés et le nombre maximal supporté par l'architecture) et une bande passante de transfert mémoire réduite.

Ainsi nous avons apporté des modifications au code afin de réduire le nombre de registres et le pourcentage des branches divergentes.

Malgré une légère amélioration des performances, le facteur d'accélération reste faible, $\times 1.01$. Le goulot d'étranglement est dû aux structures de données utilisées, triangle et edge, qui ne sont pas adaptées à l'architecture GPU.

En effet, elles ne sont pas homogènes, c'est-à-dire qu'elles contiennent soit des types flottant double, 64 bits, soit des types entiers 32 bits. C'est une des raisons qui limite la bande passante des transferts mémoire.

Étant donné que la taille des données mises en jeu en 2D peut être traitée en utilisant un CPU, nous n'avons pas amélioré les structures de données. Une première étape pour améliorer les structures de données peut être de passer d'un vecteur de structures triangle/edge à une structure de vecteurs afin de régulariser les accès mémoire grâce à l'accès des différents *threads* simultanément aux mêmes champs de la structure et donc à des adresses mémoire contiguës.

5.2 Traversée du volume composé de tétraèdres

Dans cette section, nous présentons les deux méthodes issues de celles présentées dans le Chapitre 4, proposées par Toczek [116] dans le cas du rendu volumique. Nous commençons par la méthode de comparaison de paramètres avec stockage d'équations de plans presentée dans le Chapitre 4, puis nous aborderons la méthode de traversée basée sur la classification des directions. Pour la suite, nous noterons un rayon par son point d'origine O et sa direction $\vec{d}$.

5.2.1 Méthode des paramètres avec stockage d'équations de plans

La méthode des paramètres avec stockage d'équations de plans est similaire à celle discutée dans le Chapitre 4. Dans cette méthode le vecteur tétraèdre contient quatre indices pour identifier les équations de plan de chaque facette

- tétraèdres : quatre indices pour identifier les plans de chaque face
- voisins : les indices des quatre voisins, ainsi en position i correspond le voisin opposé au sommet i.
- densités : la valeur en niveau de gris de chaque tétraèdre

Ainsi le principe de cette méthode est le suivant : pour chaque face i du tétraèdre caractérisée par le vecteur normal $\vec{n}_i$ orienté vers l'extérieur, si $\vec{d} \cdot \vec{n}_i < 0$ on calcule $p_i = \frac{P \cdot \vec{n}_i - O \cdot \vec{n}_i}{\vec{d} \cdot \vec{n}_i}$. Ainsi, la face de sortie est donnée par i telle que p_i est plus petit que tous les autres. Afin d'obtenir un vecteur normal orienté vers l'exterieur sans stocker deux équations de plan pour chaque face, une technique de stockage heuristique a été mise en place. L'astuce utilisée dans [116] est de stocker les indices des équations des plans de telle sorte qu'en partant d'un indice k présent dans le tableau de tétraèdres, on détermine l'indice k_p dans le tableau de plans comme suit :

- $k_p = \lfloor \frac{k}{2} \rfloor$
- si k est impair alors on prend le vecteur normal opposé à celui stocké dans le tableau des équations de plans

5.2.2 Méthode de classification

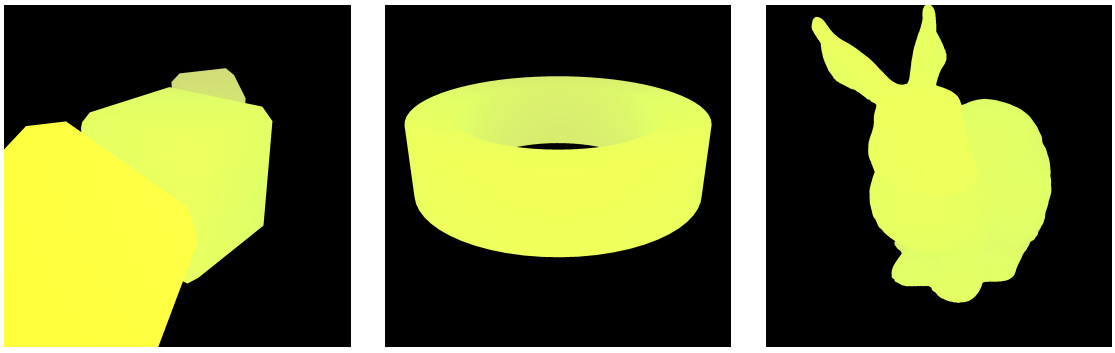
On représente le tétraèdre par les sommets A, B, C, D, où A, B, C définissent la face d'entrée du rayon :

- on calcule $\vec{n} = (B - O) \times \vec{d}$ (normal au plan contenant O, $\vec{d}$, et B)
- si $D \cdot \vec{n} > 0$, on prend $E = C$, sinon, on prend $E = A$,
- on calcule $t = -\frac{D \cdot \vec{n}}{E - D}$
- on calcule $\vec{v} = \vec{n} \times \vec{d}$
- si $((D - O) + t(E - O)) \cdot \vec{v} < 0$, avec $E = A$, alors le rayon sort par la face opposée à B, c'est-à-dire la face définie par les sommets A, D, C
- si $((D - O) + t(E - O)) \cdot \vec{v} < 0$, avec $E = C$, alors le rayon sort par la face opposée à A, c'est-à-dire la face définie par les sommets D, B, C
- enfin, si aucun des cas précédents n'est vérifié, alors le rayon sort par la face opposée à C, c'est-à-dire la face définie par les sommets A, B, D

L'idée de cette méthode est de classifier le point D par rapport au plan qui contient O, $\vec{d}$, et B, puis de classifier $(D + tE)$ par rapport au plan défini par $\vec{v}$. Ainsi, le plan de sortie ou le plan d'entrée du nouveau tétraèdre est issu de ces deux comparaisons.

5.2.3 Évaluations des performances dans le rendu volumique tétraédrique

Les performances ont été mesurées sur trois scènes en utilisant une carte GPU GeoForce GXT 285. Les scènes utilisées sont des cubes, une roue et un lapin (voir FIGURE 5.1) [116]. Le code a été compilé avec l'option *–use_fast_math* du compilateur *nvcc* de Nvidia. Cette option permet de réduire le nombre d'instructions par les opérations coûteuses (comme par exemple la multiplication, division, racine carrée, etc.) dans le code de l'*assembler*, qui est le nombre d'instructions vraiment exécutées sur GPU. Cependant les calculs effectués sont moins précis.



(a) Cubes : 131 tétraèdres (b) Roue : 1175 tétraèdres (c) Lapin : 261524 tétraèdres

FIGURE 5.1 – Scènes utilisées pour le rendu volumique tétraèdrique.

Par la suite on fera référence aux méthodes en considérant les notations suivantes :

- gpu_cmp : méthode de comparaison de paramètres

– gpu_cmp_pl : méthode de comparaison de paramètres avec stockage des équations des plans
– gpu_cl : méthode de classification

Les temps de rendu sont résumés dans le tableau (TABLE 5.1). Le type de données utilisé est le flottant sur 32 bits. En outre le *cache* de texture a été utilisé pour réduire le temps d'accès en mémoire globale.

	Temps du rendu volumique [ms]		
	Cube	Roue	Lapin
Méthode	131 tétraèdres	1175 tétraèdres	261524 tétraèdres
gpu_cmp	46	64	519
gpu_cmp_pl	47	60	183
gpu_cl	39	54	197

TABLE 5.1 – Temps de rendu des trois scènes sur carte GTX 295 en utilisant le *cache* de texture.

Dans le cas d'une tétraèdrisation avec une taille faible (nombre de tétraèdres < 1175), pour les cas des objets cubes et roue, la méthode de classification est plus rapide ou du même ordre de temps par rapport aux autres. Par contre dans le cas du lapin, constitué par un nombre plus élevé de tétraèdres la méthode gagnante est celle de comparaison des paramètres avec stockage des équations des plans. La méthode de classification est du même ordre de temps, par contre la méthode de comparaison de paramètres est bien $\times 2.5$ plus lente que les autres.

À partir des méthodes les plus performantes on a mesuré les performances pour le type flottant en double précision sur 64 bits. Du fait des contraintes du *cache* de texture, qui est disponible seulement pour les données sur 32 bits, celui-ci a été remplacé par le *cache* L1. Les performances sont résumées dans le tableau (TABLE 5.2).

Nous avons pris en compte le cas des données sur 32 bits et 64 bits, en utilisant la méthode de comparaison de paramètres avec stockage d'équations des plans et celle de classification. Dans le premier cas, avec des données sur 32 bits, nous avons obtenu les mêmes temps d'exécution comme dans le cas précédent avec *cache* de texture. Par contre, en utilisant les données sur 64 bits, la méthode de classification est plus rapide que l'autre. On a obtenu un facteur de $\times 2$ d'accélération dans le rendu de la scène du lapin.

On peut affirmer que la méthode de classification est bien plus rapide par rapport à la méthode de comparaison de paramètres avec stockage d'équations de plans. Mais, pour une utilisation comme projecteur tomographique, lorsqu'on cherche à calculer la longueur interceptée entre le rayon et le tétraèdre, un calcul supplémentaire est nécessaire pour obtenir les points d'intersection avec les faces entrantes et sortantes du tétraèdre. Ce n'est pas le cas pour les méthodes de comparaison, car la longueur interceptée est obtenue par

	Temps du rendu volumique [ms]		
	Cube	Roue	Lapin
Méthode	131 tétraèdres	1175 tétraèdres	261524 tétraèdres
gpu_cmp_pl_f32	41	53	183
gpu_cmp_pl_f64	257	349	1201
gpu_cl_f32	40	55	199
gpu_cl_f64	156	205	650

TABLE 5.2 – Temps de rendu volumique des trois scènes sur carte GTX 285.

soustraction de paramètres. De plus la méthode de classification est plus contraignante dans sa mise en œuvre par rapport aux autres méthodes.

Ainsi, pour une évaluation exhaustive des performances et une meilleure compréhension de la méthode la plus adaptée à notre cas, nous avons utilisé une carte graphique Nvidia Tesla C2070 Fermi. C'est une carte avec une architecture dédiée au calcul scientifique. Tout d'abord le cas avec le *cache* de texture a été analysé, les temps de rendu sont résumés dans le tableau TABLE 5.3.

	Temps de rendu volumique [ms]		
	Cube	Roue	Lapin
Méthode	131 tétraèdres	1175 tétraèdres	261524 tétraèdres
gpu_cmp	38	48	146
gpu_cmp_pl	39	49	133
gpu_cl	38	48	157

TABLE 5.3 – Temps de rendu volumique des trois scènes sur carte Tesla C2070 avec *cache* de texture.

Si l'on compare au cas précédent où l'on a utilisé la carte GTX 285, les temps du rendu obtenu par les trois méthodes sont équivalents pour chaque scène utilisée, à l'exception de la scène du lapin où la méthode de paramètre avec stockage d'équations des plans est un peu plus rapide.

Les mesures de performances en utilisant les types simple et double précision sont résumées dans le tableau (TABLE 5.4).

À partir des mesures de temps du rendu, on observe pour chaque scène que les deux méthodes ont des temps d'exécution équivalents : la méthode de paramètre avec stockage d'équations des plans est un peu plus performante.

	Temps de rendu volumique [ms]		
	Cube	Roue	Lapin
Méthode	131 tétraèdres	1175 tétraèdres	261524 tétraèdres
gpu_cmp_pl_f32	39	49	159
gpu_cmp_pl_f64	75	99	342
gpu_cl_f32	38	49	157
gpu_cl_f64	67	85	387

TABLE 5.4 – Temps de rendu volumique des trois scènes sur carte Tesla C2070.

D'ailleurs, l'écart entre le temps d'exécution en flottant simple précision et celui en double précision est à peu près de $\times 2$. Pour l'architecture Fermi toutes les unités de calcul (*Stream Processor*) sont équipées d'une unité arithmétique qui supporte les opérations flottantes sur 64 bits. Ce n'est pas le cas pour l'architecture GT 200, qui dispose d'une seule unité arithmétique en 64 bits par multiprocesseur (pour un ensemble de huit unités de calcul).

En conclusion, sur la base des résultats obtenus dans le cas du rendu d'un volume tétraédrique [116] et dans le cas d'utilisation de différentes architectures de cartes graphiques, nous avons retenu la méthode de comparaison de paramètres avec stockage d'équations de plans. Les points forts de cette méthode sont les performances et sa mise en œuvre simple.

5.3 Projecteur 3D

Dans cette section, nous présentons les résultats des performances de l'opérateur de projection tomographique 3D. Celui-ci est implanté par la méthode de lancer de rayons basée sur la technique de comparaison de paramètres avec stockage d'équations de plans.

Tout d'abord nous présentons les résultats concernant la validation du projecteur sur GPU, ensuite le facteur d'accélération entre le CPU et le GPU, puis les performances sur GPU entre les données sur 32 bits et sur 64 bits.

Les performances ont été mesurées en utilisant un processeur Intel Xeon E5440 @ 2.83 GHz et une carte graphique Nvidia Fermi Tesla C2070.

Par rapport aux méthodes de rendu présentées précédemment, l'opérateur de projection nécessite une étape d'initialisation qui permet de déterminer le tétraèdre qui contient le point focal à partir duquel le rayon pointe vers la caméra.

En effet dans la projection tomographique le point focal ne se trouve pas dans le volume donc l'étape d'initialisation devient plus délicate et moins évidente par rapport au cas du rendu.

Ainsi une étape d'initialisation est nécessaire afin de déterminer le point d'entrée de chaque rayon. L'initialisation n'est pas triviale car des indéterminations du point d'entrée peuvent être rencontrées. Lorsqu'un rayon entre dans le volume il faut qu'il soit initialisé avec le premier tétraèdre intercepté. Une question se pose : comment déterminer le bon tétraèdre ?

Ainsi, nous avons mis en place l'algorithme suivant : tout d'abord on conserve les indices des tétraèdres qui appartiennent aux frontières (un tétraèdre appartient aux frontières si au moins un élément du vecteur des voisins est égal à -1) dans une liste puis pour chaque tétraèdre on calcule :

- le point d'intersection entre le rayon i et une de ses faces
- $s = \vec{n}_j \cdot \vec{d}_i$ le produit scalaire entre le vecteur normal orienté vers l'extérieur de la face interceptée et la direction du rayon i
- si $s < 0$, alors le rayon est initialisé par l'indice du tétraèdre j (voir FIGURE 5.2).

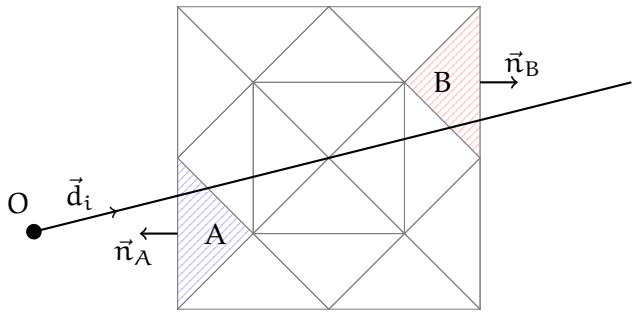


FIGURE 5.2 – Exemple d'intialisation en 2D.

5.3.1 Validation de la projection calculée sur GPU

La validation du projecteur a été faite en considérant l'objet (voir FIGURE 5.3) décrit dans le tableau (TABLE 4.3). Nous avons choisi un objet régulier qui est parfaitement décrit par le maillage afin de ne pas tenir compte de l'effet du maillage. Le maillage tétraédrique a été généré à l'aide de la librairie Tetgen [110]. La configuration d'acquisition est décrite dans le tableau (TABLE 5.5).

Nous avons analysé les résultats en utilisant la double précion et la simple précion.

Double précision

La FIGURE 5.4 montre une projection analytique et celle calculée sur GPU. Nous obtenons une erreur absolue entre la projection analytique et celle calculée sur GPU de l'ordre de $\sim 10^{-15}$, un résultat très important car proche de la précision qu'on peut attendre par un type double (égal à $\sim 10^{-16}$).

La FIGURE 5.5 montre l'erreur absolue des projections illustrées en FIGURE 5.4 et l'erreur absolue moyenne calculée sur les différents angles de projection.

Paramètre	Valeur
distance source-objet	98mm
distance objet-détecteur	132mm
taille détecteur	14.2mm $\times$ 14.2mm
résolution détecteur en pixels	256 $\times$ 256
nombre des projections	64

TABLE 5.5 – Configuration tomographique utilisée pour la validation de l'opérateur de projection sur GPU.

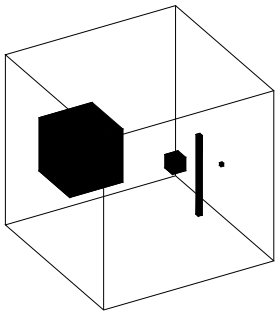
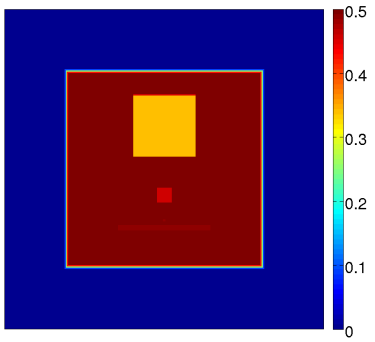
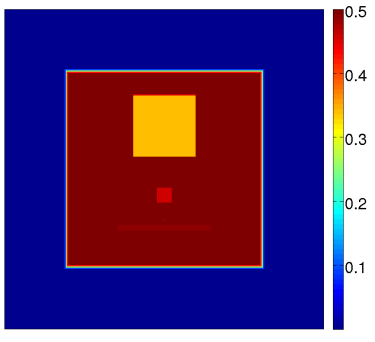


FIGURE 5.3 – Objet utilisé pour la validation de l'opérateur de projection sur GPU.

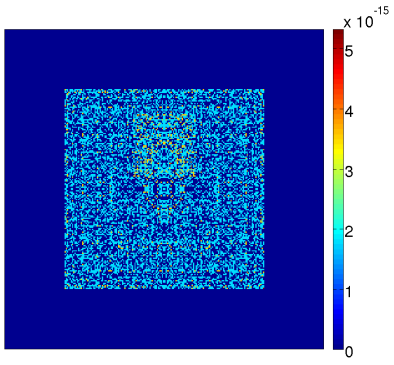


(a) Projection analytique

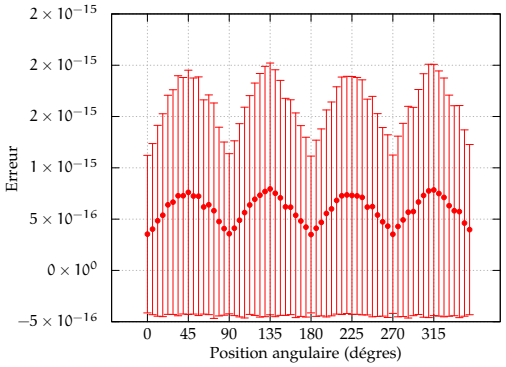


(b) Projection calculée sur GPU

FIGURE 5.4 – Projection en double précision : comparaison avec une projection analytique.



(a) Erreur absolue



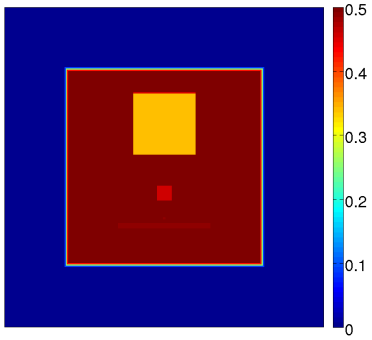
(b) Erreur absolue moyenne évaluée à chaque projection

FIGURE 5.5 – Erreur absolue entre la projection (double précision).

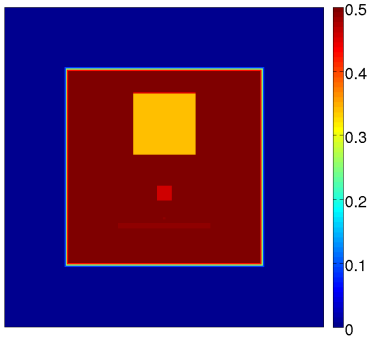
.

Simple précision

La FIGURE 5.6 montre les résultats obtenus en simple précision qui semblent en accord avec la précision maximale utilisant le type simple précision float. Par contre cela est vrai seulement à certains angles de vue, comme illustré par la FIGURE 5.7.



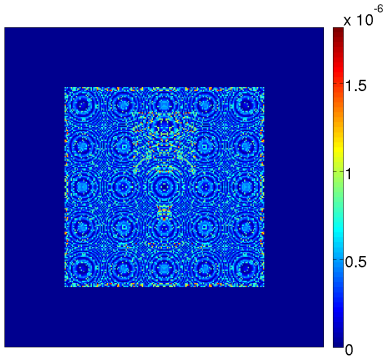
(a) Projection analytique



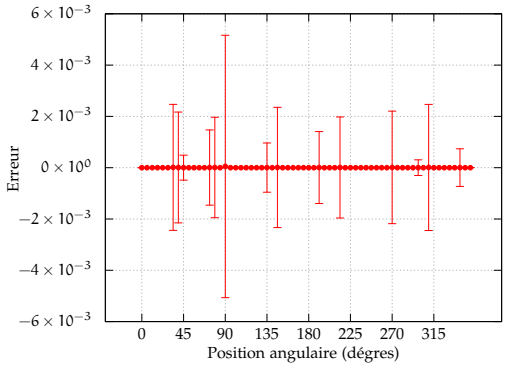
(b) Projection calculée sur GPU

FIGURE 5.6 – Projection en simple précision : comparaison avec une projection analytique.

La cause d'erreur ne peut venir que de l'étape d'initialisation, sinon nous aurions une erreur importante pour tous les angles de vue. Ainsi, nous avons décidé d'utiliser le type double pour effectuer les calculs de l'initialisation. De cette manière on obtient un résultat qui est conforme aux attentes et qui est d'ordre ~10^{-6} (voir FIGURE 5.8).



(a) Erreur absolue



(b) Erreur absolue moyenne évaluée à chaque projection

FIGURE 5.7 – Erreur absolue entre les projections (simple précision).

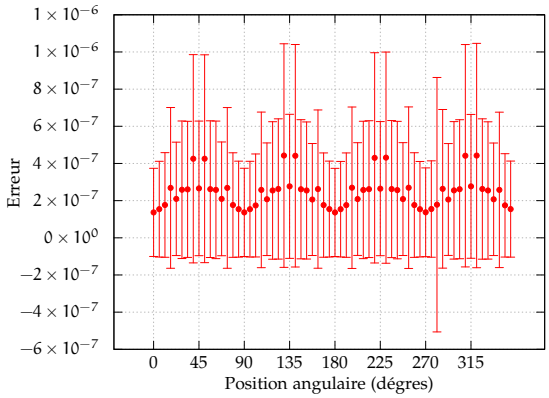


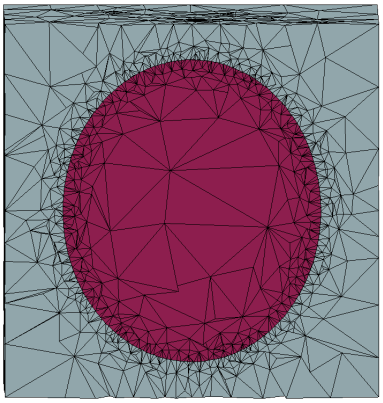
FIGURE 5.8 – Erreur absolue moyenne évaluée à chaque projection : simple précision avec l'étape d'initialisation faite en double précision.

5.3.2 Influence du maillage sur le calcul des projections

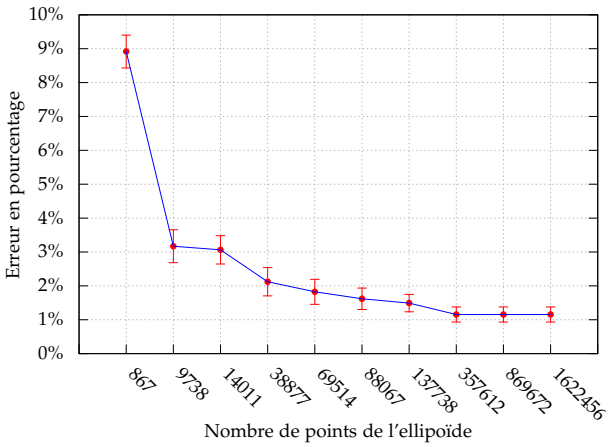
Nous avons étudié l'erreur introduit par un maillage tétraédrique qui ne décrit pas parfaitement un objet. Cette étude a été faite sur un objet de type ellipsoïde (voir FIGURE 5.9a) dont la surface est discrétisé par un nombre de points de 867 à 1622456. L'erreur en pourcentage est définie comme :

$$\mathrm{Err} = \frac{p_T - p_a}{p_a} \cdot 100\% \tag{5.1}$$

où p_T représente les projections tétraédriques et p_a représente les projections analytiques. La (voir FIGURE 5.9b) montre l'erreur Err en fonction du nombre de points qui décrit l'ellipsoïde. On observe une erreur qui décroît au fur et à mesure de l'augmentation du nombre de points qui décrit la surface de l'ellipsoïde, jusqu'à 357612 points. Après l'erreur reste constante à ~ 1%.



(a) Ellipsoïde (décrite par 14011 points)



(b) Erreur en pourcentage entre les projections tétraèdriques et les projection analytiques

FIGURE 5.9 – Objet non régulier (ellipsoïde) utilisé pour étudier l'influence du maillage tétraèdrique sur les projections.

5.3.3 Performances CPU et GPU

Les performances ont été mesurées en calculant la projection du fantôme numérique décrit précédemment (voir FIGURE 5.3). Le facteur d'accélération a été obtenu sans prise en compte du transfert des données entre la mémoire CPU et le GPU.

Taille du volume tétraédrique

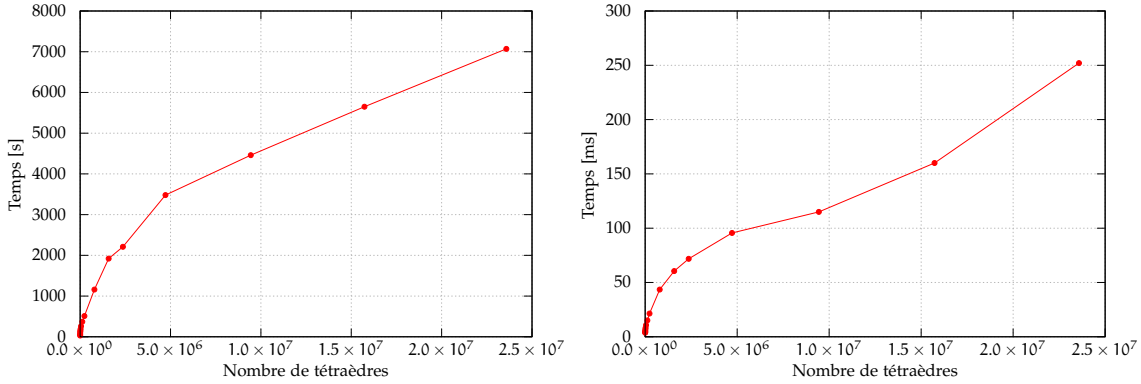
Tout d'abord nous avons évalué les performances en considérant une taille constante pour le détecteur, qui est de 256×256 pixels, en changeant le nombre de tétraèdres qui constituent le volume. Les résultats sont résumés dans le tableau (TABLE 5.6).

Nombre	Temps projection [ms]		Facteur	Temps d'initilisation
Tétraèdres	CPU	GPU (Trans. mém. [ms])	d'accélération	des rayons [ms]
194	30	1.3 (2.4)	×23	40
6890	120	3.3 (2.3)	×36	6.2×10^3
24557	190	5.6 (2.7)	×34	166.2×10^3
237443	500	20.6 (2.7)	×24	706.2×10^3
2365355	2210	68.6 (3.2)	×32	8806.2×10^3

TABLE 5.6 – Temps de calcul CPU/GPU en fonction du nombre de tétraèdres (pour le calcul du facteur d'accélération le temps de transfert mémoire n'est pas pris en compte).

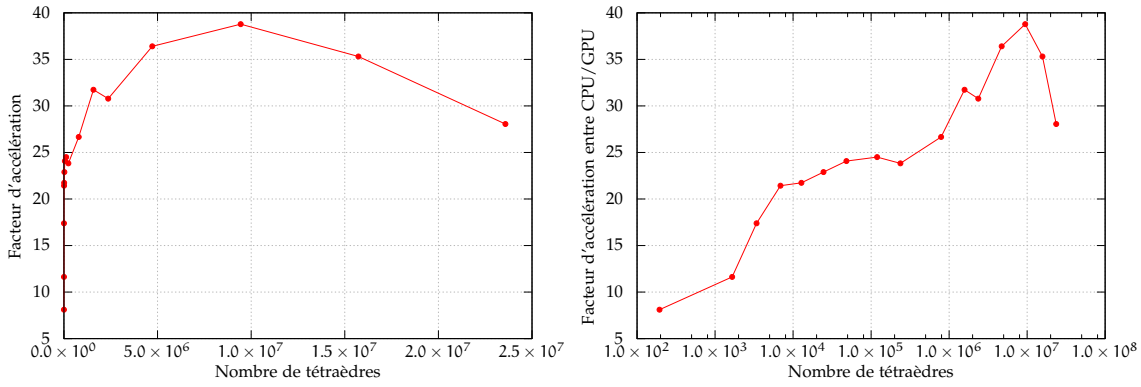
On obtient un facteur d'accélération moyen de ×30. Il n'est pas dépendant du nombre de tétraèdres. On peut constater que le temps d'initialisation n'est pas négligeable par rapport au temps d'exécution : il représente quasiment 12 fois le temps de projection lorsque le volume est constitué par 2365355 tétraèdres (voir TABLE 5.6). Nous avons également optimisé l'étape d'initialisation par la suite.

La FIGURE 5.10 illustre le temps d'exécution de la projection sur CPU et GPU à partir d'un nombre de tétraèdres égal à 194 jusqu'à environ 20 millions avec une taille du détecteur de 256×256 pixels. La FIGURE 5.11 montre le facteur d'accélération en fonction du nombre de tétraèdres obtenu à partir des données montrées en FIGURE 5.10. La courbe qui décrit le facteur d'accélération atteint son maximum pour un nombre de tétraèdres égal à 10 millions. Après cette valeur, la courbe décroît fortement.



(a) Temps CPU (b) Temps GPU

FIGURE 5.10 – Temps d'exécution de l'opérateur de projection en fonction du nombre de tétraèdres (prise en compte du temps de transfert mémoire) avec une taille du détecteur de 256×256 pixels.



(a) Échelle linéaire (b) Échelle logaritmique

FIGURE 5.11 – Facteur d'accélération de l'opérateur de projection en fonction du nombre de tétraèdres (prise en compte du temps de transfert mémoire) avec une taille du détecteur de 256×256 pixels.

Taille de la projection

Dans ce cas, nous avons analysé les performances à partir d'un nombre constant de tétraèdres, égal à 24557, en faisant varier la taille du détecteur. Les performances obtenues sont résumées dans le tableau (TABLE 5.7).

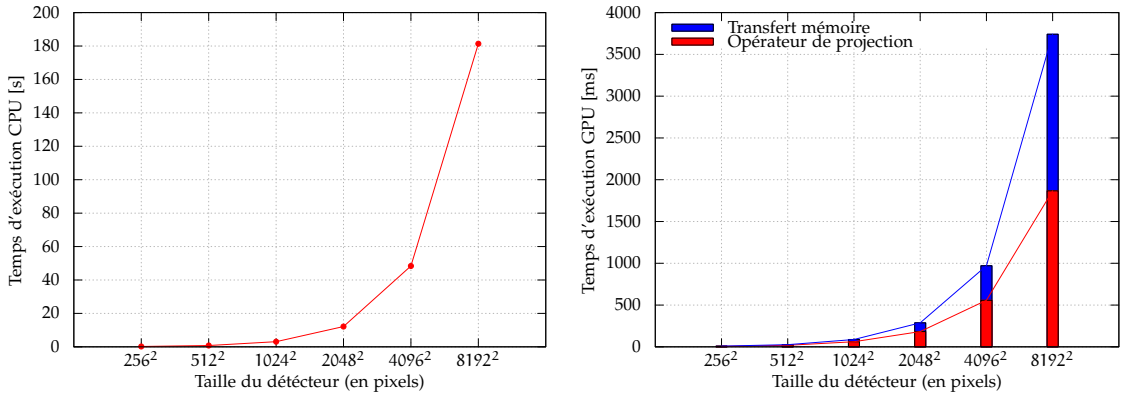
Taille (en pixels) du détecteur	Temps projection [ms] CPU	GPU (Trans. mém. [ms])	Facteur d'accélération	Temps d'initilisation des rayons [ms]
256×256	210	6.1 (2.5)	×34 (×24)	16×10^3
512×512	780	18.5 (7.5)	×42 (×30)	65×10^3
1024×1024	3080	61 (26.8)	×50 (×35)	250×10^3
2048×2048	12130	183 (104.2)	×66 (×42)	1022×10^3
4096×4096	48380	557 (413)	×86 (×50)	4100×10^3
8192×8192	184380	1867 (1174)	×98 (×48)	6544×10^3

TABLE 5.7 – Temps de calcul de l'opérateur de projection CPU/GPU en fonction de la taille du détecteur.

La FIGURE 5.12 montre le temps d'exécution de l'opérateur de projection en fonction de la taille du détecteur. Pour le temps d'exécution sur GPU, nous avons tenu compte du temps de transfert de mémoire, qui n'est pas négligeable lorsque la taille du détecteur devient importante. Ainsi, nous avons calculé le facteur d'accélération avec et sans prise en compte du transfert mémoire (voir FIGURE 5.12). Si on considère le transfert mémoire asynchrone, on obtient que le temps de projection est égal au temps d'exécution. En effet, il est possible de faire ce transfert mémoire alors que les unités de calcul effectuent des calculs par recouvrement calculs/transferts.

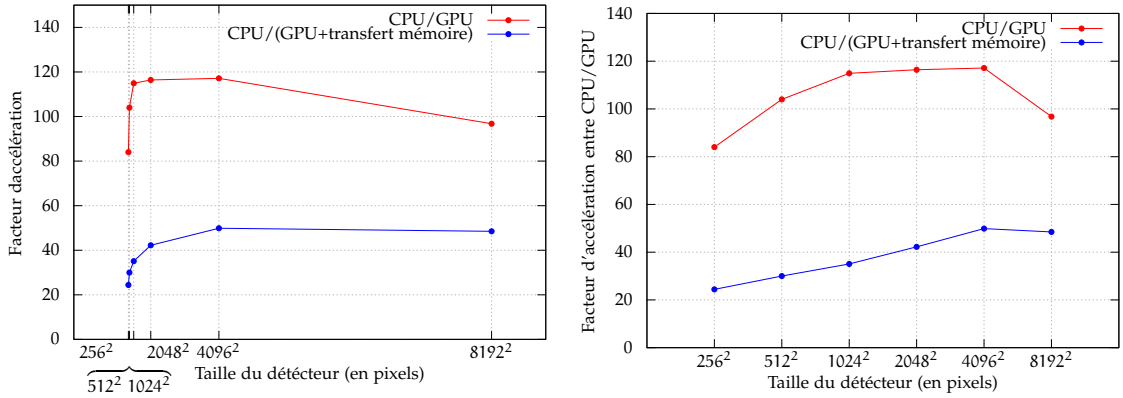
Afin de caractériser les performances obtenues sur GPU, nous avons effectué des comparaisons en considérant le temps d'exécution en fonction de la taille du détecteur et du nombre de tétraèdres. La FIGURE 5.14 présente les temps de calcul de l'opérateur de projection avec prise en compte du temps de transfert mémoire. De plus, la FIGURE 5.15 illustre le facteur d'accélération. On peut observer que le facteur d'accélération est constant par rapport à la taille du détecteur lorsqu'on considère un volume constitué par un grand nombre de tétraèdres (2365355).

Finalement, nous avons évalué les performances de l'opérateur de projection en obtenant un temps d'exécution avec une croissance linéaire en fonction soit du nombre de tétraèdres, soit de la taille du détecteur. Le transfert mémoire devient important lorsque la taille du détecteur augmente, car il faut transférer deux tableaux de taille $(8192^2 \times 3)\times 8o$ chacun pour un volume de données supérieur à 3.2 Go, contenant les points source et



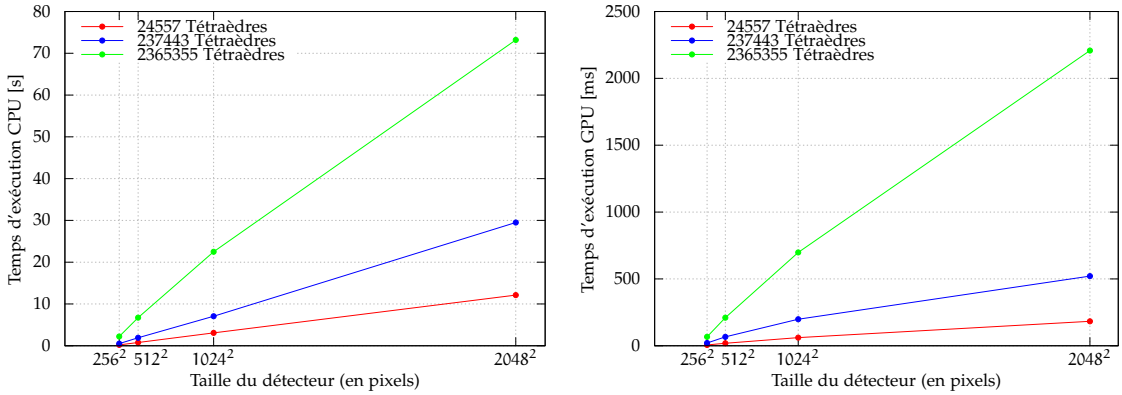
(a) Temps d'exécution CPU (b) Temps d'exécution GPU

FIGURE 5.12 – Temps d'exécution de l'opérateur de projection en fonction de la taille du détecteur avec un nombre de tétraèdres de 24577.



(a) Échelle linéaire (b) Échelle logarithmique

FIGURE 5.13 – Facteur d'accélération de l'opérateur de projection en fonction de la taille du détecteur avec un nombre de tétraèdres de 24577.



(a) Temps d'exécution CPU

(b) Temps d'exécution GPU + transfert mémoire

FIGURE 5.14 – Temps d'exécution de l'opérateur de projection CPU/GPU en fonction de la taille du détecteur avec un nombre de tétraèdres de 24577 (en rouge), 237443 (en bleu) et 2365355 (en vert).

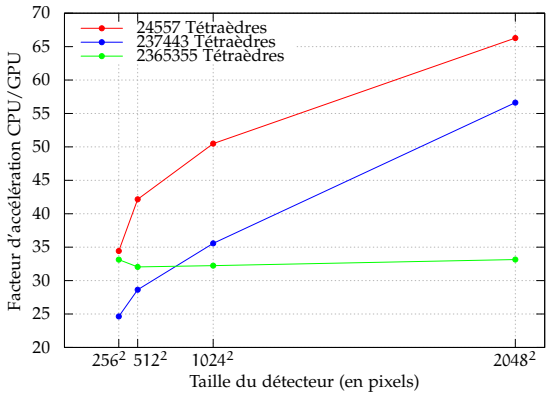


FIGURE 5.15 – Facteur d'accélération de l'opérateur de projection en fonction de la taille du détecteur avec un nombre de tétraèdres de 24577 (en rouge), 237443 (en bleu) et 2365355 (en vert).

détecteur nécessaires pour définir les rayons. Les facteurs d'accélération obtenus sont satisfaisants : $\times 48$ pour un nombre moyen de tétraèdres (par exemple 24577 tétraèdres) et une taille détecteur large (par exemple 8192×8192 pixels) (voir FIGURE 5.13), et $\times 34$ pour un nombre élevé de tétraèdres (par exemple 2365355 tétraèdres) et une taille du détecteur moyenne (par exemple 2048×20148 pixels) (voir FIGURE 5.15).

5.3.4 Performances en simple et double précision sur GPU

Nous avons étudié les performances en flottant simple précision et en double précision sur carte graphique tout d'abord en utilisant une taille de projection constante et égale à 256×256 pixels et en augmentant le nombre de tétraèdres. Les résultats sont résumés dans le tableau (TABLE 5.8).

	Temps projection GPU [ms]		
Taille (en pixels) du détecteur	simple précision	double précision	Facteur d'accélération
194	0.9	1.3	$\times 1.4$
3040	1.7	2.5	$\times 1.4$
12784	3.1	4.5	$\times 1.4$
48270	5.9	7.9	$\times 1.3$

TABLE 5.8 – Temps de calcul de l'opérateur de projection en simple et double précision avec une taille du détecteur de 256×256 pixels.

Nous avons procédé de même en faisant évoluer la taille de la projection avec un nombre de tétraèdres constant (égal à tétraèdres). Les résultats obtenus (TABLE 5.9) sont en accord avec la spécification de l'architecture GPU *Fermi*, où le facteur de performances entre le flottant simple et double précision est égal à 2 [90].

5.4 Rétroprojecteur 3D

À partir de l'implantation de l'algorithme de projection, nous avons implanté l'algorithme de rétroprojection basé sur la méthode de lancer de rayons.

Comme on peut le voir sur le pseudo-code du projecteur (voir ALGORITHME 7) et celui du rétroprojecteur (voir ALGORITHME 8) il n'y a pas de grande différence entre les deux mises en œuvre.

Cependant, dans l'algorithme de rétroprojection, la mise à jour de la valeur du tétraèdre est effectuée par des opérations atomiques. Lorsqu'on exécute des opérations atomiques, l'exécution des *threads* devient séquentielle ; ainsi on perd en partie le bénéfice de la parallélisation.

Taille (en pixels) du détecteur	Temps projection GPU [ms] simple précision	double précision	Facteur d'accélération
256×256	4.3	6.1	×1.4
512×512	13.3	18.5	×1.39
1024×1024	45	61	×1.35
2048×2048	140	183	×1.3
4096×4096	432	557	×1.28

TABLE 5.9 – Temps de calcul de l'opérateur de projection en simple et double précision avec un nombre de tétraèdres de 24557.

Nous avons testé le rétroprojecteur sur le CPU non parallélisé et nous avons observé que le temps d'exécution est très proche de celui du projecteur : le rapport de temps entre les deux est proche de 1.

À partir de ce résultat, nous avons testé le rétroprojecteur sur GPU en changeant les opérations atomiques par des opérations «normales», en sachant que le résultat final ne serait pas correct. Ainsi, nous obtenons des temps d'exécution du projecteur et du rétroprojecteur équivalents à ceux effectués sur CPU.

Algorithme 7 Pseudo code de l'opérateur de projection.

```
1 : for all n : 1 → N_rays do
2 :   ray ← Rayons(n)          // lecture en mémoire des coordonnées du rayon
3 :   tetra ← Tetras_initial(n)          // lecture en mémoire du tétraèdre de départ
4 :   p_old ← param_ent(ray, next)
5 :   intes ← 0
6 :   while (tetra! = −1) do
7 :     den ← Dens(tetra)          // lecture en mémoire de la valeur du tétraèdre
8 :     p ← param(ray, tetra)          // calcul du paramètre et du prochain tétraèdre
9 :     intes ← intes + (p − p_old) × den          // pondération de la valeur du tétraèdre
        par la longueur interceptée et mise à jour de la projection
10 :    p_old ← p
11 :  end while
12 :  projection(n) ← intes          // écriture en mémoire
13 : end for
```

Algorithme 8 Pseudo code de l'opérateur de rétroprojection.

```
1 : for all n : 1 → N_rays do
2 :   ray ← Rayons(n)          // lecture en mémoire des coordonnées du rayon
3 :   tetra ← Tetras_initial(n)          // lecture en mémoire du tétraèdre de départ
4 :   p_old ← param_ent(ray, next)
5 :   intes ← projection(n)
6 :   while (tetra! = −1) do
7 :     node ← tetra
8 :     p ← param(ray, tetra)          // calcul du paramètre et du prochain tétraèdre
9 :     Dens(node) ← Dens(node) + (p − p_old) × intes          // pondération de la
        projection du tétraèdre  par la longueur interceptée et mise à jour de la projection
        et écriture en mémoire
10 :    p_old ← p
11 :  end while
12 : end for
```

5.5 Optimisation

À partir de ces considérations deux principales étape peuvent être optimisées afin d'améliorer les performances générales de la reconstruction tomographique : l'initialisation des rayons et la rétroprojection.

5.5.1 Accélération de l'étape d'initialisation des rayons

La méthode de reconstruction est constituée, comme nous l'avons précédemment décrite, de trois étapes : reconstruction, segmentation et adaptation du maillage.

Parmi ces étapes, seule la reconstruction a été portée sur GPU en particulier la projection et la rétroprojection. Afin d'utiliser la méthode de lancer de rayons [116], une étape d'initialisation du rayon est nécessaire pour identifier le tétraèdre d'entrée dans le volume. Cette étape est très coûteuse en temps de calcul (voir TABLE 5.6 et TABLE 5.7) par rapport au temps d'exécution d'une projection (voir les graphes de la FIGURE 5.16).

L'étape d'initialisation des rayons est décrite par l'ALGORITHME 9. Il consiste en deux boucles : la première sur le nombre de rayons et la deuxième sur le nombre de tétraèdres qui appartiennent à la surface. À l'intérieur de cette boucle, il y a deux tests : un qui permet d'identifier le tétraèdre intercepté par le rayon et l'autre qui lève les ambiguïtés de localisation (voir FIGURE 5.2). Le tableau (TABLE 5.10) résume le nombre d'opérations arithmétiques nécessaires à la vérification des tests.

Ainsi, nous avons envisagé trois solutions possibles d'accélération : deux solutions algorithmiques et une solution matérielle (en utilisant une architecture parallèle multi-cœur ou GPU). La première approche est l'implémentation d'une structure hiérarchique (*quad-tree* ou grille régulière) qui nous permet de subdiviser la surface du volume (voir FIGURE 5.17).

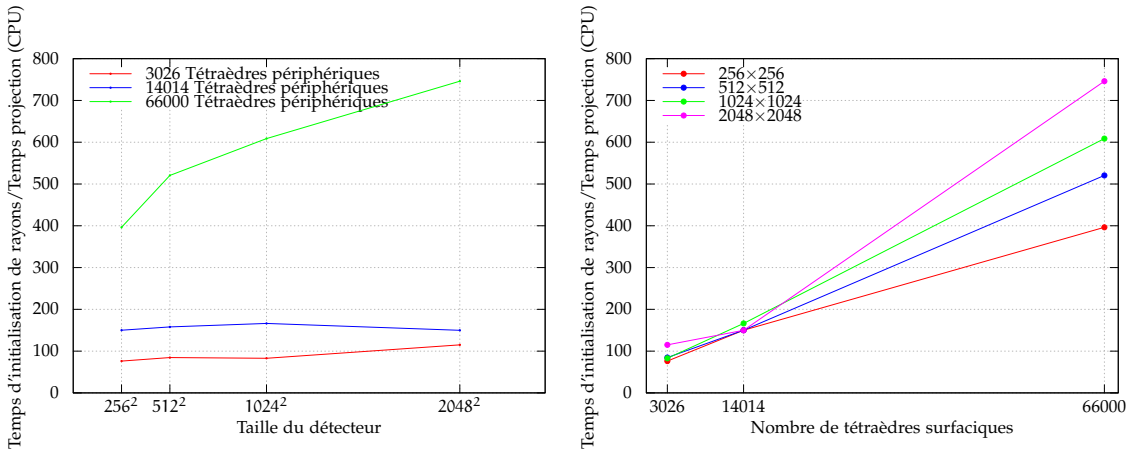


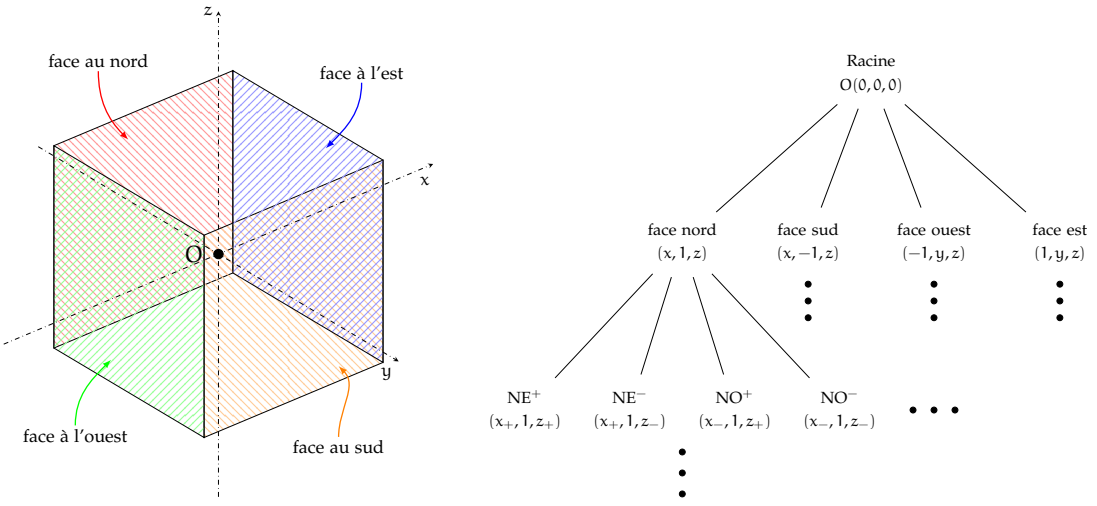
FIGURE 5.16 – Comparaison entre le temps d'exécution (CPU) d'une projection et le temps d'initialisation des rayons : (a) en fonction de la taille du détecteur, (b) en fonction de nombre de tétraèdres périfériques. 3026 tétraèdres périfériques sur un volume de 24557 tétraèdres, 14014 tétraèdres périfériques sur un volume de 237443 tétraèdres et 66000 tétraèdres périfériques sur un volume de 2365355 tétraèdres.

Algorithme 9 Initialisation des rayons.

$\mathcal{T}_s \leftarrow \{T_i\}\, i = 1, \cdots, N_s$ // création de la liste de tétraèdres périfériques
for $i : 1 \rightarrow M_{rayons}$ **do**
  $flag = false$
  while $(j : 1 \rightarrow N_f$ && $(flag == false))$ **do**
    for all $k : 1 \rightarrow 4$ **do**
      if $(intersection(rayon_m, facette_{j,k}) == true)$ **then**
        $s = \vec{d}_i \cdot \vec{n}_j$ // produit scalaire entre la direction du rayon et la normale à la facette du tétraèdre du plan orienté vers l'extérieur
        if $s < 0$ **then**
          Initialise : $rayon_i \leftarrow T_j$
          $flag = true$
        end if
      end if
    end for
  end while
end for

	Opération arithmétique		
	Somme/Soustraction (entre vecteurs)	Produit scalaire	Produit vectoriel
Test 1	4	6	3
Test 2	3	3	1

TABLE 5.10 – Nombre d'opérations nécessaire au calcul d'intersection entre le rayon et une facette d'un tétraèdre (Test 1) et la vérification (Test 2).



(a) Découpage de la surface (b) Structure hiérarchique

FIGURE 5.17 – Représentation du découpage de la surface du volume de reconstruction.

On va ainsi réduire la taille de la deuxième boucle par une recherche rapide de l'élément intersecté. Cette approche permet de diminuer le nombre de tests (voir TABLE 5.10) par de simples comparaisons de coordonnées entre la direction du rayon et les coordonnées du nœud afin de déterminer le tétraèdre intersecté. Ainsi, on minimise fortement le coût et temps de calcul. Cependant, une grille ou un *quadtree* sont des structures de données régulières, donc un problème se pose lorsque la facette d'un élément du maillage n'est pas contenue à l'intérieur d'une feuille. On aboutit à une limitation du nombre de niveaux de la structure hiérarchique. En effet un raffinement trop profond, lorsque la taille du nœud est plus petite que la taille de la facette, peut donner lieu à une répétition d'un élément à l'intérieur de plusieurs nœuds. Ainsi, la structure hiérarchique devient moins performante. Afin de ne pas affecter l'efficacité de la structure hiérarchique, on pourra utiliser un volume tétraédrique régulier, tel que le dernier nœud (niveau) de la structure hiérarchique sera de taille strictement supérieure à la taille de la plus grosse facette. De ce fait une facette sera au maximum commune à quatre nœuds et donc stockée dans ces quatre nœuds. Il est ainsi possible de définir une structure hiérarchique assez fine pour que chaque noeud contienne peu de facettes pour ainsi simplifier au maximum l'étape de recherche de la facette intersectée en la remplaçant par un simple calcul de coordonnées pour déterminer le noeud à chercher. Cette solution est efficace avec une simple grille régulière, du fait que le maillage initial sera relativement régulier aux bords du volume : les facettes seront ainsi régulièrement distribuées au sein des noeuds de la grille. Cette optimisation passe ainsi d'une complexité en $O(N^2)$ en une complexité en $O(N)$ avec N le nombre de tétraèdres ce qui peut réduire le temps d'exécution de l'initialisation par un facteur 100 à 1000.

La deuxième approche proposée, est de créer un masque autour du volume constitué par des mailles plus grossières par rapport aux mailles du volume à reconstruire afin d'obtenir un nouveau volume qui présente un nombre limité de tétraèdres sur la surface (voir FIGURE 5.18). Cette solution ne demande pas une structure de données hiérarchique supplémentaire pour chercher les tétraèdres. Par contre la création du maillage à l'intérieur du masque n'est pas trivial. En effet il faut préserver la continuité entre les deux maillages afin de rendre possible la traversée. À cause de l'insertion du masque, le volume à traverser est plus grand donc on pourrait s'attendre à une légère augmentation du temps de calcul de la projection et de la rétroprojection. En outre, les valeurs des tétraèdres qui composent le masque doivent être initialisées à zéro pour que les résultats de la (rétro)-projection ne soient pas affectés.

En résumé, la structure hiérarchique semblerait plus performante dans le cas d'une tétraèdrisation régulière. De plus elle évite la création d'un maillage additionnel. Et elle supprime les inconvénients d'initialiser les tétraèdres qui appartiennent au masque et de mettre en correspondance les deux maillages.

La dernière solution se base sur l'utilisation d'une architecture parallèle, par exemple un CPU multi-cœur ou bien une carte graphique. En effet la tâche pouvant être parallélisée

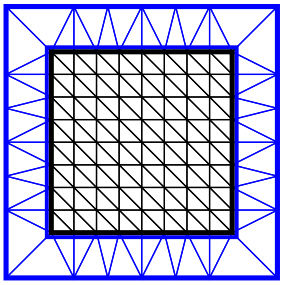


FIGURE 5.18 – Exemple de masque 2D.

est la première boucle qui concerne les rayons. Cette mise en œuvre permet une accélération linéaire avec le nombre de cœurs/GPU et elle est valable dans les deux solutions décrites ci-dessus.

5.5.2 Mise en œuvre d'accélération de l'étape d'initialisation des rayons par la structure hiérarchique

Nous avons mis en œuvre la méthode qui permet d'accélérer l'étape d'initialisation des rayons par la structure hiérarchique de type *quadtree*. Nous avons considéré l'objet régulier, qui à été utilisé dans le cas de la validation de l'opérateur de projection, dont la surface est représente par un maillage de tétraèdre régulier. De plus, nous avons considéré qu'une seule face du cube du volume à reconstruire, en particulier la face décrit par le plan yz avec $x = -\frac{d}{2}$, où d représente la longueur des arêtes de l'objet considéré. Le maillage tétraèdrique a été généré de telle façon que le dernier élément de la structure hiérarchique (la feuille de l'arbre) peut contenir deux facettes du volume tétraèdrique (voir FIGURE 5.19).

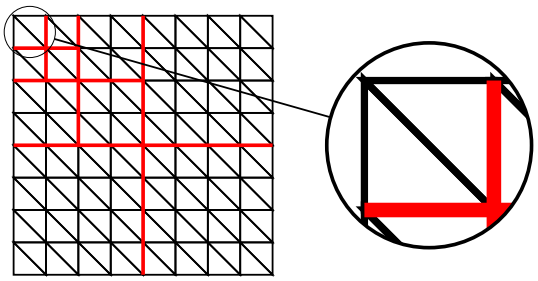


FIGURE 5.19 – Subdivision de la face du volume tétraèdrique.

Afin d'évaluer les performance de l'utilisation de la structure hiérarchique de type *quad-tree* nous avons adopté les subdivisions suivantes de la face du volume tétraèdrique :

- 16×16 (512 tétraèdres périphériques)
- 32×32 (2048 tétraèdres périphériques)
- 64×64 (8192 tétraèdres périphériques)

La FIGURE 5.20a montre les temps d'initialisation des rayons en fonction du nombre de rayons à initialiser (nombre de pixels détecteur). On observe que le temps d'exécution dépend que du nombre de rayons à initialiser pour n'importe quel nombre de tétraèdres périphériques.

De plus, comme le montre la FIGURE 5.20b l'étape d'initialisation des rayons en utilisant la structure hiérarchique est bien $\times 700$ plus rapide que l'initialisation de rayons «classique». La mise en œuvre de la structure hiérarchique permet de compenser l'écart entre le temps d'initialisation des rayons «classique» et le temps d'un projection (CPU) (voir FIGURE 5.16a).

5.5.3 Amélioration du rétroprojecteur

Les performances du rétroprojecteur sont impactées par l'usage des opérations atomiques, qui séquentialisent les accès des *threads* en mémoire partagée, lorsqu'ils se trouvent à traverser le même tétraèdre. Si le volume est constitué par un nombre faible de tétraèdres il y aura plus de *threads* qui vont accéder au même instant aux mêmes tétraèdres donc plus de conflits en mémoire (séquentialisé par les opérations atomiques) que dans le cas où le maillage est plus fin (voir FIGURE 5.21).

Comme résultat, le temps de la rétroprojection tend à se rapprocher de celui de la projection pour des tétraèdrisations de plus en plus fines (voir FIGURE 5.22).

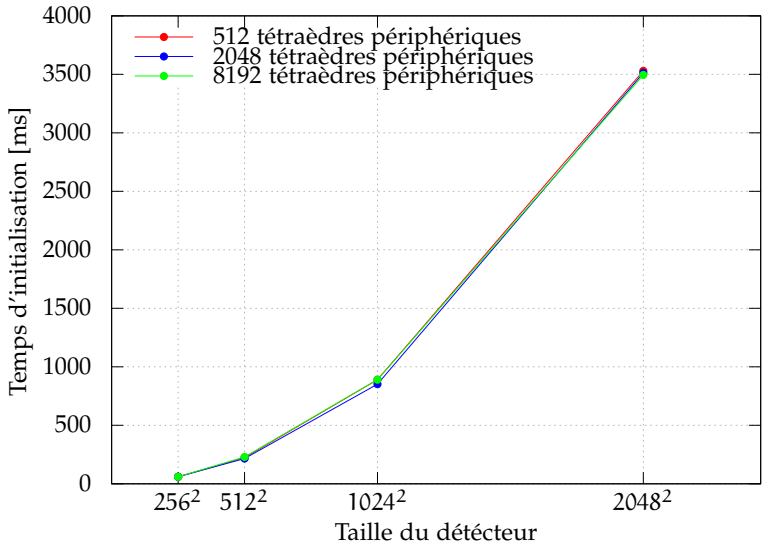
Étant donnée l'irrégularité du volume, nous avons considéré deux stratégies possibles :

- créer une matrice avec N colonnes, où N représente le nombre de tétraèdres, et M lignes , avec M le nombre de *threads*, suivi par une sommation par lignes
- modifier le plan d'exécution des *threads*.

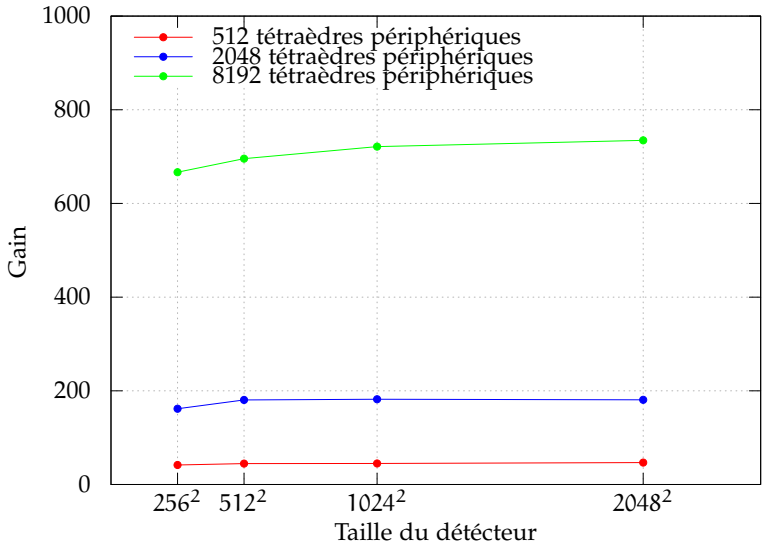
La première solution est intuitive et sa mise en œuvre est triviale. Pour chaque *thread* il faut réserver un vecteur de la taille égale au nombre de tétraèdres, ce qui représente une matrice de grandes dimensions (par exemple avec une taille de détecteur de 256×256 pixels et 24557 tétraèdres on obtient plus d'un Go). Dans cette solution on evite les accès atomiques en effectuant les accès mémoires séparément par *threads*, puis en dernière étape par une mise en commun des résultats avec quelques opérations atomiques. Cette solution supprimer quasiment complètement l'impact des opérations atomiques, au prix d'une emprunte mémoire importante.

La deuxième solution n'est pas triviale. L'idée est de réorganiser les *threads* à l'intérieur du bloc de telle façon que les *threads* dans le même bloc traversent le volume dans des régions différentes permettant ainsi de limiter ou réduire à zéro les accès simultanés et donc les opérations atomiques (voir FIGURE 5.23).

En effet, le nombre de *threads*, qui sont physiquement exécutés par le processeur graphique, est $16 \times$# MPs (nombre de multi-processeurs). S'il y a une corrélation entre les *threads* au niveau de la localité spatiale d'exécution, alors il y a une haute probabilité d'avoir des conflits en accédant en mémoire car des *threads* exécutant des rayons voisins

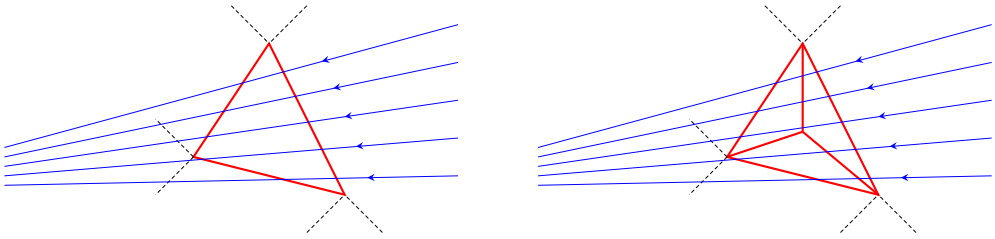


(a) Temps de l'étape d'initialisation avec la structure hiérarchique



(b) Gain entre l'étape d'initialisation des rayons sans et avec l'utilisation de la structure hiérarchique

FIGURE 5.20 – Performances obtenues par la mise en œuvre de la structure hiérarchique.



(a) Maillage grossier (b) Maillage fin

FIGURE 5.21 – Influence du maillage sur la traversée des rayons : (a) les *threads* accèdent simultanément, (b) les *threads* accèdent en instants différents aux triangles.

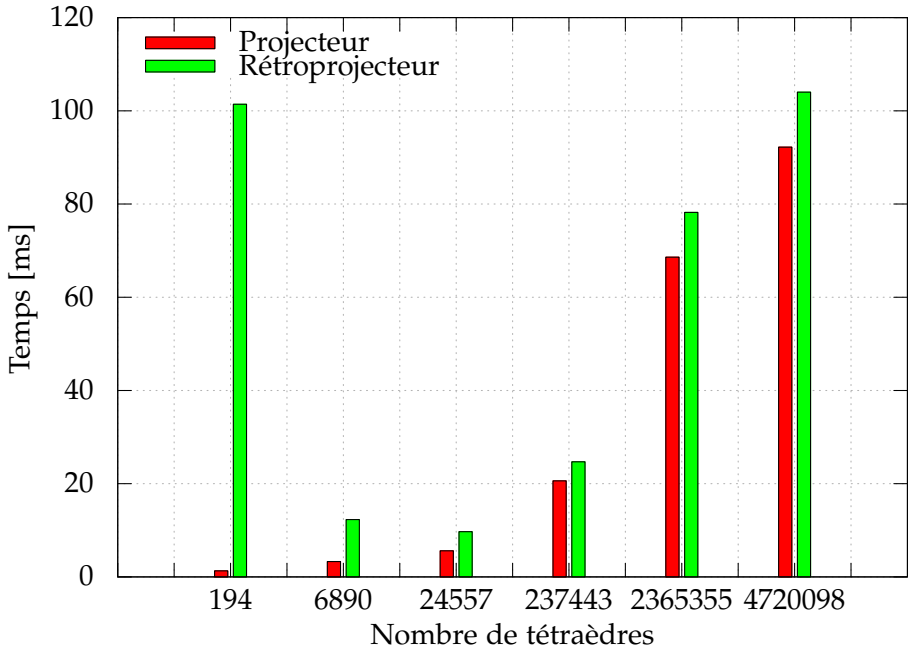


FIGURE 5.22 – Temps de calcul sur GPU de l'opérateur de projection et de rétroprojection en fonction de nombre de tétraèdres avec une taille de détecteur fixée à 256×256 pixels.

traverseront des tétraèdres voisins ou identiques (avec conflits). Au contraire, s'il n'y a pas de corrélation entre les *threads* (on perd la localité spatiale) alors on a une faible probabilité d'avoir des conflits en accédant en mémoire car le nombre de tétraèdres est largement supérieur au nombre de *threads*.

Cependant, la réorganisation comporte la perte de l'accès consécutif en mémoire globale (*coalesced memory*) et la perte de localité spatiale importante pour le fonctionnement efficace des mémoires cache. Une possibilité serait d'utiliser le préchargement en mémoire partagée afin d'améliorer les accès en mémoire globale, mais uniquement si on maitrise une certaine localité spatiale. La perte de performances du fait des accès mémoire risque de ne pas compenser le gain apporté par la réduction du nombre de conflits au niveau des opérations atomiques ! En pratique le nombre de tétraèdres est toujours relativement important par rapport au nombre de *threads* physiquement exécutés en même temps (quelques centaines), du coup l'impact des opérations reste relativement faible.

5.6 Conclusions

Dans ce chapitre nous avons présenté les travaux qui ont été réalisés sur GPU. Les performances obtenues sont très importantes et justifient le choix de l'architecture parallèle prise en compte. Un important résultat obtenu est le temps de la rétroprojection comparable à celui de la projection, contrairement au cas d'une mise en oeuvre de la rétroprojection calculée par la méthode *ray-driven*.

Nous avons analysé l'influence de la taille du problème sur le temps d'exécution de l'opérateur de projection. En particulier, l'impact en fonction du nombre de mailles du volume et de la taille du détecteur. Dans les deux cas nous avons observé une croissance linéaire. Nous n'avons pas analysé l'influence du nombre de projections (angles de veus), mais étant donné qu'à chaque position angulaire le volume et le détecteur ne changent pas, alors le temps d'exécution augmentera linéairement avec le nombre de projections.

Certains aspects de gestion de mémoire n'ont pas été pris en compte. Par exemple, chaque rayon est identifié par le point d'origine et sa direction, donc par six flottants, simple ou double précision. Mais vu que pour une géométrie *Cone-Beam* les rayons partent d'un même point on pourra stocker seulement un point d'origine pour tous les rayons pour une projection donnée. Ainsi, on pourra économiser $(N^2 - 1) \times 3$ flottants, avec N^2 la taille du détecteur.

De plus, une implémentation multi-GPU peut être envisagée aisément. Par exemple en divisant les projections et en les distribuant sur chaque carte GPU. Afin de calculer la projection et la rétroprojection dans la mémoire de chaque carte la tétraèdrisation sera distribuée dans les différentes mémoires des GPUs. Le facteur d'accélération sera ainsi linéairement proportionnel au nombre de GPU puisqu'il n'y a pas de communication entre les GPUs.

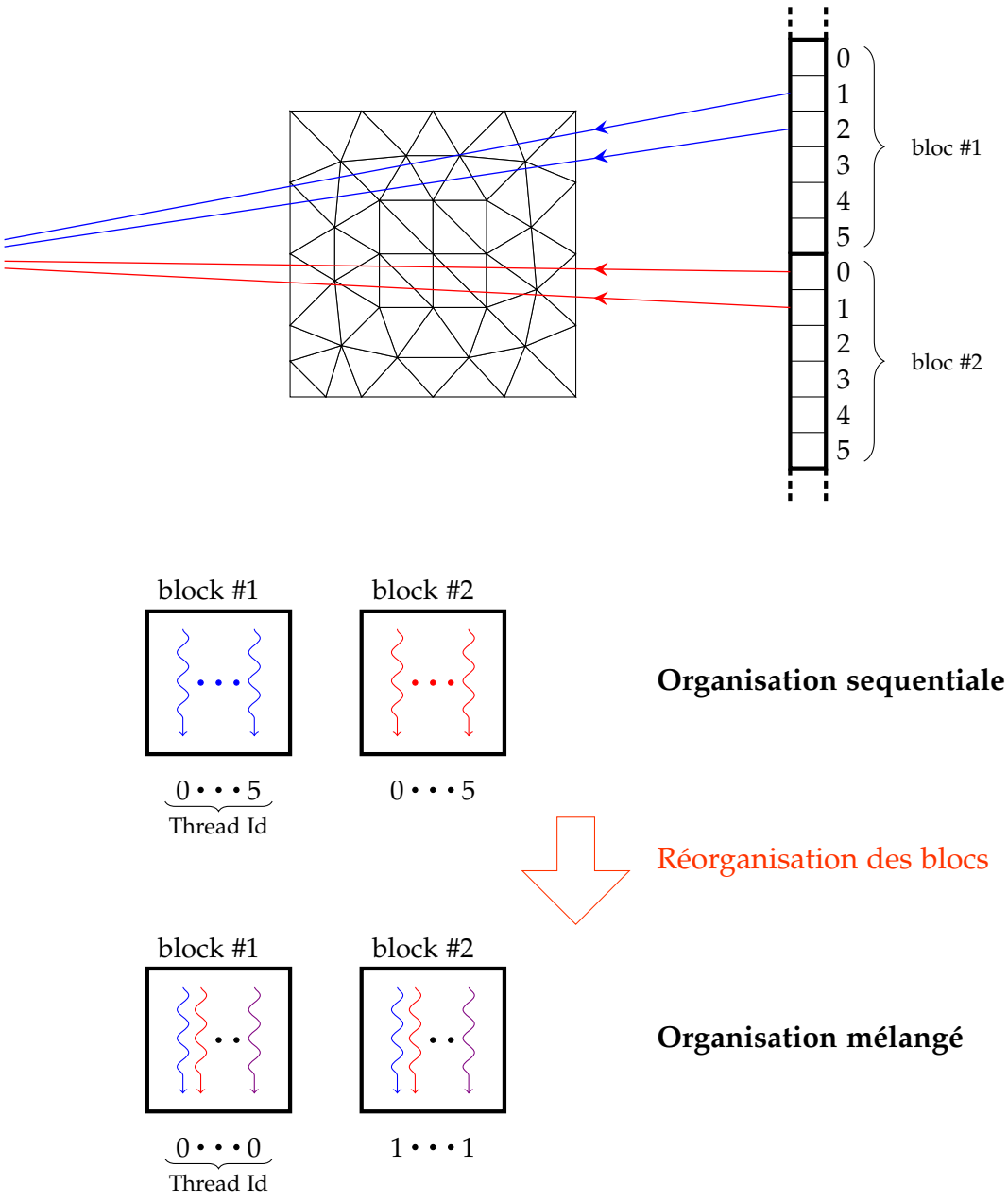


FIGURE 5.23 – Réorganisation des *threads* à l'intérieur des blocs GPU.

Conclusions et Perspectives

Les travaux de cette thèse ont pour objectif de proposer une nouvelle approche à la reconstruction d'une image tomographique. Pour cela, nous avons mis en oeuvre des algorithmes de reconstruction algébriques adaptés à une représentation non pixélisée, couplés à une méthode de segmentation usuellement appliquée en post traitement, afin de mettre à disposition une méthode globale permettant d'accéder à une image traditionnelle en niveaux de gris et à sa segmentation induite par sa représentation par un maillage adapté à son contenu.

Le travail peut être considéré à travers deux parties principales : la première concerne une approche de reconstruction tomographique à partir d'une discrétisation non conventionnelle de l'espace de reconstruction par des éléments de volume non pixélisés s'adaptant à l'objet estimé, et la deuxième partie concerne l'implantation des principales étapes de reconstruction coûteuses en termes de temps de calcul sur une architecture parallèle de type GPU.

Dans un premier temps, nous avons rappelé le principe de la tomographie par rayons X et le problème de reconstruction inverse du volume étudié à partir de ses projections. Nous avons considéré différents algorithmes de reconstruction et présenté leurs avantages et leurs inconvénients ce qui nous a permis de justifier le choix d'utiliser des méthodes itératives. Nous avons également discuté des différentes discrétisations d'une image utilisées jusqu'alors en tomographie RX, et avons fait le choix d'utiliser un modèle de discrétisation par un maillage irrégulier composé d'éléments triangulaires. Le modèle est basé sur une fonction constante par morceaux.

Dans un second temps, nous avons mené une étude bibliographique sur les algorithmes de segmentation développés en traitement d'images. Cette étude a montré que les techniques les plus efficaces pour une application sur des images de tomographie RX sont celles basées sur un modèle de contour actif, paramétrique ou géodésique. Ces techniques sont déjà employées en tomographie sur des images ou volumes reconstruits par exemple pour la segmentation d'organes en imagerie médicale. Nous avons validé le fait que la paramétrisation et le positionnement de la courbe initiale influencent le résultat final de la segmentation pour le modèle pramétrique. Ne connaissant pas *a priori* l'objet observé, nous avons opté pour le modèle géodésique basé sur une approche *level set* pour la modélisation du contour actif. Cette méthode peut être mise en œuvre sur un objet simple ou complexe, et permet de discriminer plusieurs objets ou matériaux dans l'image.

Dans le Chapitre 4, nous avons décrit la méthode de reconstruction globale intégrant la segmentation de l'objet étudié au cours des itérations. La méthode proposée consiste en une étape d'initialisation et trois étapes principales itérant séquentiellement jusqu'à convergence. Ces trois étapes sont la reconstruction tomographique, la segmentation et la génération d'un maillage adapté au contenu de l'image. Ainsi cette approche lie la reconstruction tomographique à la segmentation et permet de s'affranchir d'une étape supplémentaire de traitement d'image sur le volume reconstruit. L'initialisation du processus est un maillage irrégulier dont le nombre d'éléments peut être diminué d'au moins un facteur 10 comparé à celui d'une grille régulière usuellement utilisée. La valeur associée à ces éléments est homogène sur l'ensemble du volume. L'étape de reconstruction met en œuvre les algorithmes itératifs ML-EM et gradient conjugué adaptés à la nouvelle représentation irrégulière. L'étape de segmentation, elle, basée sur les levels sets, traite l'image estimée à l'étape précédente et fournit les contours sur lesquels s'appuiera le prochain maillage. Une étude détaillée a été menée sur la génération du maillage optimal représentant correctement les détails des différentes zones composant l'objet (contours). En particulier, dans le cas d'une image contenant plusieurs structures et/ou matériaux, nous avons établi un critère permettant d'estimer le nombre de points nécessaires à la description des différents contours de l'image. Ce critère est basé sur le rapport des longueurs de chaque contour au contour le plus petit. En adoptant cette stratégie, nous obtenons un maillage adaptatif dans lequel la densité des mailles et leur taille sont fonction de la distance aux contours. Ainsi les zones contenant de l'information sont représentées par une densité accrue de petits triangles et les zones contenant peu d'information sont au contraire représentées par des mailles grossières. Ce processus itératif simplifie le problème à chaque nouvelle itération globale, ce qui par conséquent accélère les calculs (par exemple dans la reconstruction du genou à partir de 360 projections et une taille détecteur de 1024 pixels, le temps par itérations passe de 30 minutes pour un volume de 100140 triangles à 30 secondes pour un volume de 1135 triangles).

Les résultats obtenus à partir de données simulées et de données réelles montrent des reconstructions très satisfaisantes dans de le cas de données complètes : les images en niveau de gris sont comparables aux résultats obtenus avec un algorithme analytique et le maillage final est adapté au contenu de l'objet reconstruit. Les structures d'intérêt sont correctement séparées les unes des autres. Le maillage met en évidence les contours des structures et permet de s'affranchir de l'étape de segmentation post-reconstruction.

Nous avons également reconstruit des objets à partir d'un faible nombre de projections et à partir d'un domaine angulaire restreint. La qualité des images reconstruites est médiocre surtout dans le cas d'un très faible nombre de projections, et la segmentation manque de robustesse dans les cas extrèmes (<36 projections et $<\pm 60^{\circ}$). Cependant, les résultats sont directement liés aux algorithmes de reconstruction tomographique implantés (GC et EM) qui n'incluent aucun terme de régularisation permettant de palier au manque de données. Il serait donc intéressant d'introduire des *a priori* dans ces modèles afin de

compenser ce manque d'information.

Nous nous sommes ensuite intéressés aux aspects de parallélisation de la reconstruction tomographique sur les architectures parallèles. Après une étude bibliographique conduite au Chapitre 3 détaillant les différentes architectures utilisées dans ce domaine, nous avons considéré les processeurs graphiques (GPU) pour notre application. Au Chapitre 5, nous avons présenté la mise en œuvre des opérateurs de projection et de rétroprojection sur GPU. Une version 2D de l'implantation de ces opérateurs montre la faisabilité du processus dans son ensemble. Une version optimisée a été mise en oeuvre dans le cas 3D. Nous avons étudié les techniques de lancé de rayons utilisées dans le rendu de volume tétrahédrique pour leur similarité avec l'opérateur de projection des méthodes de reconstruction itérative. Aux vues de leurs performances sur cartes graphiques, nous avons optimisé l'opérateur de projection selon l'algorithme de comparaison de paramètres avec stockage de l'équation de plans. Ainsi, les performances en terme de temps calcul ont été améliorées d'un facteur $\times 30$. L'opérateur de rétroprojection, quant à lui, fait appel à des opérations atomiques ce qui limite intrinsèquement l'amélioration des performances. Cependant, si le nombre de tétraèdres considérés est suffisamment grand (> 4700000), les opérations conflictuelles (écritures simultanées) sont moins prédominantes et les performances attendues peuvent être du même ordre que pour l'opérateur de projection (facteur $30\times$).

La phase d'initialisation du lancer de rayons peut également être optimisée avec une structure de données hiérarchique régulière. On obtient ainsi un temps d'exécution sur GPU inférieur à 100 ms pour chaque opérateur si on considère un volume composé d'environ cinq millions de tétraèdres. Les études sur les performances ont montré que le taux d'accélération dépend linéairement du nombre de tétraèdres composant le volume traité et de la taille du détecteur.

La méthode proposée dans ce travail permet d'obtenir une estimation de la distribution tomodensitométrique du volume reconstruit associée à une image topologique représentée par un maillage adapté à son contenu. Le maillage ainsi obtenu décrit l'objet étudié de façon similaire aux fichiers de type CAO usuellement utilisés (stl, off, vrml, etc). Cette caractéristique pourrait donc être utilisée dans le cadre d'études de *reverse engineering* pour obtenir des modèles de pièces industrielles ou manufacturées. Une autre application serait d'utiliser les pièces reconstruites maillées en entrée de modèles de thermodynamique ou de mécanique des fluides.

La méthode pourrait également trouver une application dans le domaine médical pour l'étude de phénomènes lents tels que l'évolution de l'arthrose. La déformation de l'os et de l'espace inter articulaire pourrait par exemple être directement étudiée à partir de l'image reconstruite superposée aux segmentations provenant des différents examens au cours du temps.

Dans chacune des applications citées, on pourrait également envisager d'initialiser le processus global de reconstruction par un fichier existant contenant les informations d'une structure maillée (fichier exitant de la pièce industrielle ou résultat d'un examen antérieur avec cette méthode). Ainsi, la grille comporterait déjà un nombre restreint d'éléments de volume ce qui permettrait de réduire considérablement les temps de calcul et d'initialiser les processus de segmentation plus près de la solution.

Des améliorations pourraient être apportées notamment dans le choix des algorithmes de reconstruction tomographique. En effet, les algorithmes utilisés dans ces travaux n'introduisent aucun *a priori* ni aucune régularisation. L'implantation d'algorithmes plus sophistiqués pourrait apporter des résultats plus satisfaisants : une meilleure estimation de l'objet impliquant alors une meilleure segmentation (les étapes alternant séquentiellement et utilisant les résultats de l'autre), en particulier dans le cas de données incomplètes.

Par ailleurs, le modèle de fonction constante par morceaux défini sur un maillage adaptatif ne permet pas d'estimer la dérivée de façon satisfaisante. On pourrait alors envisager d'utiliser une fonction continue permettant le calcul de la dérivée. Ceci éviterait tout passage d'une grille irrégulière à une grille régulière (et vice versa), et le calcul direct de la derivée permettrait de considerer un autre critère de minimisation du problème inverse (par exemple la norme TV).

Quant à la méthode de segmentation, nous avons pris en compte un modèle de segmentation simple, dans lequel le volume de reconstruction est considéré comme un objet compact ou mono-matériau. Pour traiter les cas d'objets non compacts ou multi-matériaux, la valeur des différentes lignes de niveaux sont estimées à partir de la fonction *level set*, de façon plus ou moins empirique. Afin de s'affranchir de cela, on pourrait envisager la mise en œuvre de la méthode de segmentation *multi-phase* basée sur le modèle de Mumford and Shah proposée par Vese et Chan [117]. Ceci envisage de bonnes perspectives de travail, en particulier dans le cas 3D pour lequel la parallélisation des codes serait alors une nécessité pour contenir le temps d'exécution.

En ce que concerne l'aspect de parallélisation, nous avons pris en compte l'architecture parallèle de type carte graphique. Comme nous l'avons présenté dans le Chapitre 3 d'autres architectures parallèles sont envisageables. Ainsi, des évaluations de performances plus exhaustives pourraient être effectuées en considérant les architectures de type FPGA ou cluster CPU/GPUs. Enfin, on pourrait envisager la mise en œuvre de l'algorithme de reconstruction (par exemple le gradient conjugué) entièrement sur GPU.

Finalement, pour la reconstruction de grands jeux de données, une implantation multi parallélisée devra être envisagée. La stratégie qui découpe le volume de reconstruction en sous volumes de plus petites tailles semble peu adaptée étant donné les irrégularités tant au niveau de la taille que de la densité des mailles. On pourrait considérer une autre approche en considèrant l'espace de projection et découper par rapport aux pixels détecteurs ou aux angles de vues. Une étude devra établir les bases pour passer à la

mise en œuvre sur architectures parallèles de type multi-GPUs ou server multi-cœurs CPUs / multi-GPUs puis sur cluster de multi-GPUs. Le découpage du problème et sa parallélisation permettent d'envisager un taux d'accélération proportionnel au nombre de GPUs, du fait qu'il n'y a pas de communication de résultats intermédiaires entre les GPU.

Annexe A

Courbure

Nous présentons le calcul de la courbure d'un contour à partir d'une fonction f, qui peut représenter une image ou bien une fonction implicite. D'abord nous définissons le vecteur normal orienté vers l'extérieur $\vec{N}$ comme

$$\vec{N} = \left(\frac{\nabla f}{|\nabla f|}\right) \tag{A.1}$$

En outre, le gradient de la fonction f est défini comme

$$\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right) \tag{A.2}$$

où, $|\nabla f| = \sqrt{f_x^2 + f_y^2}$, avec $f_x = \dfrac{\partial f}{\partial x}$ et $f_y = \dfrac{\partial f}{\partial y}$, respectivement.

Ainsi, la courbure est définie comme la divergence du vecteur normal unitaire $\vec{N}$:

$$\kappa = \nabla \cdot \vec{N} \tag{A.3}$$

L'équation (A.3) peut être réécrite comme

$$\begin{aligned} \kappa &= \nabla \cdot \left(\frac{\nabla f}{|\nabla f|}\right) \\ &= \nabla \left(\frac{1}{|\nabla f|}\right) \cdot \nabla f + \frac{1}{\nabla f} \nabla \cdot \nabla f \\ &= -\left(\frac{1}{|\nabla f|^2}\right) \nabla \left(|\nabla f|\right) \cdot \nabla f + \frac{1}{|\nabla f|} \cdot \Delta f \end{aligned} \tag{A.4}$$

En résolvant $\nabla \left(|\nabla f|\right)$ par rapport à $\dfrac{\partial}{\partial x}$, on obtient :

$$\begin{aligned} \frac{\partial}{\partial x} |\nabla f| &= \frac{1}{2} \frac{1}{\sqrt{f_x^2 + f_y^2}} \frac{\partial}{\partial x} \left(f_x^2 + f_y^2\right) \\ &= \frac{1}{2} \frac{1}{|\nabla f|} \left(f_x^2 + f_y^2\right) \left(2 f_x f_{xx} + 2 f_y f_{xy}\right) \\ &= \frac{1}{|\nabla f|} \left(f_x^2 + f_y^2\right) \left(f_x f_{xx} + f_y f_{xy}\right) \end{aligned} \tag{A.5}$$

De la même façon, par rapport à $\frac{\partial}{\partial y}$:

$$\begin{aligned}
\frac{\partial}{\partial y}|\nabla f| &= \frac{1}{2}\frac{1}{\sqrt{f_x^2+f_y^2}}\,\frac{\partial}{\partial y}\left(f_x^2+f_y^2\right)\\
&= \frac{1}{2}\frac{1}{|\nabla f|}\,\left(f_x^2+f_y^2\right)\left(2f_y f_{yy}+2f_x f_{xy}\right)\\
&= \frac{1}{|\nabla f|}\,\left(f_x^2+f_y^2\right)\left(f_y f_{yy}+f_x f_{xy}\right) \qquad \text{(A.6)}
\end{aligned}$$

À partir des équations précédentes (A.4), (A.5) et (A.6) après un arrangement on obtient l'équation de la courbure ainsi définie :

$$\begin{aligned}
\kappa &= -\frac{1}{|\nabla f|^3}\left\langle\left(f_x f_{xx}+f_y f_{xy}\,,\,f_x f_{xy}+f_y f_{yy}\right),\left(f_x\,,\,f_y\right)^T\right\rangle+\frac{1}{|\nabla f|}(f_{xx}+f_{yy})\\
&= -\frac{1}{|\nabla f|^3}\left[\left(f_x f_{xx}+f_y f_{xy}\right)f_x+\left(f_x f_{xy}+f_y f_{yy}\right)f_y+\left(f_{xx}+f_{yy}\right)\left(f_x^2+f_y^2\right)\right]\\
&= \frac{1}{|\nabla f|^3}\left(f_x^2 f_{yy}-2f_x f_y f_{xy}+f_y^2 f_{xx}\right) \qquad \text{(A.7)}
\end{aligned}$$

Annexe B

Fonction nœudale

Nous présentons dans cette annexe la formulation et la validation de l'opérateur de projection dans le cas d'un maillage représenté par une fonction nœudale. Cette formulation est l'adaptation de la modélisation utilisée dans le cadre de la tomographie d'émission monophotonique, SPECT (*Single Photon Emssion Computed Tomography*) [12, 118], dans le cadre de la tomographie en transmission.

Dans ce modèle une image $f(x)$, $x \in \mathbb{R}^{2,3}$ est définie sur un domaine irrégulier D. Celui-ci est représenté par un ensemble de triangles, chacun défini sur D_l, qui ne se chevauchent pas : $D = \bigcup_{l=1}^{L} D_l$. Ainsi, la fonction $f(x)$ est définie sur le domaine D comme

$$f(x) = \sum_{i=1}^{N} f(x_i)\, \varphi_i(x) \; x \in D \tag{B.1}$$

où N représente le nombre de nœuds contenu dans le domaine D. $\varphi_i(x)$ est la somme des fonctions d'interpolation de base, $\psi_{l,k}(x)$ correspond aux éléments attachés aux nœuds i. $\psi_{l,k}(x)$ est la fonction d'interpolation associée au nœud k (élément triangulaire $k = 1, \cdots, 3$) appelée *shape function* [7].

Ainsi, on peut définir l'opérateur de projection $p = Hf$, comme dans la SECTION 1.3.2 du Chapitre 1 où f est la fonction définie en (B.1). Si on considère la matrice H, où $h_{i,j}$ représente la longueur interceptée du rayon i par le triangle j, (cf. Chapitre 4 SECTION 4.4.1) alors l'élément $h_{i,j}$ est calculé comme

$$h_{i,j} = \sum_{t \in \mathcal{T}} \int_{r \cap t} \varphi_j(x)\, dx \tag{B.2}$$

où $\mathcal{T}$ représente l'ensemble des triangles qui sont attachés au nœud j, et r définit l'équation du rayon considéré. Dans ce cas la matrice $h_{i,j}$ dépend soit de la longueur d'intersection soit de la structure topologique du maillage.

La validation de l'opérateur de projection a été faite en utilisant l'image illustrée en figure (FIGURE B.1). Les projections obtenues ont été comparées à celles obtenues par l'opérateur de projection défini au Chapitre 4. Elles sont illustrées en FIGURE B.2.

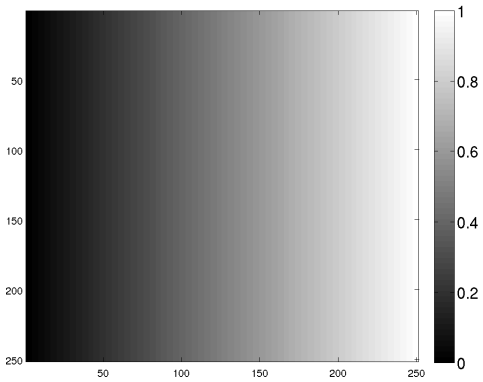
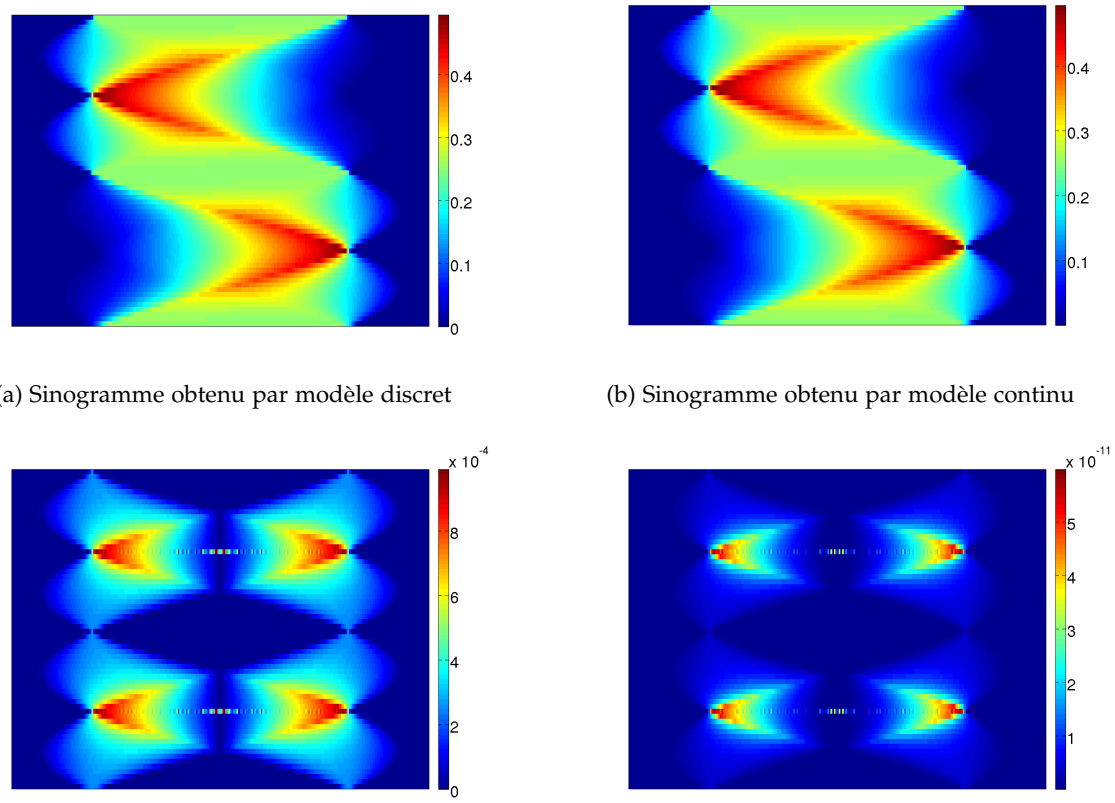


FIGURE B.1 – Image du gradient dans la direction horizontale.



(a) Sinogramme obtenu par modèle discret

(b) Sinogramme obtenu par modèle continu

(c) Erreur absolue

(d) Erreur quadratique moyenne

FIGURE B.2 – Validation du modèle continu.

Bibliographie

[1] P. Aggarwal et R. Mehra : High speed CT image reconstruction using FPGA. *International Journal of Computer Applications*, 22(4) :7–10, 2011.

[2] C. V. Alvino et A. J. Yezzi : Tomographic reconstruction of piecewise smooth images. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 576–581, 2004.

[3] M.S. Aslan, A.A. Farag et B. Arnold : Segmentation of vertebrae using level sets with expectation maximization algorithm. *In Proceedings of the International Symposium on Biomedical Imaging*, pages 2010–2013, 2011.

[4] R. Baldock et J. Graham, éditeurs. *Image Processing and Analysis*. Oxford University Press, 2000.

[5] I. N. Bankman, éditeur. *Handbook of Medical Imaging*. Acadamic Press, 2000.

[6] X. L. Battle, G. S. Cunningham et K. M. Hanson : Tomographic reconstruction using 3D deformable models. *Physics in Medicine and Biology*, 43 :983–990, 1998.

[7] E. B. Becker, G. F. Carey et J. T. Oden : *Finite Elements : an introduction*, volume I. Pretice Hall, 1981.

[8] C. Berge : Théorie des graphes et ses applications. *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik*, 40(5-6) :281–281, 1960.

[9] S. Beucher et D. Lantuejoul : Use of watershed in coutour detection. *In Proceedings International Workshop on Image Processing : Real-time Edge and Motion detection/estimation*, pages 1–12, 1979.

[10] J. A. Bondy et U.S.R. Murty : *Graph Theory*. Springer, 2008.

[11] Y. Boykov et M.-P.Jolly : Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images. *In Proceedings of the International Conference on Computer Vision*, volume 1, pages 105–112, 2001.

[12] J. G. Branvok, Y. Yang et M. N. Wernick : Tomographic image reconstruction based on a content-adaptive mesh model. *IEEE Transaction on Medical Imaging*, 23(2) :202–212, 2004.

[13] X. Bresson : A fast global minimization algorithm for active contour models based on the split-bregman method. http://www.cs.cityu.edu.hk/~xbresson/codes.html, 2009.

[14] X. Bresson, S. Esedoglu, P. Vandergheynst, J-P. Thiran et S. Osher : Fast global minimization of the active contour/snake model. *Journal of Mathematical Imaging and Vision*, 28(2) :151–167, 2007.

[15] T. M. Buzug : *Computed Tomography*. Springer, 2008.

[16] E. Candès : *Ridgelets*. Thèse de doctorat, Stanford University, 1998.

[17] J. Canny : A computational approach to edge detection. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 8(6) :679–697, 1986.

[18] V. Caselles, F. Catté, T. Coll et F. Dibos : A geometric model for active contours in image processing. *Numerische Mathematik*, 66(1) :1–31, 1993.

[19] V. Caselles, R. Kimmel et G. Sapiro : Geodesic active contours. *International Journal of Computer Vision*, 22(1) :61–79, 1997.

[20] CGAL : Cgal, Computational Geometry Algorithms Library. http://www.cgal.org, 2012.

[21] T. F. Chan, S. Esedoglu et M. Nikolova : Algorithms for finding global minimizers of image segmentation and denoising models. *Journal of Applied Mathemathics*, 66(5) :1632–1648, 2006.

[22] T. F. Chan et L. A. Vese : Active contours without edges. *IEEE Transaction on Image Processing*, 10(2) :266–277, 2001.

[23] J. Chen, J. Cong, M. Yan et Y. Zou : Fpga-accelerated 3D reconstruction using compressive sensing. *In Proceedings of the ACM/SIGDA international symposium on Field Programmable Gate Arrays*, pages 163–166. ACM, 2012.

[24] J.W. Chi, M. Brandy et N. Moore : Segmentation of the bladder wall using coupled level set methods. *In Proceedings of the International Symposium on Biomedical Imaging*, pages 1653–1656, 2011.

[25] C.-Y. Chou, Y.-Y. Chuo, Y. Hung et W. Wang : A fast forward projection using multithreads for multirays on GPUs in medical image reconstruction. *Medical Physics*, 38(7) :4052–4065, 2011.

[26] CIVA : Plateforme pour le contrôle non destructif. http://www-civa.cea.fr/, 2012.

[27] L. D. Cohen : On active contour models and balloons. *Computer Vision and Image Understanding*, 53(2) :211–218, 1991.

[28] O. Commowick et G. Malandain : Atlas-based delineation of lymph node levels in head and neck computed tomography images. *Radiotherapy and Oncology*, 87(2) :281–289, 2008.

[29] J. L. Coolidge : *A Treatise on the Geometry of the Circle and Sphere*. Chelsea Publishing Company, 1971.

[30] M. Costin : *Multiresolution Image Reconstruction in X-ray Micro- and Nano- Computed Tomography : Application in Materials Non-Destructive Testing*. Thèse de doctorat, Institut National des Sciences Appliquées de Lyon, 2010.

[31] D. Cremers, M. Rousson et R. Deriche : A review of statistical approaches to level set segmentation : integrating color, texture, motion and shape. *International Journal of Computer Vision*, 72(2) :195–215, 2007.

[32] P. J. Davis : *Interpolation and Approxination*. Dover Publications, 1975.

[33] B. De Man et S. Basu : Distance-driven projection and backprojection in three dimensions. *Physics in Medicine and Biology*, 49 :2463–2475, 2004.

[34] A. P. Dempster, N. M. Laird et D. B. Rubin : Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B*, 39(1) :1–38, 1977.

[35] M. Do et M. Vetterli : *Beyond wavelets*. Academic Press, 2002.

[36] D. Donoho : Wedgelets : Nearly minimax estination of edges. *Annals of statistics*, 27 :859–897, 1999.

[37] D. Donoho : Orthonormal ridgelet and linear singularities. *Journal on Mathematical Analysis*, 31(5) :1062–1099, 2000.

[38] H. Edelsbrunner : *Algorithms in Combinatorial Geometry*. Springer-Verlag, 1987.

[39] A Entezari, M. Nilchian et M. Unser : A box spline calculus for the discretization of computed tomography reconstruction problems. *IEEE Transaction on Medical Imaging*, pages 1532–1541, 2012.

[40] C. L. Epstein et M. Gage : The curve shortening flow. *Wave motion : theory, modelling, and computation*, 7(15-59), 1987.

[41] M. Erdt, S. Steger et G. Sakas : Regmentation : a new view of image segmentation and registration. *Journal of Radiation Oncolocy Informatics*, 4(1) :1–23, 2012.

[42] L. A. Feldkamp, L. C. Davis et J. W. Kress : Practical cone-beam algorithm. *Journal of the Optical Society of America A*, 1(6) :612–619, 1984.

[43] H. Feng, D. A. Castaño et W. C. Karl. : Tomographic reconstruction using curve evolution. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 361–366, 2000.

[44] H. Feng, W. C. Karl. et D. A. Castañon : A curve evolution approach to object-based tomographic reconstruction. *IEEE Transaction on Image Processing*, 12(1) :44–57, 2003.

[45] W. T. Freeman et E. H. Adelson : The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(9) :891–906, 1991.

[46] D. Furukawa, A. Shimizu et H. Kobatake : Automatic liver segmentation method based on maximum a posterior probability estimation and level set method. *In Proceedings of the International Conference on Medical Image Computing and Computer Assisted Intervention*, pages 117–124, 2007.

[47] N. Gac, A.Vabre et A. Mohammad Djafari : Multi GPU parallelization of 3D bayesian CT algorithm and its application on real foam reonconstruction with incomplete data set. *In Forum on recent developments in Volume Reconstruction techniques applied to 3D fluid and solid mechanics*, pages 35–38, 2011.

[48] N. Gac, A.Vabre, A. Mohammad Djafari, A. Rabanal et F. Buyens : GPU implementation of a 3D bayesian CT algorithm and its application on real foam reconstruction. *In Proceedings of the 1st International Conference on Image Formation in X-Ray Computed Tomography*, 2010.

[49] D. García-Lorenzo, L. Jérémy, D. L. Arnold, D. L. Collins et C. Barillot : Multiple sclerosis lesion segmentation using an automatic multimodal graph cuts. *In Proceedings of the International Conference on Medical Image Computing and Computer Assisted Intervention*, volume 5762, pages 584–591, 2009.

[50] G. Gaullier, P. Charbonnier et F. Heitz : Introduing shape priors in object-based tomographic reconstruction. *In Proceedings of the International Conference on Image Processing*, pages 1077–1080, 2009.

[51] T. Goldstein, Xavier Bresson et Stanley Osher : Geometric application of the split Bregmant method : Segmentation and surface reconstruction. *Journal of Scientific Computing*, 45 :272–293, 2010.

[52] R. C. Gonzalez et R. E. Woods : *Digital Image Processing*. Addison-Wesley Publishing Company, 1993.

[53] D. Grosgeorge, C. Petitjean et S. Ruan : Segmentation d'images par coupe de graphe avec a priori de forme. *In Proceedings of the Conference Reconnaissance des Formes et Intelligence Artificielle*, 2012.

[54] J. Hadamard : *Lectures on Cauchy's problem in linear partial differential equations*. New Haven Yale University Press, 1932.

[55] G. T. Herman : *Image Reconstruction from Projections : The Fundamentals of Computerized Tomography*. New York : Academic Press, 1980.

[56] L. Herraiz, S. España, J. J. Vaquero, M. Desco et J. M. Udías : FIRST : Fast iterative reconstruction software for (PET) tomography. *Physics in Medicine and Biology*, 51 :4547–4565, 2006.

[57] E. Hoetzl et W.Ring : An active contour approach for a mumford-shah model in X-Ray tomography. *In Proceedings of the International Symposium on Advances in Visual Computing*, pages 1021–1030, 2009.

[58] A. Isambert, F. Dhermain, F. Bidault, O. Commowick, P.-Y. Bondiau, G. Malandain et D. Lefkopoulos : Evaluation of an atlas-based automatic segmentation software for the delineation of brain organs at risk in a radiation therapy clinical context. *Radiotherapy and Oncology*, 87(1) :93–99, 2008.

[59] T. Jirka et V. Skala : Gradient vector estimation and vertex normal computation. Rapport technique DCSE/TR-2002-08, University of West Bohemia in Piles, 2002.

[60] P. M. Joseph : An improved algorithm for reprojecting rays through pixel images. *IEEE Transaction on Medical Imaging*, MI-1(3) :192–196, 1982.

[61] M. Kachelriess et M. Knaup : Hyperfast perspective cone-beam backprojection. *In IEEE Medical Imaging Conference Record*, pages 1679–1683, 2006.

[62] M. Kachelriess, M. Knaup et O. Bockenbach : Hyperfast parallel-beam backprojection. *In IEEE Medical Imaging Conference Record*, pages 3111–3114, 2006.

[63] M. Kachelriess, M. Knaup et O. Bockenbach : Hyperfast parallel-beam and cone-beam backprojection using the Cell general purpose hardware. *Medical Physics*, 34(4) :1474–1486, 2007.

[64] S. Kaczmarz : Angenäherte auflösung von systemen linearer gleichungen. *Bulletin International de l'Académie Polonaise des Sciences et des Lettres*, 37 :355–357, 1937.

[65] A. C. Kak et M. Slaney : *Principles of Computerized Tomographic Imaging*. IEEE, 1987.

[66] M. Kass, A. Witkin et D. Terzopoulos : Snakes : Active contour models. *International Journal of Computer Vision*, 1(4) :321–331, 1988.

[67] S. Kawata et O. Nalcioglu : Constrained iterative reconstruction by the conjugate gradient method. *IEEE Transaction on Medical Imaging*, MI-4(2) :65–71, 1985.

[68] S. Kichenassamy, A. Kumar, P. Olver, A. Tannenbaum et A. Yezzi : Gradient flows and geometric active contour models. *In Proceedings of 5th International Conference on Computer Vision*, pages 810 –815, 1995.

[69] J. K. Kim, Z. Zhang et J. A. Fessler : Hardware acceleration of iterative image reconstruction for X-Ray computed tomography. *In Proceedings of the IEEE Internation Conference on Acustic, Speech and Signal Processing*, pages 1697–1700. IEEE, 2011.

[70] M. Knaup, A. Kalender et M. Kacherlriess : Statistical cone-beam CT image reconstruction using Cell broadband engine. *In Proceedings of the IEEE Medical Imaging Conference Record*, 2006.

[71] L. Kobbelt : $\sqrt{3}$-subdivision. *In Proceedings of the Conference on Computer graphics and interactive techniques*, volume 34, pages 103–112, 2000.

[72] D.-J. Kroon : http://www.mathworks.com/matlabcentral/fileexchange/authors/29180.

[73] K. Lange, M. Bahn et R. Little : A theoretical study of some maximum likelihood algorithms for emission and transmission tomography. *IEEE Transaction on Medical Imaging*, MI-6(2) :106–114, 1987.

[74] K. Lange et R. Carson : EM reconstruction algorithms for emission and trasmission tomography. *Journal of Computer Assisted Tomography*, 8(2) :306–316, 1984.

[75] S. Lankton : Active contour segmentation. http://www.mathworks.com/matlabcentral/fileexchange/authors/30634, 2012.

[76] R. M. Lewitt : Multidimensional digit image representations using generelized Kaiser-Bessel window functions. *Optical Society of America*, 7(10) :1834–1846, 1990.

[77] R. M. Lewitt : Alternative to voxels for image representation in iterative reconstruction algorithm. *Physics in Medicine and Biology*, 37(3) :705–716, 1992.

[78] R. Malladi, J. A. Sethian et B. Vemuri : Evolutionary fronts for topology independent shape modeling and recovery. *In Proceedings of the European Conference on Computer Vision*, pages 3–13, 1994.

[79] R. Malladi, J. A. Sethian et B. C. Vemuri : Shape modeling with front propagation : a level set approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(2) :158 –175, 1995.

[80] B. De Man et S. Basu : Distance-driven projection and backprojection. *In Proceeding of the IEEE Nuclear Science Symposium and Medical Imaging Conference Record*, volume 3, pages 1477–1480, 2002.

[81] D. Marr et E. Hildreth : Theory of edge detection. *Proceedings of the Royal Society of London. Series B, Biological Sciences*, 207(1167) :187–217, 1980.

[82] T. Mcinerney et D. Teropoulos : Deformable models in medical image analysis : a survey. *Medical Image Analysis*, 1(2) :91–108, 1996.

[83] F. Momey, L. Denis, C. Mennessier, E. Thiébaut, J.-M. Becker et L. Desbat : A new representation and projection model for tomography, based on separable B-spline. *In Proceeding of the IEEE Nuclear and Science Symposium ans Medical Imagaging Conference*, pages 1–8, 2011.

[84] K. Mueller, F. Xu et N. Neophytou : Why do commodily grahics hadrware boards (GPUs) work so well for acceleration of compted tomography ? *In Proceeding of SPIE Computational Imaging V*, 2007.

[85] K. Mueller et R. Yangel : Rapid 3D cone-beam reconstruction with simultaneous algebraic reconstruction technique (SART) using 2D texture mapping hardware. *IEEE Transaction on Medical Imaging*, 19(12) :1227–1237, 2000.

[86] D. Mumford et J. Shah : Optimal approximation by piecewice smooth functions and associated variational problems. *Communication on Pure and Applied Mathematics*, 42 :577–685, 1989.

[87] G. Muralidhar, A. Bokiv et J. Giese : Snakules : A model-based active contour algorithm for the annotation of spicules on mammography. *IEEE Transaction on Medical Imaging*, 29(10) :1768–1780, 2010.

[88] F Natterer : *The mathematics of computerized tomography*. SIAM, 2001.

[89] F. Natterer et F. Wübbeling : *Mathematical Methods in Image Reconstruction*. SIAM, 2001.

[90] Nvidia : Nvidia CUDA C programming guide. http://developer.download.nvidia.com, 2012.

[91] Y. Okitsu, F. Ino et K. Hanguhara : Accelerating cone beam reconstruction using the CUDA-enambe GPU. *In Proceeding of the 14th International Conference High Performance Computing*, pages 108–119. Springer, 2008.

[92] S. Osher et R. Fedkiw : *Level Set Methods and Dynamic Implicit Surfaces*. Springer, 2003.

[93] S. Osher et J. A. Sethian : Fronts propagating with curvature dependent speed : algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 79(1) :12–49, 1988.

[94] T. Pavlidis : *Structural Pattern Recognition*. Springer-Verlag, 1977.

[95] T. Pavlidis et Y.-T. Liow : Integrating region growing and edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(3) :225–233, 1990.

[96] P. Perona et J. Malik : Detecting and localizing edges composed of steps, peaks and roofs. Rapport technique, MIT-CICS, 1990.

[97] P. Perona et J. Malik : Detecting and localizing edges composed of steps, peaks and roofs. *In Proceedings on the Third International Conference on Computer Vision*, pages 52 –57, 1990.

[98] R. Ramlau et W. Ring : A Mumford-Shah level-set approach for the inversion and segmentation of X-ray tomography data. *Journal of Computational Physics*, 221(2) :539–557, 2007.

[99] D. A. Reimann, V. Chaudhary, M. J. Flynn et I. K. Sethi : Cone beam tomography using MPI on heterogeneous workstation clusters. *In Proceedings of the Second MPI Developers Conference*, pages 142–148. IEEE Computer Society, 1996.

[100] L. I. Rudin, S. Osher et E. Fatemi : Nonlinear total variation based noise removal algorithms. *Physica D*, 60(1-4) :254–268, 1992.

[101] J. C. Russ : *Image Processing Handbook*. Taylor and Francis Group, 2011.

[102] M. Schweiger, S. R. Arridge, O. Dorn, A. Zacharopoulos et V. Kolehmainen : Reconstructing absorption and diffusion shape profiles in optical tomography by a level set technique. *Optics Letters*, 31(4) :471–473, 2006.

[103] M. Schweiger, O. Dorn et S. R. Arridge : 3D shape and contrast reconstruction in optical tomography with level sets. *Journal of Physics : Conference Series*, 124(1) :1–12, 2008.

[104] M. Senasli, L. Garnero, A. Herment et E. Mousseaux : Reconstruction 3D de vaisseaux à partir d'un faible nombre de projections à l'aide de contour déformables. *In Proceedings of the Colloque Groupement de Recherche en Traitement du Signal et des Images (GRETSI)*, 1997.

[105] M. Senasli, L. Garnero, A. Herment et E. Mousseaux : 3D reconstruction of vessel lumen from very few angiograms by dynamic contours using a stochastic approach. *Graphical Models*, 62(2) :105–127, 2000.

[106] J. A. Sethian : A review of recent numerical algorithms for hypersurfaces moving curvature dependent flows. *Journal of Differential Geometry*, 31(131-161), 1989.

[107] J. A. Sethian : *Level Set Methods and Fast Marching Mathods*. Cambridge University Press, 1999.

[108] L. A. Shepp et Y. Vardi : Maximum likelihood reconstruction for emission tomography. *IEEE Transaction on Medical Imaging*, MI-1(2) :113–122, 1982.

[109] J. R. Shewchunk : Mesh generation for domains with small angles. *In Proceeding of the 16th annual symposium on Computational geometry*, pages 1–10, 2000.

[110] H. Si : Tetgen : A quality tetrahedral mesh generator and a 3D Delaunay triangulator. http://www.tetgen.org, 2009.

[111] R. L. Siddon : Fast calculation of the exact radiological path for a three-dimensional CT array. *IEEE Transaction on Medical Imaging*, 12(2) :252–255, 1985.

[112] A. Silberschatz, G. Gagne et P. B. Galvin : *Operating System Concept*. Wiley, 2005.

[113] A. Sitek, R. H. Huesman et G. T. Gullberg : Tomographic reconstruction using an adaptive tetrahedral mesh defined by a point cloud. *IEEE Transaction on Medical Imaging*, 25(9) :1172–1179, 2006.

[114] C. Soussen et A. Mohammad-Djafari : Polygonal and polyhedral contour reconstruction in computed tomography. *IEEE Transaction on Image Processing*, 13(11) :1507–1522, 2004.

[115] D. Terzopoulos, A. Witkin et M. Kass : Constraints on deformable models : recovering 3D shape and nongrid motion. *Artificial Intelligence*, 36(1) :91–123, 1988.

[116] T. Toczek : Implemention et evaluation des méthodes de traversée de tétraèdralisations sur carte graphique (GPU). Rapport technique, Gipsa-lab Grenoble-INP, 2012.

[117] L. A. Vese et T. F. Chan : A multiphase level set framework for image segmentation using the mumford and shah model. *International Journal of Computer Vision*, 50(3) :271–293, 2002.

[118] L. Vogelsang, A. Krol, D. H. Feiglin et E. Lipson : System matrix for OSEM SPECT with attenuation compensation in mesh domain. *In Proceeding SPIE Medical Imaging 2010 : Physics of Medical Imaging*, volume 7622, pages 1–7, 2010.

[119] H. Wang : *X-ray CT Image Reconstruction Methods from Few Projections*. Thèse de doctorat, Université de Grenoble, 2011.

[120] Y. Wang et O. Lee : Use of two-dimension deformable mesh structure for video coding, Part I – the synthesis problem : Mesh-based function apporximation and mapping. *IEEE Transaction on Circuits and System for Video Technology*, 6(6) :636–646, 1996.

[121] Thomas Williams, Colin Kelley et al. : Gnuplot 4.6 : an interactive plotting program. http://www.gnuplot.info/download.html, 2012.

[122] M. Wu, C. Rosano, C. S. Carter P. Lopez-Garcia et H. J. Aizenstein : Optimum template selection for atlas-based segmentation. *NeuroImage*, 64(4) :1612–1618, 2007.

[123] C. Xu et J. L. Prince : Gradient vector flow : A new external force for snakes. *In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 66–71, 1997.

[124] C. Xu et J. L. Prince. : Snakes, shapes, and gradient vector. *IEEE Transanction on Image Processing*, 7(3) :359–369, 1998.

[125] F. Xu et K. Mueller : Real-time 3D computed tomographic reconstruction using commodily graphics hardware. *Physics in Medicine and Biology*, 52(12) :3405–3419, 2007.

[126] W. Xu et K. Mueller : Accelerating regularized iterative ct reconstruction on commodity graphics hardware (GPU). *In IEEE International Symposium on Biomedical Imaging*, pages 1287 –1290, 2009.

[127] S. Yoon, A. R. Pineda et R. Fahrig : Level set reconstruction for sparce angularly sampled data. *In Proceedings of the IEEE Nuclear Science Symposium Conference Record*, pages 3420–3423. IEEE, 2006.

[128] S. Yoon, A. R. Pineda et R. Fahrig : Simultaneous segmentation and reconstruction : A level set method approach for limited view computed tomography. *Medical Physics*, 35(5) :2329–2340, 2010.

[129] G.L. Zeng et G.T. Gullberg : A ray-driven backprojector for backprojection filtering and filtered backprojection algorithms. *In Proceeding of the IEEE Nuclear Science Symposium and Medical Imaging Conference Record*, pages 1199–1201, 1993.

[130] X. Zhao, J.-J. Hu et P. Zhang : GPU-based 3d cone-beam CT image reconstruction for large data volume. *Journal of Biomedical Imaging*, 2009 :1–8, 2009.

[131] G. Zheng et S. Schumann : 3D reconstruction of a patient-specific surface model of the proximal femur from calibrated X-ray radiographs : A validation study. *Medical Physics*, 36(4) :1155–1166, 2009.