Adéquation algorithmes et architectures parallèles pour la reconstruction 3D en tomographie X

Christophe Laurent

THESE

présentée par

Christophe Laurent

pour obtenir le titre de
DOCTEUR DE L'UNIVERSITE CLAUDE BERNARD LYON 1
(Arrêté ministériel du 30.03.1992)

Spécialité : Informatique

ADÉQUATION
ALGORITHMES ET ARCHITECTURES PARALLÈLES
POUR
LA RECONSTRUCTION 3D EN TOMOGRAPHIE X

Date de soutenance : 22 **Janvier** 1996

Composition du Jury :

Yves Robert	PRESIDENT
Pierre Grangeat	RAPPORTEUR
Denis Trystram	RAPPORTEUR
Jean-Marc Chassery	EXAMINATEUR
Serge Miguet	EXAMINATEUR
Françoise Peyrin	EXAMINATEUR

Thèse préparée au sein du Laboratoire **TIMC - Institut IMAG**

UNIVERSITE CLAUDE BERNARD - LYON I -

Président de l'Université	M. le Professeur G. Fontaine
1er Vice-Président Fédération Santé	M. le Professeur Dechavanne
1er Vice-Président Fédération Sciences	M. le Professeur Y. Lemoigne
Président du Comité de coordination des Etudes Médicales	M. le Professeur P. Zech
Secrétaire Général	M. J. Flacher

Fédération Santé

UFR de Médecine Grange-Blanche	Directeur: Mme le Pr Pellet
UFR de Médecine Alexis-Carrel	Directeur: Mr le Pr Dechavanne
UFR de Médecine Lyon-Nord	Directeur: Mr le Pr Patricot
UFR de Médecine Lyon-Sud	Directeur: Mr le Pr Gerard
Institut des Sciences Biologiques et Pharmaceutiques	Directeur: Mr le Pr Collombel
UFR d'Otonlogie	Directeur: Mr le Pr Doury
Institut des Techniques de Réadaptation	Directeur: Mr le Pr Eyssette
Département de Biologie Humaine	Directeur: Mr le Pr Bryon
Département d'Innovation et de Coordination Pédagogique	Directeur: Mr le Pr Llorca

Fédération Sciences

Institut des Sciences de la Matière	Directeur: Mr le Pr Elbaz
Institut des Sciences de l'Ingénierie et du Développement Technologiques	Directeur: Mr le Pr Grellet
Institut de Chimie et Biologie	Directeur: Mme Varagnat, Mdc
Institut D'analyse des Systèmes Biologiques et Socio-Economiques	Directeur: Mr le Pr Debouzie
Institut des Sciences de la Terre, de l'Océan, de l'Atmosphère, de l'Espace et de l'Environnement	Directeur: Mr le Pr Elmi
UF des Activités Physiques et Sportives	Directeur: Mr le Pr Camy
I.U.T A	Directeur: Mr le Pr Gielly
I.U.T B	Directeur: Mr le Pr Pivot
Département du 1er Cycle Sciences	Directeur: Mr Poncet, Mdc
Département du 2ème Cycle:	
- Sciences de la Vie et de la Terre	Directeur: Mr le Pr Blanchet
- Sciences pour l'Ingénieur	Directeur: Mr le Pr Blanchet
- Sciences de l'Analyse et de la Matière	Directeur: Mr le Pr Vialle

à Pascale ...

Remerciements

Ce travail a été effectué au sein de l'équipe INFODIS du laboratoire TIMC-IMAG de Grenoble (Techniques de l'Imagerie, de la Modélisation et de la Cognition).

Je tiens à remercier Monsieur Yves Robert, Professeur à l'ENS de LYON, pour l'honneur qu'il m'a fait en acceptant de présider mon jury de thèse et pour m'avoir donné le goût à la recherche lors de mon stage de DEA au sein de son équipe.

Je remercie Monsieur Pierre Grangeat, Directeur de Recherches au LETI-CEA de Grenoble, pour s'être intéressé à mon travail et pour avoir accepté la lourde charge d'être rapporteur de ma thèse.

Je remercie Monsieur Denis Trystram, Professeur de l'INP de Grenoble, pour m'avoir permis de collaborer avec des chercheurs de son équipe. J'ai apprécié sa disponibilité pour juger ce travail de thèse en tant que rapporteur.

Merçi à Monsieur Serge Miguet, Maître de Conférences à l'École Normale Supérieure de Lyon, pour son aide amicale, ses conseils et sa confiance. Il m'a fait le grand plaisir d'accepter d'être membre de mon jury.

J'exprime ma sincère reconnaissance à Monsieur Jean-Marc CHASSERY, Directeur de Recherches au CNRS, qui a co-dirigé mon travail de recherche. J'ai particulièrement apprécié son esprit d'ouverture, ses conseils et son soutien durant ces trois années au sein de son équipe.

Je tiens à exprimer ma profonde gratitude à Madame Françoise Peyrin, Chargée de Recherches INSERM, qui a co-dirigé cette thèse. Son soutien permanent et ses compétences dans le domaine de la tomographie 3D, m'ont permis de mener ce travail à son terme. Je remercie aussi Etienne, Elsa, Amandine et Pierre pour leur accueil chaleureux.

Je remercie tous mes compagnons de galère qui ont fait régner une bonne ambiance au sein de l'équipage et que j'ai souvent retrouver au mess. Merçi au capitaine Jean-Marc, au navigateur Françoise, au chef-mécanicien Guy, aux deux grands pilotes Paulette et Nicole, au commandant Camille et son second Philippe, au vieux loups de mer Xavier, Gérard, Dominique et à tous les autres officiers... Merçi aux matelots Franck, Jean-Baptiste, Olivier, Pascal(s), Sylvain, William, aux matelotes Christèle, Dominique, Magalie, et tous les autres qui ont partagé mes virées à chaque escale ...

Enfin, un grand merçi à mes correcteurs Pascale, Colette et Georges qui m'ont soutenu pendant ces trois années de thèse.

Résumé

Le but de cette thèse est d'étudier l'adéquation des approches parallèles à l'imagerie 3D, en prenant comme problème cible de la reconstruction d'images 3D en tomographie par rayons X. L'évolution technologique des systèmes d'acquisition, du premier scanner au Morphomètre, a généré de nouvelles problématiques au niveau des méthodes de reconstruction et au niveau des besoins informatiques.

Une première partie est consacrée aux fondements de la tomographie assistée par ordinateur et à la présentation des machines parallèles. L'évolution des machines de calcul intensif, du Cray 1 au Cray T3D, permet de résoudre des problèmes de taille croissante.

Dans la deuxième partie, nous définissons la méthodologie suivie pour paralléliser les algorithmes de reconstruction. Nous identifions les opérateurs de base (projection et rétroprojection) sur lesquel est porté l'effort de parallélisation. Nous définissons deux approches de parallélisation (locale et globale). Différentes versions optimisées de ces algorithmes parallèles sont présentées prenant en compte d'une part la minimisation des temps de communications (recouvrement calcul/communication, communication sur un arbre binaire) et d'autre part la régulation de charge (partitionnement adaptatif).

Dans la troisième partie, à partir de ces opérateurs parallélisés, nous construisons trois méthodes de reconstruction parallèles: une méthode analytique (la méthode de Feldkamp) et deux méthodes algébriques (la méthode Art par blocs et méthode SIRT). Les algorithmes sont expérimentés sur différentes machines parallèles: Maspar, Réseau de stations Sun, Ferme de processeurs Alpha, Paragon d'Intel, IBM SP1 et Cray T3D. Pour assurer la portabilité sur les différentes machines, nous avons utilisé PVM. Nous évaluons nos performances et nous montrons nos résultats de reconstruction 3D à partir de données simulées. Puis, nous présentons des reconstructions 3D effectués sur le Cray T3D, à partir de données expérimentales du Morphomètre, prototype de scanner X 3D.

Mots clés: Parallélisme, Imagerie 3D, Méthodes de Reconstruction 3D, Tomographie X

Abstract

This aim of this work is to study the adequacy of parallel approaches to 3D imaging, in the particular case of 3D image reconstruction in X-Ray tomography. The technological evolution of acquisition systems, from the first scanner to the Morphometer, has generated new problems with respect to the theory of image reconstruction and computer requirements.

In a first part, we present the background on Computed Tomography and parallel computers. The evolution of Massively Parallel Processing computers, from the Cray 1 to the Cray T3D, allows to solve increasing size problems.

In a second part, we define our methodology to parallelize the reconstruction algorithms. We identify the basic operators (projection and backprojection) on which the parallel effort is made. We define a local and a global approaches to parallelize these operators. We present different optimized versions of these algorithms taking into account, on the one hand, the minimization of communication times (overlapping communications by calculations, communications on binary tree), and one the other hand, load-balancing (adaptative partition).

In a third part, we implement three parallel reconstruction methods using these operators: Feldkamp method, and two algebraic methods, Block ART and SIRT. The algorithms have been tested on different parallel computers: Maspar, Sun workstation network, Farm of Alpha processors, Intel Paragon, IBM SP1 and Cray T3D. For portability, they have been implemented using PVM. We evaluate the performances and we show the 3D reconstruction results from simulated data. Then, we present 3D reconstructed images performed on the Cray T3D, when using data from a prototype of 3D X-Ray scanner (Morphometer).

Key words: Parallelism, 3D Imaging, 3D reconstruction methods, X-Ray Tomography

Table des matières

Table des figures

Chapitre 1

Introduction générale

L'imagerie médicale regroupe un ensemble de techniques qui permettent de visualiser l'anatomie humaine. Elle nécessite un outil informatique adapté aux différents traitements des données.

La tomographie par rayons X est un procédé d'imagerie qui permet d'obtenir des coupes de l'anatomie humaine. La tomographie assistée par ordinateur (TAO) est issue du mariage entre la tomographie et l'informatique. Elle se compose d'un système d'acquisition (source de rayons X, détecteurs) et d'un système de traitement des données acquises. Le premier prototype, le scanner X, a été développé par l'équipe de Hounsfield, au début des années 70.

L'évolution technologique des procédés d'imagerie, a engendré de nouvelles problématiques. Ainsi, les systèmes d'acquisition actuels délivrent des informations, qui sont traitées par un outil informatique, dans le but d'obtenir une image bidimensionnelle représentant le phénomène observé. Les méthodes de reconstruction, mises en œuvre lors du traitement de l'information, se sont adaptées à ces systèmes d'acquisition.

L'information bidimensionnelle résultante de ces techniques d'imagerie s'est avérée insuffisante pour l'examen de nombreuses pathologies. La tomographie tridimensionnelle est donc apparue nécessaire pour acquérir une information plus complète. Les systèmes, qui ont été développés, sont basés sur un système multi-coupes qui reconstruit un volume tridimensionnel en concaténant une série de coupes. En raison des conditions répétitives d'acquisitions et de l'échantillonnage non homogène des

données, ces systèmes génèrent de nombreuses erreurs qui doivent être corrigées lors du traitement de l'information tridimensionnelle. De nouveaux systèmes permettant l'acquisition quasi-simultanée de l'information ont donc été proposés: le DSR de la Mayo-Clinic et le Morphomètre développé par GE et le LETI-CEA. On parle ici de tomographie réellement tridimensionnelle.

Nous retracerons, dans le chapitre 2, l'évolution de la tomographie bidimensionnelle (scanner X) à la tomographie tridimensionnelle. Afin d'expliciter les méthodes de reconstruction tridimensionnelle, nous introduirons les méthodes de reconstruction bidimensionnelle. Puis, nous énumérerons les principales méthodes qui ont été proposées dans la littérature.

Le développement de la tomographie tridimensionnelle a été possible grâce aux progrès de l'outil informatique. Toutefois, les méthodes de reconstruction, qui en sont issues, requièrent un espace mémoire de grande taille et une bonne puissance de calcul, pour reconstruire un volume tridimensionnel de taille réaliste dans des temps acceptables. Suivant la taille du problème, on peut les implémenter sur des machines classiques. En raison de la faible capacité mémoire et de la puissance limitée de ce type machine, il est préférable d'utiliser des machines conçues pour le calcul intensif.

Il existe plusieurs classes de machines de calcul intensif. A l'heure actuelle, les plus performantes sont les machines parallèles. Nous introduirons, dans le chapitre 3, les notions liées au parallélisme. Nous établirons une classification des machines parallèles suivant leur modèle d'exécution. Celui-ci est implicitement relié au modèle de programmation qui sera abordé par la suite. Pour faciliter la programmation de ces machines parallèles, des bibliothèques spécifiques ont été définies. Ces bibliothèques sont dédiées soit aux calculs, soit aux communications. Nous présenterons le principe d'une bibliothèque de communication: PVM. D'autres outils ont été développés pour ces machines (langage dédié, analyseur de performances...), mais leur emploi restent assez limité. Pour justifier le choix d'une implémentation parallèle, il est classique de mesurer les performances. Nous utiliserons des critères pertinents pour évaluer les performances de nos implémentations.

De nombreux travaux ont mis en évidence l'adéquation des problèmes de reconstruction tridimensionnelle, issus de la tomographie par rayons X, aux approches parallèles. En effet, au vu des besoins informatiques nécessaires, il semble naturel de penser que les machines parallèles, par les capacités qu'elles offrent, soient le mieux adaptées pour résoudre ces problèmes.

Nous proposerons, dans le chapitre 4, une méthodologie générale pour paralléliser les méthodes de reconstruction. Deux démarches sont possibles pour paralléliser les méthodes de reconstruction. La première, l'analyse ascendante, consiste à les définir comme un problème classique d'algèbre linéaire. La parallélisation est alors basée sur l'utilisation de routines parallèles d'algèbre linéaire. La seconde que nous suivrons, l'analyse descendante, permet de conserver la structure des méthodes de reconstruction basée sur des opérateurs. La parallélisation intervient dans ce cas au niveau de l'implémentation des opérateurs.

Une première étape consistera à fixer la géométrie d'acquisition, et à identifier les opérateurs de base comme étant la projection et la rétroprojection. Puis ces opérateurs seront parallélisés suivant deux approches: l'approche locale et l'approche globale. L'implémentation de ces opérateurs parallèles utilisera un concept de machine parallèle abstraite, afin de s'assurer une bonne portabilité et pour ne pas tenir compte de l'architecture des machines cibles. Une seconde étape proposera des versions optimisées de ces algorithmes parallèles, qui prendront en compte d'une part la minimisation des temps de communications, et d'autre part la régulation de charge. L'ensemble de ces opérateurs parallèles formera alors une bibliothèque qui permettra de constituer des méthodes de reconstruction parallèles.

Notre but étant de paralléliser des méthodes de reconstruction, nous étudierons la parallélisation d'une méthode analytique, la méthode de Feldkamp dans le chapitre 5. Nous rappellerons son principe et sa formulation discrète. Sa parallélisation suivra une approche locale et sera axée sur deux types d'architectures: les machines SIMD et les machines MIMD.

Le chapitre 6 s'intéressera à la parallélisation des méthodes algébriques. Nous présenterons la méthode ART par blocs et la méthode SIRT. Elles seront parallélisées suivant deux approches (globale, locale) en raison de leur différence algorithmique. Au vu des résultats de la méthode de Feldkamp, les méthodes ART et SIRT seront implémentées sur des machines MIMD.

Pour chacune de ces méthodes, nous proposerons des versions optimisées de leurs implémentations. Nous évaluerons leurs performances après de nombreuses expérimentations sur nos machines cibles: Maspar de DEC, Réseau de stations Sun, Ferme de processeurs Alpha, Paragon d'Intel, IBM SP1 et Cray T3D.

Afin de valider nos méthodes parallèles de reconstruction, nous montrerons nos résultats d'images tridimensionnelles reconstruites à partir de données simulées. Les résultats provenant de données expérimentales, seront présentés dans le chapitre 7.

Enfin, nous conclurons par une synthèse sur l'apport du parallélisme aux méthodes de reconstruction et nous proposerons de nouvelles perspectives.

Chapitre 2

Tomographie 3D

2.1 Introduction

Nous introduirons dans ce chapitre les méthodes de reconstruction utilisées en tomographie. La tomographie est une technique employée dans le domaine médical, dans le contrôle non destructif industriel et même en astronomie. Grâce à un système d'acquisition approprié, elle recueille des informations provenant d'une source (rayonnement X, étoile ...) sur un ensemble de détecteurs. A partir de ces informations , la tomographie permet de reconstruire une tranche d'un objet, ou l'objet lui-même , qui se trouve entre la source et les détecteurs. Nous nous proposons de définir son principe dans le cadre d'applications médicales. Les méthodes de reconstruction qui permettent de reconstruire une partie ou l'objet entièrement, ont été développées en fonction du système d'acquisition mis en œuvre.
Nous retraçerons dans ce chapitre l'évolution de ces systèmes d'acquisition qui ont permis de reconstruire des signaux bidimensionnels, ou images 2D, puis des signaux tridimensionnels, ou images 3D. Nous expliciterons les méthodes de reconstruction en tomographie bidimensionnelle dans un premier temps, car elles ont servi à définir celles utilisées en tomographie tridimensionnelle. En effet, nous montrerons l'évolution des outils mathématiques qui sont la base de ces méthodes.

2.2 Origine de la tomographie 3D

2.2.1 Principe de la tomographie

La tomographie est un procédé d'imagerie médicale non invasive qui permet de visualiser une coupe d'organe, un système nerveux ou une partie du squelette d'un patient sans intervention de nature chirurgicale. L'utilisation de ces données permet au médecin de renforcer un diagnostic ou de vérifier le résultat d'une opération. Le développement de ces procédés d'imagerie est lié au développement de l'informatique. En effet, bien que le problème mathématique sous-jacent ait été résolu par Radon en 1917 [Rad17], l'essor de ces techniques est dû à l'apparition de la tomographie assistée par ordinateur ou TAO. Elle est plus communément appelée "Computerized Tomography" ou CT. La CT est basée sur deux systèmes complémentaires:

- le système d'acquisition: il est composé d'une source de rayonnement et d'un détecteur. Il fournit les données 1D (respectivement 2D) pour la reconstruction de l'image 2D (respectivement 3D). Sa géométrie définit les paramètres de la reconstruction.

- le système de traitement: à partir des données, ce système va reconstruire l'image 2D ou 3D en utilisant des méthodes de reconstruction adaptées à la géométrie d'acquisition.

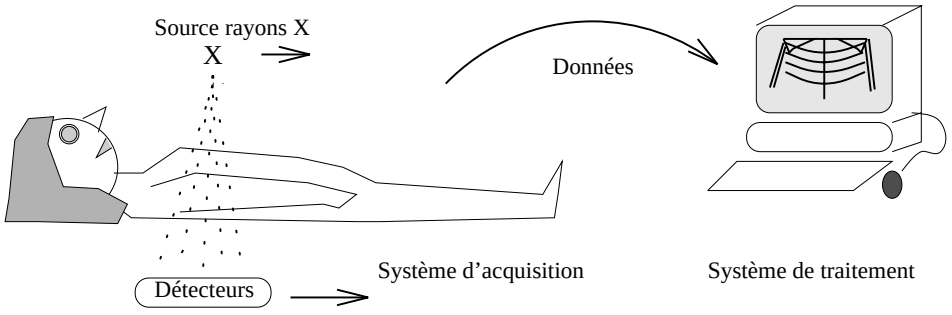


FIG. 1 - *Principe de la Tomographie assistée par ordinateur composée d'un système d'acquisition qui envoie les données au système de traitement*

La figure 1 illustre le principe de la CT. Nous nous proposons de décrire l'évolution des systèmes d'acquisition qui ont généré de nouvelles problématiques au niveau des méthodes de reconstruction. Nous rappelons tout d'abord quelques notions sur la physique du rayonnement.

2.2.2 Physique du rayonnement X

La physique du rayonnement détermine la nature des informations recueillies par les détecteurs du système d'acquisition. Nous considérons le cas de la tomographie par rayons X. Celle-ci utilise le rayonnement X, c'est-à-dire un flux de photons émis par une source X qui traverse des tissus humains, des os, des liquides intra-cellulaires ... . Chaque matière traversée atténue le flux de photons en fonction de sa nature et de son épaisseur. Ainsi, pour chaque matière traversée, nous mesurons une différence d'intensité du flux de photons entre l'entrée et la sortie de la matière qui est exprimée par la loi de Beer-Lambert pour un milieu homogène:

$$I_{\text{sortie}} = I_{\text{entrée}}.exp(-\mu.\text{épaisseur}) \tag{2.1}$$

où μ est le coefficient d'atténuation linéaire de la matière traversée.

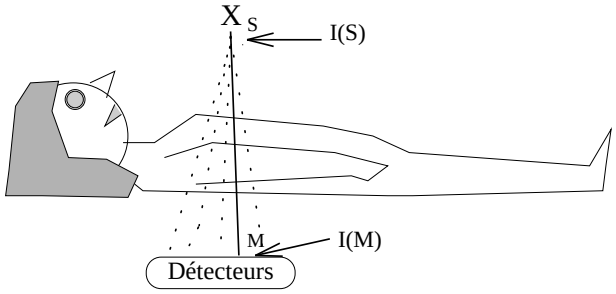


FIG. 2 - *Modélisation de l'atténuation*

Ce modèle n'est pas adapté lorsque le rayonnement traverse une partie du corps. Dans ce cas, on peut considérer que le coefficient d'atténuation μ est une fonction $\mu(l)$ définie en chaque point l de la droite (SM) où S est la source de rayonnement et M un point de détection (figure 2). La valeur de l'intensité observée au point M s'exprime par:

$$I(M) = I_0(S).exp(-\int_S^M \mu(l)dl) - \int_S^M \mu(l)dl = -ln(\frac{I(M)}{I_0(S)}) \tag{2.2}$$

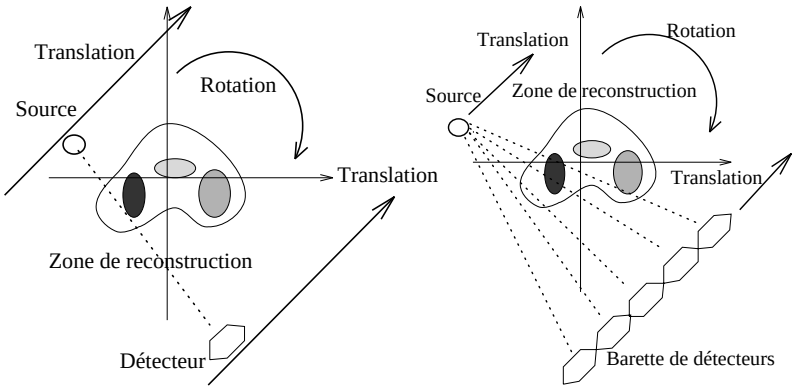
où $I_0(S)$ est la valeur de l'intensité initiale au point S. Ainsi la valeur de l'atténuation est proportionnelle au rapport entre la mesure obtenue et la valeur initiale. L'intégrale de la fonction $\mu(l)$, entre le point S et le point M, est déterminée en fonction de l'intensité du flux initial et l'intensité du flux final.

2.2.3 Evolution des systèmes d'acquisition

L'évolution technologique des systèmes d'acquisition a généré de nouveaux problèmes théoriques au niveau du choix du modèle mathématique. En effet, ceux-ci ont dû s'adapter aux géométries bidimensionnelles (2D), puis aux géométries tridimensionnelles (3D).

Système translation-rotation

Ces systèmes d'acquisition sont issus du premier prototype de scannerX 2D proposé par Hounsfield [Hou72]. Ils sont utilisés pour la tomographie 2D.



(a) Système première génération

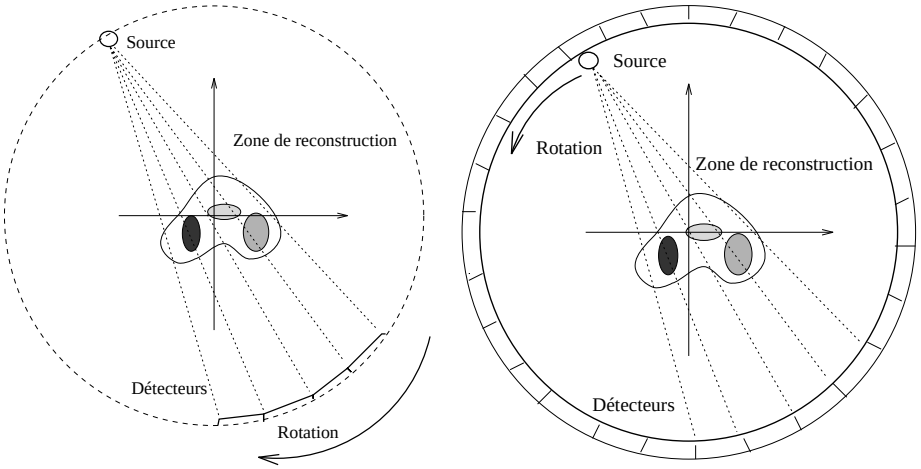
(b) Système deuxième génération

FIG. 3 - *Systèmes Translation-rotation*

Ces systèmes sont constitués d'une source de rayons X et d'un détecteur (figure 3(a)). L'ensemble source-détecteur est translaté pour balayer la section à reconstruire et obtenir une projection 1D. Puis l'ensemble effectue une rotation et le processus est alors réitéré pour une nouvelle projection. Pour accélérer les temps d'acquisition, les systèmes de deuxième génération (figure 3(b)) ont utilisé une barrette de 8 à 30 détecteurs, ce qui minimise le nombre de translations. Le faisceau, avec un angle d'ouverture assez faible, irradie cette barrette. La source et la barrette de détecteurs sont translatées simultanément, puis effectuent ensuite une rotation pour une nouvelle acquisition.

Système Eventail

Ces systèmes ont été développés pour éviter les translations des systèmes précédents. Le faisceau de rayons X utilisé est appelé faisceau divergent, "Eventail" ou " Fan-Beam". Son angle d'ouverture est compris entre 30 et 60 degrés.



(a) Système troisième génération (b) Système quatrième génération

FIG. 4 - *Systèmes Eventail*

Les détecteurs sont circulaires et ils sont, soit mobiles pour la troisième génération, soit fixes pour la quatrième génération (figure 4). Chaque détecteur représente un angle constant de détection. Le nombre de détecteurs varie d'un millier, pour les détecteurs mobiles, à une dizaine de milliers, pour les détecteurs fixes, permettant ainsi d'obtenir des projections de bonne résolution. La rapidité d'acquisition permet d'obtenir un nombre élevé de projections. On utilise le principe des systèmes de quatrième génération dans les scanners actuels.

Système Conique

Ces nouveaux systèmes ont été conçus pour la reconstruction 3D. Leur principe est d'acquérir un ensemble de projections 2D et de reconstruire une image 3D. Les sources de rayons X émettent un faisceau conique, d'où le nom de "Cone-Beam", qui intersecte un détecteur plan.

Le premier système, utilisant des rayons X est le DSR, "Dynamic Spatial Reconstruction", de la Mayo Clinic (USA). Son principe est explicité par Wood [Woo79]. Il est composé de 14 tubes de sources de rayons X associés à 14 systèmes vidéo de taille 30x30 cm. Cet ensemble tourne autour d'un patient à très grande vitesse ce qui permet un grand nombre d'acquisitions (figure 5).

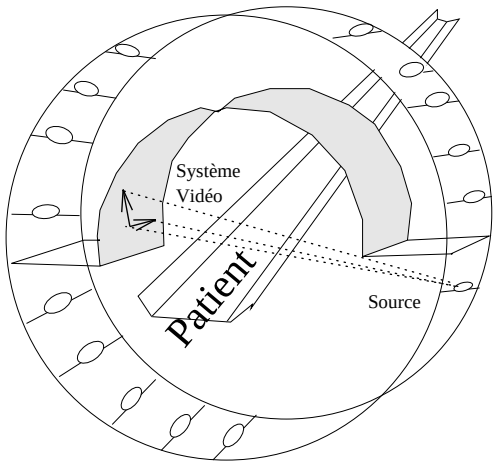


FIG. 5 - *Principe du DSR*

Dans notre chapitre 7 sur l'exploitation de données expérimentales, nous détaillerons un nouveau prototype, le "Morphomètre", qui a été développé dans une collaboration (GE-MS/Leti) et qui est basé sur la rotation rapide de deux ensembles source-détecteurs.

2.2.4 Les autres sources de rayonnement

La tomographie par rayons X est aussi appelée "tomographie par transmission de rayons X". Son utilisation principale est la description anatomique où la résolution spatiale et la densité des tissus traversés sont importantes.
Un autre type de tomographie est la "tomographie par émission", est utilisée pour visualiser les flots physiologiques ou chimiques dans un organe précis. Pour cela, une substance radioactive est injectée et émet un rayonnement au cours de sa diffusion. On mesure alors l'intensité du rayonnement par deux types de capteurs distincts en fonction de la nature des particules émises. Ces mesures correspondent aux projections de la tomographie par rayons X. Les algorithmes utilisés pour la reconstruction

d'images 2D et 3D sont similaires à ceux développés pour la tomographie par transmission aux phénomènes d'atténuation près. De nombreux systèmes d'acquisition ont été développés en fonction du type de rayonnement. On se reportera aux travaux de Budinger qui décrit les principes de la théorie de la tomographie par émission [BGH79]. Nous présentons la tomographie par émission d'un simple photon (TESP), et la tomographie par émission de Positrons (TEP).

La tomographie par émission d'un simple photon TESP

Elle utilise les émissions de photons simples: "Tomographie d'Emission d'un Simple Photon". Un détecteur muni de collimateurs recueille les photons émis dans une certaine direction. En multipliant les détecteurs, on obtient un profil 1D pour des détecteurs linéaires, ou un profil 2D, pour des détecteurs répartis sur un plan. Les images obtenues sont comparables aux images de projection. Le problème de cette tomographie est le manque de précision des mesures dû à la dispersion des photons et à leur atténuation.

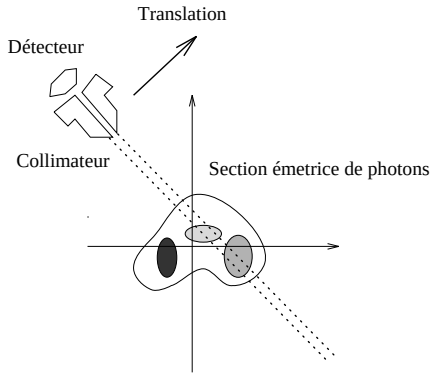


FIG. 6 - *Principe de la tomographie TESP*

La tomographie par émission de positons TEP

Elle utilise les émissions de positons β^+: "Tomographie d'Emission de Positons". L'émission des positons provient des isotopes qui ont une déficience en neutrons relativement au nombre de protons. Un proton est converti en neutron quand il perd sa charge positive qui est alors transportée par un positon de même masse qu'un électron mais de signe opposé. Quand le positon (électron positif) est éjecté du nucléon, il se combine avec un électron. Cette combinaison de matière et d'antimatière provoque une annihilation de l'électron et du positon qui sont convertis en

une énergie de 1022 kev. Cette énergie est répartie entre deux photons, de 511 kev chacun, qui sont projetés dans des directions opposées pour atteindre les détecteurs. Un test de coïncidence sur les détecteurs permet de vérifier la présence d'un positon sur la ligne entre les deux points des détecteurs. Le nombre de photons ayant atteint un détecteur détermine la concentration du mélange sur cette ligne. On obtient des images d'émission qui sont utilisées par le système de traitement de façon similaire à la tomographie de transmission.

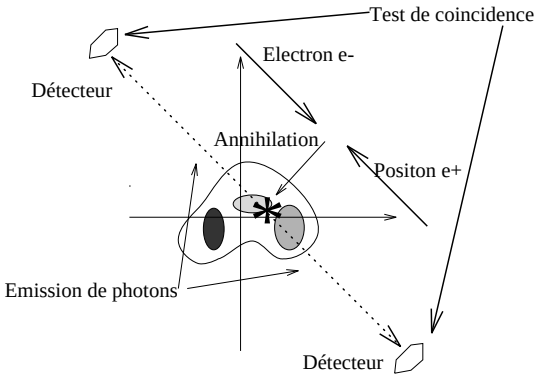


FIG. 7 - *Principe de la tomographie TEP*

2.3 Méthodes de reconstruction en tomographie 2D

Pour poser le problème de la reconstruction, nous étudions les méthodes de reconstruction en tomographie 2D, et nous nous basons sur deux articles qui font référence:

- Une synthèse des méthodes analytiques ou "Transform Methods" a été présentée par Lewitt [Lew83]. Dans cet article, il donne les principes de base des méthodes analytiques et fait référence à la plupart des méthodes utilisées en tomographie. Nous nous baserons sur ses notations pour introduire les méthodes analytiques.

- Une synthèse des méthodes algébriques ou " Finite Serie-Expansion Reconstruction Methods " a été présentée par Censor [Cen83]. Il explicite la théorie des méthodes algébriques ainsi que les algorithmes utilisés. Nous introduirons les méthodes algébriques, en utilisant la même démarche.

2.3.1 Méthodes analytiques de reconstruction

Nous présentons les principes des méthodes analytiques. Nous utilisons, tout d'abord, une géométrie d'acquisition relativement simple: la géométrie parallèle. Puis nous aborderons une géométrie d'acquisition plus complexe: la géométrie divergente.

La projection en géométrie parallèle

La tomographie assistée par ordinateur a pour objectif de former une image 2D de la distribution du paramètre étudié, dans un espace donné, à partir d'un nombre fini de mesures.
Soit une fonction f représentant la valeur de cette distribution en chaque point de l'espace étudié. La fonction f est supposée à support borné $\mathbf{D}$ et infiniment différentiable. Nous prenons pour $\mathbf{D}$, le disque unitaire pour illustrer notre définition (voir figure 8).

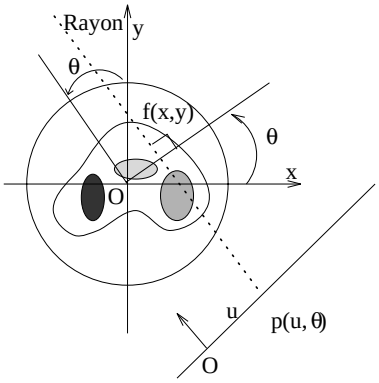


FIG. 8 - *Rayon de projection parallèle*

Une droite dans ce plan est définie par deux paramètres: u sa distance à l'origine et θ l'angle par rapport à l'axe (Oy). On définit $p(u,\theta)$ la fonction à deux variables qui associe à chaque couple (u,θ) l'intégrale de la fonction f sur la droite repérée par u et θ. Cette fonction, pour u et θ fixés, est plus communément désignée comme étant un rayon de projection parallèle. Il peut être exprimé par:

$$p(u,\theta) = \int_{-T(u)}^{+T(u)} f(u\cos\theta - v\sin\theta, u\sin\theta + v\cos\theta)dv \qquad (2.3)$$

Les limites d'intégration sont fonction de u, de θ et du domaine d'intégration. Pour notre exemple, le disque unitaire, elles s'expriment par:

$$T(u) = (1-u^2)^{1/2}, \qquad |u| \leq 1 \tag{2.4}$$

$$p(u,\theta) = 0, \qquad |u| > 1 \tag{2.5}$$

En généralisant notre domaine à $\mathbb{R}$ et en supposant que f est une fonction à support borné, l'équation 2.3 devient:

$$p(u,\theta) = \int_{-\infty}^{+\infty} f(u\cos\theta - v\sin\theta, u\sin\theta + v\cos\theta)dv \tag{2.6}$$

L'ensemble des projections $\{p(u,\theta)/\theta \in [0,\pi[, u \in \mathbb{R}\}$ est donc fonction de f, représentant l'image à reconstruire. Le problème de la reconstruction revient à exprimer de manière analytique la formule d'inversion reliant f à ses projections $p(u,\theta)$, notées aussi $p_\theta(u)$.

La transformée de Radon

Nous rappelons ici quelques notions mathématiques pour expliciter les transformations analytiques. La transformée de Fourier d'une fonction $g(x)$ notée $\mathcal{F}g(X)$ est donnée par la relation:

$$\mathcal{F}g(X) = \int_{-\infty}^{+\infty} g(x)\exp(-2\pi i x X)dx \tag{2.7}$$

La transformée de Fourier inverse, notée $\mathcal{F}^{-1}g(x)$ est définie par:

$$\mathcal{F}^{-1}g(x) = \int_{-\infty}^{+\infty} g(X)\exp(2\pi i x X)dX \tag{2.8}$$

La transformation de Fourier vérifie les égalités suivantes, pour certaines classes de fonctions:

$$\mathcal{F}(\mathcal{F}^{-1}g) = \mathcal{F}^{-1}(\mathcal{F}g) = g \tag{2.9}$$

La transformée de Fourier bidimensionnelle, notée $\mathcal{F}_2 g(X,Y)$, d'une fonction bidimensionnelle $g(x,y)$, et sa transformée bidimensionnelle inverse, notée $\mathcal{F}_2^{-1}g(x,y)$, sont définies par:

$$\mathcal{F}_2 g(X,Y) = \int\int g(x,y)\exp(-2\pi i(xX+yY)dxdy \tag{2.10}$$

$$\mathcal{F}_2^{-1} g(x,y) = \int\int g(X,Y)\exp(2\pi i(xX+yY)dXdY \tag{2.11}$$

On définit aussi pour une fonction bidimensionnelle, $h(u,\theta)$, sa transformée de Fourier par rapport à une variable $\mathcal{F}_\theta h(U,\theta)$ lorsque θ est constant et son inverse $\mathcal{F}_\theta^{-1} h(u,\theta)$:

$$\mathcal{F}_\theta h(U,\theta) = \int_{-\infty}^{+\infty} h(u,\theta)\exp(-2\pi i u U)du \tag{2.12}$$

$$\mathcal{F}_\theta^{-1} h(u,\theta) = \int_{-\infty}^{+\infty} h(U,\theta)\exp(2\pi i u U)dU \tag{2.13}$$

La transformation de Radon d'une fonction $f(x,y)$ est définie par :

$$\mathcal{R}f(u,\theta) = \int_{-\infty}^{+\infty} f(u\cos\theta - s\sin\theta, u\sin\theta + s\cos\theta)ds \tag{2.14}$$

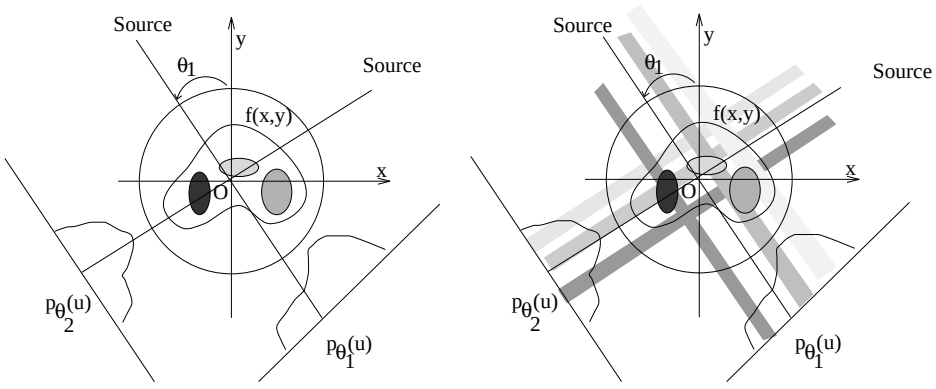
Cet opérateur exprime la relation entre les projections $p_\theta(u)$ et la fonction f à reconstruire :

$$\mathcal{R}f(u,\theta) = p_\theta(u) \tag{2.15}$$

Parallèlement on définit l'opérateur adjoint de la transformation de Radon, appelé opérateur de rétroprojection, noté $\mathcal{B}$. Il associe à chaque point du plan (x,y), la valeur de la rétroprojection de toutes les projections en ce point:

$$\mathcal{B}p(x,y) = \int_0^{\pi} p_\theta(x\cos\theta + y\sin\theta)d\theta \tag{2.16}$$

On illustre cet opérateur par la figure 9 qui représente la projection de la fonction $f(x,y)$ pour deux positions de la source (figure 9(a)), et la rétroprojection de deux projections p_{Θ_1} et p_{Θ_2} (figure 9(b)).



(a) Opération de projection (b) Opération de rétroprojection

FIG. 9 - *Opérations de base en géométrie parallèle*

Théorème de la projection

Le théorème de la projection établit une relation entre la transformée de Fourier bidimensionnelle de la fonction $f(u,\theta)$ et la transformée de Fourier de sa transformée de Radon par rapport à u:

$$\mathcal{F}_\theta \mathcal{R} f(U,\theta) = \mathcal{F}_{\mathbf{2}} f(U\cos\theta, U\sin\theta) \tag{2.17}$$

qui peut encore s'écrire:

$$\mathcal{F}_\theta p_\theta(U) = \mathcal{F}_{\mathbf{2}} f(U\cos\theta, U\sin\theta) \tag{2.18}$$

Formules d'inversion

Les méthodes de reconstruction utilisant l'approche analytique sont toutes issues d'une démarche similaire:

- Le modèle mathématique se base sur la physique du problème. La fonction f à reconstruire et les projections p_θ sont définies comme des fonctions continues à valeurs réelles,
- La fonction f à reconstruire est exprimée à partir d'une formule d'inversion utilisant les projections p_θ,
- Les formules d'inversion sont adaptées dans le cas discret pour les appliquer aux données simulées et expérimentales. La discrétisation de ces formules d'inversion permet de les traduire par des algorithmes qui reconstruisent, en fait, une approximation du modèle mathématique.

En géométrie parallèle, les formules de reconstruction sont basées sur les opérateurs définis dans le paragraphe précédent. Nous énoncerons pour chaque formule d'inversion, son principe ainsi que les problèmes liés à la discrétisation:

Formule d'inversion directe: elle est déduite directement du théorème de la projection. La transformée de Fourier de f est reconstruite à partir des transformées de Fourier des projections. La fonction f se déduit alors par inversion de sa transformée de Fourier:

$$\mathcal{F}_2 f(U,\theta) = \mathcal{F}_\theta p_\theta(u) \quad (2.19)$$

$$f = \mathcal{F}_2^{-1}\mathcal{F}_2 f \quad (2.20)$$

Le principal problème de cette méthode est d'estimer la transformée de Fourier discrète à partir des transformées de Fourier des projections [Pey90].

Formule d'inversion basée sur le théorème de la rétroprojection: ce théorème établit que la rétroprojection de toutes les projections s'exprime à l'aide d'une convolution bidimensionnelle ($*_2$) de l'image. Il s'exprime par:

$$\mathcal{B}p(x,y) = (f *_2 h)(x,y) \quad (2.21)$$

$$h(x,y) = (x^2+y^2)^{-1/2} \quad (2.22)$$

La discrétisation de cette méthode utilise les opérateurs discrets de rétroprojection et de déconvolution [Pey82].

Formule d'inversion par rétroprojection filtrée: cette formule est basée sur les deux formules précédentes [SL74]. La reconstruction de l'image est basée sur la rétroprojection des projections filtrées. Cette formule s'exprime de la manière suivante:

$$f = \mathcal{B}\tilde{p} \quad (2.23)$$

$$\tilde{p}_\theta = \mathcal{F}^{-1}(\mathcal{F}p_\theta(R).|R|) \text{ où } \Theta \in [0,\pi] \quad (2.24)$$

Dans un premier temps, les projections sont filtrées dans le domaine de Fourier avec un "bon filtre", puis elles sont rétroprojetées. Le choix du "bon filtre" détermine la qualité de la reconstruction.
Nous aborderons ce problème lors de l'implémentation de notre méthode analytique. Cette reconstruction utilise les opérateurs discrets de rétroprojection et de transformée de Fourier monodimensionelle.

Géométrie d'acquisition divergente

La géométrie divergente découle de l'utilisation des scanners de deuxième génération à géométrie "Fan Beam". Il existe deux géométries du détecteur: planaire ou circulaire. L'expression de la projection est fonction de la position de la source $S_ß$ et de la position du rayon repérée par l'angle γ, dans le cas d'un détecteur circulaire, ou par une abscisse t, dans le cas d'un détecteur plan (figure 10).

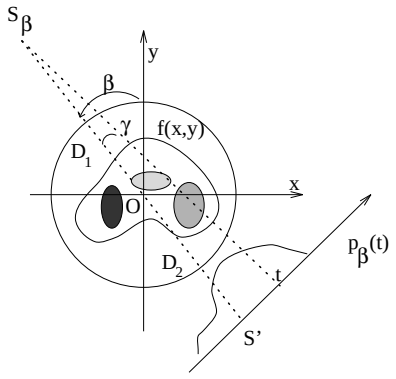


FIG. 10 - *Projection en géométrie divergente*

Les projections divergentes, $p_ß(t)$ et $p'_ß(\gamma)$, correspondent à l'intégrale de la fonction sur les rayons de projection et sont formulées à partir de la transformée de Radon, d'une manière similaire aux projections parallèles:

$$p_ß(t) = \mathcal{R}f(\frac{D_1 t}{(D^2+t^2)^{1/2}}, ß+\gamma) \quad (2.25)$$

$$p'_ß(\gamma) = \mathcal{R}f(D_1 \sin\gamma, ß+\gamma) \quad (2.26)$$

Des méthodes de reconstruction analytiques liées à cette géométrie ont été développées. De nombreux algorithmes de reconstruction par rétroprojection filtrée ont été proposés. Herman [Her80] a développé une méthode dérivée de la méthode de rétroprojection filtrée. L'algorithme obtenu est similaire à celui employé en géométrie parallèle. Son algorithme nécessite de prétraiter les projections avant la rétroprojection. Sa formulation dépend des projections $P_ß(t)$ prétraitées à partir des projections $p_ß(t)$ avec un détecteur planaire, ou des projections $P'_ß(\gamma)$ prétraitées à partir des projections $p'_ß(\gamma)$ avec un détecteur circulaire ($\tilde{P}_ß$ et $\tilde{P}'_ß$ étant les projections filtrées):

$$f(x,y) = \frac{1}{2}\int_0^{2\pi} \tilde{P}_ß(t)(\frac{D}{D_1 + x\sin ß - y\cos ß})^2 dß \quad (2.27)$$

$$f(x,y) = \frac{1}{2}\int_0^{2\pi} \tilde{P}'_ß(\gamma)\frac{D_1}{\|S_ß M\|^2} dß \quad (2.28)$$

2.3.2 Méthodes algébriques de reconstruction

Ces méthodes sont fondamentalement différentes des méthodes analytiques, car leur formulation exprime le problème de la reconstruction directement dans un espace discret.
En effet, les méthodes analytiques expriment le problème de la reconstruction dans un espace continu et leur mise en œuvre nécessite une étape de discrétisation des formules de reconstruction. Les algorithmes obtenus sont alors considérés comme de " bonnes approximations " des formules continues.

Modèle discret pour la reconstruction en projection parallèle

On discrétise l'espace de reconstruction en choisissant une base constituée d'un ensemble de fonctions de base $\{g_i / 1 \leq i \leq n\}$. La fonction à reconstruire f est alors exprimée comme une combinaison linéaire des fonctions de base:

$$f(x,y) = \sum_{i=1}^{n} f_i g_i(x,y) \tag{2.29}$$

Nous donnons l'expression des méthodes algébriques où l'espace de reconstruction 2D est discrétisé sur une grille d'éléments carrés, appelés pixels. Les pixels sont numérotés de 1 à n.

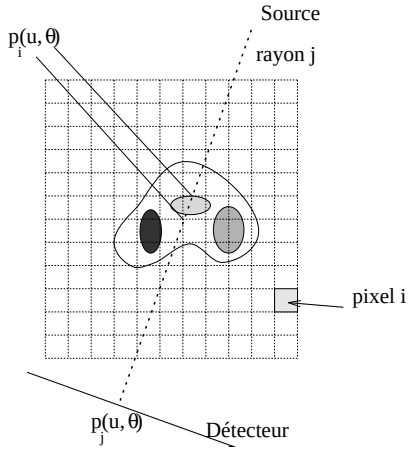


FIG. 11 - *Représentation discrète du problème*

Une projection parallèle dans cet espace discret est définie comme un ensemble de rayons parallèles traversant un nombre fini de pixels. Son expression dépend de la valeur de chaque pixel et du poids associé à chaque pixel. Ce poids est égal à la

longueur de l'intersection du rayon et du pixel traversé. La projection p_j d'un rayon j, pour u_j et θ_j fixés, se formule comme la somme des intégrales des fonctions g_i pour tous les pixels i traversés par ce rayon:

$$p_j = \sum_{i=1}^{n} f_i \int g_i(u_j \cos\theta_j - v \sin\theta_j, u_j \sin\theta_j + v \cos\theta_j) dv \tag{2.30}$$

De cette équation, on déduit le coefficient de projection R_{ji} du pixel i sur le rayon j pour u et θ fixés:

$$R_{ji} = \int g_i(u_j \cos\theta_j - v \sin\theta_j, u \sin\theta_j + v \cos\theta_j) dv \tag{2.31}$$

On peut alors reformuler l'équation:

$$p_j = \sum_{i=1}^{n} R_{ji} f_i \tag{2.32}$$

Soit R la matrice des coefficients de projection R_{ji}, appelée aussi matrice de projection, et p le vecteur des rayons de projection $\{p_j / 1 \leq j \leq m\}$ où m est égal au nombre de mesures; le système s'écrit alors:

$$p = Rf \tag{2.33}$$

On utilisera aussi la notation r_j qui définit la mesure p_j en fonction de f et de la $j$ème ligne de la matrice R:

$$p_j = r_j f \tag{2.34}$$

Cette formalisation mathématique de la projection permet d'exprimer le problème de la reconstruction comme la résolution d'un système linéaire classique. Les méthodes alors utilisées sont similaires à celles développées en analyse numérique. Le problème de reconstruction s'exprime alors de façon analogue pour des géométries d'acquisition différentes en 2D et en 3D.

La résolution d'un tel système linéaire repose sur ses spécificités:

- Le système est " très grand". Si n désigne le nombre de pixels et m désigne le nombre de rayons de projection, la taille de la matrice est égale à $n.m$. Pour obtenir des images de bonne qualité ($n = m = 512^2$), le système à résoudre est de l'ordre de 250000 équations à 250000 inconnues.

- La matrice est " très creuse". Les rayons de projection intersectent peu de pixels. On considère généralement que moins de 1% des termes de cette matrice sont non nuls.

- L'existence de la solution du système. Les acquisitions étant souvent bruitées, le système peut être inconsistant. Donc il n'admet pas de solutions. C'est pourquoi, la solution de ce système sera choisie parmi un ensemble de solutions approchées au sens d'un critère d'optimalité (moindres carrés, entropie, variance).

Pour toutes ces raisons, les méthodes algébriques développées pour résoudre ce système ne sont pas issues de méthodes d'inversion directe de la matrice R, mais de méthodes itératives qui permettent de minimiser un critère. De plus, ces méthodes ne travaillent que sur une partie de la matrice R pour éviter le problème du stockage de la matrice en mémoire.

La méthode ART

Cette méthode et ses dérivées constituent un ensemble de méthodes de reconstruction communément appelées méthodes ART pour " Algebraic Reconstruction Technics". La méthode ART est un algorithme algébrique itératif de reconstruction qui a été utilisé par Hounsfield [Hou72] pour la reconstruction sur le premier scanner EMI. Elle a été publiée pour la première fois par Gordon, Bender et Herman [GBH70], et a été identifiée à l'algorithme de Kaczmarz.

Le principe général de cet algorithme itératif est de partir d'une image initiale $f^{(0)}$ et de la corriger pas à pas pour obtenir une image satisfaisante. Dans ce processus itératif, l'image calculée à l'itération $k+1$, $f^{(k+1)}$, ne dépend que du calcul de l'image à l'itération k, $f^{(k)}$. Le calcul consiste à comparer la projection de l'image $f^{(k)}$ et le projection mesurée rayon par rayon. Les écarts entre les projections servent à corriger l'image $f^{(k)}$ pour obtenir l'image corrigée $f^{(k+1)}$.
L'algorithme initial ART se traduit donc par l'expression suivante:

$$\text{initialisation de} f^{(0)}$$
$$f_i^{(k+1)} = f_i^{(k)} + \frac{p_j - r_j f^{(k)}}{\|r_j\|^2} Rji \tag{2.35}$$

pour $i = 1...n$ et $j = k(\text{modulo } m) + 1$. Lorsque l'algorithme ART a balayé tous les rayons de projection, il a effectué un cycle. On calcule la convergence en mesurant les écarts de deux images reconstruites entre deux cycles. Pour accélérer la convergence de l'algorithme, on utilise une version de ART incluant un paramètre de relaxation $\lambda_k \in]0, 2[$. L'équation 2.35 devient alors:

$$\text{initialisation de} f^{(0)}$$
$$f_i^{(k+1)} = f_i^{(k)} + \lambda_k \frac{p_j - r_j f^{(k)}}{\|r_j\|^2} Rji \tag{2.36}$$

Exprimée sous forme matricielle, où ${}^t r_j$ correspond au vecteur r_j transposé, l'équation 2.36 devient:

$$\text{initialisation de} f^{(0)} \tag{2.37}$$
$$f^{(k+1)} = f^{(k)} + \lambda_k \frac{p_j - r_j f^{(k)}}{\|r_j\|^2} {}^t r_j \tag{2.38}$$

De nombreux algorithmes ont été proposés dans la littérature, directement déduits de l'algorithme ART. On peut citer les algorithmes SART , ART2 et MART [Her80]. Ces algorithmes calculent un rayon de projection à chaque itération ce qui correspond à une ligne de la matrice de projection R, d'où leur appellation de "Row-Action Technics".

La méthode SIRT

Les méthodes de type SIRT sont des méthodes itératives qui consistent à corriger simultanément un pixel i en utilisant tous les rayons p_j qui le traversent [P.G72]. L'algorithme qui en découle s'exprime par:

$$\text{initialisation de} f^{(0)}$$
$$f^{(k+1)} = f^{(k)} + \frac{\sum_{j=1}^{m} (p_j - r_j f^{(k)}) R_{ji}}{\sum_{j=1}^{m} \left(\sum_{l=1}^{n} R_{jl} \right) R_{ji}} \tag{2.39}$$

Ce type de méthode est surtout employé pour la reconstruction à partir de données bruitées. Dans ce cas, ces méthodes sont plus consistantes que les méthodes ART. Cependant expérimentalement, ces méthodes convergent moins vite vers une solution que les méthodes ART. On peut utiliser alors un facteur de relaxation λ_k pour accélérer la convergence. D'autre méthodes similaires ont été développées en reformulant l'équation 2.39 pour obtenir l'algorithme SMART.

2.4 Tomographie 3D

2.4.1 Apport du 3D

Les méthodes de tomographie 2D, mises au point dans le domaine médical, permettent classiquement d'obtenir des sections de l'anatomie humaine. L'information contenue dans les images 2D, obtenues par la tomographie 2D, permet de définir des critères de localité du paramètre étudié (localisation, taille, ...). Cependant, celle-ci donnant une vue partielle d'un objet 3D, l'interprétation de ces images doit tenir compte du biais lié au manque d'information. Pour pallier ce manque d'information, de nouvelles techniques ont été introduites pour obtenir des informations tridimensionnelles. Les scanners, habituellement utilisés dans nos hôpitaux, fournissent une série de coupes 2D du paramètre étudié. Par empilement de ces coupes, on obtient une image 3D. Les temps d'acquisition sont très longs, car on répète une acquisition 2D autant de fois qu'il y a de coupes. Cela entraîne des variations au niveau de l'information recherchée si, par exemple, on étudie des organes in vivo (cœur ...). De plus, la résolution obtenue pour ces images 3D n'est pas identique dans les trois dimensions en raison notamment de l'espacement des coupes. L'image 3D est reconstruite par interpolation des coupes successives, sa qualité dépend de la qualité de l'interpolation. On parlera pour cette imagerie 3D de "faux 3D " opposée à l'imagerie réellement 3D ou"Truly three-dimensional reconstruction".

La tomographie 3D consiste à reconstruire l'image 3D à partir d'acquisitions 2D prises autour du patient. Les temps d'acquisition sont plus rapides et la résolution est identique dans les trois dimensions. La rapidité d'acquisition permet des doses de rayons X et de produit de contraste moins élevées pour le patient. De tels systèmes d'acquisition ont été développés en rayons X, en PET et SPECT. Le DSR ("Dynamic Spatial Reconstruction") de la Mayo Clinic est le premier système d'acquisition réellement tridimensionnelle [RKR+80]. Dans notre chapitre sur le traitement de données expérimentales, nous décrirons le Morphomètre, un prototype développé en collaboration par GE-CGR et le LETI (CEA) [SFa92] .

Le principe de ces systèmes d'acquisition en rayons X est d'acquérir, pour chaque position de la source, une radiographie de l'objet qui corresponde à une projection 2D de l'image 3D sur un plan de détection. La série de projections 2D, ou acquisitions

2D alors obtenue, permet de reconstruire le volume par des méthodes de reconstruction. En règle générale, les méthodes de reconstruction 2D ne sont pas adaptées, et l'on doit considérer un problème de reconstruction 3D à partir des acquisitions 2D.

Dans cette section, nous rappelons le principe d'une projection 2D ainsi que les nouvelles formules des opérateurs analytiques. Nous détaillons par la suite certaines méthodes de reconstruction 3D.

2.4.2 Projection conique

Le problème de la reconstruction 3D consiste à déterminer une image 3D à partir d'un ensemble d'images 2D appelées, tantôt projections, tantôt acquisitions. La géométrie du système d'acquisition détermine le type de projection qui peut être soit divergente, soit parallèle:

- Une projection parallèle en 3D est similaire aux projections parallèles en 2D. Elle est égale aux valeurs des intégrales sur les rayons de projection parallèles entre eux.
- Une projection divergente en 3D est une projection dont la valeur est égale aux intégrales de la fonction à reconstruire sur des rayons de projection issus d'une même source . Ces rayons de projection forment un faisceau conique. On parlera alors de projection en géométrie conique.

Comme pour la tomographie 2D, on choisit f, la fonction 3D à reconstruire, comme une fonction continue, infiniment différentiable et supposée à support compact $\mathbf{D}$. Cette fonction donne la valeur en tout point M (x,y,z) de la densité du paramètre étudié. Pour déterminer la fonction f, on utilise les projections issues de la géométrie d'acquisition.

Cette géométrie est définie par la position de la source S et des plans d'acquisition ou plans de projection. Pour simplifier le modèle mathématique, on choisit comme origine du repère O le centre de la région à reconstruire. On suppose que l'axe du faisceau (OS) passe par l'origine de ce repère et que cet axe est perpendiculaire au plan de projection [Pey90].

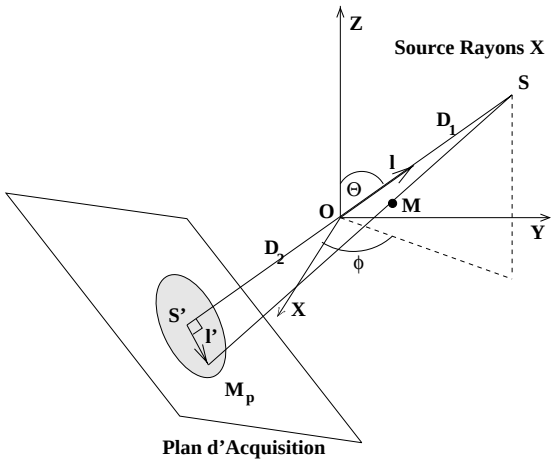


FIG. 12 - *Projection conique*

Soit le repère orthonormé $(O, \vec{i}, \vec{j}, \vec{k})$, la position de chaque point M est définie par $\vec{l} = \overrightarrow{OM} = {}^t(x, y, z)$. La position de la source conique S est déterminée par sa direction $\vec{\tau}$ sur l'axe (OS) et sa distance D_1 par rapport à l'origine: $\overrightarrow{OS} = D_1\vec{\tau}$. Le plan de projection, perpendiculaire à l'axe (OS), est à une distance D_2 de l'origine du repère. On définit par $D = D_1 + D_2$ la distance entre la source et le plan de projection. Soit S' la projection de S sur le plan de projection et M_p un point du plan de projection avec $\overrightarrow{S'M_p} = \vec{l'}$, la valeur de la projection conique, de direction $\vec{\tau}$, au point M_p est égale à l'intégrale de la fonction f sur la droite (SM_p).
Cette projection, notée $p_{\vec{\tau}}(\vec{l'})$, s'exprime par:

$$p_{\vec{\tau}}(\vec{l'}) = \int_{-\infty}^{+\infty} f\Big(\frac{D_1}{D}\vec{l'} + \lambda\frac{D\vec{\tau} - \vec{l'}}{\sqrt{D^2 + \|\vec{l'}\|^2}}\Big)d\lambda \tag{2.40}$$

2.4.3 Opérateurs analytiques en tomographie 3D

Ces opérateurs analytiques sont issus des opérateurs définis pour la tomographie 2D. Les relations établies entre ces opérateurs en 2D ne sont pas toujours vérifiées en 3D. De plus, la nature de la trajectoire de la source implique des hypothèses différentes de reconstruction. Par analogie avec le 2D, des opérateurs ont été développés dans le cas où la source décrit la surface d'une sphère [DFR79]. Sous certaines conditions, on peut formuler ces opérateurs lorsque la source décrit deux

circonférences orthogonales [Tuy83]. Pour simplifier la plupart des méthodes analytiques utilisent une trajectoire circulaire. Cela implique que les méthodes reconstruisent une approximation de la fonction f. On peut citer les opérateurs suivants:

Transformation de Radon 3D La transformée de Radon 3D est égale aux intégrales de la fonction f sur des plans de l'espace. Ces plans sont définis par une distance d à l'origine et par un vecteur unitaire $\vec{n}$ orthogonal. La transformée de Radon 3D est notée $\mathcal{R}f$ comme en 2D et s'exprime par:

$$\mathcal{R}f(d,\vec{n}) = \int\int\int f(\vec{l})\delta(\vec{l}.\vec{n} - d)d\vec{l} \tag{2.41}$$

On notera $\mathcal{R}'f$ la dérivée partielle de la transformée de Radon par rapport à d.

Opérateur de rétroprojection Cet opérateur est beaucoup utilisé par les méthodes de reconstruction analytiques. Il associe à l'ensemble des projections coniques $p_{\vec{\tau}}$ l'image 3D, notée $\mathcal{B}p$, définie par :

$$\mathcal{B}p(\vec{l}) = \int_{\Omega} p_{\vec{\tau}}\Big(D\frac{\vec{l} - (\vec{l}.\vec{\tau})\vec{\tau}}{D_1 - \vec{l}.\vec{\tau}}\Big)d\omega_{\vec{\tau}} \tag{2.42}$$

Ω étant le domaine d'intégration angulaire et $\omega_{\vec{\tau}}$ la variable d'intégration angulaire définie par la trajectoire de la source S.

2.4.4 Classification des méthodes de reconstruction 3D

Les méthodes de reconstruction 3D sont dérivées des méthodes 2D utilisant une géométrie particulière et des opérateurs 3D. Elles ont été classifiées dans le cadre d'un rapport de synthèse du GDR TDSI[1] [GP93]. Ce rapport présente quatre classes de méthodes:

- les méthodes analytiques dérivées de la discrétisation d'une formule continue d'inversion 3D,
- les méthodes algébriques qui calculent la solution discrète, par des algorithmes itératifs ou directs adaptés de la tomographie 2D,

1. Groupe de Recherche du Traitement du Signal et Images du CNRS

- les méthodes statistiques qui sont des méthodes où la fonction à reconstruire et les projections sont considérées comme des variables aléatoires. Ces méthodes permettent d'introduire des informations statistiques sur l'objet à reconstruire et sur le bruit des données,

- les méthodes structurelles recherchant des structures de l'objet à reconstruire (surface, contour, ensemble de points).

Nous présentons un aperçu succinct des méthodes les plus utilisées en tomographie 3D.

2.4.5 Méthodes analytiques 3D

Formule d'inversion de Denton

Cette formule est démontrée dans le cas où la source se déplace sur la surface d'une sphère [DFR79]. Elle exprime la fonction f comme la rétroprojection pondérée des projections corrigées et pseudo-convoluées. En géométrie divergente, elle est utilisée sous la forme suivante:

$$f(\vec{l}) = \int \Big(\int p_{\vec{\tau}}(\vec{l'}) h(\vec{l}_{\vec{\tau}}, \vec{l'}) \frac{D}{\sqrt{D^2 + \|\vec{l'}\|^2}} d\vec{l'} \Big) \frac{D^3}{8\pi^3 (D - \vec{\tau}.\vec{l'})^3} d\omega_{\vec{\tau}} \tag{2.43}$$

avec $\vec{l}_{\vec{\tau}}$ le projeté de $\vec{l}$ et pour noyau de convolution $h(\vec{l}_{\vec{\tau}}, \vec{l'})$:

$$\vec{l}_{\vec{\tau}} = \frac{D(\vec{l} - (\vec{\tau}.\vec{l})\vec{\tau})}{D - \vec{\tau}.\vec{l}} \tag{2.44}$$

$$h(\vec{l}_{\vec{\tau}}, \vec{l'}) = \frac{1}{\|\vec{l}_{\vec{\tau}} - \vec{l'} - (\vec{l'} \Lambda \vec{l}_{\vec{\tau}})/D\|^3} \tag{2.45}$$

Filtrage de la rétroprojection

Dans le même cas d'acquisition que précedemment, on peut montrer que l'image peut être obtenue par déconvolution de la rétroprojection des projections corrigées [PRA88].
Cette méthode se formule ainsi :

$$\mathcal{B}'p(\vec{l}) = \int \frac{D_1}{\|D_1\vec{\tau} - \vec{l}\|} p_{\vec{\tau}}(\vec{l}_{\vec{\tau}}) d\omega_{\vec{\tau}} \tag{2.46}$$

$$\mathcal{B}'p(\vec{l}) = (f *_2 h)(\vec{l}) \qquad \text{avec } h(\vec{l}) = 1/\|\vec{l}\|^2 \tag{2.47}$$

La formule de la reconstruction est alors donnée par:

$$f = \mathcal{F}_3(\mathcal{F}_3^{-1}(\mathcal{B}'p)(\vec{R}).\|\vec{R}\|) \tag{2.48}$$

Algorithme TTR (True Three-dimensional Reconstruction)

Son principe est basé sur l'utilisation de la rétroprojection des projections parallèles qui est égale à l'image 3D convoluée [NC78]. Pour sa mise en œuvre, les projections parallèles sont calculées approximativement à partir des projections coniques. En utilisant une généralisation du théorème de la projection 3D, la fonction f peut être exprimée par:

$$f(\vec{l}) = \mathcal{B}(p *_2 k)(\vec{l}) \qquad \text{où k est un filtre bidimensionnel} \tag{2.49}$$

L'algorithme, déduit de cette relation, se formule par:

- Estimation des projections parallèles $p_{//}$ à partir des projections coniques p,
- Convolution 2D des projections $p_{//}$ avec un filtre k vérifiant $\mathcal{F}_2 k(\vec{R}) = \|\vec{R}\|$:

$$p^c_{//}(\vec{l}) = (p_{//} *_2 k)(\vec{l}) \tag{2.50}$$

- Rétroprojection des projections parallèles convoluées $p^c_{//}$:

$$f(\vec{l}) = \mathcal{B}(p^c_{//})(\vec{l}) \tag{2.51}$$

Méthode de Feldkamp

Cette méthode est une généralisation en 3D de l'algorithme de reconstruction 2D en géométrie éventail, adaptée par Feldkamp [FDK84] dans le cas où la source décrit un cercle. La fonction f reconstruite est alors une approximation de l'image originale. Elle est basée sur la rétroprojection des projections pondérées et filtrées. C'est une méthode simple à mettre en œuvre, qui est largement utilisée en tomographie 3D. Elle se formule en trois étapes:

- Pondération des projections p:

$$p'(\vec{l'}) = p(\vec{l'})\frac{D}{\sqrt{D^2 + \|\vec{l'}\|^2}} \tag{2.52}$$

- Filtrage monodimensionnel des projections p' effectué, suivant les directions transverses à l'axe de rotation, par un filtre k vérifiant $\mathcal{F}k(Y) = |Y|$:

$$\tilde{p}(\vec{l'}) = p'(x', y') *_{y'} k(y') \tag{2.53}$$

- Rétroprojection pondérée des projections $\tilde{p}$ où $\vec{l}_{\vec{\tau}}$ est la projection sur le plan de projection de $\vec{l}$:

$$f(\vec{l}) = \frac{1}{2}\int_{\theta=0}^{2\pi} \tilde{p}(\vec{l}_{\vec{\tau}}) \frac{D^2}{(D_1 - \vec{l}.\vec{\tau})^2} d\theta \tag{2.54}$$

Méthode de Grangeat

Cette méthode a été développée à partir d'une relation entre la dérivée de Radon $\mathcal{R}'f$ et la transformée en rayons X pour chaque plan passant par la source. Cette transformée en rayons X est notée $\mathcal{X}f(S, A)$ où S est la position de la source et A le point du plan de détection sur la droite d'acquisition. Si la source décrit une trajectoire curviligne telle que tout plan passant par le support de l'objet rencontre la trajectoire, alors il est possible de décrire tout le domaine de Radon. Cette méthode introduite par Grangeat [Gra87] se caractérise par quatre étapes:

- Calcul de $\mathcal{R}'f$ dans le repère d'acquisition,
- Réarrangement pour passer au paramètrage de $\mathcal{R}'f$ en coordonnées sphériques
- Convolution et rétroprojection de la dérivée de la transformée de Radon qui permet d'obtenir les projections sur les plans méridiens du domaine de Radon,
- Rétroprojection de ces projections sur les plans transverses.

Méthode DC

C'est une méthode similaire à l'algorithme de Feldkamp. La différence notoire est l'étape de filtrage qui est un filtrage bidimensionnel dans le domaine de Radon. On pourra se reporter aux travaux de Defrise et Clack à l'origine de cette méthode [DC94].

2.4.6 Méthodes algébriques 3D

Les méthodes algébriques utilisées en tomographie 3D sont identiques à celles développées pour la tomographie 2D. Le modèle de base de ces méthodes n'est plus une grille de pixels, mais un volume composé d'éléments volumiques ou voxels. Les voxels sont numérotés de 1 à n et l'on définit une base $\{g_i/1 \leq i \leq n\}$ similaire à celle utilisée en 2D.

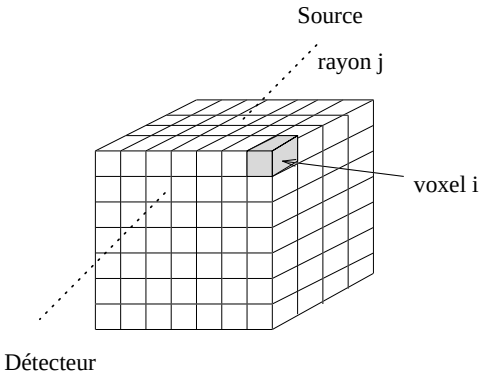


FIG. 13 - *Discrétisation du problème en 3D*

La fonction à reconstruire f est exprimée de la même façon comme une combinaison linéaire de ces fonctions de base:

$$f(x,y,z) = \sum_{i=1}^{n} f_i g_i(x,y,z) \tag{2.55}$$

Le système linéaire $Rf = p$ reste inchangé et il sert de base à toutes les reconstructions algébriques en tomographie 3D. La plupart des méthodes développées pour le 2D sont transposables en 3D. Nous présentons quelques méthodes qui décomposent le système linéaire $Rf = p$ en blocs.

A l'opposé des méthodes dites "Row-Action Technics" qui travaillent sur un rayon de projection, les méthodes par blocs travaillent sur une projection entière. Le système linéaire est reformulé en partitionnant la matrice de projection R et le vecteur p. Cette approche permet d'envisager l'implémentation parallèle de ces méthodes où la distribution des données sur les processeurs est calquée sur la décomposition du système en blocs [Pey90]. Le vecteur p contient l'ensemble des projections. On fait apparaître les projections P_j qui correspondent à l'ensemble

des projections p_j pour θ fixé. La matrice R est donc décomposée en blocs R_j. Le système devient:

$$\begin{pmatrix} R_1 \\ \vdots \\ R_j \\ \vdots \\ R_m \end{pmatrix} f = \begin{pmatrix} P_1 \\ \vdots \\ P_j \\ \vdots \\ P_m \end{pmatrix} \tag{2.56}$$

L'algorithme itératif issu de ce système fait intervenir une matrice Ω_j et un paramètre de relaxation λ_k:

$$\begin{aligned} &\text{initialisation de} f^{(0)} \\ f^{(k+1)} &= f^{(k)} + \lambda_k \, {}^t R_j \Omega_j (P_j - R_j f^{(k)}) \end{aligned} \tag{2.57}$$

Les algorithmes issus de cette décomposition se distinguent par le choix de la matrice Ω_j. Lorsque $\Omega_j = (R_j \, {}^t R_j)^{-1}$ le calcul d'inversion de matrice coûte très cher. Des algorithmes basés sur des matrices Ω_j moins coûteuses en terme de calcul ont été proposé. Nous rappelons deux critères de convergence:

- lorsque la matrice Ω_j est une matrice symétrique qui vérifie $R_j \, {}^t R_j \Omega_j R_j = R_j$, l'algorithme 2.57 converge pour $\lambda_k \in]0, 2[$ [Elf80].
- l'algorithme 2.57 converge si Ω_j vérifie l'inégalité suivante [EH81]:

$$\lim_{k \to \inf} \| \, {}^t R_{l_k} (I - R_{l_k} \, {}^t R_{l_k} \Omega_{l_k}) R_{l_k} \| < 1 \tag{2.58}$$

  où $l_k = k(\mathrm{mod} m) + 1$

Nous avons choisi d'utiliser les deux méthodes suivantes.

La méthode ART par Blocs

L'algorithme issu de cette méthode, défini par l'équation 2.59, peut s'interpréter comme la rétroprojection de la différence entre la projection mesurée P_j et la projection calculée $R_j f^{(k)}$.

$$\begin{aligned} f^{(k+1)} &= f^{(k)} + \lambda_k \, {}^t R_j \Omega_j (P_j - R_j f^{(k)}) \\ \Omega_j &= [\mathrm{diag}(R_j \, {}^t R_j)]^{-1} \end{aligned} \tag{2.59}$$

La méthode SIRT par Blocs

Cette méthode, déduite de la méthode SIRT, rétroprojete à chaque itération les différences entre chaque projection mesurée P_j et chaque projection calculée $R_j f^{(k)}$. La matrice W est égale à la somme de toutes les matrices $\Omega_j = \mathrm{tr}(R_j\, {}^t R_j)$. Chaque itération de l'agorithme 2.60 correspond à m itérations de l'algorithme précédent.

$$\begin{aligned} f^{(k+1)} &= f^{(k)} + \lambda_k W \sum_{l=1}^{m} {}^t R_j (P_j - R_j f^{(k)}) \\ W &= [\sum_{l=1}^{m} \mathrm{tr}(R_j\, {}^t R_j]^{-1} I \end{aligned} \tag{2.60}$$

2.5 Conclusion et évolution des méthodes de reconstruction 3D

Nous avons montré l'évolution des systèmes d'acquisitions qui ont généré des nouvelles problématiques de reconstruction. De ce fait, les méthodes de reconstruction 3D découlent souvent des méthodes 2D. Nous n'avons pas abordé, dans ce chapitre, le problème de la mise en œuvre de ces méthodes. On peut noter que certaines méthodes nécessitent des géométries d'acquistion qui ne sont pas envisagables d'un point de vue pratique et d'autres requièrent un outil informatique permettant de résoudre des problèmes de grande taille. Nous avons choisi de nous intéresser à des méthodes de reconstruction utilisables dans le cas d'une trajectoire circulaire.

Parmis les méthodes analytiques, nous avons choisit d'implanter la méthode de Feldkamp en raison de sa simplicité. Au niveau des méthodes algébriques, nous avons choisi celles basées sur une résolution par blocs et sur des matrices Ω_j facilement calculables pour satisfaire à ces conditions.

Pour situer cette étude au niveau de la tomographie 3D, on peut identifier plusieurs axes de recherche au vu des différents travaux présentés lors du congrès sur la tomographie 3D intitulé:" International Meeting on Fully Three-Dimensional Image reconstruction in Radiology and Nuclear Medecine" qui a eu lieu en 1995 à Aix-les-Bains. Nous présentons quelques uns de ces axes de recherche:

Amélioration de la quantification De nouvelles méthodes essayent de modéliser

la diffusion pour la corriger ou pour mieux la compenser. On peut citer les travaux en SPECT de Bouchard [BCD95] et Frey[FJT95], et en PET de Beyer[BKTS95]et Ott[Oa95].

Nouvelles géométries de reconstruction Ces géométries sont issues de l'utilisation d'échantillonnages efficaces [Des95], de géométrie hélicoïdale [ED95], de l'utilisation de détecteurs haute et basse résolution [WGB95] ou de l'utilisation de données provenant d'une géométrie Cone-Beam et Fan-Beam [GZ95].

Rapidité des algorithmes De nouveaux algorithmes sont développés pour accélérer les temps de reconstruction. On peut distinguer deux approches: les algorithmes issus de nouveaux modèles mathématiques ([Smi95, DKT95, WOC95, MB95]), et les algorithmes implémentés sur des machines offrant de grandes puissances de calcul, les machines parallèles ([LPC95b, OEM95]).

C'est dans cette dernière optique que s'inscrit notre étude: la parallélisation d'algorithmes de reconstructions 3D qui permettent de reconstruire des images 3D de tailles réalistes dans des temps acceptables.

Chapitre 3

Parallélisme

3.1 Introduction

Notre travail est basé sur l'utilisation des machines parallèles mais l'évolution rapide de leurs architectures et de leurs environnements logiciels, nous a conduit à concevoir nos approches sur une machine parallèle abstraite. Dans ce chapitre, nous rappelons les fondements du parallélisme. Puis nous détaillons les modèles d'exécution des machines parallèles. Nous développons par la suite les modèles de programmation associés à ces modèles d'exécution. Pour faciliter la mise en œuvre des programmes sur des machines parallèles, de nombreux projets de recherche ont eu pour but de créer des environnements de programmation. Nous présentons les principales bibliothèques de communication et de calcul utilisées par les grandes applications scientifiques. Pour valider l'utilisation de machines parallèles, les performances obtenues peuvent être évaluées suivant différents critères: mesures des temps, accélération et efficacité. Nous définissons ces critères à la fin de ce chapitre.

3.2 Origine du parallélisme et son évolution

Les besoins croissants des grandes applications scientifiques et militaires ont été le principal moteur économique du développement de ces nouvelles architectures. Les premières machines de calcul intensif, les machines vectorielles, sont apparues au milieu des années 70, notamment avec la création de Cray Research en 1974. Leur coût technologique au niveau du développement des architectures comme au niveau de leur utilisation, a freiné leur emploi d'une façon systématique. Aujourd'hui, un

simple réseau d'ordinateurs individuels (stations de travail, PC) supportant une bibliothèque de communication (PVM, MPI, P4 ...) est considéré comme une machine parallèle. Le rapport coût/performance de ces pseudo-machines parallèles est très intéressant. Cette évolution des performances résulte de la progression exponentielle de la vitesse des microprocesseurs, et de la densité d'intégration des microprocesseurs et des mémoires. Le tableau 1 montre, pour trois processeurs caractéristiques, l'évolution technologique réalisée depuis 20 ans et la croissance annuelle de chaque paramètre (fréquence d'horloge, transistor, mémoire).

Année	1974	1983	1994	Croissance par an
Processeur caractéristique	8080	80286	Alpha	
Fréquence d'horloge (MHz)	2	10	300	1.24
Nombre de Transistors	10^3	10^4	3.10^6	1.35
Mémoire associée (Kbytes)	4	512	64000	1.5

TAB. 1 - *Évolution de la vitesse et de la densité d'intégration des processeurs*

Cette évolution technologique ne doit pas dissimuler que les machines parallèles sont réservées à un petit nombre d'applications et que le coût des logiciels est souvent supérieur au coût des architectures. Ainsi, les modèles de programmation ne peuvent pas être dissociés des modèles d'exécution de ces machines et réciproquement [Eti94]. Nous présentons ces différents modèles d'exécution basés sur la classification des architectures de M.J. Flynn [Fly79].

3.3 Classification des architectures

Les architectures des machines parallèles sont classiquement définies suivant deux critères de localité:

- le séquencement des instructions, ou flot de données,
- les données ou blocs mémoires.

Une machine exécute un programme constitué d'un flot d'instructions. Pour une machine séquentielle, le flot des instructions qui arrive au processeur est séquencé par une horloge unique.

Pour une machine multiprocesseurs, on peut distinguer si le flot d'instructions est géré par un contrôle centralisé ou par un contrôle distribué. Dans le premier cas, les processeurs exécutent un programme constitué d'un seul flot d'instructions sous la direction d'un unique séquenceur. Dans le deuxième cas, chaque processeur effectue un flot d'instructions sous la direction de son propre séquenceur: le contrôle est donc distribué.

Pour la gestion de la mémoire ou des données, on distingue les machines à mémoire partagée et celles à mémoire distribuée. Dans le premier cas, cette mémoire centralisée génère des conflits au niveau de ses accès. Dans le cas d'une mémoire distribuée, la localisation de la mémoire entraîne des temps d'accès aux informations dépendant de l'architecture de la machine.

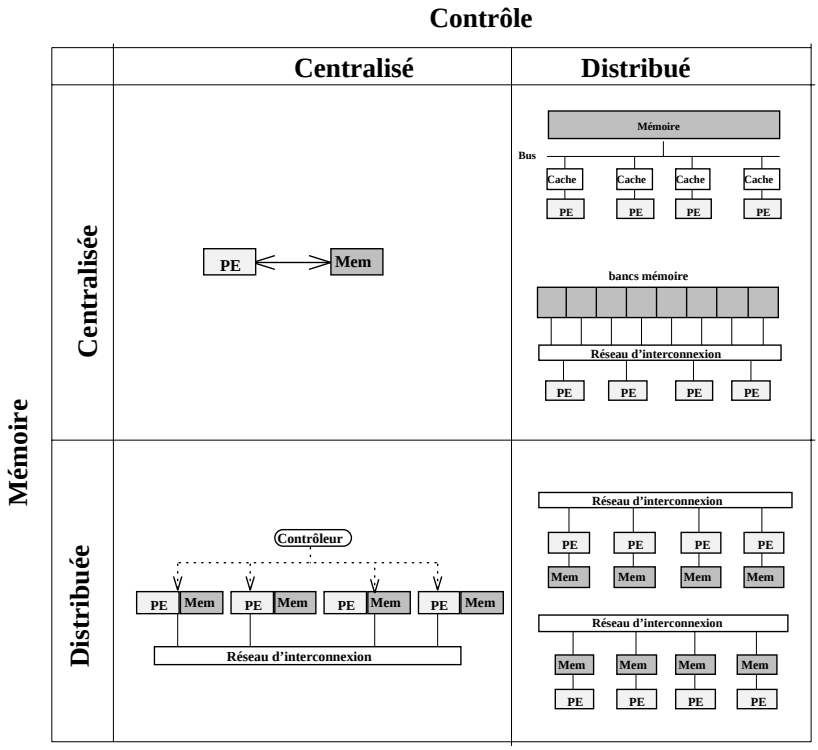


FIG. 14 - *Classification suivant Flynn*

La figure 14 présente les différents modèles d'exécution. Les machines, que nous utilisons dans notre étude, ont des modèles d'exécution basés sur un contrôle centralisé ou distribué avec des mémoires distribuées. Une machine séquentielle classique

correspond dans notre tableau à un contrôle et à une mémoire centralisée. D'autres machines parallèles ont un mode de contrôle et de gestion de la mémoire qui leur est spécifique: les machines vectorielles, systoliques, ou les réseaux neuronaux.
Nous décrivons les principes généraux des machines parallèles rentrant dans notre classification.

3.3.1 Machine parallèle SIMD à mémoire distribuée

Les machines parallèles SIMD (Single Instruction stream Multiple Data Stream) sont classiquement définies par le modèle d'exécution présenté en figure 15.

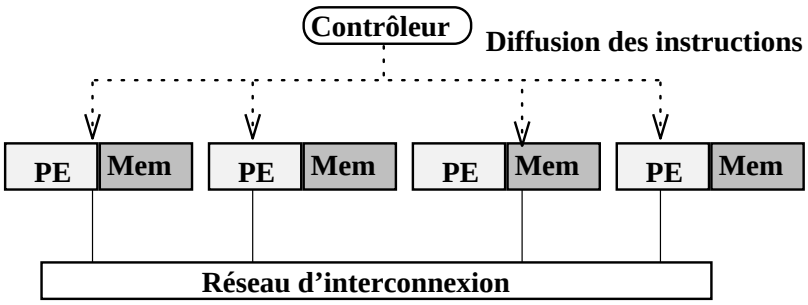


FIG. 15 - *Modèle d'exécution des machines SIMD*

Les processeurs de ces machines n'ont pas de séquenceur et ont une puissance de calcul très faible par rapport aux machines séquentielles classiques. Ces processeurs sont constitués d'unités arithmétiques et logiques programmables. Chaque processeur possède une mémoire de faible capacité et exécute les instructions une à une, sous l'impulsion de l'unique contrôleur ou séquenceur. La puissance de calcul de ce type de machine est obtenue grâce au grand nombre de processeurs.

La Connexion Machine (CM) de TMC [CM-87] est un exemple représentatif de cette classe de machine. Elle est constituée de 65536 processeurs élémentaires 1 bit. Un mécanisme de masquage permet d'exécuter l'instruction courante sur certains processeurs, alors dénommés actifs par rapport aux autres processeurs qui attendent la fin de l'instruction pour passer de l'état passif à celui d'actif. Le réseau d'interconnexion est un hypercube. Les machines qui lui ont succédé sont, la CM-2 et la CM-200 constituées de processeurs élémentaires plus puissants.

La Maspar (ou Massively Parallel) est un autre exemple caractéristique de cette classe de machines. Elle est constituée de processeurs 4 bits. Nous décrivons son architecture et son utilisation dans le chapitre 4.

3.3.2 Machine parallèle MIMD à mémoire centralisée

L'architecture de ces machines se caractérise par le faible nombre de processeurs lié à la complexité du réseau d'interconnexion. Leur puissance repose sur la puissance de chaque processeur de base et non sur sur le nombre de processeurs. Cette classe est constituée de deux sous-classes de machines parallèles: les multiprocesseurs à bus et les multiprocesseurs vectoriels.

Multiprocesseur à bus

Ces machines parallèles utilisent des processeurs classiques et une mémoire hiérarchisée. L'accès aux données se fait par les caches qui sont reliés à la mémoire principale via un bus (figure 16).

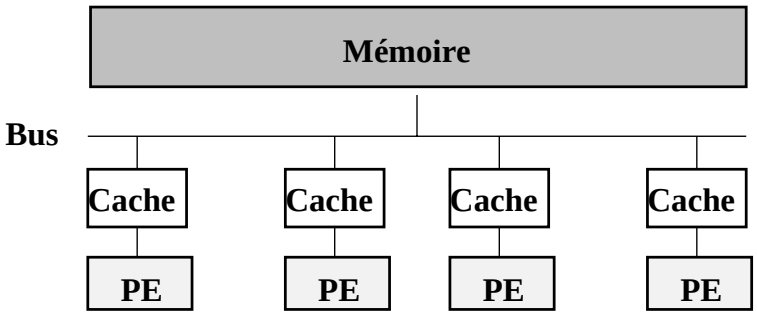


FIG. 16 - *Multiprocesseur à bus*

Le principal problème de ce type de machine est la modélisation du temps d'accès aux données puisqu'il dépend de leurs présences ou non dans le cache local. De plus, la cohérence des caches amène un surcoût pour la gestion de la mémoire centrale. Les récents serveurs de station sont des exemples de ce type de machine. Ils sont dotés de puissants microprocesseurs délivrant une puissance de calcul très appréciable. On peut citer les machines DEC10000 à 6 processeurs Alpha AXP, les machines Power-Challenge de SGI constituées de 36 microprocesseurs MIPS R4400 et la nouvelle machine Examplar de Convex [Exe94].

Multiprocesseurs vectoriels et accès mémoire uniforme

Ces machines tantôt considérées comme vectorielles ou parallèles sont constituées d'un petit nombre de processeurs vectoriels et d'une mémoire divisée en banc mémoire (figure 17).

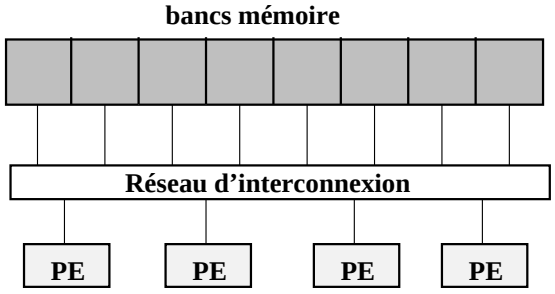


FIG. 17 - *Multiprocesseurs vectoriels*

Elles sont les héritières des premières machines vectorielles mono-processeurs comme le Cray 1. Leur architecture provient de l'évolution des machines mono-processeurs vers les machines multiprocesseurs. Le problème de ce type de machine est la rapidité d'accès à la mémoire via le réseau d'interconnexion et la gestion des conflits d'accès. Les machines Cray XMP et YMP ainsi que les machines vectorielles de Nec et Hitachi font partie de cette classe de machines. Elles ont été conçues pour les applications de calcul scientifique et l'apport de compilateurs pour la parallélisation automatique laisse présager un bel avenir à ce type de machines.

3.3.3 Machine parallèle MIMD à mémoire distribuée

Ces machines sont constituées de nœuds élémentaires et d'un réseau d'interconnexion. Le nœud élémentaire est composé de trois éléments:

- Le processeur: les premières machines utilisaient des processeurs spécifiques comme le Transputer. Pour diminuer les coûts de développement, les constructeurs utilisent des processeurs standards: Alpha de Dec, i860 d'Intel, Sparc, RS6000 d'IBM, Power 2 ...,

- La mémoire: elle est propre à chaque processeur et est associée parfois à un contrôleur. Son espace d'adressage est soit local soit global.

- Le routeur: il permet de transmettre les données aux autres processeurs. Cette fonctionnalité est soit le fait du processeur, dans le cas du Transputer, soit d'un processeur dédié aux communications.

Nous pouvons distinguer deux sous-classes de machines en fonction de l'espace d'adressage.

Machine MIMD à passage de message

Chaque processeur possède son propre espace d'adressage sur la mémoire locale. Ainsi si un processeur PE_i veut travailler avec des données contenues dans la mémoire M_j du processeur PE_j, il doit envoyer un message au processeur PE_j et attendre la réception d'un message contenant les données. Son modèle d'exécution est présenté par le schéma 18.

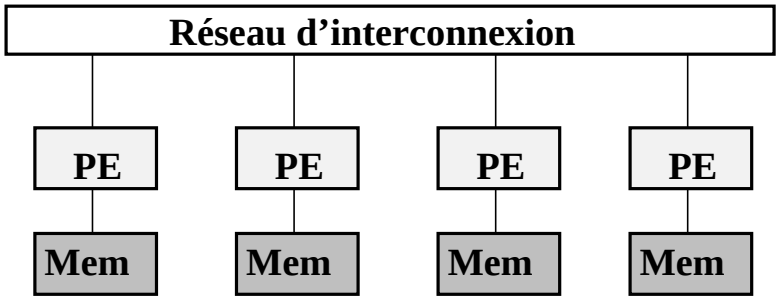


FIG. 18 - *MIMD à passage de message*

Plusieurs types de machines rentrent dans cette classification:

Les machines massivement parallèles Ces machines sont considérées comme massivement parallèles par la nature spécifique de leur réseau d'interconnexion et par la gestion de leurs accès. En effet, les nœuds de ces machines ne sont accessibles que par l'intermédiaire d'un frontal qui gère les entrées-sorties et les allocations de nœuds aux utilisateurs: de ce fait, un utilisateur est propriétaire d'une partition des nœuds de la machine à un instant donné.
Les machines les plus représentatives sont la CM-5 de TMC [CM-93], la Paragon d'Intel [Par91], la machine NCube3. La CM5 est équipée de processeurs standards RISC SPARC et de trois réseaux d'interconnexion distincts: un réseau de contrôle, un réseau de données et un réseau de diagnostic. Le réseau de données a une topologie statique en arbre élargi ("fat tree"). Il

permet d'effectuer les transferts de données entre les processeurs et gère les entrées-sorties. Les caractéristiques de la Paragon seront développées dans le chapitre 4.

Les machines moyennement à massivement parallèles Les architectures de ces machines sont définies comme des stations de travail reliées par un réseau d'interconnexion, efficace au contraire des machines précédentes qui se définissent plus comme un ensemble de nœuds reliés par un réseau d'interconnexion. On peut citer comme exemple de ce type de machines les SP1 et SP2 d'IBM [SP193] et la CS2 de Meiko [CS-93]. Ce type de machine se situe entre les machines massivement parallèles et les réseaux de stations de travail. Ce sont des machines relativement ouvertes: un utilisateur n'est pas propriétaire d'un nœud à un instant donné. Ceci permet une meilleure utilisation des capacités de la machine mais entraîne des possibilités de surcharge. Leur réseau d'interconnexion est souvent un réseau multi-étages et leurs processeurs sont des processeurs standards: RS6000 (SP1), Power2(SP2) et SuperSPARC (CS2). Nous présenterons l'utilisation d'une machine SP1 dans le chapitre 4.

Les réseaux de stations de travail On peut considérer ces machines comme faisant partie des machines MIMD à mémoire distribuée car elles sont composées de stations de travail possédant un processeur et une mémoire, et sont reliées via un réseau classique Ethernet ou un réseau spécifique: un "Giga Switch". On désigne ces machines soit comme une ferme de processeurs, soit comme un réseau de stations de travail. Le principal problème des réseaux de stations de travail est dû à la disparité des nœuds élémentaires qui les composent et, qui impliquent de gérer cette disparité au niveau des algorithmes. Le grand avantage avec ce type de machine est le rapport coût/performance qui permet de paralléliser de petites applications à moindre coût.
Nous utiliserons deux machines de cette classe: un réseau de cinq stations SUN à processeur SPARC1 et SPARC2 et une ferme de 16 processeurs Alpha reliés par un " Giga Switch" en fibre optique.

Machine MIMD à espace d'adressage unique

Ces machines se caractérisent par une mémoire physiquement distribuée mais logiquement répartie. Leur schéma est presque identique à celui des machines MIMD

à passage de messages, la différence se situant au niveau des nœuds: le réseau d'interconnexion relie les mémoires et non les processeurs (schéma 19). Soit la mémoire est virtuellement partagée à l'aide de caches mémoire, soit elle est accessible à l'aide d'un contrôleur mémoire.

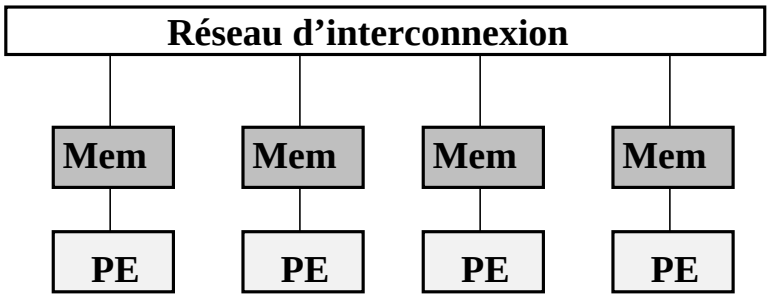


FIG. 19 - *MIMD à espace d'adressage unique*

La machine KSR [KSR92] utilise une mémoire virtuellement partagée. Chaque nœud est constitué d'un processeur et d'un cache primaire. La mémoire est une mémoire locale paginée et constitue un cache secondaire. La mémoire est accessible par les processeurs à travers ces deux caches mémoires. Le réseau d'interconnexion sert alors à gérer la cohérence des caches et à transférer les copies multiples.
La machine T3D de CRAY [Cra94] a une approche différente. Chaque mémoire est locale et l'accès à la mémoire est géré par un contrôleur mémoire. Ainsi, si un processeur veut accéder à des données, il envoie une requête au contrôleur. Si les données se trouvent en mémoire locale elles sont tout de suite transférées, sinon le contrôleur se charge d'aller chercher ces données sur une autre mémoire locale via le réseau d'interconnexion. Ces types de machines offrent de nouvelles solutions architecturales qui permettent une gestion de la mémoire simplifiée pour l'utilisateur donc une programmation "plus facile". Nous présenterons plus en détail l'utilisation du T3D dans le chapitre 4.

3.3.4 Réseaux d'interconnexion

Les réseaux d'interconnexion sont à la base de tous les modèles d'exécution présentés précédemment. Leur rapidité est un facteur de performance des machines parallèles. Leur modélisation permet à l'utilisateur d'optimiser ses implémentations

en fonction des caractéristiques du réseau d'interconnexion. Leurs topologies sont définies en fonction de plusieurs critères:

- le diamètre du réseau qui définit la plus grande distance entre deux processeurs,
- la connectivité du réseau liée au nombre de liens de communication de chaque processeur aux autres processeurs. Plus le nombre de liens est important, plus les communications peuvent être effectuées en parallèle,
- les sous-topologies contenues dans le réseau permettant à l'utilisateur de définir ses algorithmes de communication.

Le tableau 2 présente les topologies les plus usuelles des réseaux d'interconnexion.

Réseau	Nombre de nœuds	Diamètre	Nombre de voisins
Anneau	N	$\lfloor \frac{N}{2} \rfloor$	2
Grille 2D	$N.M$	$M + N - 2$	2 ou 4
Hypercube	$N = 2^d$	$\log N$	d
Tore 2D	$N.M$	$\lfloor \frac{N}{2} \rfloor + \lfloor \frac{M}{2} \rfloor$	4
Tore 3D	$N.M.L$	$\lfloor \frac{N}{2} \rfloor + \lfloor \frac{M}{2} \rfloor + \lfloor \frac{L}{2} \rfloor$	6
Multi-étage (p)	$N = 4^p$	constant	4 sur le premier étage

TAB. 2 - *Les topologies usuelles*

3.3.5 Modes de communication

Parallèlement aux nombreux réseaux d'interconnexion, les machines parallèles utilisent des modes de communication s'appuyant sur des stratégies différentes. Les communications sont effectuées sur un lien entre deux processeurs. Elles peuvent être soit dans un seul sens à un instant donné (mode "half-duplex"), soit dans les deux sens (mode "full-duplex"). Pour communiquer un message d'un processeur à un processeur distant, les techniques utilisées sont:

Store and Forward Le message est émis comme une succession de communications entre deux processeurs. Il est stocké par le processeur en réception puis il est à nouveau émis vers un nouveau destinataire. Ce mode est utilisé sur les premières machines parallèles comme l'IPSC/1 d'Intel et les machines à base de Transputers.

Circuit-Switched Le processeur qui émet le message, ne transmet que l'en-tête du message au destinataire. L'en-tête établit le parcours de l'acheminement du message. Une fois que le processeur destinataire a reçu l'en-tête, il envoie un accusé de réception qui déclenchera le transfert.

Wormhole L'algorithme est identique à celui du "Circuit-Swiched". La seule différence vient du corps du message qui est envoyé par petits paquets à la suite de l'en-tête. En cas de blocage, le message est stocké temporairement le long du parcours.

Virtual Cut Through Son principe, similaire au "Wormhole", permet de libérer le chemin en cas de blocage par regroupement du message sur le dernier processeur.

Les deux derniers modes de communication sont les plus utilisés sur la dernière génération de machines parallèles. Pour calculer le coût des algorithmes de communication utilisant ces différents modes, de nombreuses études ont été réalisées pour modéliser l'envoi d'un message de taille variable. Le modèle classique pour l'acheminement d'un message de taille L entre deux processeurs est défini par:

- un temps d'initialisation ou "start-up" β,
- un temps de propagation $L\tau$ linéairement proportionnel à L.

Le temps d'acheminement est donc égal à : $T_L = \beta + L\tau$. Cette modélisation nous permet d'évaluer le coût des communications sur les machines que nous utilisons. Les valeurs de β et τ sont définies pour chaque machine parallèle. Pour des communications entre plus de deux processeurs, ces valeurs dépendent des possibilités de la machine. Des surcouches logicielles sont utilisées pour permettre des communications globales performantes. On pourra se référer aux travaux de Saad et Schultz [SS89] qui présentent des algorithmes optimaux de communication sur la plupart des architectures.

3.4 Modèles de programmation

La parallélisation d'un algorithme peut être définie par la maxime "diviser pour régner". En effet, si l'on considère un algorithme séquentiel comme une tâche T à effectuer sur des données D, le propre du parallélisme est de définir un ensemble de

tâches T_i à partir de T et un ensemble de données D_j à partir de D et de répartir ces tâches et ces données sur les différents nœuds des machines parallèles. Cette division sur les tâches et sur les données a été formalisée par deux modèles de programmation [RT94]:

- Le modèle de division sur les données est appelé le *parallélisme de données*. Son principe est d'effectuer des actions successives sur des données réparties sur les processeurs.

- Le modèle de division par les tâches est appelé *parallélisme de contrôle*. Son principe est de décrire une application comme un ensemble de tâches indépendantes pour pouvoir les effectuer en parallèle.

Pour une parallélisation efficace, une application parallèle issue de l'un de ces deux modèles de programmation doit prendre en compte les problèmes liés à l'équilibrage de charge, au placement des données et au regroupement des résultats. Nous donnons un bref aperçu de ces modèles de programmation dans les paragraphes suivants.

3.4.1 Parallélisme de données

Le parallélisme de données consiste à découper les données et à les répartir sur les mémoires des processeurs. Deux types de partitionnement sur les données sont envisageables: le partitionnement sur les données initiales ou le partitionnement sur les données résultats. Dans le premier cas, on distribue l'espace des données initiales en sous-domaines sur chaque processeur; dans le second cas, chaque processeur est responsable d'un sous domaine de l'espace résultat.
Dans la plupart des applications, les espaces de données initiales et de données résultats peuvent être divisés par un découpage similaire, ce qui simplifie la parallélisation.
Après la répartition, chaque processeur effectue un calcul séquentiel sur les données dont il est propriétaire. Le séquencement des instructions peut être soit centralisé, soit distribué:

- Dans le cas d'un séquencement centralisé, le parallélisme de données a pour modèle d'exécution les machines SIMD. En effet, ce type de machine est par essence une machine "data parallel", car une même instruction est effectuée sur des données réparties sur les mémoires locales.

- Dans le cas d'un séquencement distribué, on utilise le modèle de programmation SPMD (“ Single Programme Multiple Data ”). Ce modèle distribue à chaque nœud une copie d'un programme. Ce programme est constitué de tâches de calcul et de tâches de communication. Contrairement aux machines SIMD où chaque instruction est synchronisée, ce modèle effectue les tâches de calcul en parallèle de manière asynchrone, ce qui rend la parallélisation potentiellement plus efficace. Les tâches de communication servent à la synchronisation des calculs et aux transferts éventuels des données. L'équilibrage des tâches de calcul reste l'un des principaux problèmes des applications développées sur le modèle SPMD. De nombreuses techniques ont été mises en œuvre pour résoudre le problème d'équilibrage de charge, nous en présenterons quelques unes lors de l'implémentation de nos algorithmes.

Ce modèle est utilisé pour la parallélisation automatique de “gros” codes industriels. Dans cette optique, des langages parallèles ont été développés comme HPF. On peut représenter le parallélisme de données en utilisant le formalisme du parallélisme de contrôle. Si on découpe le programme en un ensemble fini tâches T_i, alors chaque processeur effectue la même tâche T_i sur des données différentes $D_1, D_2 ... D_j$ (figure 20).

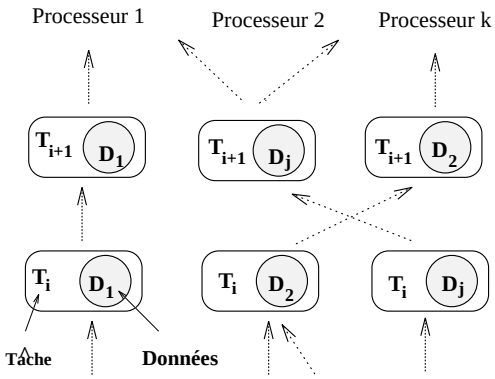


FIG. 20 - *Parallélisme de données*

3.4.2 Parallélisme de contrôle

Le parallélisme de contrôle définit un programme parallèle comme étant l'expression d'un algorithme sous la forme d'un ensemble de calculs pouvant être effectués en parallèle. On formalise cet ensemble de calcul par un ensemble de tâches. Chaque

tâche est un programme séquentiel classique auquel on ajoute des primitives de communication qui définissent les interactions d'une tâche avec les autres. L'algorithme peut être représenté sous la forme d'un graphe orienté où les nœuds formalisent les tâches et les arcs orientés, les communications. Dans ce cas, les processeurs effectuent des tâches différentes sur des données différentes.

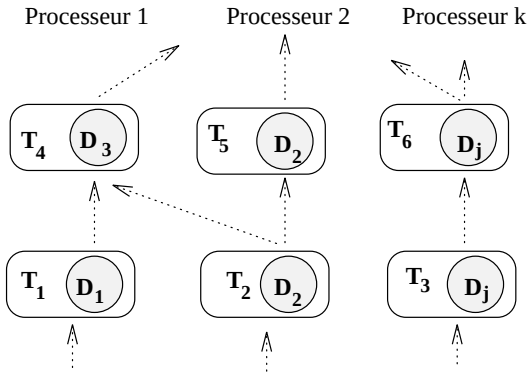


FIG. 21 - *Parallélisme de contrôle*

Le principal problème de cette classe d'applications est le placement optimal des tâches sur les processeurs d'une machine parallèle [AP88]. Dans le cas idéal, on souhaiterait disposer d'une machine parallèle composée d'autant de processeurs qu'il y a de tâches. Ce type de parallélisme est implémenté sur les machines de MIMD.

3.5 Environnement de programmation

L'apparition du parallélisme a permis d'augmenter la puissance des machines dédiées au calcul. Mais pour obtenir la puissance théorique des machines parallèles, l'implémentation des applications doit être efficace.
Cette remarque exprime le fait qu'il y a toujours une différence notoire entre la puissance théorique d'une machine parallèle et son utilisation. Pour essayer d'obtenir une puissance maximale, les utilisateurs doivent programmer leurs applications en fonction des caractéristiques des machines. Pour cela les constructeurs de machines ont développé des langages parallèles, comme Occam dédié à l'utilisation des Transputers. Ces premiers langages avaient un handicap lié à leur dépendance vis à vis de l'architecture utilisée. A l'heure actuelle, il n'existe pas de langage parallèle standard. Les constructeurs se sont basés sur des langages standards comme C ou

Fortran, pour définir leurs outils de programmation. Ces outils sont principalement des bibliothèques de communication et des bibliothèques de calculs parallèles.
Nous allons présenter les bibliothèques les plus utilisées actuellement et nous développerons l'une d'elles, PVM, que nous avons utilisée pour nos implémentations. Avant de les présenter, rappelons les principales difficultés liées au développement d'applications parallèles:

- Le développement efficace nécessite l'intervention d'un spécialiste de l'informatique parallèle car actuellement le code produit par les outils de parallélisation automatique n'est pas assez performant. Le futur langage HPF permettra d'intégrer le parallélisme d'une manière presque transparente pour l'utilisateur,

- Le manque de standardisation des langages utilisés génère des efforts de réécriture lors du portage d'une application d'une machine parallèle sur une autre. Des tentatives de standardisation sont en cours pour l'élaboration de bibliothèques de communication, comme la bibliothèque MPI héritière de la bibliothèque PVM disponible sur presque toutes les machines parallèles,

- Les outils d'aide à la programmation sont rudimentaires. En raison de la complexité des architectures et des communications, on ne peut utiliser que des traces d'exécution post-mortem.

3.5.1 Bibliothèques de communication

Les premières bibliothèques de communication ont été conçues par les constructeurs qui, ne désirant pas investir dans un langage propre à leur machine, ont créé des bibliothèques contenant leurs propres routines de communication. Nous avons utilisé l'une de ces bibliothèques NX [Par91], développée par Intel pour l'utilisation de la Paragon.

En raison de la multiplicité des machines et donc des bibliothèques constructeurs, de nombreux groupes de recherche ont défini des bibliothèques de communication compatibles avec un grand nombre de machines parallèles. Ces bibliothèques ont été mises au point sur des réseaux d'ordinateurs, car ceux-ci sont une bonne modélisation des machines distribuées actuelles. Cette modélisation a permis de

considérer un réseau hétérogène d'ordinateurs comme une machine parallèle. Ces bibliothèques offrent ainsi un double avantage: le développement d'applications parallèles à moindre coût sur des machines constituées de machines séquentielles, le portage de telles applications sur un bon nombre de machines parallèles. On peut citer les bibliothèques les plus utilisées: PVM[BDJ+94],PICL[GHPW90], P4[BL92]. A l'heure actuelle PVM est considéré comme un standard du fait de sa large diffusion. Un nouveau standard a été défini par les constructeurs et les groupes de recherche: MPI[MPI93].

MPI (Message Passing Interface) est un projet de standardisation des systèmes de communication pour le passage de messages sur les architectures parallèles à mémoire distribuée. L'un des objectifs de MPI est d'avoir une interface portable et facile d'utilisation et ceci, sans sacrifier les performances. MPI est basé sur une synthèse des travaux déjà réalisés. Cette bibliothèque propose de nombreux mécanismes de communication efficaces reposant sur la "bufferisation" des données échangées, permettant un codage des données et donc de pouvoir travailler éventuellement avec une machine hétérogène. De plus MPI, par la nature même de ses communications, peut intégrer les dernières techniques d'optimisation d'un code parallèle comme le recouvrement des calculs par les communications. Depuis la sortie de ce projet, un certain nombre d'implémentations ont vu le jour, notamment sur la SP1 d'IBM.

3.5.2 PVM (Parallel Virtual Machine)

PVM est un environnement de programmation développé par des chercheurs de l'université du Tennessee et du Oak Ridge National Laboratory [BDJ+94]. Cet environnement est l'un des plus utilisés actuellement et on le considère souvent comme le standard de facto des bibliothèques de "message passing". Son succès peut se mesurer par le nombre de constructeurs qui ont implémenté une version sur leur machine et par le nombre de participants toujours croissant aux deux conférences annuelles qui lui sont consacrées, l'une au USA et l'autre en Europe.

Philosophie de PVM

Lors de l'élaboration d'un programme parallèle, un programmeur détermine les phases de communication et les phases de calcul de son application. Les phases de calcul peuvent être assimilées à des tâches élémentaires qui se communiquent des informations lors de phases de communication. Pour une application séquentielle sur un système Unix, des tâches identiques sont identifiées à des processus, et elles communiquent entre elles par des outils de communication comme les sockets, les sémaphores et l'émission et la réception de signaux.
PVM a repris cette philosophie. Il est défini comme étant un système de gestion de processus associé à des routines de communication de messages entre ces processus. Cette gestion de processus est gérée par des "démons" qui collectent les messages et qui les transmettent du processus émetteur au processus destinataire. De plus, PVM est un système qui supporte l'hétérogénéité des applications et des machines: il permet de construire une machine parallèle composée de stations de travail et de nœuds d'une machine parallèle. La transmission d'informations entre ces nœuds hétérogènes est possible grâce à un système de codage des messages nommé XDR.
Le succès de PVM peut ainsi s'expliquer par:

- le langage de programmation, constitué de C ou Fortran augmenté des routines de communication, ne nécessite pas l'apprentissage d'un nouveau langage,
- la diffusion de cet environnement sur la plupart des stations de travail et des machines parallèles assure la portabilité des applications,
- la souplesse d'utilisation de la gestion des processus permet une programmation avec peu de contraintes.

Fonctionnalités de PVM

La machine virtuelle est constituée d'un certain nombre de nœuds qui sont soit des stations de travail (SUN, SGI, IBM, HP...), soit des nœuds d'une machine parallèle (Paragon, SP1, T3D,...), soit une machine parallèle complète (Cray YMP, Maspar...).
A chaque nœud correspond un démon PVM propriétaire de chaque utilisateur. Cela permet d'avoir plusieurs applications sur un même ensemble de nœuds.
Chaque tâche d'une application PVM, représentée par un processus PVM, est définie

par un numéro de processus appelé "tid" pour "Task Identifier". Ce numéro de processus est donné par le "démon" pour la gestion des processus. L'utilisateur dispose pour sa part d'une numérotation qui permet d'identifier chaque processus. Cette numérotation correspond aux "tids" par l'intermédiaire d'une table de correspondance.
La gestion de processus, laissée à l'utilisateur, est dynamique. En cours d'utilisation de nouveaux processus peuvent être créés et détruits de la même façon. Il est donc possible d'avoir plusieurs processus séquentiels sur un même nœud PVM. Si un nœud PVM est constitué d'un processeur avec sa mémoire d'une puissance théorique P, la présence de plusieurs processus divise d'autant leur puissance théorique égale, pour PE processus, à $\frac{P}{PE}$. C'est pourquoi, l'on considère souvent un processus PVM comme un nœud PVM.

Les communications entre processus sont définies par les routines de communication. Deux types de communications sont possibles entre processus: les communications point à point et les communications globales. La notion de groupe, définie pour rassembler un ensemble de processus, permet d'effectuer des communications globales sur ce groupe. Nous développons l'utilisation des communications dans le paragraphe suivant.

Pour utiliser un programme utilisant PVM, on exécute l'interface de contrôle pour initialiser la machine virtuelle. Cette interface permet de gérer les différents nœuds de la machine virtuelle et de contrôler les processus PVM actifs. Ces fonctions de contrôle de processus sont aussi disponibles dans le langage et permettent à l'application de collecter elle-même des informations sur les processus actifs, d'ajouter ou de détruire un nœud et d'envoyer des signaux aux autres processus.

Les Communications

Les communications entre les différents processus se font par échange de message. Les messages sont souvent constitués de valeurs ayant différents types. Pour cela, le programmeur définit la forme du message contenant plusieurs types et alloue la zone mémoire correspondant à l'ensemble des valeurs. Cette zone mémoire, appelée buffer, sert à l'émission et à la réception des messages. Une communication est

toujours définie de la façon suivante:

Envoi d'un message Il se déroule en trois phases. La première consiste à initialiser le buffer d'émission. Ensuite les données sont rangées dans ce buffer par un empaquetage. Enfin le buffer est envoyé en direction du ou des récepteurs du message. L'envoi est toujours une fonction asynchrone non bloquante.

Réception d'un message Le processus attend de recevoir le message d'un émetteur désigné ou quelconque. Une fois le buffer reçu, il est désempaqueté. La réception est par essence bloquante. On peut utiliser des primitives non-bloquantes en testant si le message a été reçu.

Pour comprendre le mécanisme de gestion des communications, la figure 22 indique ce qui se passe lors de l'envoi d'un message du processus A vers les processus B et C, où A et B appartiennent au même nœud. Le message AB est envoyé au démon $D1$ qui le redirige vers B. Le message AC est envoyé au démon $D1$ qui lorsqu'il a trouvé le démon destinataire lui envoie le message AC. Puis le démon $D2$ transmet le message à C.

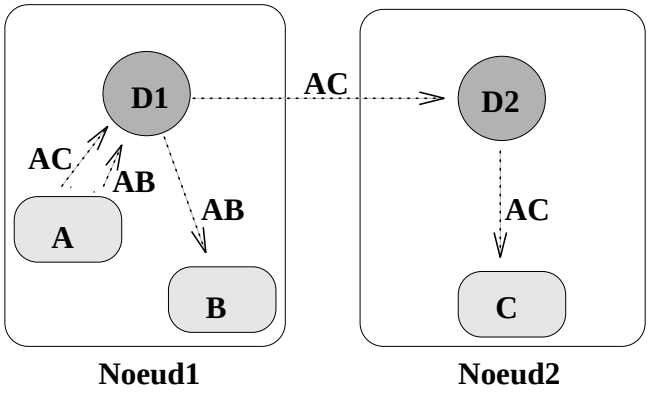


FIG. 22 - *Communication PVM entree différents processus A,B et C*

Les communications point à point utilisent des routines d'envoi, "send" et de réception, "recv". Les communications globales peuvent être construites à partir de ces primitives. Il existe néanmoins des routines globales de diffusion "cast", de groupement "gathering" et de réduction "reduce", utilisées avec une liste de processus ou un groupe de processus.

Programmation

Il existe des outils d'aide à la programmation comme Hence, Xab et XPVM. Ces outils permettent de visualiser les traces de PVM et de suivre l'évolution d'une application. On peut ainsi suivre le comportement de chaque processus et des communications inter-processus. La portabilité de PVM permet de tester les applications sur des petits réseaux de stations de travail, ou même sur une simple station de travail avant de les porter sur des machines parallèles. On peut ainsi vérifier que le schéma de communication est cohérent et valider l'application sur des petits jeux de données.

3.5.3 Bibliothèques de calcul

Parallèlement aux développements de bibliothèques de communication, des bibliothèques de calcul, basées sur des noyaux de calcul, ont été programmées de façon efficace sur un bon nombre de machines. L'objectif de ces bibliothèques est de proposer à l'utilisateur un certain nombre de primitives d'algèbre linéaire parallèle, pour que celui-ci puisse développer des applications complexes et efficaces sans tenir compte de l'implémentation effective des opérateurs de base. Ces bibliothèques sont issues des premières bibliothèques mises en œuvre pour les ordinateurs vectoriels.
La première bibliothèque, LINPACK[DMBS88], contient des programmes de factorisation matricielle: LU,Cholesky, QR et SVD. Elle a été écrite en Fortran et a introduit la programmation modulaire. Elle utilise les notions de BLAS ("Basic Linear Algebra Subroutines") qui se définissent suivant trois niveaux, ou α désigne un scalaire, x et y des vecteurs et A, B et C des matrices:

- BLAS 1: opérations vecteur-vecteur: $x \rightarrow \alpha * x + y$,
- BLAS 2: opérations matrice-vecteur: $x \rightarrow \alpha * A * x + y$,
- BLAS 3: opérations matrice-matrice: $C \rightarrow \alpha * A * B$.

La bibliothèque EISPACK est constituée de routines parallèles dédiées aux problèmes de valeurs propres et de vecteurs propres. Elle utilise aussi les BLAS. La plus répandue actuellement est la bibliothèque ScaLAPACK[CDPW92] qui est basée sur l'utilisation des BLAS et des BLACS qui sont la réécriture des routines BLAS sur des machines de type MIMD. Pour de plus amples informations, on pourra se reporter aux nombreux travaux de synthèse [Col94].

3.5.4 Langages parallèles

Les premiers langages parallèles dédiés aux processeurs comme Occam apparenté à un pseudo-assembleur, ont été conçus par les constructeurs qui ont offert des langages dédiés à leurs machines. L'avènement des machines SIMD a favorisé l'apparition de ces langages car chaque machine a une architecture et un système d'exploitation qui lui est propre. La CM2 utilise un langage dérivé du Lisp le * Lisp. Les machines Maspar utilisent un langage basé sur le C ou le Fortran appelé MPL. On peut définir cet outil de programmation comme un langage car il intègre des notions de localité des variables, globales à tous les processeurs ou locales à chacun, ainsi que des routines de communications propres à son architecture. Nous l'avons utilisé dans le cadre de notre travail, nous en présenterons les aspects généraux.

Face à ces langages figés par une architecture, un langage dédié au parallélisme de données est en cours de réalisation: HPF (" High Performance Fortran") [HPF93]. Ce langage s'appuie sur Fortran 90. Pour sa mise au point, il a bénéficié des acquis de la vectorisation automatique. L'utilisateur peut spécifier au compilateur le partage des données et désigner les boucles à effectuer en parallèle. Ce langage propose également des opérations de réduction transparentes pour le programmeur. C'est le compilateur qui se charge de générer un code, souvent à échange de messages, pour la machine cible. Les premiers compilateurs commencent à apparaître, mais il reste encore beaucoup de problèmes non résolus.

3.5.5 Outils d'aide à la programmation

Au niveau des outils d'aide à la programmation, le classique débogage des algorithmes permet de comprendre les erreurs syntaxiques de programmation. Le développement de débogueurs parallèles s'est avéré beaucoup plus difficile que les débogueurs classiques, principalement à cause des problèmes liés aux communications. Néanmoins certains constructeurs proposent des débogueurs pour leurs machines, comme le débogueur associé au langage MPL pour les Maspar ou Cray-Tool associé au Cray T3D.
L'utilisation de bibliothèques de communication comme PVM permet d'utiliser les outils classiques sous Unix. D'autres outils sont toutefois nécessaires pour l'analyse des programmes parallèles. Le parallélisme ayant pour objectif d'optimiser les

temps d'exécution, des outils ont été développés pour analyser les communications et le problème de la répartition de charge. Ces outils utilisent des traces d'exécution post-mortem pour visualiser le comportement des algorithmes. On peut citer deux outils: PARAGRAPH utilisé principalement par Intel et Athapascan développé par une équipe de recherche grenobloise [APA94].

3.6 Evaluation des performances

L'emploi du parallélisme a pour but de diminuer le temps de résolution d'un problème. Comme nous l'avons évoqué au paragraphe précédent, des outils ont été développés dans ce sens. Ils sont basés sur des critères définissant si la parallélisation d'un algorithme est efficace. La simple mesure du temps d'exécution d'un programme parallèle ne permet pas de juger de la qualité de son implémentation. De plus, il est nécessaire de disposer de critères globaux permettant de comparer les temps d'exécution. De tels critères sont utilisés pour modéliser le comportement de l'algorithme en faisant varier la taille du problème ou le nombre de processeurs. Cette modélisation permet d'évaluer les performances de l'algorithme pour la résolution de problèmes de plus grande taille.
Nous utilisons dans notre étude les critères suivants:

Le facteur d'accélération Il est calculé en considérant un problème de taille fixe. Soit un algorithme qui résout un problème de taille N en un temps $T_{seq}(N)$. Soit sa version parallèle sur PE processeurs qui résout le même problème en un temps $T_{//}(N, PE)$.
On appelle facteur d'accélération, ou "Speed Up", le rapport suivant:

$$Acc(N, PE) = \frac{T_{seq}(N)}{T_{//}(N, PE)} \tag{3.61}$$

Généralement, on étudie ce critère lorsque N est fixé et que PE varie. On prend comme temps séquentiel $T_{seq}(N)$, le temps du meilleur algorithme séquentiel. Dans la plupart des cas, on prend $T_{seq}(N) = T_{//}(N, 1)$ pour chaque machine utilisée.

Le facteur d'accélération relative Nous introduisons ce critère pour comparer les implémentations des algorithmes sur les différentes machines que nous utilisons. Il nous permet de juger si une implémentation est plus performante

sur une machine que sur une autre. On se sert d'une machine de référence séquentielle pour calculer ce critère en fonction du temps de référence $T_{ref}(N)$ et du temps d'exécution sur une machine parallèle. Nous utilisons ce critère essentiellement pour comparer nos implémentations sur une machine SIMD et sur les machines MIMD. Il se formule par:

$$Acc_{ref}(N, PE) = \frac{T_{ref}(N)}{T_{//}(N, PE)} \tag{3.62}$$

Le facteur d'efficacité Il permet de mesurer les performances de l'algorithme parallèle pour un nombre fixé de processeurs. Ce critère indique si les processeurs sont utilisés de façon optimale. Il se formule par:

$$Eff(N, PE) = \frac{Acc(N, PE)}{PE} \tag{3.63}$$

Plus ce critère est proche de 1, plus l'implémentation est optimale sur PE processeurs. Un algorithme parallèle efficace est fonction de la partie purement séquentielle de l'algorithme, des temps de communication des données entre les processeurs et des temps d'attente dus à des synchronisations pour effectuer un calcul en parallèle.

Nous présenterons des techniques de répartition de charge et de recouvrement des calculs par les communications permettant d'optimiser les algorithmes parallèles.

Pour certaines implémentations, il est possible d'obtenir des efficacités supérieures à 1, ce qui équivaut à dire que le facteur d'accélération est supérieur au nombre de processeurs. On parlera, dans ce cas, d'efficacité et d'accélération "superlinéaires". Ce phénomène se produit avec des architectures utilisant des mémoires caches de taille importante. Si l'on désire résoudre un problème de taille N sur une machine parallèle à PE processeurs, on répartit le problème en PE sous-problèmes de taille $\frac{N}{PE}$ dans le cas d'une application de parallélisme de données. On peut supposer que la taille de la mémoire cache du processeur est supérieure à $\frac{N}{PE}$ mais inférieure à N. Ainsi, en diminuant le nombre d'accès à la mémoire, l'implémentation parallèle est accélérée de façon artificielle.

Cela montre la limitation de ces critères qui permettent toutefois de donner des informations sur le comportement de l'implémentation parallèle. D'autres critères ont

été proposés par Gustafson [Gus88] qui propose un facteur d'accélération intégrant une notion de la taille de la machine et, par Desbat et Colombet, qui étend les notions d'accélération et d'efficacité pour les réseaux de processeurs hétérogènes [Col94]

3.7 Conclusion

Notre introduction sur le parallélisme nous a permis de présenter les principaux modèles d'exécution et de programmation. Ces modèles complémentaires forment la base de toute application parallèle. Un programmeur doit construire son application en fonction du caractère de celle-ci et de l'implémentation de son algorithme. Pour notre étude, nous utilisons des machines parallèles appartenant à plusieurs modèles d'exécution.

Nous expliciterons, pour chaque machine utilisée, son mode de fonctionnement et ses principales caractéristiques. L'évolution rapide de l'informatique parallèle a généré de nombreuses machines qui sont aujourd'hui considérées comme dépassées. La tendance actuelle au niveau des architectures est l'utilisation de machines MIMD à mémoires distribuées. Les processeurs utilisés sont souvent des processeurs standards pour limiter les coûts de développement. Leur réseau d'interconnexion est soit un tore 2D ou 3D soit un réseau multi-étage. Au niveau des logiciels, on peut noter l'effort de standardisation pour les bibliothèques de communication comme MPI ou pour les langages, comme HPF. Nous utilisons, pour la plupart de nos implémentations, le standard actuel des bibliothèques de communication: PVM. Notre choix a été déterminé par la souplesse de cette bibliothèque et la portabilité des applications sur les machines que nous utilisons.

Nous avons explicité les critères de performance qui nous permettront de juger nos implémentations parallèles. Ces critères permettent d'apprécier le taux de parallélisme qu'un algorithme peut obtenir.

Nous avons évoqué dans le chapitre précédent, le problème de la mise en œuvre des méthodes de reconstructions 3D. On peut noter que la plupart des méthodes effectuent une série d'opérations similaires sur un ensemble discrétisé de points. La dimensionalité de cet ensemble de points (68 millards de points pour une image

3D de taille 512^3) est un frein pour la reconstruction d'images de taille réaliste. Les approches parallèles, qui ont été développées, ont tenté de reduire ce problème par l'utilisation d'un parallélisme de données. Dans le chapitre suivant, nous présenterons de nombreux travaux basés sur ce modèle de programmation et implémentés sur des machines de modèles d'exécutions très différents.

De ce fait, nos approches sont basées sur un modèle de parallélisme de données. Nous détaillerons notre méthodologie pour la parallélisation des algorithmes de reconstruction 3D en tomographie X.

Chapitre 4

Méthodologie

4.1 Introduction

Nous pouvons aborder la parallélisation des méthodes de reconstruction 3D par deux approches différents. La première consiste à traiter le problème de reconstruction 3D comme un problème classique d'algèbre linéaire. La parallélisation peut alors être basée sur l'utilisation de routines parallèles d'algèbre linéaire. La seconde approche consiste à conserver la structure des méthodes de reconstruction, et à identifierles noyaux de bases les plus coûteux. La parallélisation intervient dans ce cas au niveau de l'implémentation des opérateurs.

Nous présentons dans ce chapitre les différentes approches pour implémenter les opérateurs de base des méthodes de reconstruction 3D. Nous définissons dans un premier temps le cadre de notre étude pour fixer la géométrie d'acquisition, et pour identifier les opérateurs de base. Les opérateurs discrets déterminés à partir de ces opérateurs sont parallélisés suivant deux approches: une approche locale et une approche globale.

Les architectures de nos machines cibles étants très différentes, nous utilisons le concept d'une machine parallèle abstraite pour paralléliser nos algorithmes. Son modèle d'exécution est définie dans ce chapitre. Pour minimiser les temps de communication, nous optimisons nos implémentations en proposant des algorithmes basés sur des recouvrements des calculs par les communications, des schémas de communication plus efficaces, et un partitionnement adaptatif des données. Nous détaillerons

l'organisation modulaire des programmes pour mettre en œuvre les méthodes sur nos machines cibles.

4.2 Objectifs de la parallélisation

4.2.1 Contraintes de la tomographie 3D

Les nouveaux systèmes d'acquisition et les méthodes qui découlent de ces systèmes, posent de nouveaux enjeux pour la reconstruction des images 3D. Nous décrivons les problèmes liés à ces méthodes par un exemple significatif de reconstruction 3D. Soit la reconstruction d'une image 3D de taille 256^3 à partir de 256 acquisitions 256^2, avec des images codées sur 4 bytes pour chaque pixel ou voxel, les contraintes de la reconstruction sont:

La gestion des données: la taille des données 3D amène par son volume des problèmes de stockage et de gestion des entrées-sorties. Ainsi, pour notre exemple, la taille des données est de $64Mbytes$, pour l'image 3D à reconstruire, et de $256.256^2 Kbytes = 64Mbytes$, pour les acquisitions, ce qui représente un volume total de $128Mbytes$ de données à gérer.

La taille du système discret de reconstruction: ces nouveaux systèmes d'acquisition génèrent des grands systèmes linéaires souvent creux. Nous évaluons la taille du système discret de reconstruction pour quelques méthodes:

- Algorithme de Feldkamp: cette méthode analytique a une complexité en $m.N^3$ avec m le nombre d'acquisitions, de taille M^2, et N^3 la taille de la matrice de l'image 3D. L'opération de filtrage est en $M.log(M)$, ce qui donne au total un algorithme en $M.N^3 + m.M.log(M)$. Donc pour notre exemple, le système discret de reconstruction a pour taille totale $16Gbytes$. Elle est cependant très creuse.
- Méthode ART: Cette méthode algébrique a une coût en fonction du nombre d'itérations (NI) et du coût de chaque itération. Une itération est composée, pour chaque acquisition, d'une projection et d'une rétroprojection. Le coût d'une projection et d'une rétroprojection étant en N^3, le coût total est de $NI.2.m.N^3$. Pour notre exemple, nous obtenons un système discret de reconstruction de taille $NI.32Gbytes$.

La taille des systèmes discrets de reconstruction est démesurée par rapport aux capacités de la mémoire d'une station de travail classique qui a au plus $128 Mbytes$ de mémoire.

Les temps d'exécution: les nouveaux systèmes d'acquisition ont été développés en partie pour donner des informations pré-opératoires. Le tableau 3 présente des temps d'exécution de l'algorithme de Feldkamp pour reconstruire une image 256^3 à partir de 512 acquisitions 256^2 [Jac88]. En analysant ce tableau, on peut remarquer que les implémentations sur machine séquentielle nécessitent des temps de calcul prohibitifs pour une utilisation de ces techniques en milieu hospitalier.

Machines	SUN4-260	DEC VAX 8530	DEC VAX 8700	IBM 3090
Temps (heures)	44	42	28	5,5

TAB. 3 - *Temps d'exécution de l'algorithme de Feldkamp [Jac88]*

Ces contraintes montrent que la reconstruction d'une image 3D sur une machine séquentielle requièrt des mémoires de grandes tailles et des temps de calcul très importants. Les approches séquentielles classiques ne sont donc pas adaptées pour la reconstruction d'image 3D lorsque $N > 64$.

Pour pallier ces contraintes qui sont identiques dans la plupart des applications scientifiques, les numériciens se sont tournés vers des méthodes développées sur des machines parallèles pour améliorer les temps de reconstruction et pour envisager la reconstruction d'image 3D de taille acceptable.

4.2.2 Approches parallèles existantes

Nous présentons les approches parallèles qui ont déjà été mises en œuvre pour paralléliser les problèmes de reconstruction en tomographie. Nous les avons classées en fonction de leur modèle d'exécution:

Les architectures dédiées

Un grand nombre de projets en tomographie ont eu pour but de construire des machines dédiées à la tomographie. Une première étude [TP81], basée sur l'algorithme

de rétroprojection, a proposé un circuit pour accélérer les calculs d'adresse. A la suite de cette étude, l'emploi de circuits spécialisés basés sur des microprocesseurs programmables [HBM85] s'est généralisé. Des architectures VLSI ont été spécialement développées pour des algorithmes de reconstruction. En analysant les étapes d'un programme de reconstruction algébrique pour la tomographie PET, Jones [JBC88] a défini une machine dont l'architecture tient compte des flots de données du programme. Dans un autre projet [JBC90], il a proposé une architecture utilisant des circuits VLSI pouvant reconstruire, soit une image 3D de taille 128x128x32, soit 4096 vues 2D. La reconstruction d'une image 3D nécessite une réorganisation des rayons de projection. Cette architecture obtient des performances de l'ordre de la minute pour reconstruire une image 3D. Une autre architecture a utilisé le concept de pipe-line des architectures vectorielles pour définir un processeur performant pour l'implémentation physique de la plupart des algorithmes de reconstruction [CA89].

La conception de processeurs spécialisés s'est ensuite orientée sur une architecture de machines ayant pour modèle d'exécution le modèle SIMD. Dans ce contexte, Borges [BdA89] montre que la meilleure architecture est une machine constituée de processeurs dédiés, connectés suivant une topologie en grille. En utilisant ce résultat, Lattard [Lat88] a développé une machine massivement parallèle dédiée. Elle est composée de processeurs élémentaires fonctionnant en mode asynchrone. Chaque processeur réalise des traitements élémentaires sur ses données locales et communique des données aux autres processeurs par passage de message. Cette parallélisation est identifiable à celle utilisée par les machines MIMD à passage de message. D'un point de vue général, les performances obtenues avec ce type de machine sont souvent supérieures à celles obtenues sur des machines vectorielles et sur des machines SIMD. Le problème reste le développement de ces architectures qui sont souvent figées au niveau des algorithmes de reconstruction et au niveau de la taille des données à reconstruire. Ainsi le coût de ces machines est acceptable si l'on considère le rapport coût/performances, mais il devient prohibitif en raison de la spécificité de chaque architecture. Cela explique le développement des applications de reconstruction sur d'autres modèles d'exécution.

Les architectures vectorielles

Les approches utilisant ce modèle d'exécution ont utilisé les spécificités des machines vectorielles pour obtenir de bonnes performances. Des implémentations ont été vectorisées avec les outils de vectorisation automatique fournis avec ce type de machine. Le langage utilisé, généralement le Fortran, permet de mettre à jour des boucles qui sont vectorisées automatiquement lors de la compilation du programme. Peyrin [Pey90] a implémenté une méthode de reconstruction analytique sur un Cyber 205 et sur un Eta 10 de Control Data, en utilisant un Fortran vectoriel. Elle reconstruit des images 3D de petites tailles $(32, 64)$ dans des temps acceptables: 3,9 secondes pour une reconstruction d'une image de taille 32^3 à partir de 36 projections et 25,7 secondes pour une image de taille 64^3 à partir de 100 projections.

D'autres approches consistent à identifier les noyaux de calculs parallélisables et à utiliser au mieux les routines parallèles fournies avec la machine. Kaufman [Kau87] propose d'accélérer un algorithme issu d'une méthode analytique en développant des algorithmes basés sur des BLAS. Ces implémentations réalisées sur Cray 1 sont efficaces car cette machine utilise un langage assembleur basé sur les BLAS . Une autre solution, proposée par Guerrini [GS89], est de découper le programme séquentiel en fonction des opérations déjà parallélisées comme la FFT (Fast Fourier Transform). Il obtient des temps de l'ordre de deux secondes pour reconstruire une image 2D (128^2) à partir de 128 projections de taille 256.

Le problème de la reconstruction avec ce type de machine est résolu en adaptant les algorithmes aux caractéristiques des machines vectorielles. Les performances obtenues en 2D ou en 3D sur des problèmes de petite taille sont satisfaisantes. Cependant ces machines utilisent des langages spécifiques limitant souvent la portabilité des algorithmes. De plus, étant limité au niveau de la puissance de calcul, cela empêche le traitement de problèmes de grande taille. Pour résoudre ce problème de puissance, les utilisateurs de ce type de machine se sont tournés vers les machines MIMD vectorielles.

Les machines SIMD

les premières approches, développées sur cette classe de machine, reconstruisent des images 2D pour la tomographie SPECT. Mc Carty [MM91] utilise l'architecture de la machine SIMD, une grille 2D, pour simuler la propagation du photon sur un espace discrétisé représenté par une grille. Il reconstruit des images 2D de tailles 32^2 et 64^2 par des méthodes itératives sur des machines SIMD composées de 32^2 ou 64^2 processeurs. Cette similitude entre l'espace discrétisé et l'architecture de la machine permet d'obtenir des bonnes performances (moins de 10 secondes par itération) et une bonne simulation de l'atténuation. Pour reconstruire l'image 2D, il suffit de faire tourner les données sur la grille de processeur pour simuler chaque projection. En tomographie SPECT 3D, Miller [MB93] propose une reconstruction semblable en utilisant la mémoire de chaque processeur pour simuler la troisième dimension. Les opérations de projection consistent en une sommation des effets d'atténuation et de flou sur chaque plan de projection. La rotation du système source-détecteurs est effectuée par rotation de chaque plan en mémoire. La rétroprojection consiste alors à répartir la différence entre les projections calculées et les projections mesurées pour chaque plan. Ces implémentations utilisent l'architecture de la machine pour paralléliser les opérations de projection et de rétroprojection. Il reconstruit des images de taille 64^3 à partir de 96 acquisitions en 400 secondes sur une machine composée de 64^2 processeurs et en 150 secondes sur une machine composée de 128^2 processeurs. Cette parallélisation est efficace car le système d'acquisition utilise une géométrie parallèle. En effet, la géométrie parallèle permet d'effectuer les opérations de base de manière synchrone.

Dans le cas d'une géométrie conique, les opérations divergentes de projection et de rétroprojection ne peuvent pas être simulées sur l'architecture d'une machine classique. Une solution est de redéfinir le problème en opérations élémentaires et de considérer le problème de la reconstruction comme un problème d'analyse numérique. Les algorithmes sont développés autour de routines efficaces de la machine utilisée. On peut citer les travaux de Cook [CD92] qui a utilisé cette approche pour paralléliser une méthode temps réel en tomographie quantitative. Les temps de reconstruction sur une machine parallèle dépendent souvent de la répartition des données. Rajan [RPR94] a étudié le problème de l'équilibrage des charges sur une

topologie en hypercube étendue, pour une reconstruction en tomographie PET. Il compare, pour deux répartitions de données possibles, le coût des communications en fonction de chaque partitionnement. Il montre qu'un partitionnement optimal est souvent un facteur d'accélération des implémentations parallèles. Nous le vérifierons dans le paragraphe suivant.
La plupart des études proposées avec ce modèle d'exécution sont efficaces pour des systèmes d'acquisition à géométrie parallèle pour des reconstructions d'image 3D de taille maximale de 128^3 voxels. En raison de la faible capacité des mémoires des processeurs élémentaires, les machines SIMD ne permettent pas de travailler sur des données de grande taille. De plus, la géométrie conique n'est pas adaptée à ce type de machine. L'utilisation de machines MIMD d'une grande capacité mémoire permet de résoudre des problèmes en géométrie conique.

Les machines MIMD

Le temps total d'exécution d'une application parallèle dépend du temps d'exécution sur chaque processeur. L'équilibrage de charge des processeurs permet d'optimiser la parallélisation. Dans cette optique, Atkins [AMH93] a proposé une architecture MIMD basée sur un réseau de 17 Transputers. Elle est formée de deux sortes de processeurs: les Workers au nombre de 16, qui calculent chacun la reconstruction d'une partie de l'image 3D, et le Master, qui gère l'équilibrage de charge entre les Workers. Une autre solution est de définir des partitionnements optimaux. Chen [CLC91] a étudié des solutions pour la répartition sur une machine à mémoire partagée et sur une machine à mémoire distribuée. Il reconstruit une image 3D en PET par une méthode algébrique. La matrice de projection est répartie par colonnes ou par lignes sur chacun des processeurs. Il met en évidence que les implémentations sur machines à mémoire partagée sont moins efficaces que celles sur machines à mémoire distribuée. Dans une autre étude, il utilise des partitionnements non-homogènes qui permettent de réduire le coût des communications en utilisant des techniques de recouvrement de calcul par les communications [CL94]. Pour l'implémentation de méthodes analytiques, il découpe l'algorithme en opérateurs de base. Le schéma de communication est un arbre binaire plongé sur une topologie en hypercube [CLC90].

Pour minimiser les communications et tirer partie de la topologie en hypercube, Zapata [ZBR+90] répartit les données suivant un code de Gray. Cet adressage des

données par le code de Gray permet de définir sur quelle dimension de l'hypercube les communications sont réalisées. Des projets se sont intéressés au mode de projection en PET. Barresi [BBG90] utilise un réseau de Transputers pour paralléliser une méthode de reconstruction en mode "Ray-tracing". Ce mode impliquant la communication des processeurs se trouvant sur une même ligne de projection, le coût des communications en est donc très important.

Pour conclure on peut citer les résultats de Charles [CLM93] qui a implémenté un algorithme de rétroprojection sur un IPSC, en utilisant un anneau comme schéma de communication. Il reconstruit des images 3D de taille 32^3 à 256^3 par une méthode analytique. Les résultats montrent que le temps de reconstruction décroit inversement au nombre de processeurs. Cependant pour un problème de taille fixe, l'accélération ne croit pas en fonction du nombre de processeurs. Cela indique qu'il existe un nombre optimal de processeurs pour implémenter un problème de taille fixe sur une machine parallèle. De plus, pour un nombre fixé de processeurs, l'efficacité augmente quand la taille du problème augmente. Ces remarques signifient que les problèmes de grandes tailles de reconstruction 3D semblent bien adaptés aux machines MIMD.

4.2.3 Classification des approches

Ces approches parallèles mettent en évidence le modèle de programmation utilisé pour implémenter les algorithmes de reconstruction. Le problème de la reconstruction peut se ramener au problème suivant:
soit deux ensembles de données 3D liés par une relation, les données résultats sont obtenues par des opérations sur les données initiales; on peut représenter schématiquement le problème de la reconstruction:

- Soit $\{D^0, D^1, D^k, D^n\}$, l'ensemble des espaces de données. D^0 représente l'espace des données initiales, D^k un espace de données intermédiaires et D^n l'espace de données résultats.

- Soit $\{Op^0, Op^1, Op^{n-1}\}$, l'ensemble des opérateurs qui transforment des données d'un espace de données à un autre. Par exemple la transformation Op^i s'exprime par $D^i \overset{Op^i}{\rightarrow} D^{i+1}$.

Le problème de la reconstruction se formalise alors par une série d'opérations:

$$D^0 \overset{Op^0}{\rightarrow} D^1 \cdots D^i \overset{Op^i}{\rightarrow} D^{i+1} \cdots D^{n-1} \overset{Op^{n-1}}{\rightarrow} D^n \tag{4.64}$$

Les opérations Op^i sont des opérations de projection, de rétroprojection, de filtrage, de convolution, de transformée de Fourier... . Ces opérations sont assimilables à un ensemble de tâches. Dans l'ensemble des méthodes, on ne peut effectuer ces tâches que suivant un ordre chronologique. Pour cette raison, le modèle de parallélisme mis en œuvre dans la plupart de ces approches est un parallélisme de données.

Ces approches utilisant des machines basées sur des modèles d'exécution très différents, peuvent être toutefois implémentées de façons similaires. Cependant on peut distinguer deux classes d'approches qui sont fonction de la granularité du parallélisme utilisé. La notion de granularité du parallélisme est définie ici suivant la parallélisation des méthodes. Les implémentations qui utilisent le parallélisme au niveau du traitement de chaque pixel ou chaque voxel, sont définies comme des implémentations parallèles à grain fin. Réciproquement, les implémentations basées sur la parallélisation des opérateurs de base sont définies comme des implémentations à gros grain. Le parallélisme à grain fin est utilisé dans des approches issues d'une analyse ascendante du problème de la reconstruction, tandis qu'un parallélisme à gros grain est utilisé pour des approches issues d'une analyse descendante.

Analyse ascendante

Un grand nombre de méthodes de reconstruction 3D, parmi les méthodes analytiques discrétisées et les méthodes algébriques sont assimilables à la résolution de systèmes linéaires creux. Une approche pour paralléliser ces méthodes est d'utiliser des noyaux de calculs parallèles ou de développer des processeurs dédiés aux traitements d'un petit nombre de voxels. Ces noyaux de calculs parallèles sont souvent déterminés par les caractéristiques de la machine utilisée et par l'emploi de bibliothèques de calculs. Cette approche consiste donc à construire les opérateurs de base à partir de routines parallèles déjà programmées. Les implémentations sur machines dédiées, sur machine vectorielles et pour certaines d'entre elles sur machine SIMD, sont basées sur une analyse ascendante des problèmes de reconstruction. Généralement pour ce type d'approche, l'efficacité de l'implémentation est privilégiée au détriment de sa portabilité.

Analyse descendante

Une autre approche part du constat suivant: la plupart des méthodes de reconstruction sont fondées suivant des modèles mathématiques différents mais utilisent des opérateurs similaires comme la projection, la rétroprojection, la convolution, les transformations de Fourier... . Cette approche consiste donc à paralléliser ces opérateurs et à les intégrer dans les différentes méthodes. Ce type d'analyse amène à définir une bibliothèque d'opérateurs parallèles. Ces opérateurs sont parallélisés suivant des approches différentes en fonction de leur utilisation. Ainsi, les méthodes de reconstruction sont construites à partir de cette bibliothèque qui se veut portable et adaptable à toutes les méthodes. On retrouve cette analyse descendante des problèmes de reconstruction dans la plupart des implémentations MIMD. Nous l'utilisons donc pour paralléliser des méthodes de reconstruction sur nos machines cibles, qui sont principalement des machines MIMD.

Notre étude consiste, dans un premier temps, à identifier et à paralléliser les opérateurs de base éléments d'une telle bibliothèque et, dans un deuxième temps à développer des méthodes à partir de ces opérateurs de base [LPC95c]. Les algorithmes présentés ont pour modèle d'exécution notre machine parallèle abstraite qui englobe la plupart des modèles d'exécution de nos machines parallèles.

4.3 Cadre de l'étude

4.3.1 Identification des opérateurs de base

En tomographie par transmission comme en tomographie par émission, les acquisitions sont considérées comme des projections de l'image à reconstruire. L'expression de cette opération de projection est fonction de la géométrie d'acquisition. On modélise le processus physique par un opérateur continu de projection. Les mesures ou acquisitions sont effectuées lors d'une projection sur un ensemble de cellules appelé détecteur. Ce détecteur est modélisé par un ensemble de points définis lors de l'échantillonnage des mesures sur ce détecteur. L'opération de projection analytique est donc estimée par un opérateur de projection discret. La qualité de l'approximation dépend du pas d'échantillonnage. On assimile souvent cet opérateur de projection analytique à son expression discrète, car ce dernier est utilisé lors de l'implémentation des méthodes de reconstruction.

Les méthodes de reconstruction utilisent des opérateurs qui effectuent une opération duale à la projection. Nous utilisons l'opérateur de rétroprojection discret. Celui-ci peut être considéré comme l'opérateur dual de l'opérateur de projection car il permet de passer d'un espace 2D à un espace 3D, tandis que l'opérateur de projection discret permet de passer d'un espace 3D à un espace 2D. Nous détaillons leurs expressions dans le cadre d'une géométrie d'acquisition conique.

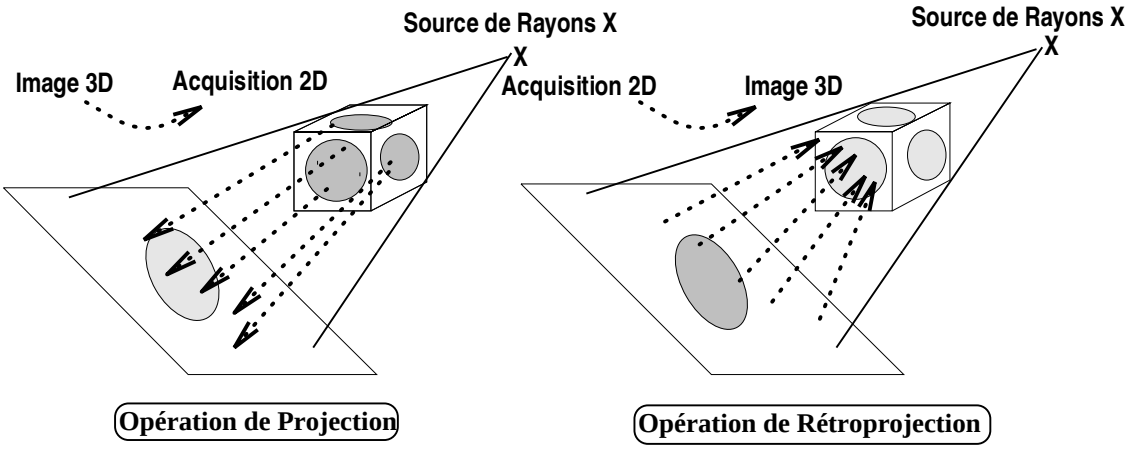


FIG. 23 - *Dualité des opérateurs*

4.3.2 Géométrie d'acquisition

Les méthodes de reconstruction 3D présentées dans le chapitre 2 sont basées sur des systèmes d'acquisition où la source décrit soit une sphère, soit deux cercles dans des plans orthogonaux, soit un seul cercle. Nous avons constaté que la plupart des systèmes d'acquisition se limitent à des géométries simples à mettre en œuvre pour des raisons technologiques. Nous choisissons une géométrie d'acquisition modélisant celles utilisées par les prototypes d'acquisitions en tomographie 3D comme le DSR ou le Morphomètre. Celle-ci est donc constituée d'une source et d'un détecteur plan tournant autour du patient suivant une trajectoire circulaire. Notre étude se limite alors à la tomographie par rayons X pour simplifier l'expression des formules de reconstruction que nous mettons en œuvre. Cependant les algorithmes que nous développons sont aussi applicables dans le cas de la tomographie par émission.

Le repère orthonormé $(O, \vec{i}, \vec{j}, \vec{k})$ est le repère de l'espace de reconstruction. On choisit comme origine de ce repère le centre de la zone à reconstruire. On suppose

que l'axe du faisceau de rayons X passe toujours par l'origine O et est déterminé par la droite (OS) où S est la position de la source. Le plan des détecteurs est orthogonal à cette droite, et on appelle S' le point d'intersection de cette droite avec le plan. S' est la projection de S sur le plan des détecteurs. L'ensemble source-détecteur effectue une rotation autour de l'objet à reconstruire. Cette rotation de la source est repérée par deux angles Θ et Φ. Le vecteur directeur $\vec{l}$ de la rotation est déterminé par:

$$\vec{l} = \begin{pmatrix} \cos\Phi \sin\Theta \\ \sin\Phi \sin\Theta \\ \cos\Theta \end{pmatrix} \tag{4.65}$$

On exprime pour une position initiale de la source S, ses coordonnées et les coordonnées de sa projection S' avec $D_1 = \|\overrightarrow{OS}\|$, $D_2 = \|\overrightarrow{OS'}\|$ et $D = D_1 + D_2$.

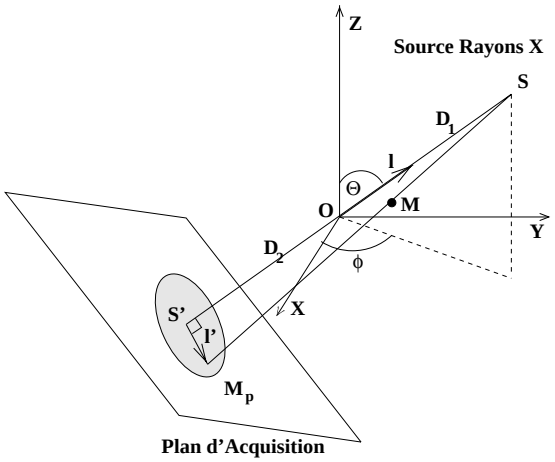


FIG. 24 - *Projection en géométrie conique*

A un instant donné, la position de la source est repérée par les angles Θ et Φ. Le système source-détecteurs effectue deux rotations. Par un simple changement de variables, l'expression de chaque projection peut être ramenée au cas initial ce qui permet d'exprimer chaque projection avec le même repère. Pour simplifier l'écriture on se ramènera au cas initial en effectuant la rotation inverse: au lieu de faire tourner l'ensemble source-détecteurs, on fait tourner la zone à reconstruire. Un point M de cet espace est défini dans le repère $(O, \vec{i}, \vec{j}, \vec{k})$ par ses coordonnées ${}^t(x, y, z)$ ou par son vecteur $\vec{r} = \overrightarrow{OM}$. Ses nouvelles coordonnées ${}^t(X, Y, Z)$ après la rotation inverse

se calculent, d'après le système suivant:

$$\begin{cases} X &= & x\sin\Phi & -y\cos\Phi & \\ Y &= & x\cos\Phi\cos\Theta & +y\sin\Phi\cos\Theta & -z\sin\Theta \\ Z &= & x\cos\Phi\sin\Theta & +y\sin\Phi\sin\Theta & +z\cos\Theta \end{cases} \tag{4.66}$$

Ainsi S et S' conservent les mêmes coordonnées, et les points du plan de détection P ont pour coordonnées: ${}^t(u, v, -D_2)$. La fonction f est supposée nulle en dehors d'une sphère centrée en l'origine. Cette sphère est l'intersection des cônes de projection déterminés par l'angle α du faisceau de rayon X. On peut établir la relation reliant la taille du cube (T_{cube}) inscrit dans cette sphère, qui détermine la région de reconstruction de la fonction f, en fonction de la taille des plans de détection T_{plan}:

$$T_{plan} = 2D\tan(\alpha/2) \tag{4.67}$$

$$\Rightarrow \alpha/2 = \tan^{-1}(T_{plan}/2D) \tag{4.68}$$

$$T_{cube} = \sqrt{3}D_1\sin(\alpha/2) \tag{4.69}$$

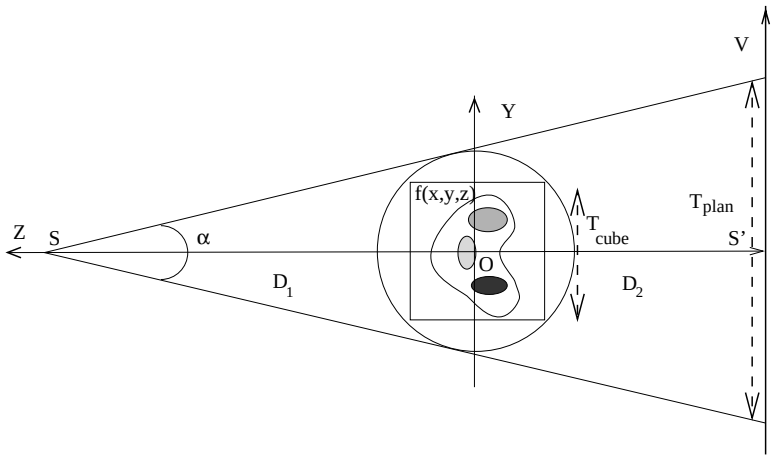


FIG. 25 - *Zone de reconstruction*

La géométrie de notre système d'acquisition étant fixée, nous définissons dans les paragraphes suivants l'expression et les algorithmes des opérateurs de base: la projection et la rétroprojection.

4.3.3 Discrétisation des données

Les méthodes de reconstruction permettent d'établir une relation reliant l'espace des mesures à l'espace résultat contenant l'image 3D à reconstruire. Pour

implémenter ces méthodes, on modélise ces espaces par des espaces discrets de dimension finie.

L'espace des mesures est représenté par un ensemble de m acquisitions 2D. Ces acquisitions sont effectuées sur un plan de détection qui se modélise par une grille de cellules détectrices. Chaque acquisition 2D est définie comme la projection 2D de l'image 3D sur la grille de détecteurs. La grille est formée de M^2 pixels chacun de dimension $\triangle_p^2$. L'ensemble des mesures est alors formalisé par l'ensemble discret $\{Im_j / 1 \leq j \leq m\}$ où chaque image Im_j représente une acquisition. Ces images sont constituées de M^2 pixels notés im_{ja} avec $1 \leq a \leq M^2$. Chaque pixel a pour coordonnées (p, q) sur l'image de projection: $im_{ja} = Im_j(p, q)$ avec $1 \leq p \leq M$ et $1 \leq q \leq M$.

La fonction f qui appartient à l'espace résultat est échantillonnée sur un volume cubique V suivant un pas d'échantillonnage $\triangle_v$. Le volume V est constitué de N^3 éléments unitaires appelés voxels notés v ou $v(i, j, k)$ avec $1 \leq i \leq N$; $1 \leq j \leq N$ et $1 \leq k \leq N$. Chaque voxel $v(i, j, k)$ représente la valeur de la fonction égale à $f(i\triangle_v, j\triangle_v, k\triangle_v)$. On définit ainsi une fonction f_d qui est la fonction discrétisée de f sur l'ensemble des voxels.

$$f_d(x, y, z) = \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{N} f(i\triangle_v, j\triangle_v, k\triangle_v)\delta(x - i\triangle_v)\delta(y - j\triangle_v)\delta(z - k\triangle_v) \quad (4.70)$$

où $\delta(x)$ est la distribution de Dirac. Pour un pas d'échantillonnage assez fin, on peut considérer que la fonction f_d est une bonne approximation de f. Nous utilisons cette fonction pour nos reconstructions. La relation entre l'ensemble des acquisitions et la fonction à reconstruire dépend du mode de projection utilisé:

- Le mode le plus naturel est le "Ray-tracing". Il consiste à partir d'un point P du plan de projection et à sommer tous les voxels qui se trouvent sur la droite reliant P à la source S. Chaque voxel sommé est affecté d'un poids égal à la longueur de l'intersection entre la droite (PS) et le voxel,

- Un autre mode est le "Voxel-driven". Il consiste à projeter chaque voxel sur la grille des détecteurs. Il nécessite de choisir une méthode d'interpolation afin de répartir la valeur du voxel sur les points discrétisés du détecteur. Nous avons choisi une interpolation bilinéaire qui permet de répartir la valeur du voxel sur

les quatre points du détecteur les plus proches du point où le voxel est projeté. Cela revient donc à sommer sur un point du détecteur tous les voxels qui sont projetés à une distance inférieure au pas d'échantillonnage de la grille.

Nous comparons ces deux modes de projection en 2D. On projette les pixels d'une grille, de pas d'échantillonnage $\triangle_{grille}$, sur un détecteur ligne de pas d'échantillonnage $\triangle_{ligne}$. Ces figures montrent le nombre de pixels intervenant dans le calcul d'un point du détecteur lorsque $\triangle_{grille} = 2 * \triangle_{ligne}$ (figure 26.a), $\triangle_{grille} = \triangle_{ligne}$ (figure 26.b), $2 * \triangle_{grille} = \triangle_{ligne}$ (figure 26.c).

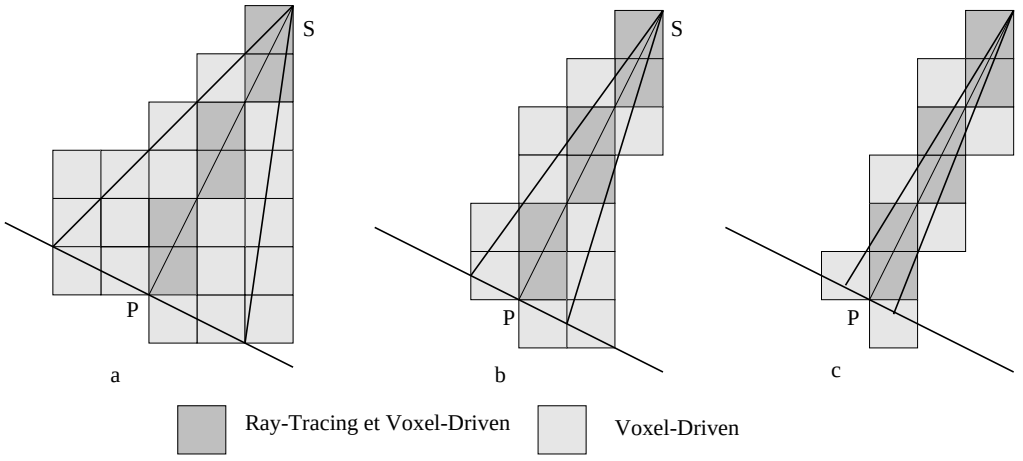


FIG. 26 - *Comparaison du mode Ray-tracing et du mode Voxel-driven dans le cas où a)* $\triangle_{grille} = 2 * \triangle_{ligne}$ *b)* $\triangle_{grille} = \triangle_{ligne}$ *et c)* $2 * \triangle_{grille} = \triangle_{ligne}$

Dans chacun des cas le nombre de pixels intervenant pour le calcul d'un point est plus important avec le mode "Voxel-driven". L'estimation discrète définie par ce mode est donc meilleure que le "Ray-tracing". Nous l'utilisons pour discrétiser les opérateurs de base. Au point de vue de la parallélisation, nous avons montré que les méthodes algébriques se parallélisent mieux si on emploie le mode de projection "Voxel-Driven" plutôt que le "Ray-Tracing"[DLR95].

4.3.4 Opérateurs discrets de projection et de rétroprojection

Les projections coniques sont déterminées suivant la position de l'ensemble source-détecteurs. Dans notre modèle discret, on effectue m projections de l'image à reconstruire sur les images de projections Im_j où j est fonction des angles Θ et Φ.

Le changement de variable permet de considérer chaque projection comme une projection sur le plan défini par $Z = -D_2$ et par son repère $(S', \vec{i}, \vec{j})$. Dans ce nouveau repère, la projection conique $p_{\Theta\Phi}$ au point P est égale à l'intégrale de la fonction f sur la droite (SP) et peut être exprimée d'après l'équation:

$$p_{\Theta\Phi}(u,v) = \int_{-\infty}^{+\infty} f\Big(\frac{D_1 - Z}{D}u, \frac{D_1 - Z}{D}v, Z\Big)\frac{\sqrt{D^2 + u^2 + v^2}}{D}dZ \tag{4.71}$$

En utilisant le mode de projection "Voxel-driven", pour discrétiser l'opérateur de projection conique, on définit pour chaque opération de projection $p_{\Theta\Phi}$ une opération de projection interpolée $\tilde{p}_{\Theta\Phi}$ sur la grille de détecteurs. Chaque projection interpolée $\tilde{p}_{\Theta\Phi}$ s'exprime comme une convolution de la projection $p_{\Theta\Phi}$ avec une fonction d'interpolation $I(u,v)$:

$$\tilde{p}_{\Theta\Phi}(u,v) = \sum_{p=1}^{M}\sum_{q=1}^{M} p_{\Theta\Phi}(p\triangle_p, q\triangle_p) I(u - p\triangle_p, v - q\triangle_p) \tag{4.72}$$

En utilisant une interpolation bilinéaire pour $I(u,v)$, l'équation précédente devient:

$$\begin{aligned}\tilde{p}_{\Theta\Phi}(u,v) &= (1-\lambda)(1-\mu)\, p_{\Theta\Phi}(p,q) + \lambda(1-\mu)\, p_{\Theta\Phi}(p+1,q) + \\ &\quad + \mu(1-\lambda)\, p_{\Theta\Phi}(p,q+1) + \lambda\mu\, p_{\Theta\Phi}(p+1,q+1)\end{aligned} \tag{4.73}$$

avec $\lambda = p + 1 - u/\triangle_p$ et $\mu = q + 1 - v/\triangle_p$.

De l'équation précédente, on peut établir que l'opérateur de projection conique discret relie l'ensemble des points (i,j,k) de f_d à l'ensemble des points (p,q) de chaque projection conique $\tilde{p}_{\Theta\Phi}$. La valeur en chaque point du plan de détection (p,q) est égale à la somme des valeurs des voxels (i,j,k), projetés à une distance inférieure au pas du plan $\triangle_p$.

Par analogie, on définit l'opérateur de rétroprojection conique discret qui exprime la valeur de la fonction f_d en chaque point (i,j,k). On associe à chaque voxel $v(i,j,k)$ la somme des valeurs des points du plan de détection (p,q) sur chaque projection $\tilde{p}_{\Theta\Phi}$. Sur chaque plan de projection, on choisit les points (p,q) se trouvant à une distance inférieure au pas du plan $\triangle_p$ de la projection (u,v) du voxel.

L'implémentation des opérateurs de projection et de rétroprojection en mode "Voxel-Driven" consiste alors pour chaque voxel en trois étapes:

- Rotation du voxel $v(i,j,k)$ qui a pour nouvelles coordonnées (X,Y,Z):

$$\left\{\begin{array}{lcccc} X & = & i\triangle_v \sin\Phi & -j\triangle_v \cos\Phi & \\ Y & = & i\triangle_v \cos\Phi\cos\Theta & +j\triangle_v \sin\Phi\cos\Theta & -k\triangle_v \sin\Theta \\ Z & = & i\triangle_v \cos\Phi\sin\Theta & +j\triangle_v \sin\Phi\sin\Theta & +k\triangle_v \cos\Theta \end{array}\right. \tag{4.74}$$

- Calcul de l'adresse du point de projection de coordonnées (u,v) sur le plan de détection et détermination de la cellule p,q contenant le point de projection:

$$\left\{\begin{array}{cc} u = \frac{DX}{D_1 - Z}, & v = \frac{DY}{D_1 - Z} \\ p = \lfloor \frac{u}{\triangle_p} \rfloor, & q = \lfloor \frac{v}{\triangle_p} \rfloor \\ \lambda = p + 1 - u/\triangle_p, & \mu = q + 1 - v/\triangle_p \end{array}\right. \tag{4.75}$$

- Sommation des valeurs. Dans le cas d'une projection discrète, on répartit la valeur du voxel (i,j,k) sur les quatre points du détecteur par interpolation bilinéaire. Dans le cas d'une rétroprojection, on effectue la somme des valeurs des quatres points du plan de détection. On affecte à chaque valeur le coefficient de l'interpolation bilinéaire:

$$Projection \left\{\begin{array}{rcl} \tilde{p}_{\Theta\Phi}(p,q,) & = & (1-\lambda)(1-\mu)\, f_d(i\triangle_v, j\triangle_v, k\triangle_v) \\ \tilde{p}_{\Theta\Phi}(p+1,q) & = & \lambda(1-\mu)\, f_d(i\triangle_v, j\triangle_v, k\triangle_v) \\ \tilde{p}_{\Theta\Phi}(p,k+1) & = & \mu(1-\lambda)\, f_d(i\triangle_v, j\triangle_v, k\triangle_v) \\ \tilde{p}_{\Theta\Phi}(p+1,q+1) & = & \lambda\mu\, f_d(i\triangle_v, j\triangle_v, k\triangle_v) \end{array}\right. \tag{4.76}$$

$$Retroprojection \left\{\begin{array}{rcl} f_d(i\triangle_v, j\triangle_v, k\triangle_v) & = & (1-\lambda)(1-\mu)\, \tilde{p}_{\Theta\Phi}(p,q) \\ & & +\lambda(1-\mu)\, \tilde{p}_{\Theta\Phi}(p+1,q) \\ & & +\mu(1-\lambda)\, \tilde{p}_{\Theta\Phi}(p,q+1) \\ & & +\lambda\mu\, \tilde{p}_{\Theta\Phi}(p+1,q+1) \end{array}\right. \tag{4.77}$$

Nous formalisons les opérations élémentaires de projection d'un voxel sur un plan de détections et rétroprojection des valeurs des points de projection sur un voxel. Nous désignons par v_i un élément unitaire de la fonction à reconstruire f_d et par Im_j une image de projection composée de pixels im_{ja}. Nous définissons:

$\xrightarrow{\mathcal{P}}$ comme une opération élémentaire de projection:

$$v_i \xrightarrow{\mathcal{P}} im_{ja}, im_{jb}, im_{jc}, im_{jd} \tag{4.78}$$

où le voxel v_i contribue à la valeur des quatre pixels a, b, c et d de Im_j. Cette opération élémentaire de projection a une complexité de 15 opérations d'addition/soustraction, de 28 multiplication/division et de 4 opérations d'affectation sur l'image de projection Im_j.

$\overset{\mathcal{R}}{\leftarrow}$ comme une opération élémentaire de rétroprojection:

$$vi \overset{\mathcal{R}}{\leftarrow} im_{na}, im_{nb}, im_{nc}, im_{nd} \tag{4.79}$$

où les quatres pixels a, b, c et d de Im_j contribuent à la valeur du voxel v_i. Cette opération élémentaire de rétroprojection a une complexité de 15 opérations d'addition/soustraction, de 28 multiplication/division et d'une opération d'affectation sur le voxel v_i.

On peut remarquer que les opérations élémentaires $\overset{\mathcal{P}}{\rightarrow}$ et $\overset{\mathcal{R}}{\leftarrow}$ peuvent se calculer d'une façon similaire, seules les affectations des valeurs changent. Leurs algorithmes seront développés sur une même base d'opérations de changement d'espace, d'adressage et d'interpolation.

4.3.5 Algorithmes de projection et de rétroprojection

On présente ici les algorithmes séquentiels de la projection et de la rétroprojection. Pour la projection, on suppose que la fonction f_d est discrétisée sur un ensemble de voxels $\{v_i / 1 \leq i \leq N^3\}$ formant un volume V. Ce volume est projeté sur chaque image de projection Im_j. Pour la rétroprojection, on reconstruit l'image 3D discrétisée f_d sur le volume V composé de voxels unitaires à partir des images de projections Im_j.

Algorithme 1 *Projection*

lire(V)
Pour $j = 1$ à m
  créer(Im_j)
  Pour $i = 1$ à N^3
    $v_i \overset{\mathcal{P}}{\rightarrow} im_{ja}, im_{jb}, im_{jc}, im_{jd}$
  écrire(Im_j)

Algorithme 2 *Rétroprojection*

créer(V)
Pour $j = 1$ à m
  lire(Im_j)
  Pour $i = 1$ à N^3
    $v_i \overset{\mathcal{R}}{\leftarrow} im_{ja}, im_{jb}, im_{jc}, im_{jd}$
écrire(V)

Pour définir la complexité des algorithmes de projection et de rétroprojection, on peut utiliser deux approches, nous avons choisi de comptabiliser le nombre d'opérations élémentaires effectuées ($\overset{\mathcal{P}}{\rightarrow}$, $\overset{\mathcal{R}}{\leftarrow}$). Ce calcul de complexité permet d'exprimer le coût de ces algorithmes en fonction d'un seul type d'opération:

- Complexité de la projection: $m.N^3.T(\overset{\mathcal{P}}{\rightarrow})$
- Complexité de la rétroprojection: $m.N^3.T(\overset{\mathcal{R}}{\leftarrow})$

où $T(\overset{\mathcal{P}}{\rightarrow})$ est le coût de l'opération élémentaire $\overset{\mathcal{P}}{\rightarrow}$ et $T(\overset{\mathcal{R}}{\leftarrow})$ est le coût de l'opération élémentaire $\overset{\mathcal{R}}{\leftarrow}$L'expression de la complexité des opérateurs et leurs algorithmes montrent que leur implémentation est similaire et que leur coût en terme d'opérations élémentaires est de l'ordre de $m.N^3$ opérations.

4.4 Méthodologie pour la parallélisation

4.4.1 Machine parallèle abstraite

Pour paralléliser les méthodes de reconstruction, nous définissons une machine parallèle abstraite. Cette machine parallèle abstraite est une machine parallèle à mémoire distribuée. Son modèle d'exécution englobe les modèles d'exécution des machines parallèles de notre étude. Elle est composée de PE nœuds homogènes reliés par un réseau d'interconnexion. Chaque nœud est constitué d'un processeur élémentaire PE_i associé à sa mémoire PEM_i.
On suppose que la topologie de ce réseau permet à chaque processeur de communiquer avec un autre processeur. Cette hypothèse ne tient pas compte des techniques de routage implémentées sur chaque machine parallèle. Lors d'un échange de données, on considèrera que les données sont transférées de la mémoire du processeur émetteur à la mémoire du processeur récepteur.

Notre analyse du problème de la reconstruction nous a permis d'identifier les opérations les plus coûteuses dans les méthodes de reconstruction. Nous nous attacherons donc à paralléliser ces opérateurs de base sur cette machine parallèle abstraite. Nous décomposons cette tâche en un certain nombre d'étapes suivantes:

- la répartition des données qui distribue les données sur chaque processeur,
- le choix du schéma de communication qui génère deux approches,
  - la parallélisation par une approche locale,
  - la parallélisation par une approche globale
- l'évaluation de la complexité des algorithmes
- l'optimisation de la parallélisation.

Nous explicitons chaque étape en prenant comme exemple la parallélisation de l'opérateur de projection.

4.4.2 Répartition des données

Les données sont constituées de deux ensembles: un volume V composé de N^3 voxels et m images de projection composées elles mêmes de M^2 pixels. Les relations entre toutes ces données sont établies par les opérateurs de base. Pour une projection, la valeur d'un voxel contribue aux valeurs de 4 pixels sur chaque image de projection. Pour une rétroprojection, la valeur de chaque pixel contribue aux valeurs des voxels se projetant à une distance inférieure au pas d'échantillonnage $\triangle_p$.

Le calcul de chaque pixel (voxel) est indépendant des autres pixels (voxels). Cependant un pixel est rétroprojeté sur un ensemble de voxels déterminé par la géométrie d'acquisition et, de la même façon un voxel est projeté sur un ensemble de pixels. Ces ensembles sont définis lors des phases de calculs. Cela signifie que chaque pixel peut être en relation avec quasiment n'importe quel voxel: il doit donc avoir la possibilité de "voir" l'ensemble du volume V. De même un voxel doit avoir la possibilité de "voir" l'ensemble des images de projection Im_j.

Ces relations indiquent qu'un découpage homogène des données sur l'ensemble des mémoires des processeurs est le mieux adapté. L'image 3D (le volume V) et les *images de projection* Im_j sont donc réparties sur les différents processeurs. Nous proposons deux répartitions en fonction de la taille de la mémoire des processeurs.

- Quand la taille des données est supérieure à la taille des mémoires de tous les processeurs, l'image 3D est décomposée en T tranches de taille respective $N_T.N^2$ avec $T.N_T = N$. Dans ce cas de figure, nous travaillons avec une seule image de projection à la fois. Nous considérons que nous avons, à chaque instant, une sous-image 3D de taille $N_T.N^2$ et une image de projection 2D de taille M^2 répartie sur les PE mémoires.

- Quand la taille des données est inférieure à la taille des mémoires de tous les processeurs, nous supposons que l'image 3D (V) est décomposée en PE sous-images et que les m images de projection sont distribuées sur les PE processeurs (figure 28). Chaque sous-image de V, notée V_i, a une taille égale à $N_{PE}.N^2$ avec $PE.N_{PE} = N$. De plus chaque processeur contient m_{PE} images de projection ($m = m_{PE}.PE$).

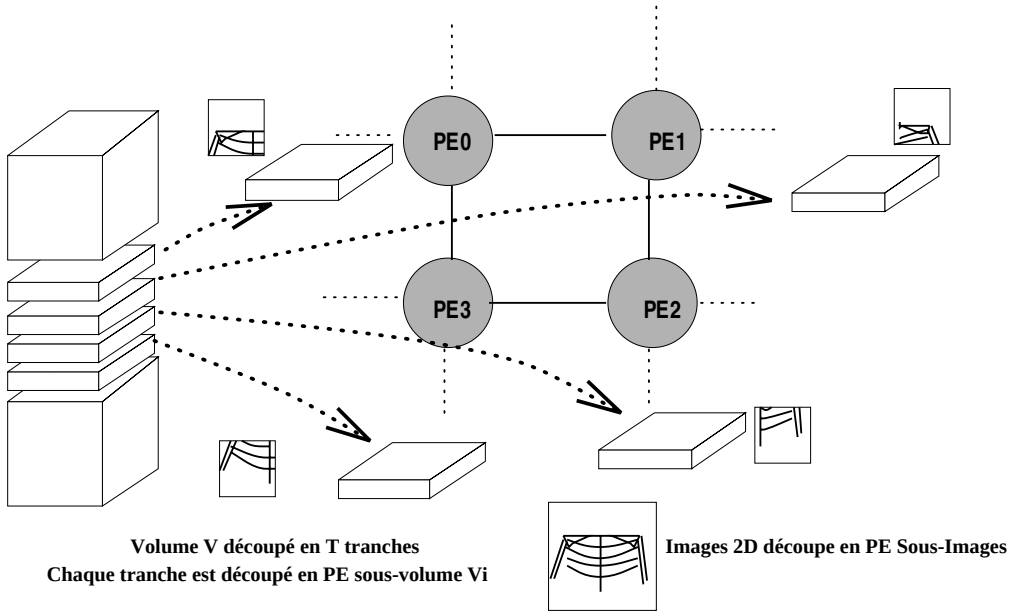


FIG. 27 - *Répartition d'une tranche et d'une image*

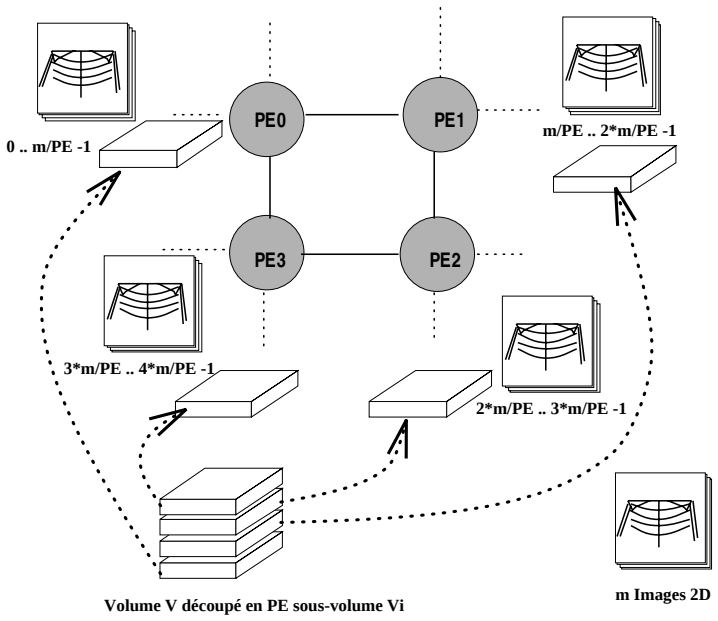


FIG. 28 - *Répartition de l'ensemble des données*

Les données étant réparties sur les processeurs, le problème est de choisir un schéma de communication pour le transfert des données. Pour la plupart des algorithmes que nous présenterons, nous supposerons que l'ensemble des données est réparti sur les processeurs.

4.4.3 Schéma de communications

Avant de définir le schéma de communication qui détermine le transfert des données, il faut connaitre la nature des données à transférer. Pour savoir quelles données transférer, on étudie l'opérateur de projection utilisé par deux méthodes: ART par blocs et SIRT. On calcule le coût de communication dans le cas où le volume et les images de projection sont transférés.
Avec la méthode ART par blocs, que l'on étudiera dans le chapitre 6, la projection s'effectue sur une seule image à un instant donné.
Si le processeur, possédant l'image de projection, la communique aux autres processeurs, on effectue une opération de diffusion-réduction. Le coût des communications, T_{com}, s'exprime par:

$$T_{com}(Im_j) = 2 \times (PE - 1)(\beta + M^2\tau) \tag{4.80}$$

où β et τ sont les constantes de notre modèle de communication. Dans le cas où les sous-images V_i sont communiquées vers le processeur possédant l'image de projection, le coût de communication est égal à:

$$T_{com}(V_i) = (PE - 1)(\beta + \frac{N^3}{PE}\tau) \tag{4.81}$$

Avec la méthode SIRT, que l'on étudiera dans le chapitre 6, la projection peut s'effectuer sur plusieurs images en même temps. Ainsi chaque image Im_j ou chaque sous-image V_i doit parcourir le réseau d'interconnexion pour effectuer la projection de m images en parallèle. Lorsque une image Im_j, ou une sous-image V_i, est transférée entre deux processeurs, $(PE - 1)$ images Im_j, ou sous-images V_i, sont transférées en parallèle. Ainsi le calcul des communications revient à calculer soit le transfert de V_i (noté $T_{com}(V_i)$), soit le transfert des m_{PE} images Im_j (noté $T_{com}(Im_j)$), entre tous les processeurs:

$$T_{com}(Im_j) = (PE - 1)(\beta + \frac{m.M^2}{PE}\tau) \tag{4.82}$$

$$T_{com}(V_i) = (PE - 1)(\beta + \frac{N^3}{PE}\tau) \tag{4.83}$$

Généralement on peut admettre que $N^3 \approx m.M^2$. Cela signifie que le coût de communication des sous-images V_i est presque toujours supérieur ou égal au coût de communication des images Im_j, comme nous le prouvent les deux exemples précédents. Les phases de communication transfèrent donc les images Im_j pour réaliser les opérations de projection et de rétroprojection et les V_i restent dans la

mémoire de chaque processeur PE_i.

En examinant les coûts de communication des deux exemples précédents, nous avons choisi de définir deux schémas de communications différents correspondant respectivement à une approche locale et à une approche globale [LPC95d]. Le premier schéma de communication utilisera une topologie en anneau, le deuxième schéma se servira d'arbres binaires pour effectuer les opérations de réductions et de diffusions. Ces deux schémas de communication ont l'avantage de pouvoir se plonger sur la plupart des topologies des machines SIMD et MIMD.

4.4.4 Approche locale

L'approche locale consiste à effectuer les opérateurs de base $\xrightarrow{\mathcal{P}}$ et $\xleftarrow{\mathcal{R}}$ sur des données se trouvant sur un même processeur. Les communications vont, dans ce cas, servir à mettre en relation chaque image de projection Im_j avec l'ensemble des sous-images V_i. Soit l'opération $V_i \xrightarrow{\mathcal{P}} Im_j$ qui effectue la projection de tous les voxels v_i sur l'image de projection Im_j, la formule de la projection discrète de V sur l'ensemble des projections $Im_m = \{Im_j / 1 \leq j \leq m\}$, notée $P(V, Im_j)$ peut s'écrire:

$$P(V, Im_j) = \sum_{j=1}^{m} \sum_{i=1}^{PE} V_i \xrightarrow{\mathcal{P}} Im_j \tag{4.84}$$

$$\begin{aligned} - = {} & V_1 \xrightarrow{\mathcal{P}} Im_1 + V_2 \xrightarrow{\mathcal{P}} Im_2 + \cdots + V_P E \xrightarrow{\mathcal{P}} Im_{PE} \\ & + V_1 \xrightarrow{\mathcal{P}} Im_{PE} + V_2 \xrightarrow{\mathcal{P}} Im_1 + \cdots + V_P E \xrightarrow{\mathcal{P}} Im_{PE-1} \\ & + V_1 \xrightarrow{\mathcal{P}} Im_2 + V_2 \xrightarrow{\mathcal{P}} Im_3 + \cdots + V_P E \xrightarrow{\mathcal{P}} Im_1 \\ & + V_1 \xrightarrow{\mathcal{P}} Im_{PE+1} + V_2 \xrightarrow{\mathcal{P}} Im_{PE+2} + \cdots + V_P E \xrightarrow{\mathcal{P}} Im_{2PE} \\ & + V_1 \xrightarrow{\mathcal{P}} Im_{2PE} + V_2 \xrightarrow{\mathcal{P}} Im_{PE+1} + \cdots + V_P E_P P Im_{2PE-1} \\ & + V_1 \xrightarrow{\mathcal{P}} Im_{PE+2} + V_2 \xrightarrow{\mathcal{P}} Im_{PE+3} + \cdots + V_m \xrightarrow{\mathcal{P}} Im_{PE+1} \\ & + \cdots \\ & + V_1 \xrightarrow{\mathcal{P}} Im_{m-PE} + V_2 \xrightarrow{\mathcal{P}} Im_{m-PE+1} + \cdots + V_P E \xrightarrow{\mathcal{P}} Im_m \\ & + V_1 \xrightarrow{\mathcal{P}} Im_m + V_2 \xrightarrow{\mathcal{P}} Im_{m-PE+1} + \cdots + V_P E \xrightarrow{\mathcal{P}} Im_{m-1} \\ & + V_1 \xrightarrow{\mathcal{P}} Im_{m-PE+1} + V_2 \xrightarrow{\mathcal{P}} Im_{m-PE+2} + \cdots + V_m \xrightarrow{\mathcal{P}} Im_{m-PE} \end{aligned} \tag{4.85}$$

On reformule cette équation comme une suite de projections parallèles des V_i sur PE images Im_j en notant $P^l_{//}(Im_k, Im_{k+PE})$ la projection parallèle des V_i sur les

$Im_k, \ldots, Im_{k+PE}$, et en supposant que chaque processeur PE_i effectue la projection $V_i \xrightarrow{\mathcal{P}} Im_{(k+l) \bmod (k+PE)}$:

$$P(V, Im_j) = \sum_{k=1}^{\frac{m}{PE}} \sum_{l=1}^{PE-1} P^l_{//}(Im_k, Im_{k+PE}) \tag{4.86}$$

Les projections parallèles $P^l_{//}(Im_k, Im_{k+PE})$ pour $0 \leq l \leq PE-1$ peuvent s'interpréter comme étant la projection cyclique des images $Im_k, \ldots, Im_{k+PE}$. Pour chaque cycle, le processeur PE_i effectue les projections $V_i \xrightarrow{\mathcal{P}} Im_k$, $V_i \xrightarrow{\mathcal{P}} Im_{k+PE}$ jusqu'à $V_i \xrightarrow{\mathcal{P}} Im_{k+1}$, cela pour chaque $k = \frac{m}{PE}$. Après chaque projection, le processeur PE_i envoie au processeur PE_{i+1}, l'image Im_k pour qu'il puisse effectuer la projection $V_{i+1} \xrightarrow{\mathcal{P}} Im_k$ et attend, du processeur PE_{i-1}, l'image Im_{k-1} pour effectuer la projection $V_i \xrightarrow{\mathcal{P}} Im_{k-1}$. Le schéma de communication issu de ce cycle est donc l'anneau où chaque processeur communique avec son prédécesseur et son suivant sur l'anneau.

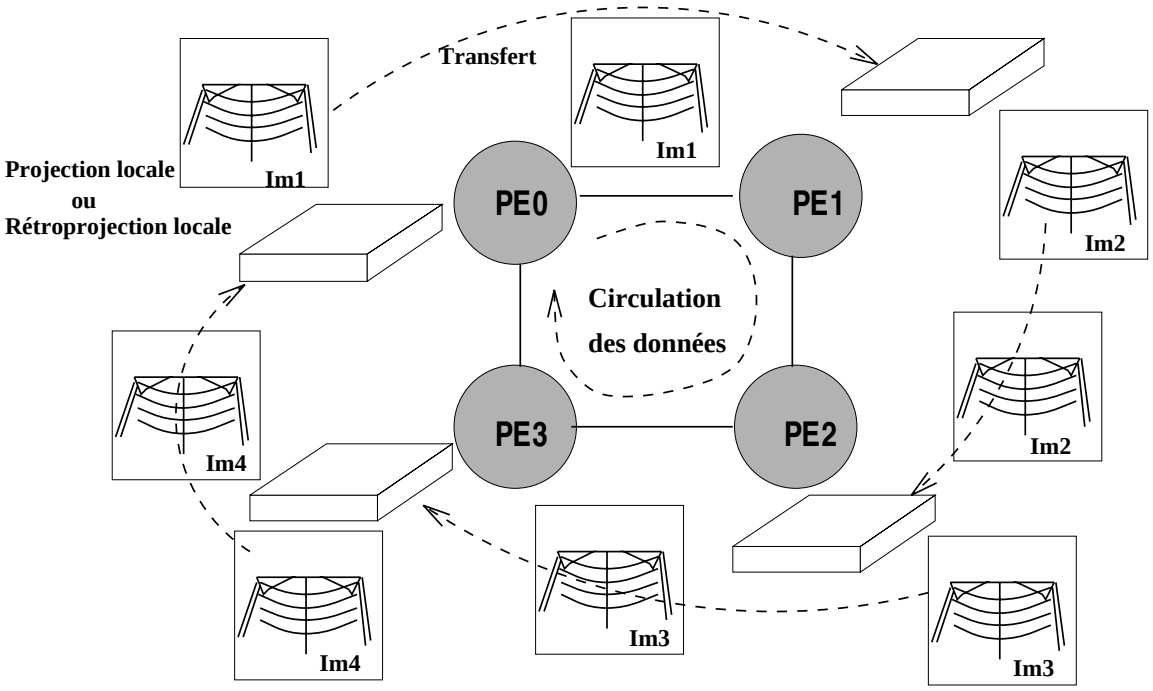


FIG. 29 - *Approche locale*

L'algorithme parallèle qui découle de cette formule est découpé en phases de calcul, où les opérations $V_{i+1} \xrightarrow{\mathcal{P}} Im_k$ sont effectuées et en phases de communications, où les images Im_j sont transférées (figure 29). On utilise les primitives de communication: *send*(PE_{i+1}, Im_j) qui envoie l'image Im_j au processeur PE_{i+1} et *recv*(PE_{i-1}, Im_j) qui reçoit l'image Im_j du processeur PE_{i-1}.

Algorithme 3 *Projection Parallèle Locale*

lire(V_i)
Pour $j = 1$ à $\frac{m}{PE}$
  créer(Im_j) avec $Im_j \in \{Im_{j*\frac{m}{PE}} \dots Im_{(j+1)*\frac{m}{PE}-1}\}$
  Pour $n = 1$ à PE
    Pour $v_i \in V_i$
      $v_i \xrightarrow{\mathcal{P}} im_{ja}, im_{jb}, im_{jc}, im_{jd}$
    send(PE_{i+1}, Im_j)
    recv(PE_{i-1}, Im_j)
  écrire(Im_j)

Cet algorithme est implémenté sur les machines MIMD. Nous présentons une version adaptée à une architecture SIMD dans le chapitre 5.

4.4.5 Approche globale

Au contraire de l'approche locale, les opérateurs de base implémentés par une approche globale utilisent des données se trouvant sur plus d'un processeur pour effectuer les opérations élémentaires $\xrightarrow{\mathcal{P}}$ et $\xleftarrow{\mathcal{R}}$. En prenant comme exemple l'opérateur $\xrightarrow{\mathcal{P}}$, nous exposons les différentes possibilités pour l'implémenter dans le cadre d'une approche globale. Dans ce type d'approche, l'implémentation utilise des phases de communications pour réaliser les opérateurs élémentaires. Les implémentations diffèrent suivant la granularité des communications.

Communication pixel par pixel

Le cas le plus simple consiste à calculer pour chaque voxel v_i son adresse sur l'image Im_j et à envoyer au processeur, contenant l'image Im_j, un message contenant la valeur du voxel et son adresse de projection. Ce message peut être considéré comme le pixel de projection du voxel considéré. Le processeur récepteur se charge alors de recevoir les messages et d'accumuler les valeurs en fonction des adresses de projection. Si l'on suppose que l'adresse est codée sur deux entiers et la valeur sur un réel, la taille du message est égale à deux réels, et le coût des communications d'une projection est égal à:

$$T_{com} = N^3 \frac{PE - 1}{PE}(\beta + 2\tau) \tag{4.87}$$

Dans ce cas, l'initialisation de la communication est effectuée à chaque transfert de voxel, ce qui pénalise ce type d'approche. De plus, on ne prend pas en compte, dans ce calcul, les temps d'attente dus au processeur récepteur.

Communication par groupe de pixels

Pour minimiser les communications et pour optimiser les opérations de sommation sur le processeur récepteur, on utilise des opérations de réduction-sommation sur chaque pixel de l'image Im_j. Une opération de réduction-sommation permet d'envoyer des données à un même processeur et de les sommer. Dans ce but, chaque processeur précalcule les adresses de projection des voxels et les trie par adresse croissante en sommant sur un même pixel de projection la valeur des voxels ayant une même adresse de projection. On effectue ensuite séquentiellement une opération de réduction-sommation sur chaque pixel de l'image Im_j. Le coût de la communication est égal à: $T_{com} = M^2 T_{reduc-som}(1)$ et dépend du coût d'une réduction-sommation $T_{reduc-som}(1)$ pour un pixel. Ce mode de communication est adapté aux architectures SIMD. Nous détaillons l'algorithme implémenté sur la Maspar dans le chapitre 5.

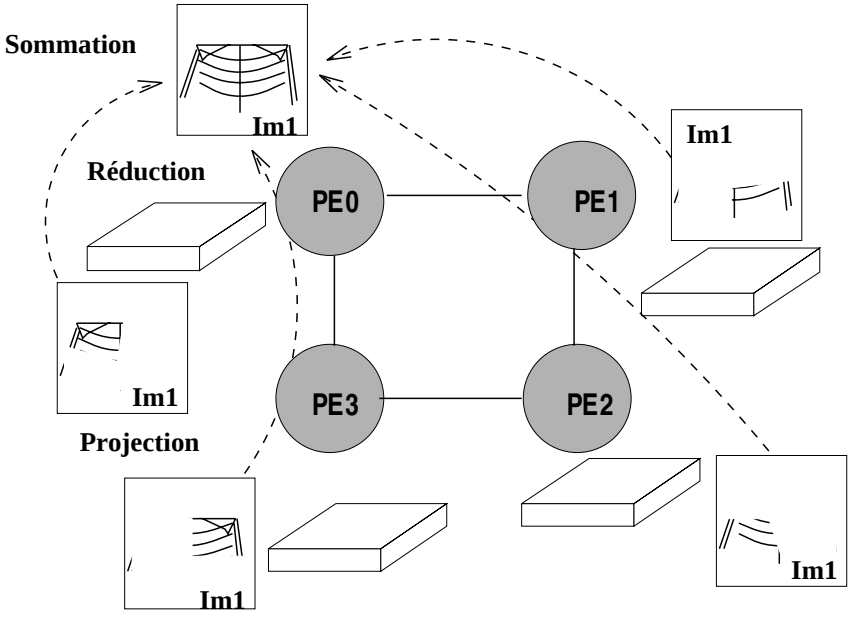


FIG. 30 - *Approche globale*

Communication par image partielle

Nous reprenons l'approche définie précédemment en communiquant une image au lieu de communiquer des valeurs de voxels (figure 30). Cette approche consiste à calculer localement les projections élémentaires $\xrightarrow{\mathcal{P}}$ pour obtenir sur chaque processeur l'image de la projection du volume V_i notée Im'_j. Puis on effectue une réduction-sommation des images Im'_j sur l'image Im_j. Le coût des communications est alors égal à: $T_{com} = T_{reduc-som}(M^2)$. Cette approche permet de minimiser le nombre de communications. Cependant, le coût de communication dépend du schéma de communications utilisé pour implémenter l'opération de réduction-sommation. Nous utilisons cette approche sur les machines MIMD. L'algorithme de projection du volume V sur m images Im_j, utilise l'opérateur de réduction-sommation *reduc-som*(PE_i, Im_j) des images Im_j sur le processeur PE_i. On présente l'algorithme exécuté sur chaque PE_i.

Algorithme 4 *Projection Parallèle Globale*

lire(V_i)
Pour $j = 1$ à m
  si $(Im_j \in PE_i)$
    créer(Im_j)
    Pour $v_i \in V_i$
      $v_i \xrightarrow{\mathcal{P}} im_{ja}, im_{jb}, im_{jc}, im_{jd}$
    reduc-som(PE_i, Im_j)
    écrire(Im_j)
  sinon $\{Im_j \in PE_k\}$
    créer(Im'_j)
    Pour $v_i \in V_i$
      $v_i \xrightarrow{\mathcal{P}} im'_{ja}, im'_{jb}, im'_{jc}, im'_{jd}$
    reduc-som(PE_k, Im'_j)
  finsi

4.4.6 Évaluation des coûts

Le coût d'un algorithme parallèle est fonction du temps de calcul sur chaque processeur et du temps de communication entre les processeurs. Comme pour les algorithmes séquentiels, les temps de calcul T_{cal} sont fonction du nombre d'opérations élémentaires effectuées ($\xrightarrow{\mathcal{P}}, \xleftarrow{\mathcal{R}}$). Les temps de communication T_{com} sont fonction de la taille des images transférées et de l'algorithme de communication utilisé. On évalue

les coûts de la projection parallèle des V_i sur m images Im_j en comparant l'approche locale et l'approche globale.

Approche locale Chaque processeur effectue séquentiellement une phase de calcul et une phase de communication. Le temps de calcul sur chaque processeur est égal à la somme des temps de calcul des projections $V_i \xrightarrow{\mathcal{P}} Im_j$ pour $1 \leq j \leq m$. De même le temps de communication est égal à la somme des temps de communication de chaque image Im_j. Le temps de communication se compose d'un temps d'émission et d'un temps de réception. Dans la plupart des librairies de communication comme PVM, l'émission est non-bloquante. On suppose que le temps de l'émission d'une image est négligeable devant le temps de réception d'une autre image, chaque processeur recevant $m - \frac{m}{PE}$ images Im_j. Le temps de réception est égal au temps de transfert de l'image entre deux processeurs.

$$T_{cal} = \sum_{j=1}^{m} N_{PE}.N^2.T(\xrightarrow{\mathcal{P}}) = \frac{m.N^3}{PE}.T(\xrightarrow{\mathcal{P}}) \tag{4.88}$$

$$T_{com} = \sum_{j=1}^{m-\frac{m}{PE}} (\beta + M^2.\tau) = m.\frac{PE-1}{PE}(\beta + M^2.\tau) \tag{4.89}$$

Approche globale le coût de cette approche est fonction des temps de calcul et des temps de communication. Le temps de communication d'une opération de réduction-sommation dépend de son algorithme de communication. Dans la plupart des bibliothèques de communication, l'algorithme consiste à recevoir séquentiellement les données et à les sommer. On peut considérer que les m projections sont effectuées sur le même processeur.

Le temps de calcul des processeurs émetteurs est égal à la somme des temps de calcul des projections $V_i \xrightarrow{\mathcal{P}} Im_j$. Le temps de calcul du processeur récepteur est augmenté du temps des opérations de sommation des images Im'_j. Donc le temps de calcul total, qui est pris en considération, est le temps de calcul du processeur récepteur. On définit par $Som(im_{ja}, im'_{ja})$ l'opération qui somme la valeur d'un pixel de Im'_j sur un pixel de Im_j. Pour chaque projection, le processeur récepteur effectue $(PE-1)$ sommations d'images de taille M^2. Le temps de calcul total est pour chaque projection égal à :

$$T_{cal(1)} = N_{PE}.N^2.T(\xrightarrow{\mathcal{P}}) + \sum_{i=1}^{PE-1} M^2.Som(im_{ja}, im'_{ja}) \tag{4.90}$$

On en déduit le temps de calcul total des m projections:

$$T_{cal} = \sum_{j=1}^{m} N_{PE}.N^2.T(\overset{\mathcal{P}}{\rightarrow}) + \sum_{j=1}^{m}\sum_{i=1}^{PE-1} M^2.Som(im_{ja}, im'_{ja}) \quad (4.91)$$

$$- = \frac{m.N^3}{PE}.T(\overset{\mathcal{P}}{\rightarrow}) + m.(PE-1).M^2.Som(im_{ja}, im'_{ja}) \quad (4.92)$$

Pour les communications, les processeurs émetteurs envoient une image Im'_j pour chaque projection. On peut négliger le coût de ces envois des images Im'_j par rapport à leur temps de transfert sur le processeur récepteur. Le temps de communication d'une réduction est donc égal à:

$$T_{com(1)} = \sum_{i=1}^{PE-1} (\beta + M^2.\tau) \quad (4.93)$$

On en déduit le temps de communication total des m projections:

$$T_{com} = \sum_{j=1}^{m}\sum_{i=1}^{PE-1} (\beta + M^2.\tau) = m.(PE-1).(\beta + M^2.\tau) \quad (4.94)$$

Cette analyse montre que l'approche locale est plus performante au niveau des temps de calcul tout comme au niveau des temps de communication. Cependant on utilise ces deux approches car elle servent, chacune, à paralléliser les différentes méthodes de reconstruction.

Ce calcul théorique permet de donner une estimation du temps total de calcul. Pour mesurer les performances réelles des algorithmes, nous avons implémenté des routines qui déterminent le temps de chaque calcul et le temps de communication de chaque processeur. Pour faciliter les mesures on considère l'activité du processeur en deux phases: l'une de calcul et l'autre de communication. La phase de communication regroupe le temps de toutes les communications du processeur et les temps d'attente dus aux autres processeurs. Chaque algorithme est donc découpé en deux parties rassemblant, d'une part toutes les phases de calculs et d'autre part, les phases de communication. On peut ainsi comparer les temps de communication et les temps de calcul de tous les processeurs. Soit $T_{cal(i)}$ le temps de calcul du processeur PE_i et $T_{com(i)}$ son temps de communication, le temps d'exécution parallèle $T_{//}$ que l'on mesure est défini par:

$$T_{//} = \max_{1 \leq i \leq PE} (T_{cal(i)} + T_{com(i)}) \quad (4.95)$$

4.4.7 Optimisation du parallélisme

Pour améliorer l'efficacité des implémentations, nous utilisons deux méthodes complémentaires [LCPC95]. La première permet de minimiser les coûts de communication, la seconde permet de réguler au mieux la charge de chaque processeur.

Minimisation des temps de communication Pour chaque approche que nous avons explicitée, il existe des techniques permettant de minimiser les communications. Pour l'approche locale, nous utilisons une méthode de recouvrement des communications par les calculs. Pour l'approche globale, nous utilisons un algorithme "Global-Combine" pour implémenter l'opération de réduction.

Régulation de la charge Il existe plusieurs méthodes pour réguler la charge de travail de chaque processeur. Le modèle "Master-Worker" présenté par Atkins [AMH93] est utilisé par un grand nombre d'applications. Ce modèle nécessite de bloquer un processeur le "Master" qui sert à réguler la charge. Ce processeur n'effectue pas de calcul. Le nombre de processeurs alors utilisés pour la reconstruction est $PE - 1$ ce qui n'est pas très efficace. Une autre technique consiste à estimer avant chaque phase de calcul sa charge de calcul. Les informations sont alors communiquées aux autres processeurs. En fonction de ces informations, on effectue une nouvelle répartition des données. Cette méthode est en fait une redistribution [MR91] des données.

Notre méthode consiste à réguler la charge après exécution d'une phase de calcul. Elle permet de prendre en compte la puissance de chaque processeur.

Recouvrement des communications par les calculs

De nombreux travaux se sont intéressés aux recouvrements des communications par les calculs pour minimiser le coût des communications. La bibliothèque LOCCS [DT93] est une bibliothèque de routines d'algèbre linéaire parallèle utilisant les recouvrement calculs/communications. Une technique classique de recouvrement consiste à calculer la taille optimale des données à transférer pendant laquelle des calculs sont effectués. Pour mettre en œuvre ces techniques, les routines de communication utilisées sont des routines non-bloquantes.

Notre approche consiste à calculer localement les données en attendant que de nouvelles données arrivent. Si à la fin du traitement des données Im_j et de leur envoi, le processeur PE_i n'a toujours pas reçu de nouvelles données du processeur PE_{i-1}, il ira en chercher d'autres sur sa mémoire (Im_{j+1}).

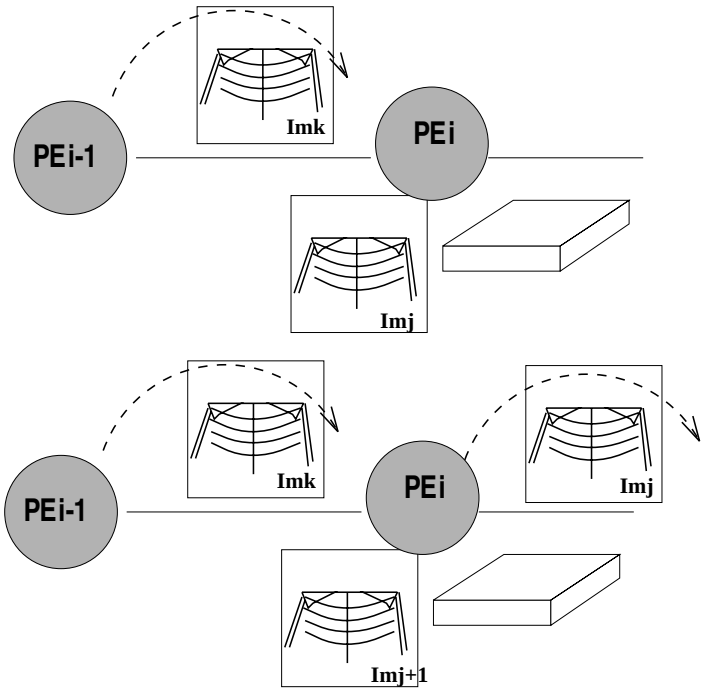


FIG. 31 - *Recouvrement des communications par des calculs*

Dans l'approche locale les calculs et les communications étaient synchronisés; ici chaque processeur effectue les calculs tout en recevant de nouvelles données d'une façon totalement asynchrone par rapport aux autres processeurs. L'algorithme de projection utilise donc une routine de réception non-bloquante *n-recv*(PE_i, Im_j) qui indique si l'image Im_j reçue du processeur PE_i est entièrement arrivée. Le traitement de la projection s'arrête lorsque tous les V_i ont été projetés sur tous les Im_j ($NIma_i = m$). Chaque image est stockée lorsqu'elle a reçu la projection de tous les V_i ($NVol_j = PE$).

Algorithme 5 *Projection Parallèle Locale en mode asynchrone*

lire(V_i)
$NIma_i \leftarrow 0$
Tant que $NIma_i \leq m$
  si (n-recv(PE_i, Im_j)=faux)
    créer(Im_j)
    $NVol_j \leftarrow 0$
  Pour $v_i \in V_i$
    $v_i \xrightarrow{\mathcal{P}} im_{ja}, im_{jb}, im_{jc}, im_{jd}$
  $NVol_j \leftarrow NVol_j + 1$
  $NIma_i \leftarrow NIma_i + 1$
  si($NVol_j = PE$)
    écrire(Im_j)
  sinon
    send(PE_{i+1}, Im_j)
  finsi

Cette approche permet de recouvrir au mieux les communications par les calculs. Tant qu'il y a suffisamment d'images Im_j à calculer, le recouvrement est optimal. Puis il dépend du rapport entre le calcul d'une projection et la communication d'une image de projection pour recouvrir les communications des images restantes. L'évaluation du nombre d'images dont le transfert n'est pas recouvert est fonction de l'architecture de la machine. On note ϵ le pourcentage de communications non recouvertes. Le coût de l'algorithme est égal à:

$$T_{//} = \frac{m.N^3}{PE}.T(\xrightarrow{\mathcal{P}}) + \epsilon(m - \frac{m}{PE}).(\beta + M^2.\tau) \tag{4.96}$$

Algorithme Global Combine

Pour améliorer l'opération de réduction-sommation, notre algorithme utilise un schéma de communication en arbre binaire. De nombreuses études ont montré que ce schéma de communication était efficace sur différentes topologies [BLPvdG93]. Son principe est de découper l'opération à effectuer en arbre binaire suivant la topologie de la machine.
La réduction est effectuée en n étapes avec $PE = 2^n$. A l'étape k, les processeurs, présents, à l'étape $k-1$ et vérifiant $PE_i/2^k = 1$, envoient leur image Im'_j aux processeurs qui vérifient $PE_i/2^k = 0$. Ces derniers calculent la somme de l'image reçue et de l'image présente dans sa mémoire.

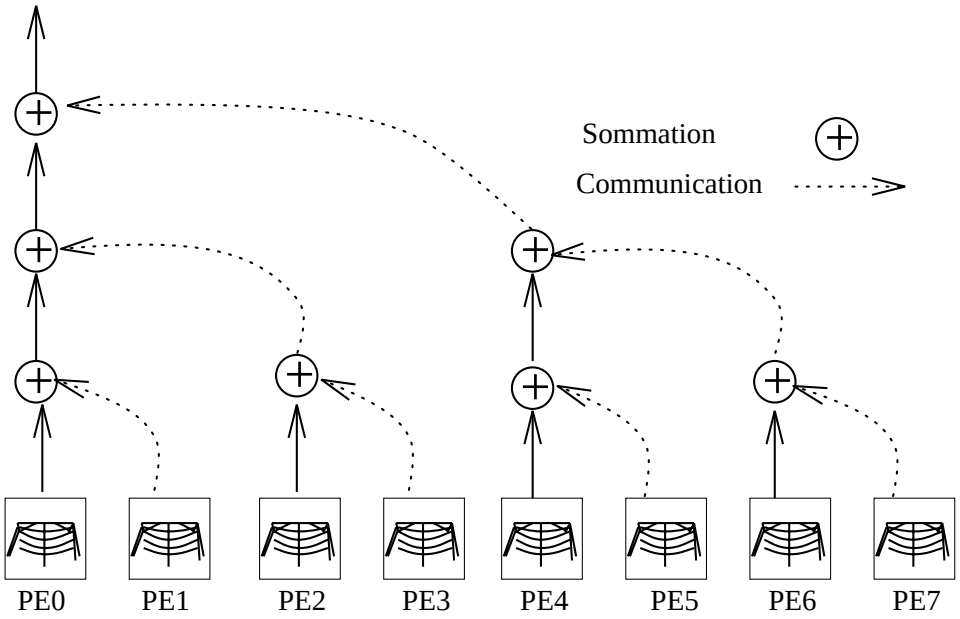


FIG. 32 - *Réduction-Sommation sur un arbre binaire*

L'algorithme se formule ainsi:

Algorithme 6 *Réduction-Sommation des images de projection* Im'_j

$k \leftarrow 0$
Tant que $2^k < PE + 1$
  $reste \leftarrow PE_i / 2^k$
  si $(reste = 0)$
    recv$(PE_{i+2^{k-1}}, Im''_j)$
    Pour $l = 1$ à M^2
      $Im'_{jl} \leftarrow Im'_{jl} + Im''_{jl}$
    $k \leftarrow k + 1$
  sinon
    send$(PE_{i-reste}, Im_j)$
    $k \leftarrow PE + 1$
  finsi

Le coût d'une réduction-sommation pour une projection avec $PE = 2^n$ est de:

$$T_{reduc-som} = n((\beta + M^2.\tau) + M^2.Som(im_{ja}, im'_{ja})) \tag{4.97}$$

Donc le coût total de l'algorithme de projection par une approche globale devient:

$$T_{cal} = \frac{m.N^3}{PE}.T(\xrightarrow{\mathcal{P}}) + n.m.M^2.Som(im_{ja}, im'_{ja}) \tag{4.98}$$

$$T_{com} = m.n.(\beta + M^2.\tau) \tag{4.99}$$

Partitionnement adaptatif

Nous développons ici une méthode simple pour équilibrer la charge des processeurs. Cette méthode est applicable dans deux cas de figure, au niveau de la machine ou au niveau de l'image 3D à reconstruire:

- L'architecture de la machine cible peut avoir des processeurs hétérogènes, comme le réseau de stations, ce qui implique des puissances de calcul propres à chaque processeur. La machine parallèle peut être une machine ouverte. Cela signifie que plusieurs applications peuvent être exécutées en même temps sur un même nœud. Dans ce cas les puissances des processeurs ne sont plus homogènes. Pour certaines topologies, les temps de communication entre deux processeurs quelconques ne sont pas toujours identiques.

- L'image 3D à reconstruire est de dimension N^3. Une réduction de la taille de l'image est possible soit par un seuillage sur l'image soit par un à priori sur la zone à reconstruire. La répartition de l'image 3D n'est donc plus homogène.

L'algorithme d'équilibrage de charge, que nous décrivons, est basé sur les temps d'exécution après une itération pour les algorithmes itératifs ou après une exécution pour les algorithmes des méthodes directes. Nous l'utilisons de façon complémentaire à la méthode de recouvrement des communications par les calculs. Comme les communications sont asynchrones, les processeurs les plus rapides terminent leur travail en premier. On définit V_i^0 la vitesse initiale du processeur PE_i et D_i^0 la taille de ses données initiales. Les données D_i^0 représentent le nombre de plans du volume V_i sur chaque processeur: $\sum_{i=1}^{PE} D_i^0 = N$. Après chaque itération k ou exécution, le processeur PE_i a un temps d'exécution $T_{//(i)}$ égal à la somme du temps de communication $T_{com(i)}^k$ et du temps de calcul $T_{cal(i)}^k$. La nouvelle vitesse du processeur peut s'exprimer par:

$$V_i^k = \frac{D_i^k}{T_{com(i)}^k + T_{cal(i)}^k} \tag{4.100}$$

On normalise la vitesse de chaque processeur en fonction du processeur le plus rapide: $V_{max}^k = \max_{1 \leq i \leq PE} V_i^k$. On obtient une vitesse relative Vr_i^k:

$$Vr_i^k = \frac{V_i^k}{V_{max}^k} \tag{4.101}$$

Grâce à cette vitesse relative, la nouvelle partition des données, pour l'itération ou l'exécution suivante, est égale à:

$$D_i^{k+1} = \frac{N}{\sum_{i=1}^{PE} Vr_i^k}.Vr_i^k \tag{4.102}$$

4.4.8 Méthodes implémentées

Les méthodes sont implémentées à partir des opérateurs parallélisés. Nous utilisons pour chacune d'elles, soit une approche locale, soit une approche globale.

la méthode de Feldkamp Cette méthode analytique permet une reconstruction à partir des Im_j sans tenir compte de la chronologie d'acquisition. Chaque V_i peut être reconstruit simultanément. Cette méthode est donc parallélisée par une approche locale. Elle utilise un opérateur parallèle de rétroprojection auquel se rajoute une opération de pondération et de filtrage.

la méthode ART par blocs Cette méthode nécessite de reconstruire l'image 3D à partir de l'image de différence entre chaque projection mesurée et chaque projection calculée. Ce traitement ne peut être fait que de manière séquentielle. Nous utilisons un opérateur de projection globale pour calculer la projection de l'image 3D, et après avoir diffusé l'image de différence, on effectuera en local une rétroprojection de celle-ci.

la méthode SIRT Cette méthode a l'avantage, comme la méthode de Feldkamp, de pouvoir calculer simultanément les projections de l'image 3D. Les images de différence sont calculées en parallèle et sont ensuite rétroprojetées. La méthode SIRT est programmée avec des opérateurs parallèles de projection et de rétroprojection implémentés par une approche locale.

4.4.9 Machines cibles

Nous avons utilisé de nombreuses machines parallèles pour implémenter les méthodes de reconstruction à partir des opérateurs parallèles de projection et de rétroprojection. Nous présentons pour chacune d'elles ses processeurs de base, son réseau d'interconnexion, ses performances crêtes, et le langage de programmation utilisé .

La machine de référence: un SUN 4

Notre machine séquentielle de référence est un SUN 4. Elle est constituée d'un processeur RISC Sparc 2 et d'une mémoire de 16 Mbytes. Sa puissance crête est de 5 Mflops (Million d'opérations flottantes par seconde). Nous avons implémenté nos algorithmes séquentiels sur cette machine.

la Maspar de DEC

Il nous a semblé intéressant d'utiliser une machine SIMD pour implémenter nos algorithmes parallèles. La Maspar [Bla90] est un exemple représentatif des machines SIMD actuelles. Nous utilisons la Maspar du laboratoire Lip de l'ENS-Lyon. Son architecture est un tore 2D composé de 32x32 processeurs élémentaires. La mémoire de chaque processeur est de 16 Kbytes, ce qui donne 16Mbytes pour la mémoire de la Maspar. Cette machine a deux réseaux d'interconnexion:

- Le réseau "XNet" qui permet à chaque processeur de communiquer avec ses huit voisins.
- Le réseau "Global Router" qui permet de communiquer avec les autres processeurs à l'aide d'un réseau cross-bar hiérarchique.

La puissance crête de cette machine est de 88 Mflops. Nous utilisons le langage parallèle MPML développé pour cette machine. Il permet de déclarer des variables locales et globales à tous les processeurs. Ce langage est fourni avec un débogueur qui permet de visualiser l'état de chaque variable sur tous les processeurs.

Le réseau de stations

Nous avons choisi de nous intéresser aux réseaux de stations car ce sont des machines parallèles ayant un ratio coût/performances très intéressant. Notre réseau est constitué de deux SUN à processeur Sparc 2 et de trois SUN à processeur Sparc 1. Ces processeurs ont respectivement une puissance crête de 5 Mflops et de 2,5 Mflops. La puissance crête théorique de cette machine est de 17,5 Mflops. Chaque station constituant cette machine possède une mémoire de 16 Mbytes. Ces machines sont reliées par un réseau Ethernet classique. Nous utilisons la librairie PVM qui permet de considérer ce réseau de stations de travail comme une machine MIMD.

Comme les processeurs de cette machine ne sont pas homogènes, nous utilisons les techniques de partitionnement adaptatif pour implémenter nos algorithmes.

La Ferme de processeurs Alpha

Cette machine s'apparente à un réseau de stations de travail dédié exclusivement au parallélisme, contrairement à la machine précédente qui peut être utilisée de façon optimale lorsque personne ne travaille sur l'une des machines du réseau. Cette machine parallèle fait partie du pool de machines du Laboratoire Informatique Fondamental de Lille. Elle est constituée de 16 nœuds reliés par deux réseaux d'interconnexion:

- Un réseau Ethernet en fibre optique basé sur un Giga Switch de bande passante 200Mbytes/sec. Ce réseau sert aux transferts de messages entre les processeurs.
- Un Ethernet classique qui permet de transférer des fichiers et d'effectuer des opérations de gestion du système sur chacun des nœuds.

Chaque nœud de cette machine est composé d'un processeur RISC Alpha de DEC, de puissance crête de 150 Mflops, et d'une mémoire de 64 Mbytes. La puissance théorique de cette machine est de 2,4 Gflops. La librairie de routines de communication installée sur cette machine est la librairie PVM.

Le SP1 d'IBM

La machine SP1 que nous utilisons est composée de 16 processeurs IBM RS6000. Elle fait partie du "pool" de machines parallèles de l'IMAG. Elle est administrée par le laboratoire LMC de Grenoble. Chaque processeur a une puissance crête de 125 Mflops et est associé à une mémoire de 64 Mbytes. Au total cette machine a une puissance de 2 Gflops et une mémoire de 1 Gbytes. Cette machine a plusieurs réseaux d'interconnexion: deux dédiés à l'administration et, deux autres partagés par les utilisateurs. Ces derniers permettent de transmettre les messages des utilisateurs entre les différents processeurs:

- Un réseau multi-étages avec une bande passante de 20 Mbytes/sec qui sert surtout à la programmation.
- Un Ethernet qui peut servir à la programmation mais qui est surtout dédié aux transferts de fichiers entre les différents disques.

Les librairies de "message passing" servant à la programmation de cette machine sont des librairies standards comme PVM, PVMe une version de PVM développée par IBM pour ses machines, et MPI.

La Paragon d'Intel

Cette Paragon appartient au "pool" des machines parallèles de l'Université Claude Bernard de Lyon. Cette machine MIMD, considérée comme une machine massivement parallèle, dispose de 32 nœuds. Deux nœuds sont dédiés aux entrées-sorties. La topologie de cette machine est une grille 2D. Chaque nœud est composé de deux processeurs i860 XP, l'un destiné au calcul et l'autre aux communications, d'une mémoire de 64 Mbytes et de deux DMA. Chaque processeur a une puissance crête de 75 Mflops, ce qui donne une puissance théorique pour la machine de 2,25 Gflops. Les nœuds sont connectés au réseau d'interconnexion par l'intermédiaire d'un PMRC ("Paragon Mesh Routing Chip"). Ces PMRC permettent de router les informations dans le réseau en utilisant le mode "Wormhole". Chaque PMRC possède quatre liens avec ses voisins.

Pour exécuter des programmes sur cette machine parallèle, l'utilisateur doit constituer une partition de l'ensemble des processeurs. Le principe de partitionnement exclusif des processeurs, assure à l'utilisateur une puissance théorique fonction du nombre de processeurs. Intel a développé sa propre librairie de communication NX qui intègre les opérations classiques de communication mais aussi des routines efficaces d'entrée-sortie.

Le Cray T3D

Le Cray T3D est la première machine massivement parallèle développée par Cray. La machine que nous étudions, appartient au centre de calcul du CEA Grenoble qui regroupe un Cray C90 et un Cray T3D. Son architecture est constituée de 64 nœuds. Chaque nœud est formé de deux noyaux de calcul et de stockage reliés au réseau d'interconnexion par un contrôleur mémoire. Le noyau de calcul est composé d'un processeur Alpha de Dec et d'une mémoire DRAM de 64 Mbytes. La topologie de cette machine est un tore 3D (8x4x4) de 64 nœuds ou 128 processeurs. La puissance crête de cette machine est de 19,2 Gflops.

Cette machine est une machine MIMD à mémoire distribuée mais logiquement partagée. Le système d'adressage des données utilise une adresse sur 36 bits, 12 pour la mémoire locale et 24 bits de déplacement. Grâce à ce système d'adressage unique, un processeur peut accéder à une donnée locale ou à une donnée distante via le contrôleur mémoire. Celui-ci génère un message qui est routé dans le réseau d'interconnexion dans la direction x, y puis z. Le protocole de transfert utilisé est le "Wormhole". La librairie de communication ShMem ("Shared Memory") utilise ce principe de la mémoire virtuellement partagée. La librairie PVM, que nous utilisons, a été adaptée à ce mode d'adressage. Des travaux ont permis d'évaluer les performances des différentes librairies de communication du T3D.

4.5 Conclusion

Nous avons présenté une méthodologie pour implémenter les méthodes de reconstruction 3D. Elle est basée sur une programmation parallèle efficace des opérateurs de projection et de rétroprojection. Elle privilégie la portabilité des algorithmes parallèles pour leur implémentation sur nos machines cibles (tableau 4).

Machine	Type	Processeur	Nombre	Topologie
SUN4	séquentiel	Sparc2	1	
MasPar	SIMD	élémentaire	1024	Tore 2D
Réseau de stations	MIMD	Sparc2 et Sparc1	2 et 3	Ethernet
Paragon (Intel)	MIMD	i860	32	Grille
T3D (Cray)	MIMD	AXP	128	Tore 3D
SP1 (IBM)	MIMD	RS6000	32	Réseau multi-étages
Ferme de Processeurs	MIMD	AXP	16	Giga-Switch

TAB. 4 - *Machines Cibles*

Nous développons dans les chapitres suivants trois méthodes de reconstruction basées sur ces opérateurs parallèles: une méthode analytique, l'algorithme de Feldkamp, et deux méthodes algébriques, la méthode ART par blocs et la méthode SIRT. Pour chacune d'elles, on présentera:

- Sa formulation analytique ou algébrique en fonction de la géométrie d'acquisition.

- Son implémentation séquentielle.
- Son implémentation parallèle par une approche locale ou globale.
- Une optimisation de son implémentation en utilisant les techniques de minimisation des temps de calculs et de répartition de charge.
- Ses résultats au niveau de la qualité de l'image reconstruite et au niveau des performances atteintes sur chaque machine parallèle.

Pour tester nos algorithmes, nous utiliserons des fantômes construits à partir de notre interface. De plus, nous comparerons les résultats de ces trois méthodes de reconstruction au niveau de la qualité d'image obtenue, comme au niveau des performances. Cette comparaison nous servira à définir des critères de qualité et de performance pour reconstruire une image 3D à partir de données expérimentales.

Chapitre 5

Parallélisation d'une méthode analytique

5.1 Introduction

Nous présenterons dans ce chapitre la parallélisation d'une méthode analytique: la méthode de Feldkamp. Nous commencerons par exposer la méthode dans un domaine continu et donner sa formulation discrète. Nous décrirons ensuite les différents filtres que nous mettrons en œuvre lors de la reconstruction. Puis, nous montrerons que cette méthode est basée sur un opérateur discret qui est similaire à celui défini dans le chapitre 4. Enfin, nous expliciterons l'algorithme séquentiel associé à la formulation discrète de la méthode de Feldkamp.

Nous présenterons ensuite la parallélisation de cette méthode. Nous nous intéresserons dans un premier temps à la parallélisation sur une machine SIMD. Nous redéfinirons les opérateurs de base parallèles pour les adapter aux machines SIMD. Contrairement aux machines MIMD, nous montrerons qu'il est plus efficace d'utiliser des opérateurs parallélisés suivant une approche globale plutôt qu'une approche locale. Nous présenterons des résultats expérimentaux de performances de l'implémentation de cette méthode sur une machine SIMD: une Maspar composée de 1024 processeurs.

Dans un deuxième temps, nous expliciterons la parallélisation de la méthode de Feldkamp sur une machine MIMD. La méthode sera alors implémentée avec un opérateur de rétroprojection parallélisé suivant une approche locale. Nous utiliserons

les techniques de recouvrement calculs/communications pour améliorer l'efficacité de nos implémentations. De plus, nous utiliserons un partitionnement adaptatif pour équilibrer la charge des processeurs dans le cas d'une machine MIMD à processeurs hétérogènes. Nous définirons ainsi trois implémentations parallèles de cette méthode sur machine MIMD: une version en mode synchrone, une version asynchrone et une version asynchrone utilisant un partitionnement adaptatif. Nous présenterons des résultats expérimentaux de performances de chacune de ces implémentations sur nos différentes machines MIMD.

Enfin, nous montrerons des résultats de reconstructions d'images 3D définies à partir de données simulées. Nous reconstruirons des images simples composées d'une sphère, et des images plus complexes composées de plusieurs sphères imbriquées. Nous évaluerons la qualité de la reconstruction en fonction du filtre utilisé à l'aide d'un critère de distance entre l'image initiale et l'image reconstruite.

5.2 Méthode de Feldkamp

5.2.1 Formulation en géométrie conique

Son principe est basé sur une rétroprojection pondérée des projections préalablement filtrées et pondérées. La géométrie d'acquisition, liée à la méthode de Feldkamp, est une trajectoire circulaire de l'ensemble source-détecteur (figure33); l'angle Φ a donc une valeur constante. La fonction reconstruite est donc une approximation de la fonction f définie au chapitre 2.

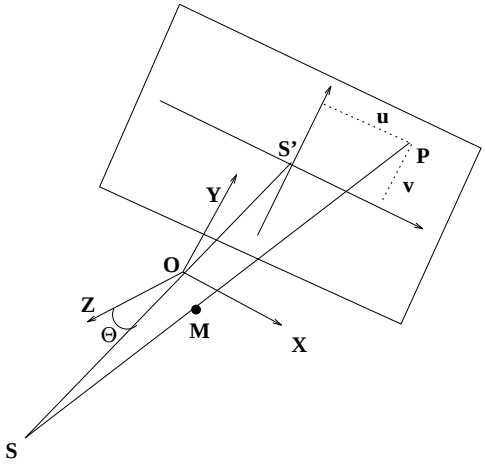


FIG. 33 - *Géométrie d'acquisition de la méthode de Feldkamp*

Pour corriger la divergence du faisceau, les projections de l'image à reconstruire sont pondérées en chaque point P du plan de projection par le rapport entre la distance source-détecteur et la distance entre la source et P: $\frac{\|\overrightarrow{SS'}\|}{\|\overrightarrow{SP}\|}$. Les projections pondérées correspondent à la transformation en rayons X pondérée de l'image 3D projetée. Elles s'expriment en fonction des projections $p_{\Theta\Phi}$ de l'image 3D. Les projections pondérées, notées p'_Θ sont alors exprimées en fonction des projections p_Θ en tout point P du plan de projection de coordonnées (u, v):

$$p'_\Theta(u,v) = p_\Theta(u,v)\frac{D}{\sqrt{D^2+u^2+v^2}} \tag{5.103}$$

Les projections p'_Θ sont préalablement filtrées avant d'être rétroprojetées. Les filtres utilisés classiquement par les algorithmes de rétroprojection sont des filtres monodimensionnels $K(V)$ qui filtrent chaque projection p'_Θ, ligne par ligne, dans le domaine des fréquences.

Le passage dans le domaine des fréquences nécessite des hypothèses supplémentaires sur la fonction f à reconstruire car le domaine d'intégration de l'opérateur de transformée de Fourier est infini. On suppose donc, que la fonction f a un spectre limité dans le domaine des fréquences inclus dans un cercle de rayon V_β. Cela permet de réduire le domaine d'intégration des opérateurs de transformée de Fourier utilisés lors du filtrage des projections. Ainsi le filtre $K(V)$ mis en œuvre doit vérifier la condition suivante: $K(V) = |V|$ si $|V| < V_\beta$.

L'opération de filtrage peut s'écrire :

$$\tilde{p}_\Theta(u,v) = \mathcal{F}^{-1}\Big(\mathcal{F}(p'_\Theta(u,V)).K(V)\Big) \tag{5.104}$$

La fonction f est reconstruite à partir de ces projections $\tilde{p}_\Theta$ par une opération de rétroprojection pondérée en chaque point $M(x,y,z)$:

$$f(x,y,z) = \frac{1}{2}\int_{\theta=0}^{2\pi} \tilde{p}(u,v,-D_2)\frac{D^2}{(D_1-Z)^2}d\theta \tag{5.105}$$

où $(u,v,-D_2)$ sont les coordonnées de projection du point M de coordonnées (X,Y,Z) après une rotation d'angle Θ. Le facteur de pondération s'interprète de la façon suivante: sur une même ligne de projection (SP), la valeur de la fonction f est corrigée en chacun des points M de cette droite par le rapport $\frac{\|\overrightarrow{SP}\|^2}{\|\overrightarrow{SM}\|^2}$.

5.2.2 Choix de l'opérateur de filtrage

Comme nous l'avons supposé précédemment, les filtres utilisés par la méthode de Feldkamp vérifient la condition: $K(V) = |V|$ si $|V| < V_\beta$. Le choix d'un "bon filtre" est souvent fonction de la nature des acquisitions. Dans le cas idéal où les acquisitions ne sont pas bruitées, le choix se portera sur un filtre rampe. Dans la plupart des cas, les acquisitions sont bruitées et les hautes fréquences génèrent de nombreuses erreurs; le choix d'un bon filtre se portera, alors, vers des filtres réduisant l'influence des hautes fréquences grâce à une fenêtre adaptée à la nature du bruit (fenêtre cosinus, fenêtre sinus, fenêtre de Hamming ...). Leur formulation s'exprime, dans le domaine fréquentiel, par la multiplication de cette fenêtre $W(V)$ par la fonction de base $|V|$: $K(V) = |V|W(V)$.

Pour chacun des filtres, que nous avons testé, nous donnons sa définition dans le domaine fréquentiel. Nous comparerons ces différents filtres lors de l'analyse de nos résultats de reconstruction à partir de données simulées et expérimentales.

Le filtre rampe

Ce filtre, appelé aussi filtre de Ram-Lak [RL71] tient son nom de sa représentation caractéristique dans le domaine des fréquences (figure 34).

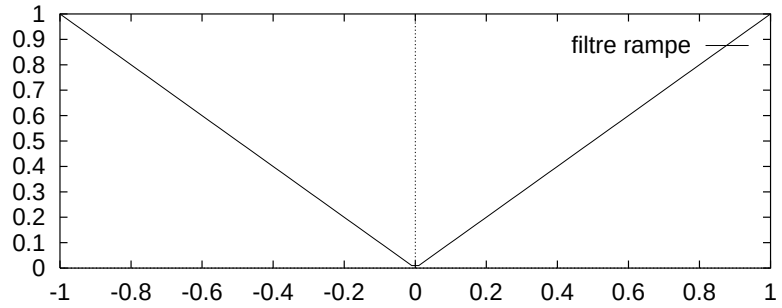


FIG. 34 - *Représentation fréquentielle du filtre rampe*

Il vérifie la condition:

$$\mathcal{F}K(V) = |V| rect_{V_\beta}(V) \tag{5.106}$$

où $rect_{V_\beta}$ est la fonction rectangulaire égale à 1 sur l'intervalle $[-V_\beta, V_\beta]$ et nulle en dehors.

Le filtre avec une fenêtre sinus

Le filtre de Shepp et Logan [SL74] utilise une fenêtre permettant d'atténuer les hautes fréquences et vérifiant:

$$\mathcal{F}K(V) = |V| rect_{V_\beta}(V) \frac{\sin(\pi V/V_\beta)}{2\pi} \tag{5.107}$$

La figure 35 montre de quelle manière ce filtre atténue davantage les hautes fréquences que le filtre rampe.

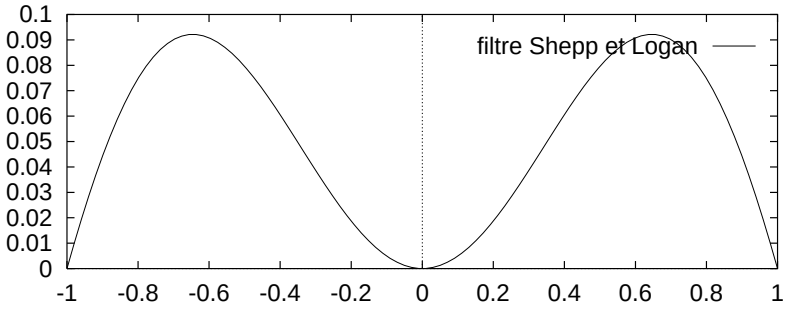


FIG. 35 - *Représentation fréquentielle du filtre de Shepp et Logan*

Le filtre avec une fenêtre cosinus

Nous utilisons aussi un filtre utilisant une fenêtre cosinus qui permet, suivant la valeur du coefficient d'apodisation α, de modifier la fenêtre. La définition fréquentielle de ce filtre s'exprime par :

$$\mathcal{F}K(V) = |V| rect_{V_\beta}(V) \Big(\frac{1 + \cos(\pi V / V_\beta)}{2} \Big)^\alpha \tag{5.108}$$

Les figures (36,37,38,39) sont les représentations fréquentielles de ce filtre pour différentes valeurs du coefficient d'apodisation α.

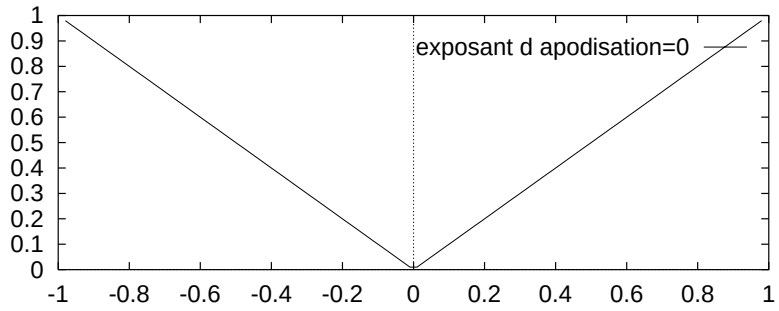


FIG. 36 - *Filtre avec fenêtre cosinus et* $\alpha = 0$

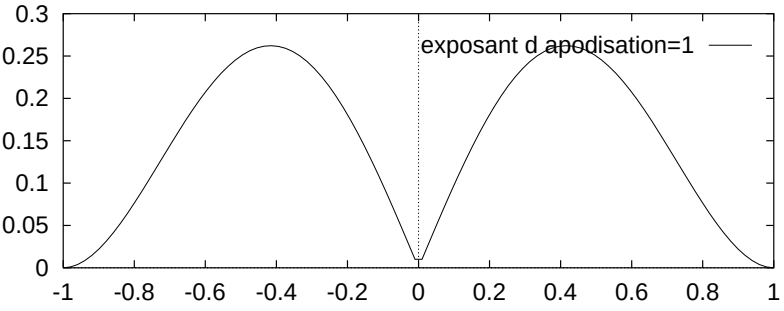


FIG. 37 - *Filtre avec fenêtre cosinus et* $\alpha = 1$

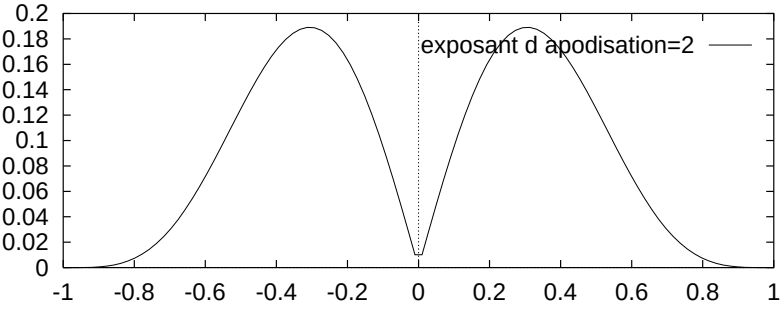


FIG. 38 - *Filtre avec fenêtre cosinus et* $\alpha = 2$

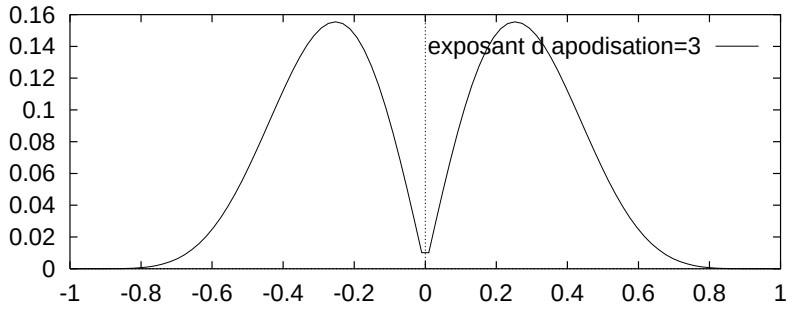


FIG. 39 - *Filtre avec fenêtre cosinus et* $\alpha = 3$

5.2.3 Discrétisation de la méthode

La discrétisation de l'algorithme de Feldkamp peut être formulée grâce à l'opérateur discret de rétroprojection qui met en relation chaque point échantillonné de la fonction f_d avec l'ensemble de ses projections. Nous explicitons, dans le cas discret, la pondération et le filtrage des projections p_{Θ_n}, effectués avant la rétroprojection.

La pondération consiste à affecter à chaque point (p, q) des projections p_{Θ_n}, une contribution qui est fonction de la distance du point (p, q) à la source S. Nous obtenons ainsi des projections pondérées p'_{Θ_n} qui sont déterminées à partir des projections p_{Θ_n} par la relation suivante:

$$p'_{\Theta_n}(p,q) = p_{\Theta_n}(p,q)\frac{D}{\sqrt{D^2 + (p\triangle_p)^2 + (q\triangle_p)^2}} \tag{5.109}$$

Les projections p'_{Θ_n} sont ensuite filtrées dans le domaine des fréquences. Le passage dans le domaine des fréquences nécessite un opérateur de transformée de Fourier discret. Soit $p'_{\Theta_n}(.,l)$ ($1 \leq l \leq M$), l'ensemble des M lignes de chaque projection, l'opération de filtrage s'effectue ligne par ligne et peut s'écrire:

$$\tilde{p}_{\Theta_n}(.,l) = \mathcal{F}_{\mathbf{d}}^{-\mathbf{1}}\Big(\mathcal{F}_{\mathbf{d}} p'_{\Theta_n}(L).K(L)\Big) \tag{5.110}$$

où $\mathcal{F}_{\mathbf{d}}$ est la transformée de Fourier discrète et $\mathcal{F}_{\mathbf{d}}^{-\mathbf{1}}$ son inverse. La réponse impulsionnelle du filtre est une fonction supposée finie. Afin, d'améliorer le résultat du filtrage, le calcul de la FFT est effectué pour $2M$ points aprés avoir ajouté $M/2$ points à gauche et $M/2$ à droite de la ligne à filtrer.

La méthode de Feldkamp étant basée sur une rétroprojection pondérée des projections $\tilde{p}_{\Theta_n}$, la fonction f_d est donc reconstruite à partir des projections $\tilde{p}_{\Theta_n}$, par un opérateur de rétroprojection discret similaire à celui défini dans le chapitre 4. Ici, l'opération de rétroprojection discrète est pondérée en chaque point M de f_d. Cette pondération définie par le rapport des distances, d'une part entre la source et la projection P de M, et d'autre part la source et le point M, est égale à:

$$Pond(i\triangle_v, j\triangle_v, k\triangle_v) = \frac{\|\overrightarrow{SP}\|^2}{\|\overrightarrow{SM}\|^2} = \frac{D^2}{(D_1 - \triangle_v(i\cos\Phi\sin\Theta + j\sin\Phi\cos\Theta + k\cos\Theta))^2} \tag{5.111}$$

Dans le cas où $\Phi = 0$, on obtient:

$$Pond(i\triangle_v, j\triangle_v, k\triangle_v) = \frac{D^2}{(D_1 - \triangle_v(i\sin\Theta + k\cos\Theta))^2} \tag{5.112}$$

La formulation discrète de la méthode de Feldkamp s'exprime alors en tout point M de f_d par:

$$f_d(i\triangle_v, j\triangle_v, k\triangle_v) = Pond(i\triangle_v, j\triangle_v, k\triangle_v)\Big(\sum_{n=1}^{m}\Big((1-\lambda)(1-\mu)\,\tilde{p}_{\Theta_n}(p,q) + \\ + \lambda(1-\mu)\,\tilde{p}_{\Theta_n}(p+1,q) + \mu(1-\lambda)\,\tilde{p}_{\Theta_n}(p,q+1) + \lambda\mu\,\tilde{p}_{\Theta_n}(p+1,q+1)\Big)\Big) \tag{5.113}$$

où λ et μ sont les coefficients de l'interpolation bilinéaire.

L'algorithme séquentiel de la méthode de Feldkamp est déduit de ces formulations discrètes.

5.2.4 Algorithme séquentiel

L'algorithme séquentiel de la méthode de Feldkamp consiste à effectuer les opérations de pondération, de filtrage et de rétroprojection pondérée sur chaque projection.

L'opération de pondération et de filtrage ne fait intervenir que les *images de projections* Im_j, tandis que l'opération de rétroprojection pondérée met en relation les Im_j avec le volume V. L'algorithme séquentiel est donc articulé autour de deux algorithmes: l'algorithme de pondération-filtrage et l'algorithme de rétroprojection pondérée.

Algorithme de pondération-filtrage

Comme le filtrage est effectué ligne par ligne dans la méthode de Feldkamp, on pondère les images ligne par ligne pour conserver une homogénéité dans le prétraitement des Im_j. Cet algorithme traite chaque ligne en quatre étapes:

- pondération de la ligne,
- passage dans le domaine des fréquences de la ligne par un algorithme de transformée de Fourier,
- multiplication de la ligne avec le filtre défini dans le domaine des fréquences,
- passage dans le domaine direct par un algorithme de transformée de Fourier inverse.

L'algorithme de transformée de Fourier que nous avons choisi est un algorithme de transformée de Fourier rapide ("Fast Fourier Transform") [CT65].

Algorithme 7 *Pondération-Filtrage*

Pour $q = 1$ à M
  Pour $p = 1$ à M (pondération)
    $Im_j(p,q) \leftarrow Im_j(p,q).\frac{D}{\sqrt{D^2+(p\triangle_p)^2+(q\triangle_p)^2}}$
  $Im_j(.,q)' \leftarrow (FFT(Im_j(.,q))$
  Pour $p = 1$ à M (filtrage)
    $Im'_j(p,q) \leftarrow Im'_j(p,q).Filtre(p)$
  $Im_j(.,q) \leftarrow (FFT^{-1}(Im_j(.,q)')$

où $Im_j(.,q)$ représente la $q$ième ligne de Im_j.

Cet algorithme montre que le coût de la pondération d'une *image de projection* est de l'ordre de $6M^2$ opérations. Le coût du filtrage dépend du coût de l'algorithme de FFT. L'algorithme de Cooley et Tukey a un coût en $n \log n$. Cependant, pour améliorer le filtrage de chaque ligne des images Im_j, on ajoute $M/2$ zéros, en début et en fin de chaque ligne. De plus, cet algorithme travaille à partir de nombres complexes: le filtrage et le calcul des FFT de chaque ligne sont effectués sur $4M$ points. Le coût du filtrage est donc de $8M^2 \log 4M + 16M^2$. Au total, la pondération et le filtrage d'une image Im_j coûte: $8M^2 \log 4M + 22M^2$.

Algorithme de rétroprojection pondérée

Cet algorithme est directement déduit de celui de la rétroprojection. Il effectue pour chaque voxel $v(i,j,k)$ la même série de quatre opérations:

- Calcul des nouvelles coordonnées (X,Y,Z) du voxel $v(i,j,k)$ après la rotation d'angle Θ.
- Calcul de l'adresse de projection (u,v) et détermination de la cellule (p,q) contenant le point de projection.
- Pondération de la rétroprojection calculée d'après l'équation 5.112.
- Sommation de la valeur des quatre pixels par interpolation bilinéaire affectée par la valeur de la pondération.

Cette série d'opérations est représentée par l'opération élémentaire de la rétroprojection, notée $vi \overset{\mathcal{R}}{\leftarrow} im_{na}, im_{nb}, im_{nc}, im_{nd}$, car elle diffère de celle-ci seulement par le calcul de la pondération. Ainsi, l'algorithme de la rétroprojection pondérée est le même que celui de l'opérateur projection. Le coût de cet algorithme en terme d'opérations élémentaires est donc égal à: $N^3.T(\overset{\mathcal{R}}{\leftarrow})$ pour chaque projection Im_j.

Algorithme de Feldkamp

On construit à partir des deux algorithmes précédent, l'algorithme de Feldkamp:

Algorithme 8 *Feldkamp*

créer(V)
Pour $j = 1$ à m
  lire(Im_j)
  Pondération-Filtrage(Im_j)
  Pour $i = 1$ à N^3
    $v_i \overset{\mathcal{R}}{\leftarrow} im_{ja}, im_{jb}, im_{jc}, im_{jd}$
écrire(V)

Si l'on prend en compte le coût de chaque algorithme, on obtient un coût théorique total de: $m.N^3 \overset{\mathcal{R}}{\leftarrow} +8m.M^2 \log 4M + 4m.M^2$. Lorsque $N^3 >> m.M^2$, on peut négliger le coût du filtrage et de la pondération.

5.2.5 Expérimentation séquentielle

L'algorithme de Feldkamp a été implémenté sur notre machine séquentielle de référence: un Sun Sparc 3. Le tableau 5 présente les temps de reconstruction d'un volume V de taille N^3 à partir de 100 images Im_j de taille 128^2. La complexité de l'algorithme de Feldkamp est alors de: $100.N^3.T(\overset{\mathcal{R}}{\leftarrow})$. Nous comparons les temps de reconstruction avec les temps d'exécution des opérateurs de base: la projection et la rétroprojection. Les opérateurs de base ont une complexité de $100.N^3.T(\overset{\mathcal{P}}{\rightarrow})$ pour la projection et de $100.N^3.T(\overset{\mathcal{R}}{\leftarrow})$ pour la rétroprojection.

Algorithme	$N = 32$	$N = 64$	$N = 128$
Projection	162	961	7639
Rétroprojection	118	923	7334
Feldkamp	437	1565	10606

TAB. 5 - *Temps d'exécution de l'algorithme de Feldkamp et des opérateurs de base (sec)*

Les résultats sont en accord avec les coûts théoriques des algorithmes. En effet, lorsque N évolue, les temps d'exécution des opérateurs de bases sont multipliés pour la plupart par 8. De plus, ils montrent bien que les opérateurs de base ont un coût similaire. Le coût de la pondération et du filtrage pénalisent les performances de l'algorithme de Feldkamp par rapport à l'algorithme de rétroprojection.

Les performances obtenues par l'algorithme de Feldkamp indiquent que les problèmes de reconstruction pour $N \geq 128$ nécessitent des temps de calcul très importants. La parallélisation de cet algorithme est une bonne solution pour réduire les temps de calcul.

5.3 Parallélisation de la méthode de Feldkamp

Une bonne parallélisation nécessite d'identifier les noyaux de calculs parallélisables. L'algorithme séquentiel comporte deux opérations: une opération de pondération-filtrage et une opération de rétroprojection pondérée. Pour obtenir de bonnes performances pour chaque opération, on doit définir le placement optimal des données et leur schéma de communication induit.

L'opérateur de pondération-filtrage peut être parallélisé en utilisant les algorithmes parallèles de FFT. Ceux-ci permettent de calculer les transformées de Fourier des lignes des Im_j distribuées sur les processeurs [CD93]. Bien que cette parallélisation soit efficace, elle entraîne un sur-coût de communication dû au réarrangement des données nécessaires pour exécuter l'opérateur de rétroprojection. Nous avons donc choisi de paralléliser cet opérateur en utilisant la répartition des données de l'opérateur de rétroprojection.

L'opérateur de rétroprojection est parallélisé suivant le modèle d'exécution de la machine cible. Nous présentons plusieurs approches pour l'implémenter sur les machines SIMD. Pour les machines MIMD, nous utilisons un opérateur de projection parallèle, développé l'approche locale présentée dans le chapitre 4.

Pour chaque implémentation, nous évaluons les coûts théoriques des algorithmes et nous analysons les performances obtenues. Nous montrerons que les architectures MIMD semblent mieux adaptées aux problèmes de reconstructions que les architectures SIMD.

5.4 Parallélisation sur machine SIMD

Nous avons implémenté cette méthode sur une machine SIMD: une Maspar composée de 32x32 processeurs et décrite au chapitre 4. La taille de la mémoire de chaque processeur étant trop faible: $16Kbytes$, l'ensemble des données ne peut pas tenir en même temps sur la mémoire de tous les processeurs. Le volume V est donc divisé en T tranches de taille $\frac{N^3}{T}$. L'image 3D est alors reconstruite par tranche, séquentiellement à partir des *images de projection* Im_J.

A tout instant, une tranche du volume V et une *image de projection* Im_j sont réparties sur les mémoires des processeurs. Chaque processeur PE_i a en mémoire un sous-volume V_i de taille $\frac{N^3}{T.PE}$ et une sous-image $Im_{(j,PE_i)}$ de taille $\frac{M^2}{PE}$. L'algorithme de Feldkamp, que l'on déduit de cette répartition, est implémenté en parallèle à partir de l'opérateur parallèle de rétroprojection:

Algorithme 9 *Feldkamp Général*

créer(V)
Pour $t = 1$ à T
  Pour $j = 1$ à m
    lire(Im_j)
    Pondération-Filtrage(Im_j)
    Rétroprojection(Im_j)
écrire(V)

Comme chaque projection Im_j est répartie en PE sous-images $Im_{(j,PE_i)}$, nous redéfinissons la parallélisation de l'opérateur de rétroprojection suivant l'approche locale et l'approche globale.

5.4.1 Approche locale

L'approche locale consiste à effectuer le calcul local de la rétroprojection des $Im_{(j,PE_i)}$ sur V_i puis à transférer les $Im_{(j,PE_i)}$. Nous évaluons différents schémas de communication des $Im_{(j,PE_i)}$.

Schéma de communication classique: l'anneau

Avec un schéma de communication en anneau, la rétroprojection d'une image Im_j est effective lorsque toutes les $Im_{(j,PE_i)}$ ont été rétroprojetées sur les V_i. Cependant, on peut constater que chaque voxel v_i a une unique projection sur Im_j. Ainsi un sous-volume V_i est reconstruit par rétroprojection d'une région de Im_j contenue dans un petit nombre de sous-images $Im_{(j,PE_i)}$. Il n'est donc pas nécessaire de communiquer toutes les sous-images $Im_{(j,PE_i)}$ au processeur PE_i. Ce schéma de communication n'est donc pas adapté car il génère des communications inutiles pour la reconstruction.

Schéma de communication dynamique

L'objectif est d'effectuer seulement les transferts des sous-images $Im_{(j,PE_i)}$ nécessaires au calcul local des V_i. Dans ce but, on construit pour chaque V_i la liste des sous-images $Im_{(j,PE_i)}$ contenant la région permettant de reconstruire V_i. Pour ordonner ces transferts, on se sert de la position de chaque sous-image $Im_{(j,PE_i)}$ sur la grille des processeurs par rapport à la position des V_i.
Chaque processeur PE_i étant repéré par sa position (x, y) sur le tore 2D, chaque transfert d'une sous-image $Im_{(j,PE_{x_1,y_1})}$ vers le processeur PE_{x_2,y_2} est exprimé comme un déplacement $D_{\triangle_x,\triangle_y}$ où $\triangle_x = x_2 - x_1$ et $\triangle_y = y_2 - y_1$. On obtient donc sur chaque processeur la liste des déplacements D_i à effectuer. L'ensemble D de ces déplacements permet de réaliser la rétroprojection des Im_j sur V_i.

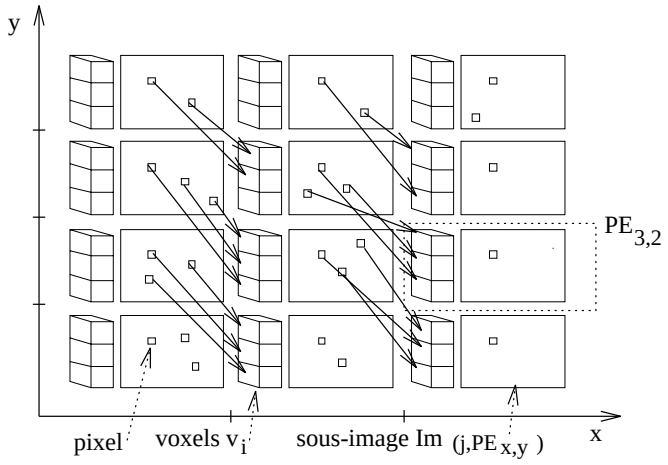


FIG. 40 - *Rétroprojection locale sur un machine SIMD*

Sur tous les processeurs $PE_{x,y}$, on associe à chaque déplacement $D_{\triangle_x,\triangle_y}$ la liste des voxels qui se projettent sur l'image se trouvant sur les processeurs $PE_{x+\triangle_x,y+\triangle_y}$. Ainsi, après le déplacement des sous-images, on effectue les opérations élémentaires de rétroprojection $\overset{\mathcal{R}}{\Leftarrow}$ à partir de cette liste de voxels. La figure 40 montre les pixels contribuant à la valeur des voxels lorsque l'on déplace les sous-images $Im_{(j,PE_{x_1,y_1})}$ d'un déplacement $D_{\triangle_{+1},\triangle_{-1}}$.

Pour minimiser les communications, on ordonne cette liste D en calculant pour chaque déplacement $D_{\triangle_x,\triangle_y}$ un déplacement relatif $Dr_{\delta_x\delta_y}$ fonction du déplacement précédent. $D = \cdots D_{2,3}D_{2,4}D_{3,4}\cdots \Rightarrow D = \cdots D_{2,3}Dr_{0,1}Dr_{1,0}\cdots$
Ainsi, si la liste contient les déplacements $D_{1,\triangle_y}$ et $D_{2,\triangle_y}$ cela signifie que l'ensemble des processeurs doit réaliser les transferts suivants:

1. communication des sous-images $Im_{(j,PE_{x,y})}$ au processeur $PE_{x+1,y}$,
2. calcul de la rétroprojection sur chaque processeur à partir de $Im_{(j,PE_{x-1,y})}$,
3. communication des sous-images $Im_{(j,PE_{x,y})}$ vers leur processeur d'origine,
4. communication des sous-images $Im_{(j,PE_{x,y})}$ au processeur $PE_{x+2,y}$,
5. calcul de la rétroprojection sur chaque processeur à partir de $Im_{(j,PE_{x-2,y})}$,
6. communication des sous-images $Im_{(j,PE_{x,y})}$ vers leur processeur d'origine.

En calculant le déplacement relatif entre deux déplacements consécutifs, on montre que l'on peut remplacer les communications 3 et 4 de notre exemple par une seule communication. Ici le déplacement relatif est égal à :$Dr_{\delta_x\delta_y} = D_{2,\triangle_y} - D_{1,\triangle_y} = Dr_{1,0}$. Cela signifie que les communications 3 et 4 peuvent être remplacées par la communication entre les processeurs $PE_{x+1,y}$ et $PE_{x+2,y}$ des sous-images $Im_{(j,PE_{x,y})}$. Globalement on effectue un glissement de Im_j sur la grille des processeurs lorsque l'on communique les sous-images $Im_{(j,PE_{x,y})}$.

L'algorithme de rétroprojection d'une image Im_j sur une tranche de V, qui découle de ce schéma de communication dynamique, est le suivant:

Algorithme 10 *Rétroprojection locale*

création des listes D_i
Pour $\triangle_y = -PE/2$ à $PE/2$
  Pour $\triangle_x = -PE/2$ à $PE/2$
    si $D_{\triangle_x \triangle_y} \in \bigcup_{i=0}^{PE} D_i$
      calcul $Dr_{\delta_x \delta_y}$
      envoi des $Im_{(j,PE_{x,y})}$ aux $PE_{x+\delta_x, y+\delta_y}$
      rétroprojection locale des V_i

Grâce à l'ordonnancement des déplacements, les communications établies par les déplacements relatifs $Dr_{\delta_x \delta_y}$ se limitent aux communications entre processeurs voisins car δ_x et δ_y prennent pour valeur $(-1, 0, +1)$. Les communications sont effectuées à travers le réseau "XNet" de la Maspar qui relie chaque processeur à ses huit voisins.

5.4.2 Approche globale

Cette approche consiste à rétroprojeter les pixels en effectuant des diffusions globales de leur valeur à travers le réseau. Nous utilisons pour cette approche le réseau "Global Router" qui permet de relier chaque processeur aux autres par l'intermédiaire d'un"cross-bar hiérarchique".

Une première étape permet de déterminer l'adresse des pixels de l'image Im_j qui contribuent à la valeur de chaque voxel v_i. A partir de ces adresses, on calcule les adresses relatives qui sont définies en fonction de la position du pixel sur chaque sous-image $Im_{(j,PE_{x,y})}$. Ces adresses relatives permettent, lors de la diffusion, d'éviter les conflits d'adressage sur chaque processeur. Comme le montre la figure 41, le schéma de communication de cette approche va consister à relier à chaque instant, des pixels et des voxels: on effectue en paralléle des diffusions. Cela permet de commmuniquer seulement la valeur des pixels intervenant dans la rétroprojection. Pour les déterminer, on crée sur chaque processeur des listes L_i triées sur les adresses relatives de ces pixels.

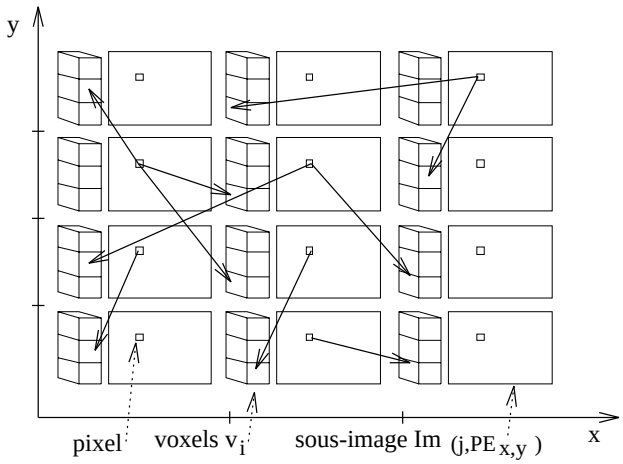


FIG. 41 - *Rétroprojection globale sur un machine SIMD*

L'algorithme parallèle effectue séquentiellement des multi-diffusions à partir de chaque adresse relative, jusqu'à ce que toutes les listes soient vides.

Algorithme 11 *Rétroprojection globale*

création des listes L_i
Tant que les L_i sont non vides
  Pour $j = 0$ à M^2/PE
    si $L_i(j)$ non vide
      transfert des pixels
      rétroprojection des pixel

5.4.3 Evaluation de ces approches

On examine pour chacune des approches leur coût théorique respectif.

Approche locale

Dans le cas d'une approche locale, l'algorithme effectue K étapes pour rétroprojeter une image Im_j. Le temps de calcul sur chaque processeur est fonction du temps de calcul de chaque étape $T^k_{cal(i)}$. Nous supposons que globalement le temps de calcul total est égal à $T_{cal(i)} = \sum_{k=1}^{K} T^k_{cal(i)} = \frac{N^3}{T.PE}.T(\overleftarrow{\mathcal{R}})$. Entre chaque étape de calcul, les processeurs communiquent les sous-images $Im_{(j,PE_{x,y})}$. Pour évaluer le temps de communication total, on peut considérer que le nombre d'étapes K est majoré par PE^2 et est minoré par $\alpha.PE$, si l'on suppose que chaque tranche se projette sur une

partie de l'image Im_j de largeur $\alpha.\frac{M^2}{PE}$ (figure 42). Ainsi le temps de communication total $T_{com(i)}$ vérifie l'inégalité suivante:

$$\alpha.PE.\frac{M^2}{PE} \leq T_{com(i)} \leq PE^2.\frac{M^2}{PE} - \alpha.M^2 \leq T_{com(i)} \leq PE.M^2 \tag{5.114}$$

Approche globale

Pour le deuxième algorithme, la rétroprojection parallèle est plus ou moins efficace en fonction du nombre total de diffusions parallèles à effectuer. On peut considérer, dans une hypothèse basse, que le nombre de diffusions parallèles est égal au nombre de voxels dans chaque tranche. Le coût de communication est donc égal à la taille d'un pixel multiplié par le nombre de voxels sur chaque tranche. Dans une hypothèse haute, toutes les diffusions sont parallèles, le nombre de diffusions est donc égal au nombre de voxels. Le coût de communication total vérifie l'inégalité suivante, si on suppose que la diffusion de n pixels sur une machine SIMD est égal à 1:

$$\frac{N^3}{T.PE} \leq T_{com(i)} \leq \frac{N^3}{T} \tag{5.115}$$

Comparaison des coûts de communications

Le coût des calculs de l'approche globale étant similaire à celui de l'approche locale, nous comparons le coût des communications pour différentes valeurs de N et M. Expérimentalement, nous avons observé que le nombre maximal de voxels en mémoire sur chaque processeur de la Maspar était de 128. On peut ainsi déterminer le nombre de tranches T pour chaque valeur de N et en déduire les valeurs encadrant le coût des communications pour la rétroprojection:

$$T_{com(min)} \leq T_{com(i)} \leq T_{com(max)} \tag{5.116}$$

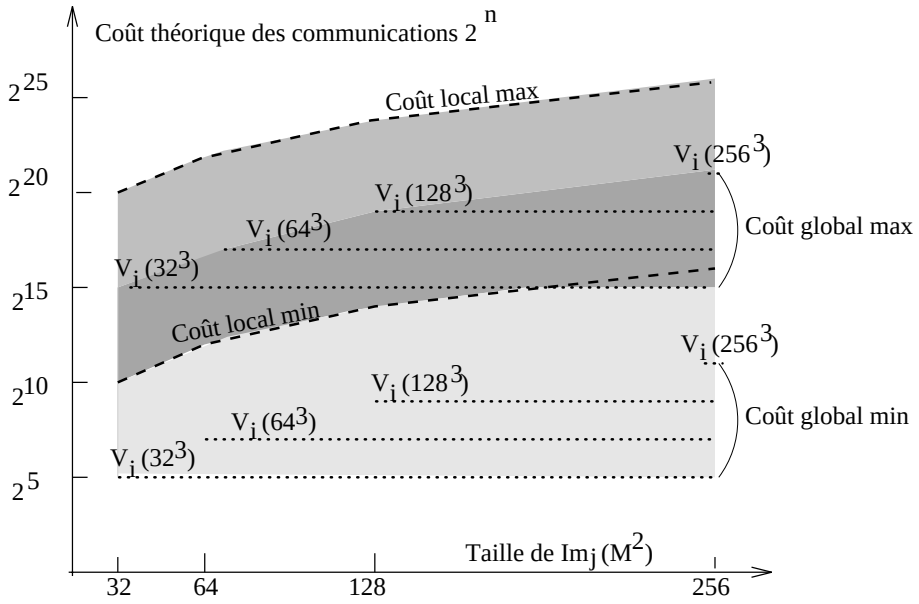


FIG. 42 - *Comparaison des coûts théoriques des communications suivant l'approche utilisée*

Les courbes de la figure 42 représentent les valeurs maximales et minimales du coût des communications des deux approches pour quelques valeurs de M et N. On peut remarquer que les coûts (max et min) de l'approche globale évoluent suivant la valeur de N par paliers, tandis que pour l'approche locale, ils génèrent deux courbes.

En délimitant les zones des coûts des communications, on peut remarquer qu'il existe trois zones distinctes: la zone définie par l'approche globale, la zone définie par l'approche locale et leur intersection. Pour une taille d'image donnée, sa communcation par une approche globale est moins couteuse que part une approche locale. Ainsi, cette figure montre que l'approche locale est théoriquement moins efficace que l'approche globale. Nous choisissons donc l'approche globale pour implémenter la méthode de Feldkamp.

5.4.4 Expérimentations et discussion

L'implémentation de la projection et de la rétroprojection étant similaires, nous avons validé notre choix en comparant les temps d'exécution d'une projection implémentée par une approche locale et par une approche globale [LPC94a]. Le tableau 6 présente quelques temps d'exécution d'une projection d'un volume V (N^3) sur une image Im_j (N^2).

Approche	$N = 32$	$N = 64$	$N = 128$
Locale	12.61	58.78	161.45
Globale	1.34	7.95	31.5

TAB. 6 - *Comparaison des temps d'exécution d'une projection locale ou globale (sec)*

Ces résultats montrent bien que l'approche globale permet d'obtenir de meilleures performances que l'approche locale pour implémenter la méthode de Feldkamp sur une machine SIMD. Nous avons testé notre algorithme développé par l'approche globale avec les mêmes données que notre implémentation séquentielle. Le tableau 7 permet de comparer les performances obtenues sur la Maspar et celles obtenues sur la machine séquentielle de référence.

Machine	$N = 32$	$N = 64$	$N = 128$
Sun4	437	1565	10606
Maspar	245	664	4276

TAB. 7 - *Temps d'exécution de Feldkamp sur un Sun4 et sur une Maspar(sec)*

La parallélisation sur machine SIMD, comme la Maspar, permet de diminuer sensiblement le temps de reconstruction. Les accélérations relatives obtenues avoisinent 2,5. Si l'on compare ces résultats avec le nombre de processeurs (1024), on peut considérer que l'implémentation sur une telle machine n'est pas efficace. Cela est dû principalement à la taille des mémoires des processeurs et aux schémas de communication qui ne sont pas efficaces sur une telle architecture.

Une solution pour améliorer les performances est de minimiser les communications. Des techniques de minimisation comme le recouvrement des communications par les calculs ne peuvent être utilisées sur des machines SIMD comme la Maspar. Ces limitations nous ont dirigés vers des machines plus puissantes qui permettent une gestion plus efficace des communications: les machines MIMD.

5.5 Parallélisation sur machine MIMD

Les machines MIMD actuelles offrent des puissances de calcul et des capacités mémoires qui permettent de traiter des problèmes de grande taille. Cependant, chacune de nos machines MIMD cibles a sa propre architecture. Pour éviter une réécriture des algorithmes, l'algorithme de Feldkamp a été parallélisé sur une machine MIMD abstraite. Dans un premier temps, nous avons testé cet algorithme sur notre pseudo-machine parallèle, le réseau de stations, et ensuite nous l'avons implémenté sur des machines parallèles plus performantes. Cette démarche nous a permis de tester notre schéma de communication et d'éviter de "gaspiller" le temps machine qui nous a été alloué sur certaines machines.

Nous présentons, dans cette section, l'algorithme implémenté sur nos machines MIMD. Il est basé sur l'opérateur parallèle de rétroprojection développé suivant une approche locale. Nous avons donc plongé l'architecture de notre machine abstraite sur chaque topologie. Nous supposons que l'ensemble des données est réparti sur nos processeurs. Le schéma de communication mis en œuvre pour leur transfert est l'anneau. Nous optimisons son implémentation suivant les caractéristiques de chaque machine MIMD par des techniques de recouvrement de communication et d'équilibrage de charges.

Les performances de nos implémentations sont évaluées en prenant le temps de chaque phase de calcul et de communication par l'appel à des routines systèmes. Nous identifions les phases de calcul par les symboles $\lceil, \rfloor, \lfloor, [$, et les phases de communication par les symboles $\Uparrow, \|, \Downarrow, \Updownarrow$. Le temps de calcul $T_{cal(i)}$ sur chaque processeur est égal à la somme des temps de chaque phase de calcul et, le temps de communication $T_{com(i)}$ est égal à la somme des temps de chaque phase de communication. Le temps total d'exécution $T_{//}$ est mesuré en fonction du temps de calcul $T_{cal(i)}$ et du temps de communication $T_{com(i)}$ de chaque processeur PE_i. Nous rappelons sa définition:

$$T_{//} = \max_{1 \leq i \leq PE} (T_{cal(i)} + T_{com(i)}) \tag{5.117}$$

5.5.1 Communication en mode synchrone

Nous avons implémenté cette méthode par une approche locale, car les projections Im_j peuvent être rétroprojetées sur les sous-volumes V_i sans ordre précis. Cette approche se caractérise par une succession de phases de calcul et de communication. Les communications sont effectuées de façon synchrone.

Algorithme synchrone

L'algorithme synchrone est développé à partir de l'algorithme de rétroprojection locale. Comme $m > PE$, il calcule parallèlement PE rétroprojections locales à partir de PE Im_j sur l'ensemble des processeurs. Avant d'être rétroprojettées, les *images de projection* Im_j sont pondérées et filtrées. Ce processus est réitéré $\frac{m}{PE}$ fois pour rétroprojeter l'ensemble des Im_j.

Algorithme 12 *Feldkamp Synchrone*

création des V_i
Pour $j = 1$ à $\frac{m}{PE}$
  lire(Im_j)
[Pondération-filtrage(Im_j)
  Pour $n = 1$ à PE
⌈ Pour $v_i \in V_i$
⌊ $v_i \xleftarrow{\mathcal{R}} im_{ja}, im_{jb}, im_{jc}, im_{jd}$
⇑ send(PE_{i+1}, Im_j)
⇓ recv(PE_{i-1}, Im_j)
écrire(V_i)

On suppose qu'initialement chaque processeur possède, dans sa mémoire de stockage, un ensemble Im_{PE_i} de $\frac{m}{PE}$ *images de projection* Im_j . Ainsi, les temps d'accès aux données sont identiques sur l'ensemble des processeurs. Sans cette hypothèse, les temps d'accès aux données sont non-homogènes et génèrent des temps d'attente qui doivent être pris en compte lors de l'évaluation des performances. Avec notre hypothèse, on ne comptabilise que les temps de calcul et de communication.

La rétroprojection parallèle de PE *images de projection* Im_j, a pour coût de calcul: $T_{cal(i)} = PE.\frac{N^3}{PE}.T(\overleftarrow{\mathcal{R}}) + 8M^2 \log 4M + 22M^2$ et pour coût de communication: $T_{com(i)} = (PE-1)(\beta + M^2\tau)$. Le coût total est donc égal à:

$$T_{cal(i)} = m\frac{N^3}{PE}.T(\overleftarrow{\mathcal{R}}) + \frac{m}{PE}(8M^2 \log 4M + 22M^2) \quad (5.118)$$

$$T_{com(i)} = \frac{m}{PE}(PE-1)(\beta + M^2\tau) \quad (5.119)$$

On peut remarquer que le coût total des calculs est égal à la taille des données réparties sur chaque processeur et que le coût total des communications est égal à la taille de toutes les *images de projection* Im_j.

5.5.2 Communication en mode asynchrone

Nous avons remarqué lors de nos premières expérimentations que la part du temps de communication est très importante par rapport au temps de calcul. Pour améliorer les performances de nos implémentations, nous utilisons donc une technique permettant de réduire la part des communications en effectuant un recouvrement des communications par les calculs.

Son principe défini dans le chapitre 4 consiste, pour chaque processeur, à rétroprojeter soit l'image Im_j reçue du processeur précédent sur l'anneau, soit une image provenant de sa mémoire de stockage qui sera préalablement pondérée et filtrée. Le calcul étant effectué, l'image Im_j est envoyée au processeur suivant sur l'anneau. L'émission et la réception de chaque Im_j sont programmées avec des routines de communication non-bloquantes, ce qui permet de rétroprojeter une autre image Im'_j pendant que Im_j est communiquée.

Le recouvrement des communications par les calculs est total tant que le processeur peut calculer une nouvelle rétroprojection à partir, soit d'une image reçue, soit d'une image de la mémoire de stockage. Lorsque toutes les images Im_j de l'ensemble Im_{PE_i} ont été rétroprojetées par le processeur PE_i sur V_i, les calculs, alors effectués sur le processeur PE_i, concernent des images Im_j venant d'autres processeurs. Après une phase de calcul, il se peut qu'un processeur PE_i attende une nouvelle *image de projection* Im_j à rétroprojeter, dans ce cas, la communication n'est pas entièrement

recouverte. Ces phases d'inactivité des processeurs dépendent du rapport entre le temps d'une phase de calcul et le temps d'une phase de communication.

Algorithme asynchrone

L'algorithme asynchrone de la méthode de Feldkamp s'inspire de l'algorithme de projection asynchrone. Cet algorithme doit gérer les *images de projection* Im_j pour s'assurer que chaque image a été rétroprojetée sur l'ensemble des V_i: on associe à chaque Im_j un compteur $NVol_j$ de rétroprojection. La rétroprojection est complète lorsque chaque V_i a reçu la contribution de toutes les Im_j ($NIma_i = m$).

Algorithme 13 *Feldkamp Asynchrone*

création des V_i
$NIma_i \leftarrow 0$
Tant que $NIma_i \leq m$
⇳ si (n-recv(PE_i, Im_k)=faux)
lire(Im_j), $NVol_j \leftarrow 0$
[Pondération-filtrage(Im_j)
sinon
$Im_j \leftarrow Im_k$
finsi

⌈ Pour $v_i \in V_i$
| $v_i \overset{\mathcal{R}}{\leftarrow} im_{ja}, im_{jb}, im_{jc}, im_{jd}$
⌊ $NIma_i \leftarrow Nima + 1$, $NVol_j \leftarrow NVol_j + 1$
si $NVol_j \leq PE$
⇳ send(PE_{i+1}, Im_j)
finsi
écrire(V_i)

Dans la section suivante, on montre qu'expérimentalement le recouvrement permet de diminuer le temps d'exécution. Cependant, le gain de performance obtenu ne peut être évalué théoriquement car il est fonction des caractéristiques de chaque machine. On se contente donc de l'équation suivante pour exprimer le coût théorique:

$$T_{//,i} = m\frac{N^3}{PE}.T(\overset{\mathcal{R}}{\leftarrow}) + \frac{m}{PE}(8M^2 \log 4M + 22M^2) + \epsilon(\beta + M^2\tau) \qquad (5.120)$$

où ϵ représente le pourcentage de communications non-recouvertes. Nous avons implémenté sur nos machines MIMD une version asynchrone de l'algorithme de Feldkamp.

5.5.3 Partitionnement adaptatif

Cette technique de régulation de charge permet d'optimiser les implémentations parallèles de l'algorithme de Feldkamp sur des machines parallèles à processeurs hétérogènes. Nous l'avons développée pour prendre en compte l'hétérogénéité des processeurs formant notre réseau de stations. Ce mécanisme de régulation de charge permet de tirer partie de la puissance maximale du réseau de stations.

Lors de l'examen des performances de l'algorithme asynchrone, nous nous sommes aperçus que les processeurs les plus puissants terminent la reconstruction bien avant les autres. Le temps de reconstruction était donc pénalisé par ces derniers. L'objectif de cette répartition est de distribuer les données en fonction de la vitesse relative de chaque processeur. Il est donc indépendant de la taille des données, ce qui permet de calculer les vitesses relatives sur un exemple de petite taille pour, ensuite, reconstruire des images de plus grande taille. Après quelques exécutions sur notre réseau de stations, les vitesses relatives de chaque processeur ont été définies. Nous présentons dans la section suivante le gain de performance généré par ce partitionnement adaptatif.

5.5.4 Expérimentations et discussion

Nous testons l'algorithme parallèle de Feldkamp à partir de plusieurs jeux d'essais. Chaqu'un d'eux permet de reconstruire une boule incluse dans un volume V de taille N^3 à partir de N *images de projection* Im_j de taille N^2. Le problème à résoudre a donc un coût séquentiel de l'ordre de N^4. Nous présentons les performances obtenues sur nos machines MIMD. Nous avons calculé, pour chaque expérimentation, l'accélération et l'efficacité pour évaluer les performances de nos implémentations.

Réseau de stations (sous PVM)

Nous avons choisi pour tester notre algorithme sur le réseau de stations de faire varier uniquement la taille du volume V à reconstruire à partir d'un même lot de 100 acquisitions. Nous avons implémenté la méthode de Feldkamp par une approche locale suivant différents modes [LPC94b]:

- le mode synchrone: le programme peut se découper en phases de calcul et en phases de communication. Nous utilisons des communications de réception bloquantes.
- le mode asynchrone: c'est la version optimisée du programme précédent. Nous mettons en œuvre le recouvrement des communications par des calculs à l'aide de communications non-bloquantes.
- le mode asynchrone adaptatif: nous effectuons un régulation des charges pour prendre en compte l'hétérogénéité des stations formant le réseau. Après plusieurs exécutions, les vitesses relatives des différentes stations ont été établies. Nous présentons les résultats issus du meilleur partitionnement.

Le tableau 8 présente les résultats d'exécutions pour les différentes implémentations. Nous les comparons avec les temps obtenus sur notre machine séquentielle qui est la machine la plus puissante de notre réseau de stations.

	PE	Feldkamp		
N		32	64	128
SUN4	1	437	1565	10606
Mode synchrone	5	199	628	3395
Mode asynchrone	5	191	541	3216
Mode asynchrone adaptatif	5	171	446	2593

TAB. 8 - *Temps d'exécution de Feldkamp sur le réseau de stations (sec)*

Les courbes de la figure 43 représentent l'efficacité des différentes implémentations en fonction de la taille du problème. Ces résultats montrent bien que la version asynchrone permet de gagner seulement 5% d'efficacité par rapport à la version synchrone. Cela peut s'expliquer par l'hétérogénéité des stations qui génèrent des phases d'inactivité pour les processeurs les plus "puissants".

Grâce au partitionnement adaptatif, on régule la charge des processeurs et on réduit de ce fait les temps d'inactivité. En effet, le gain d'efficacité par rapport à la version synchrone est ici de 20%. De plus, on peut remarquer que l'efficacité augmente en fonction de la taille du problème. En raison de la petite taille mémoire de notre machine de référence, celle-ci doit se servir de sa partition de "swap" pour résoudre les problèmes de grande taille. Ainsi, en multipliant les processeurs, on multiplie la taille de la mémoire.

Pour conclure, nous pouvons dire que les implémentations de la méthode de Feldkamp sont assez efficaces sur cette machine parallèle: nous obtenons en effet une efficacité maximale de 80%. Les implémentations sur ce type de machine sont donc très intéressantes, si on compare le ratio coût/performances par rapport à celui des autres machines parallèles.

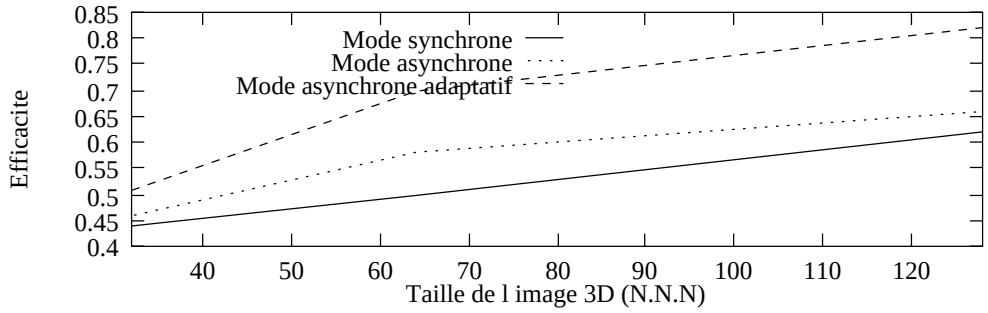


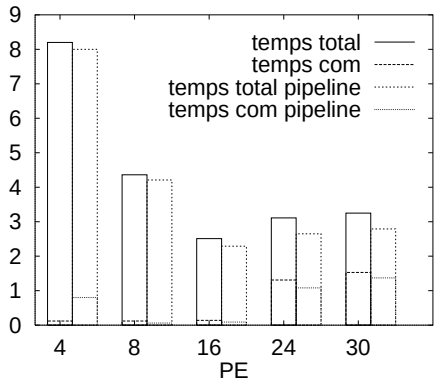
FIG. 43 - *Efficacité des implémentations de Feldkamp sur le réseau de stations*

Paragon

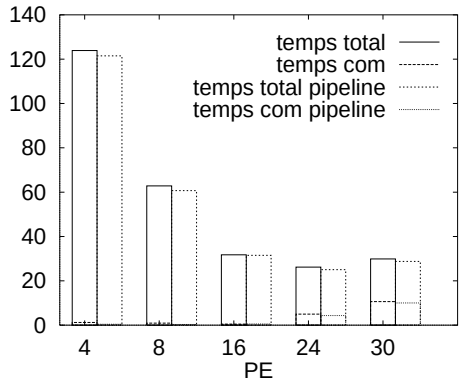
Nous avons testé nos implémentations en faisant varier la taille du problème (N^4) et le nombre de processeurs (PE). Nous avons implémenté une version en mode synchrone et une version en mode synchrone pipelinée. En nous basant sur différents travaux [Col94] et après une série de tests d'envoi d'images entre deux processeurs, nous nous sommes aperçus que l'envoi pipeliné d'une image est plus efficace. Le mode de communication de la Paragon étant le "Whormhole", l'envoi d'une image est donc effectué par paquets après l'envoi d'un en-tête. Pour accélérer la communication, nous divisons l'image par paquets de taille inférieure à la taille de l'en-tête. Pour chaque paquet, on effectue l'envoi d'un seul en-tête. L'envoi d'une image revient donc à l'envoi pipeliné d'un ensemble de paquets qui sont constitués par un certain nombre de lignes de l'image.

Les figures 44(a),44(b) et 44(c) présentent les temps d'exécution pour $N = 32$, $N = 64$ et $N = 128$. Sur chaque figure on indique le temps total et le temps de communication pour chaque expérimentation. On peut remarquer que globalement le temps de la version pipeline est similaire au temps sans pipeline. Cela peut s'expliquer par la part relativement faible des communications dans le temps total d'exécution. Ces figures montrent que lorsque la taille du problème n'est pas en adéquation avec le nombre de processeurs ($PE = 24$ et $PE = 30$), certains processeurs sont alors inactifs ce qui pénalise les performances. La figure 44(d) permet de comparer les performances obtenues avec le mode pipeline en fonction de la taille du problème et du nombre de processeurs. Si on calcule la charge de calcul par processeur que l'on peut exprimer par $\frac{N^4}{PE}$, on peut voir sur cette courbe que les performances se dégradent lorsque la charge par processeur est supérieure à 2^{16}.

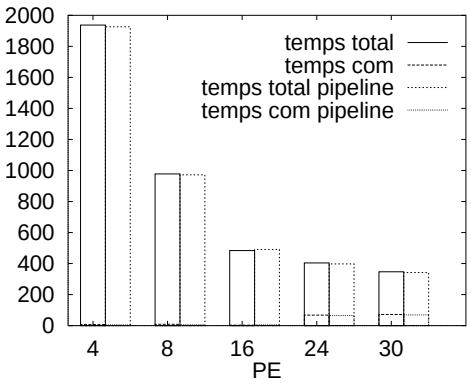
Nous avons calculé l'efficacité de notre implémentation en mode pipeline (figure 44(e)). Nous obtenons de bonnes efficacités pour un problème de taille 128^4. Cependant, pour des problèmes de plus petite taille, on doit adapter le nombre de processeurs à la taille du problème pour obtenir des efficacités satisfaisantes.



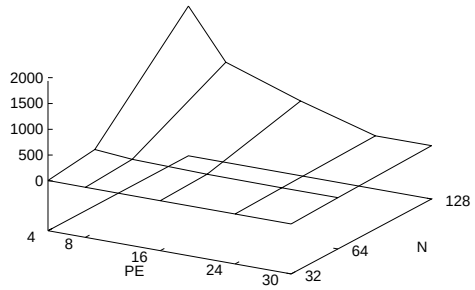
(a) Temps pour $N = 32$ (sec)



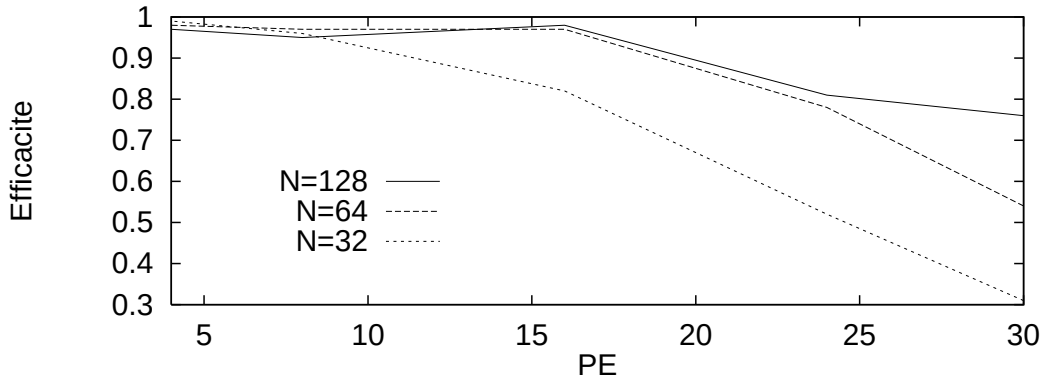
(b) Temps pour $N = 64$ (sec)



(c) Temps pour $N = 128$ (sec)



(d) Temps version pipeline



(e) Efficacité des implémentations

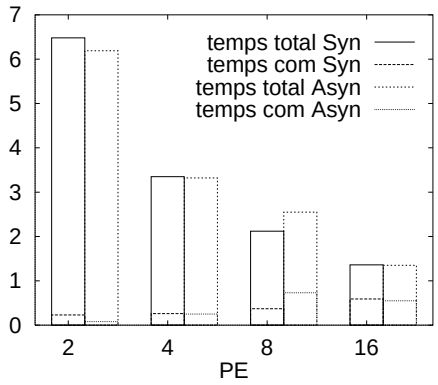
FIG. 44 - *Temps et efficacités des implémentations de Feldkamp sur la Paragon*

Ferme de Processeurs

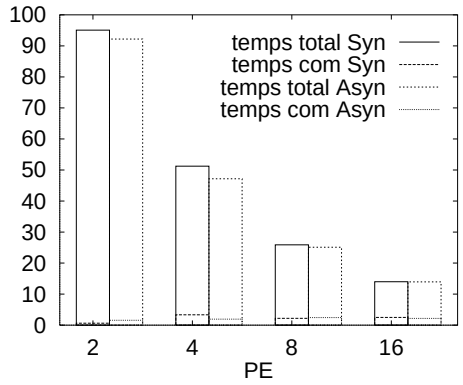
Sur cette machine composée de 16 processeurs homogènes, nous avons expérimenté la version de l'algorithme en mode synchrone et la version en mode asynchrone. Nous utilisons sur cette machine la librairie de communication PVM pour communiquer les données entre les processeurs. Nous avons effectué des tests de performances pour des problèmes de taille N^4 avec $N = 32, 64, 128$. Pour chaque problème de reconstruction, nous comparons les résultats de performances entre la version synchrone (Syn) et la version asynchrone (Asyn). Les figures 45(a),45(b) et 45(c) présentent les résultats de performances des implémentations pour chaque taille de problème. Pour chaque expérimentation, nous indiquons la part du temps de communication dans le temps total d'exécution.

Au vu de ces résultats nous pouvons constater que la version synchrone et la version asynchrone ont un comportement similaire. La version asynchrone permet de diminuer sensiblement les temps d'exécution. Cependant, la part des communications étant assez faible, le gain de performance n'est pas très important. Lorsque le nombre de processeurs est égal à seize, on peut remarquer que la part des communications est importante pour chaque expérimentation. Cela semble indiquer que les temps de communications ne sont pas linéaires entre une exécution sur huit ou sur seize processeurs. La figure 44(b) présente le comportement général de la version asynchrone sur cette machine. Elle montre que l'implémentation de cette méthode est bien adaptée à cette machine parallèle car les temps d'exécution diminuent lorsque l'on augmente le nombre de processeurs.

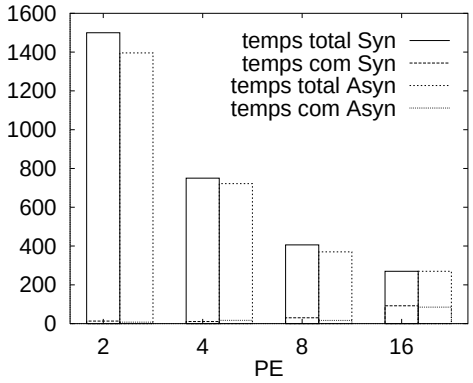
La figure 45(e) présente les efficacités des implémentations en mode synchrone et en mode asynchrone. Elle montre que l'efficacité croît en fonction de la taille du problème. Les implémentations en mode asynchrone ($N = 128a, N = 64a, N = 32a$) sont plus efficaces que les implémentations en mode synchrone ($N = 128, N = 64, N = 32$). Pour des problèmes de petite taille, il est souhaitable d'adapter le nombre de processeurs à la taille du problème pour obtenir une bonne efficacité comme le prouve les courbes pour $N = 32$.



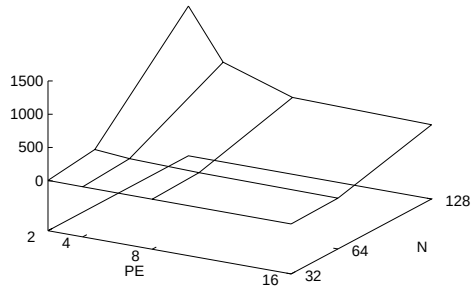
(a) Temps pour $N = 32$ (sec)



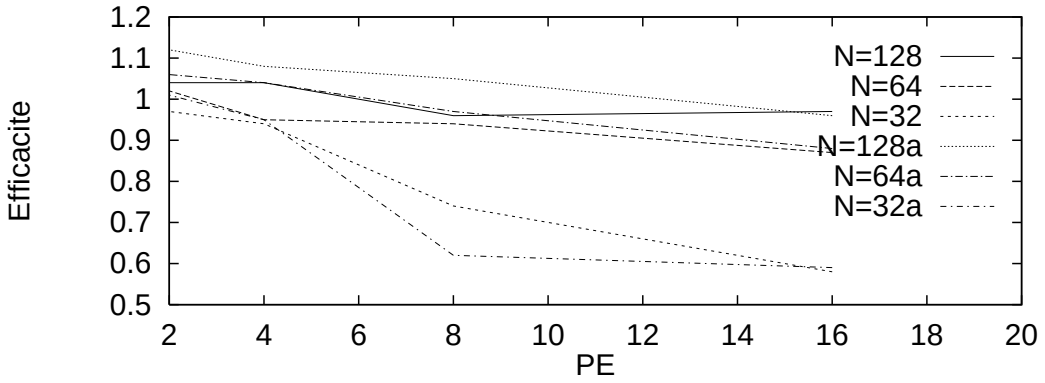
(b) Temps pour $N = 64$ (sec)



(c) Temps pour $N = 128$ (sec)



(d) Temps version asynchrone



(e) Efficacité des implémentations

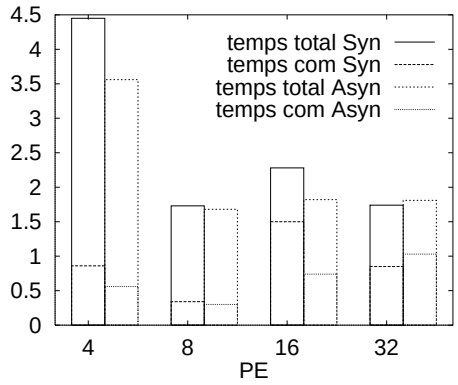
FIG. 45 - *Temps et efficacités des implémentations de Feldkamp sur la ferme de processeurs*

SP1

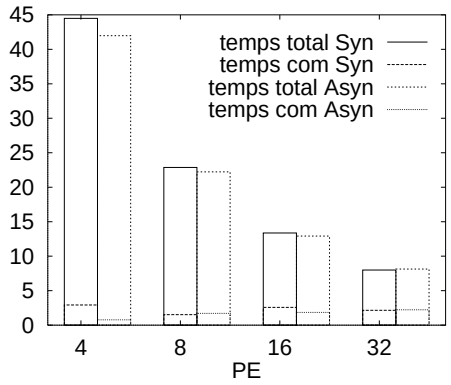
Cette machine est composée de 32 processeurs reliés par deux réseaux de communication: un réseau classique Ethernet et un réseau multi-étages. Nous avons effectué plusieurs tests de performances de nos programmes sur les deux réseaux, et nous nous sommes rendus compte que les performances obtenues sur le réseau multi-étages étaient bien supérieures à celles sur le réseau Ethernet. Nous présentons les performances obtenues sur le réseau multi-étages.

Les figures 46(a),46(b) et 46(c) présentent les temps d'exécution des implémentations parallèles en mode synchrone et en mode asynchrone. Comme sur la ferme de processeurs, on peut constater que les performances en mode asynchrone (Asyn) sont similaires à celles en mode synchrone (Syn). Le débit du réseau multi-étages étant de 20Mbytes/sec, le coût des communications est alors très faible. Ainsi pour des problèmes de taille supérieure à 64^4, le temps de communication représente un faible pourcentage du temps total d'exécution. Le gain de performance des implémentations en mode asynchrone est donc relativement faible. Cependant lorsque le nombre de processeurs est égal à trente-deux, les performances s'écroulent pour les problèmes de petite taille ce qui montre qu'il faut adapter le nombre de processeur à la taille du problème. La figure 46(d) présente les performances obtenues en mode asynchrone. On peut remarquer que lorsque le nombre de processeurs est supérieur à quatre, les courbes de performances pour un nombre de processeurs fixé sont similaires. Cela montre que les performances ne se détériorent pas lorsque l'on augmente la taille du problème ou lorsque l'on diminue le nombre de processeurs si la charge de calcul par processeur exprimée par $\frac{N^4}{PE}$ est inférieure à 2^{17}.

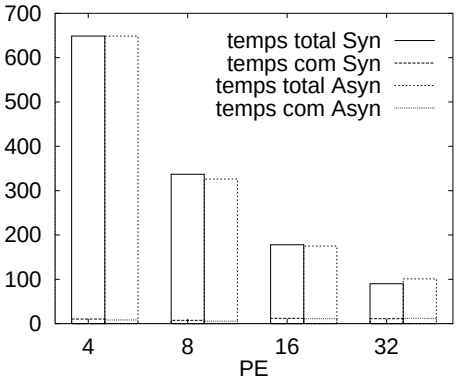
Nous avons calculé les efficacités de chaque expérimentation (figure 46(e)). Si on compare cette courbe avec la courbe d'efficacité de la ferme de processeurs (figure 45(e)), on peut remarquer que cette machine à un comportement semblable à la ferme de processeurs lorsque la taille du problème est supérieure à 64^4. Ici aussi on peut constater que l'efficacité augmente lorsque la taille du problème augmente. Pour conclure, on peut dire que les implémentations de nos méthodes sur cette machine sont très efficaces pour des problèmes de grande taille supérieure à 32^4.



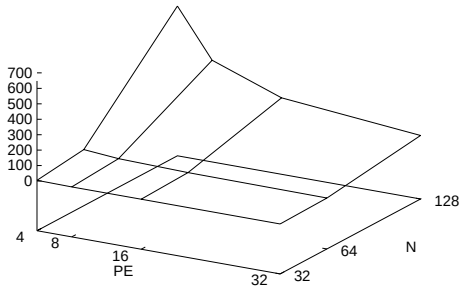
(a) Temps pour $N = 32$ (sec)



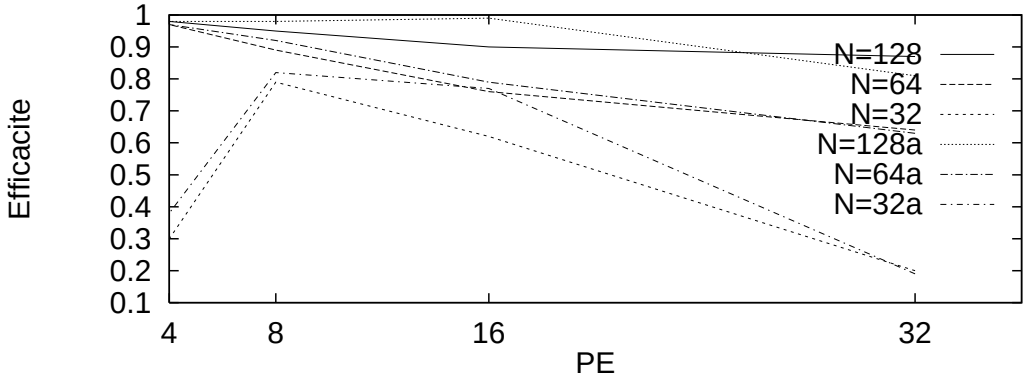
(b) Temps pour $N = 64$ (sec)



(c) Temps pour $N = 128$ (sec)



(d) Temps version asynchrone



(e) Efficacité des implémentations

FIG. 46 - *Temps et efficacités des implémentations de Feldkamp sur le SP1*

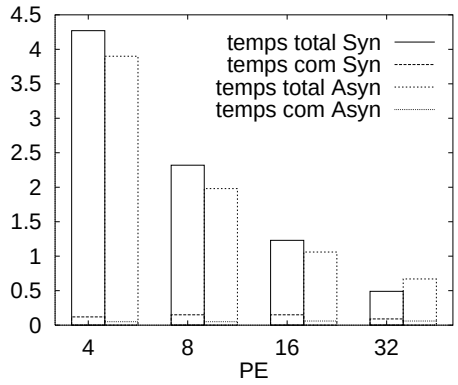
T3D

Sur cette machine constituée de 128 processeurs, nous avons testé notre algorithme en mode synchrone et en mode asynchrone pour des tailles de problèmes allant de 32^4 à 512^4. Pour chaque test, nous avons adapté le nombre de processeurs à la taille du problème. Les figures 47(a),47(b) et 47(c) présentent les temps d'exécution de notre algorithme en mode synchrone (Syn) et en mode asynchrone (Asyn). On peut constater que sur cette machine les communications sont quasi-inexistantes, puisque la part des communications n'apparaît même pas pour $N = 128$. Cependant la version asynchrone réduit les temps d'attente dus à la réception des images. Ainsi, nous obtenons un gain de performance relativement faible entre les deux versions synchrone et asynchrone de l'algorithme parallèle. La figure 47(d) présente les temps d'exécution de la version en mode asynchrone. Les performances se dégradent quand la charge de calcul de chaque processeur est supérieure à 2^{17}. Le tableau 9 présente des temps d'exécution pour des problèmes de grande taille ($N = 256^4$ et $N = 512^3$). En raison des coûts importants pour résoudre ces problèmes de grandes taille, on s'est limité à quelques expériences. On peut noter que l'on reconstruit une image 3D de taille 512^3 à partir de 256 acquisitions en moins d'une heure.

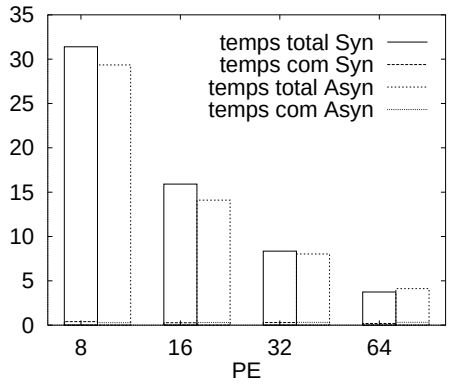
	$N = 256^4$				$N = 256.512^3$
Nombre de processeurs	16	32	64	128	128
Mode synchrone	4748	931	1235	329	3019
Mode asynchrone	5212	924	1231	304	

TAB. 9 - *Temps d'exécution sur le T3D pour des problèmes de grande taille (sec)*

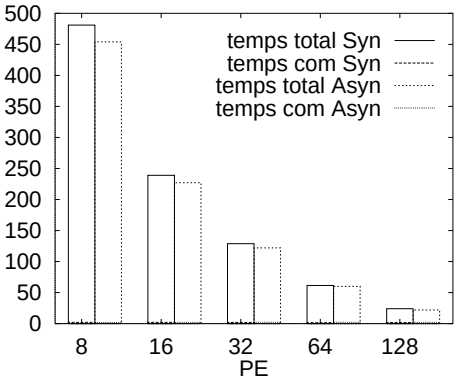
Nous calculons les efficacités seulement pour les problèmes de petite taille. En effet, il n'a pas été possible de résoudre les problèmes de grande taille sur un seul processeur pour calculer les efficacités, en raison de l'espace mémoire nécessaire à ces problèmes. Si on examine les courbes de d'efficacité (figure 47(e)), on peut noter que l'efficacité augmente avec la taille du problème. Nous soulignons aussi que pour $N = 128$, nous obtenons des efficacités quasiment toujours supérieures à 1. Nous atteignons les limites du modèle de calcul de l'accélération et de l'efficacité que nous avons défini dans le chapitre 3. Cela indique aussi que pour des problèmes de grande taille une mesure d'efficacité n'est pas nécessaire, car ce sont des problèmes qui ne peuvent pas être résolus en séquentiel dans des temps acceptables.



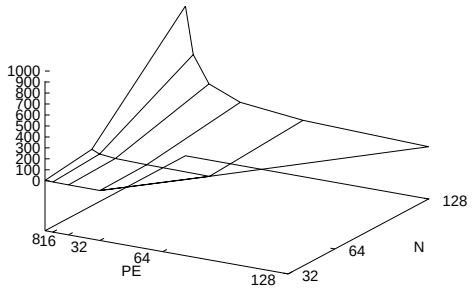
(a) Temps pour $N = 32$ (sec)



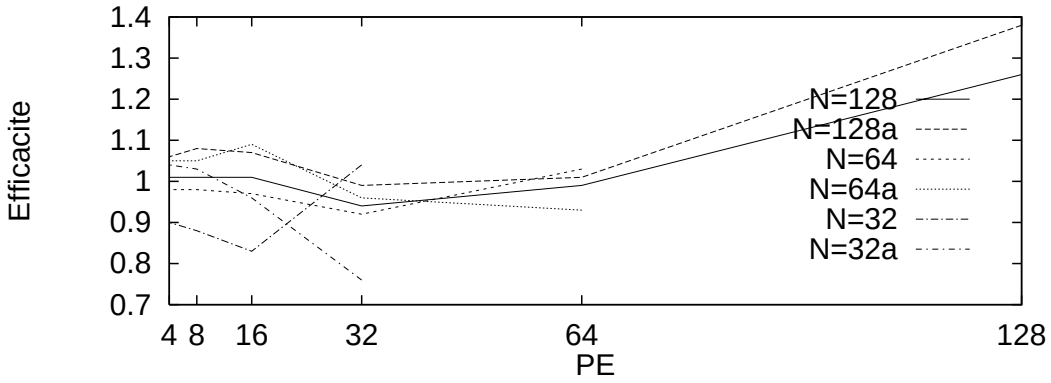
(b) Temps pour $N = 64$ (sec)



(c) Temps pour $N = 128$ (sec)



(d) Temps version asynchrone



(e) Efficacité des implémentations

FIG. 47 - *Temps et efficacités des implémentations de Feldkamp sur le T3D*

5.6 Reconstruction d'images 3D

5.6.1 Mode de calcul des projections

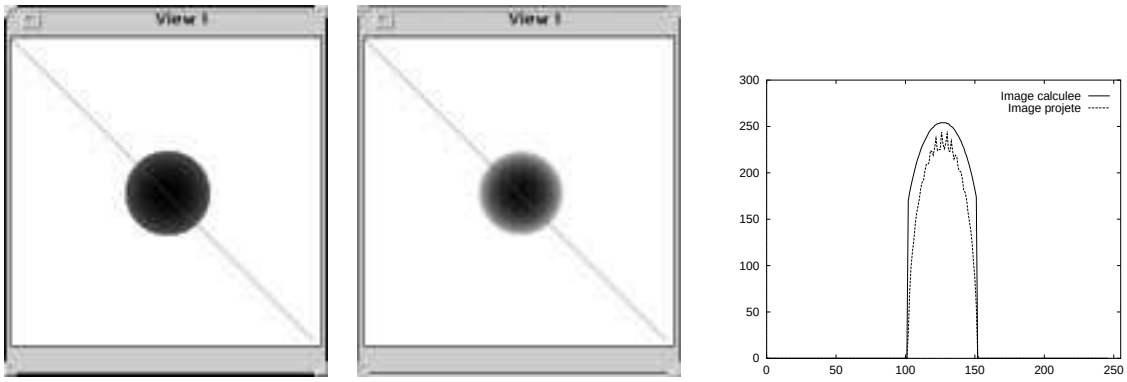
Nous avons testé nos algorithmes de reconstruction sur différents jeux de données. On reconstruit une sphère de densité uniforme pour les deux premiers jeux de données de tailles 32^3 et 64^3, et un ensemble de sphères imbriquées pour le troisième jeux de données. Ces jeux de données ont été définis à l'aide de notre logiciel Tomotool (annexe A). L'ensemble des reconstruction d'images 3D utilise les paramètres d'acquisition du Morphomètre que nous décrivons dans le chapitre 7.

Pour reconstruire ces images 3D, on doit d'abord définir les acquisitions. Pour calculer les acquisitions ou *images de projection*, il existe deux méthodes:

- soit on définit la densité en chaque point de projection à l'aide de formulation définissant la projection d'une sphère, et la densité en chaque point de cette sphère. On élimine ainsi les erreurs dues à la projection du cube original sur chaque *image de projection*. On obtient des projections simulées (figure 48(a)).

- soit on crée un cube original et on le projette pour créer les *images de projection*: on obtient des projections calculées (figure 48(b)).

La figure 48(c) présente les profils d'une projection calculée et d'une projection simulée. On peut constater que la projection simulée permet de reconstruire une image 3D dans le cas idéal où les projections sont parfaites, c'est à dire sans les erreurs générées par l'étape de projection.



(a) Image simulée (b) Image calculée (c) Profils des images

FIG. 48 - *Image de la projection d'une sphère*

5.6.2 Définition des écarts

Pour mesurer la qualité des reconstructions d'une image reconstruite f_r par rapport à l'image initiale f, on peut définir différents critères d'écart:

- L'écart quadratique moyen défini par:

$$q(f, f_r) = \frac{\|f - f_r\|_2}{N^3} \tag{5.121}$$

- L'écart quadratique moyen normalisé par la variance σ_f^2 de l'objet original:

$$qn(f, f_r) = \frac{1}{N^3} \frac{\|f - f_r\|_2}{\sigma_f^2} = \frac{q(f, f_r)}{\sigma_f^2} \tag{5.122}$$

- Le coefficient de corrélation:

$$co(f, f_r) = \frac{1}{N^3} \frac{(f - \overline{f}).(f_r - \overline{f}_r)}{\sigma_f \sigma_{f_r}} \tag{5.123}$$

où $\overline{f}$, $\overline{f}_r$ σ_f et σ_{f_r} sont les moyennes et les écarts type de f et f_r.

On mesure pour chacun de nos jeux d'essais, le coefficient de corrélation lorsque l'on reconstruit l'image 3D à partir soit des projections calculées, soit des projections simulées.

5.6.3 Reconstruction de sphères

Nous comparons les résultats des reconstructions suivant le filtre utilisé (rampe, Shepp et Logan, fenêtre cosinus avec différentes valeurs du coefficient d'apodisation α) et suivant le mode de calcul des projections (projection calculée, projection simulée). Pour chaque volume reconstruit, nous présentons une coupe du volume initial, une coupe du volume reconstruit avec un filtre rampe à partir de projections simulées et à partir de projections calculées. Nous montrons la qualité de la reconstruction suivant le filtre utilisé en visualisant les profils des coupes des images reconstruites. De plus, nous calculons les coefficients de corrélation entre l'image originale et l'image reconstruite qui nous permet d'évaluer un critère de qualité.

Sphère comprise dans un volume de taille 32^3

Nous reconstruisons une sphère uniforme qui est incluse dans un volume de 32^3 voxels. Ses paramètres géométriques sont les suivants:

- centre de la sphère:(0,0,0),
- rayon de la sphère: 10 * taille d'un voxel,
- densité en chaque point de la sphère: 100, le reste du volume étant à 0.

Nous exposons les résultats de la reconstruction de cette image suivant le filtre utilisé. Les figures 49(c) et 49(b) présentent des coupes des boules reconstruites en utilisant un filtre rampe à partir, soit des projections de l'image initiale 49(a), soit des projections simulées. Pour chaque type de projections (calculées, simulées), nous comparons les profils des images reconstruites (figures 50(a) et 50(b)). Les filtres sont référencés de la façon suivante: rampe pour le filtre rampe; shepp pour le filtre de Shepp et Logan; h0,h1,h2 et h3 pour le filtres utilisant une fenêtre cosinus avec pour coefficient d'apodisation $\alpha = 0, 1, 2, 3$. Le meilleur filtre pour la reconstruction de cette sphère est le filtre rampe pour chaque type de projection. Nous retrouvons ces résultats, si on évalue les coefficients de corrélation (tableau 10).

Ecart	Filtre	Rampe	Shepp	$\alpha = 0$	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$
$co(f, f_r)$	Projection simulée	0.968	0.960	0.968	0.962	0.955	0.948
	Projection calculée	0.959	0.954	0.959	0.955	0.942	0.942

TAB. 10 - *Coefficients de corrélation des reconstructions*

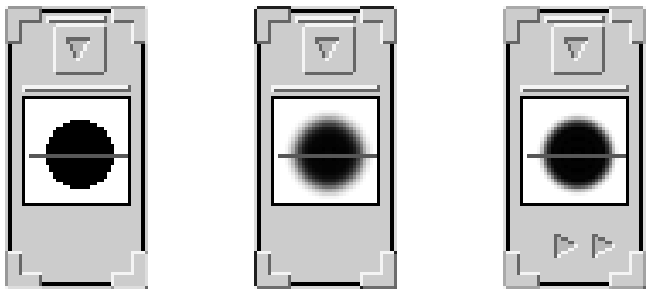
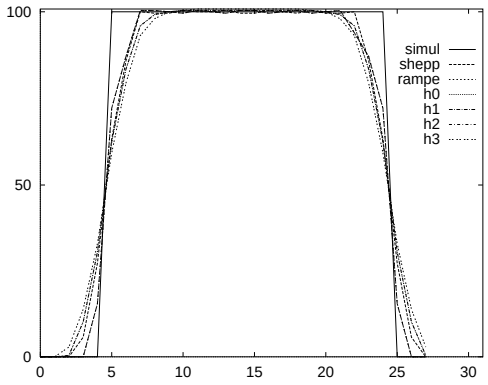
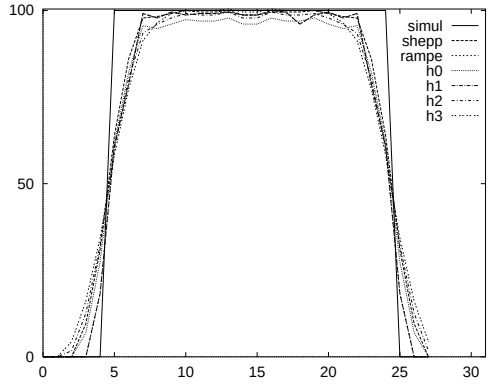


FIG. 49 - *Coupe médiane de la sphère initiale (a) et des sphères reconstruites à partir, soit des projections simulées (b), soit des projections calculées (c)*



(a) Profils des reconstructions à partir de projections simulées



(b) Profils des reconstructions à partir de projections calculées

FIG. 50 - *Profil des reconstructions par Feldkamp de la sphère* 32^3

Sphère comprise dans un volume de taille 64^3

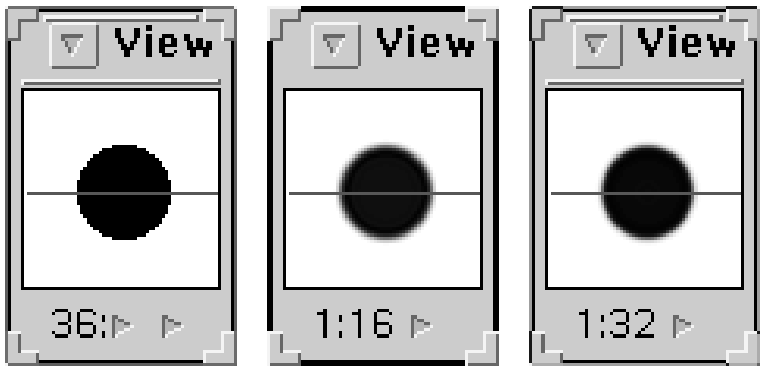
Nous reconstruisons une sphère uniforme excentrée qui est incluse dans un volume de 64^3 voxels. Ses paramètres géométriques sont les suivants:

- centre de la sphère:(1,-10,-10),
- rayon de la sphère: 15 * taille d'un voxel,
- densité en chaque point de la sphère: 150, le reste du volume étant à 0.

Les figures 51(b) et 51(c) présentent des coupes des boules reconstruites en utilisant un filtre rampe à partir soit des projections de l'image initiale 51(a), soit des projections simulées. Les figures 52(b) et 52(a) exposent les profils des reconstructions. On peut remarquer que l'on obtient dans les deux cas une meilleure reconstruction avec le filtre rampe car nous utilisons des données non bruitées. En se référant aux profils des projections calculées et simulées, il est intéressant de noter que les profils des sphères reconstruites leur sont similaires. Si on compare les coefficients de corrélation suivant le mode de calcul des projections (tableau 11), on s'apercoit que la reconstruction à partir de projections calculées obtient en moyenne des meilleurs coefficients de corrélation que l'autre mode de calcul des projections.

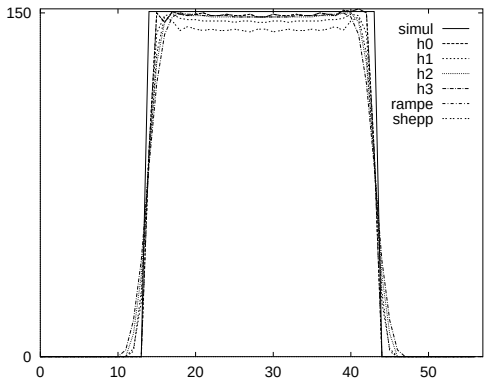
Ecart	Filtre	Rampe	Shepp	$\alpha = 0$	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$
$co(f, f_r)$	Projection simulée	0.913	0.9113	0.913	0.911	0.910	0.908
	Projection calculée	0.976	0.972	0.976	0.972	0.968	0.964

TAB. 11 - *Coefficients de corrélation des reconstructions*

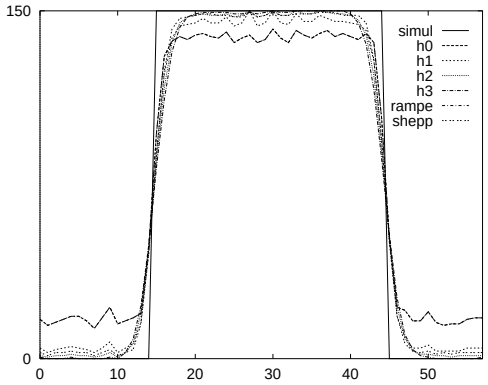


(a) (b) (c)

FIG. 51 - *Coupe de la sphère initiale (a) et des sphères reconstruites à partir, soit des projections simulées (b), soit des projections calculées (c)*



(a) Profils des reconstructions à partir de projections simulées



(b) Profils des reconstructions à partir de projections calculées

FIG. 52 - *Profil des reconstructions par Feldkamp de la sphère* 64^3

Sphères imbriquées comprises dans un volume de taille 128^3

Nous avons choisi de reconstruire un ensemble de sphères imbriquées pour simuler un objet complexe. Le tableau 12 présente les caractéristiques de chaque sphère.

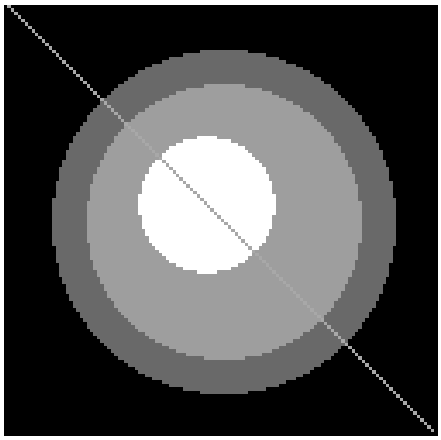
Sphère	Centre	Rayon*taille-voxel	Densité
sphère1	(0,0,0)	50	100
sphère2	(0,0,0)	40	150
sphère3	(15,15,15)	10	200
sphère4	(-5,-5,-5)	20	240

TAB. 12 - *Description des sphères imbriquées*

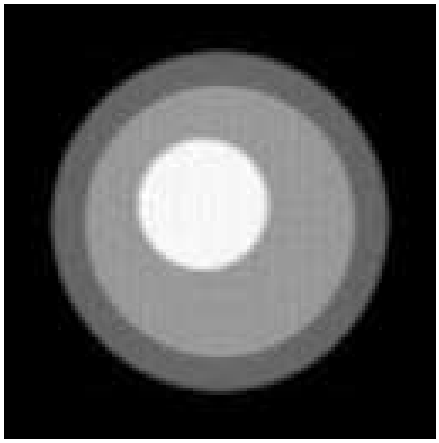
Nous reconstruisons ces sphères à partir de données calculées. Le filtre de Shepp et Logan obtient les meilleurs résultats de reconstruction au vu des coefficients de corrélation du tableau 13. Nous comparons les profils des reconstructions suivant deux coupes des images 3D. On peut noter que la nature du filtre ne modifie pas sensiblement la reconstruction dans le cas de données non bruitées (figures 53(e) et 53(f)). Chacune de ces coupes permet de visualiser les boules imbriquées. La première coupe qui représente le plan XY pour Z=0, permet de visualiser les sphères 1,2 et 3 (figures 53(a) et 53(b)). La deuxième coupe qui représente le plan XY pour Z=-13, permet de visualiser les sphères 1,2 et 4 (figures 53(d) et 53(f)).

Ecart	Rampe	Shepp	$\alpha = 0$	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$
$co(f, f_r)$	0.9924	0.9927	0.9924	0.9929	0.9919	0.9904

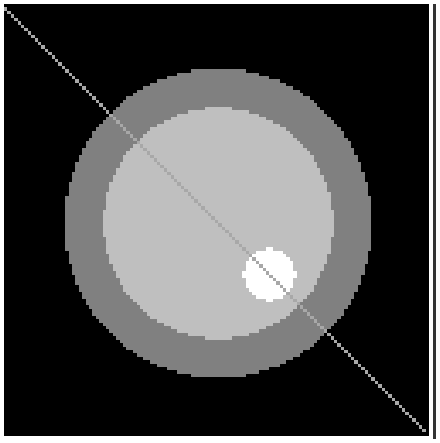
TAB. 13 - *Coefficients de corrélation des reconstructions*



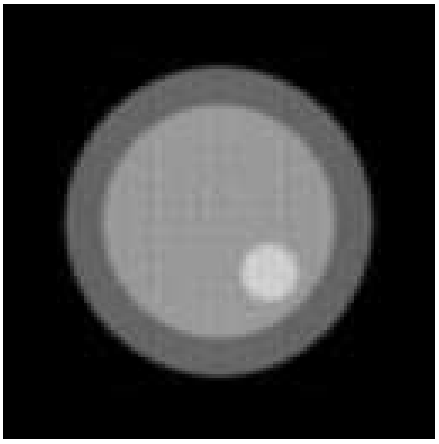
(a) Image initiale(Z=0)



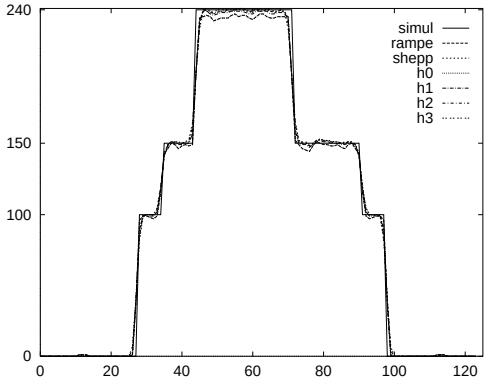
(b) Image reconstruite (Z=0)



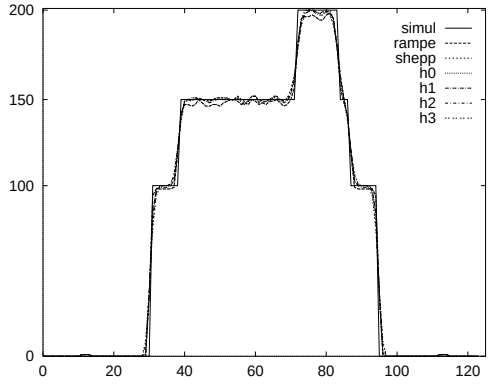
(c) Image initiale(Z=-13)



(d) Image reconstruite (Z=-13)



(e) Profil pour Z=0



(f) Profil pour Z=-13

FIG. 53 - *Coupes Z=0 et Z=-13 des sphéres imbriquées reconstruites par Feldkamp et profils des reconstructions suivant le filtre utilisé*

5.7 Conclusion

Nous avons présenté, dans ce chapitre, la parallélisation d'une méthode analytique: la méthode de Feldkamp. Nous avons proposé plusieurs parallélisations de cette méthode suivant l'architecture de la machine cible. Nous avons montré que pour une machine SIMD, il était préférable d'utiliser un opérateur de rétroprojection parallélisé suivant une approche globale. Alors que sur les machines MIMD, nous utilisons un opérateur de rétroprojection parallélisé suivant une approche locale. Nous avons mis en œuvre des techniques de recouvrement des communications par des calculs et de partitionnement adaptatif pour améliorer les performances des implémentations de cette méthode sur les machines MIMD.

Nous obtenons de bonnes performances sur les machines MIMD et une bonne efficacité de nos implémentations qui avoisine en moyenne les 90%. Sur machine SIMD, nous obtenons des performances moyennes dues à la faible capacité mémoire des processseurs. De plus, la géométrie d'acquisition conique que nous utilisons, génère sur ce type de machine de nombreuses phases d'inactivé ce qui pénalise les performances.

En conclusion, à partir des nombreuses expérimentations que nous avons effectuées, nous avons constaté qu'il faut adapter le nombre de processeurs à la taille du problème pour obtenir de bonnes performances et des implémentations efficaces [LPC95a].

Au niveau de la reconstruction d'images 3D, nous obtenons de bons résultats au regard du critère de la qualité que nous avons défini. Le choix d'un bon filtre nécessite souvent la connaissance du bruit à filtrer. Dans notre cas, les données n'étant pas bruitées, nous retrouvons comme meilleur filtre soit le filtre rampe, soit le filtre de Shepp et Logan.

Nous avons calculé l'accélération relative de nos implémentations sur les machines parallèles, que nous utilisons, par rapport au temps sur notre machine de référence. La figure 54 représente les résultats obtenus pour la reconstruction d'une image 3D (128^3) à partir de 128 projections 2D (128^2). L'accélération relative est exprimée suivant une échelle logarithmique pour pouvoir comparer les performances de chaque machine. On obtient la plus forte accélération sur le Cray T3D, ce qui était prévisible. On peut noté que le réseau de stations obtient une meilleure accélération que notre machine SIMD: la Maspar. Cela tend à prouver que les machines MIMD sont mieux adaptées aux problèmes de reconstruction d'images 3D.

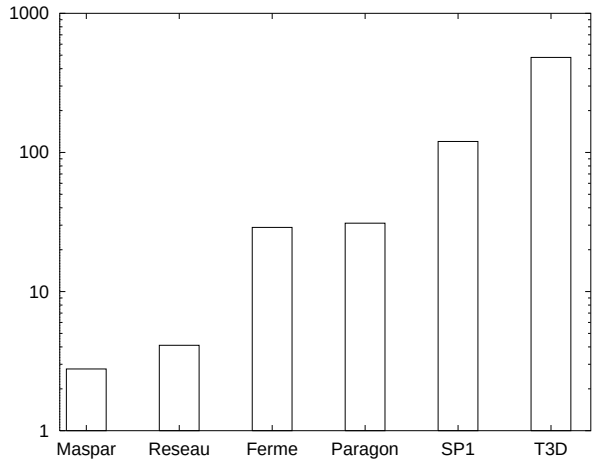


FIG. 54 - *Accélération relative des implémentations de Feldkamp*

Chapitre 6

Parallélisation des méthodes algébriques

6.1 Introduction

Nous présenterons dans ce chapitre la parallélisation de méthodes algébriques: la méthode ART par blocs et la méthode SIRT. Nous expliciterons leurs formulations dans le cas d'une géométrie d'acquisition conique. Nous montrerons que la méthode SIRT peut être exprimée comme une méthode par blocs. Cette structure par blocs, nous permet d'utiliser les opérateurs de projection et de rétroprojection développés dans le chapitre 4.

La parallélisation de ces méthodes sera alors basée sur une version parallèle de ces opérateurs. Cependant, nous montrerons que nous utilisons une approche différente pour paralléliser chaque méthode. En effet, la méthode ART utilise des opérateurs de projection et de rétroprojection parallélisés suivant une approche globale, tandis que la méthode SIRT utilise des opérateurs parallélisés suivant une approche locale.

Après avoir présenté une première version parallèle de ces méthodes, nous nous sommes intéressés à une version optimisée de ces algorithmes parallèles.
L'algorithme parallèle de la méthode ART utilise des opérations de communication globale comme la réduction ou la diffusion. Sa version optimisée met en œuvre des communications globales plus efficaces basées sur un schéma de communication en arbre binaire.

Pour optimiser l'algorithme parallèle de la méthode SIRT, nous utiliserons des techniques de recouvrement des communications par des calculs.

Nous présenterons les résultats de performances que nous avons obtenues sur trois machines parallèles: la ferme de processeurs, le SP1 et le T3D. Nous avons choisi ces machines parallèles, car elles utilisent la même bibliothèque parallèle de communications: PVM. De plus, les implémentations de méthodes de reconstruction sur ces machines semblent être très efficaces, d'après les résultats du chapitre 5.

Pour finir, nous montrerons les résultats d'images 3D reconstruites à partir de données simulées. Nous utiliserons les mêmes jeux de données que ceux présentés dans le chapitre 5. Pour chaque méthode, nous évaluerons la qualité de la reconstruction en fonction des critères de distance. Nous nous intéresserons à l'évolution de ces critères en fonction du nombre d'itérations nécessaires pour reconstruire l'image 3D et lorsque l'on modifie le paramètre de relaxation pour accélérer la convergence.

6.2 Présentation des méthodes algébriques

Nous avons choisi les méthodes algébriques qui résolvent le système $Rf = p$ en le subdivisant en un ensemble équivalent de sous-systèmes $R_j f = P_j$ pour $j = 1...m$. Ces méthodes par blocs reconstruisent la fonction f en traitant, à chaque cycle, une ou plusieurs projections coniques au lieu de traiter une ou plusieurs lignes pour les méthodes traditionnelles. On peut ainsi résoudre des systèmes de grande taille en les subdivisant en sous-systèmes.

De plus, ces méthodes utilisent les opérateurs de base que nous avons développés dans le chapitre 4. Leur implémentation nécessite d'adapter ces opérateurs discrets et permet ainsi, de ne pas réécrire entièrement leurs algorithmes. Leur schéma itératif met en œuvre les matrices de projection R_j et les matrices Ω_j ou W, dont nous explicitons les constructions dans le paragraphe suivant.

6.2.1 Expression des matrices R_j

Pour implémenter les méthodes algébriques par blocs, on doit tout d'abord connaître la structure de chaque matrice R_j. Nous nous intéressons au calcul de ces matrices dans le cadre de notre étude.

Le faisceau de la projection conique est discrétisé sur un ensemble de droites reliant la source à chaque point discrétisé du plan de projection. Chaque ligne l de la matrice R_j représente donc les coefficients des voxels v_i qui contribuent à la valeur du point l . L'expression de la projection au point l est égale à:

$$p_l = \sum_{i=1}^{N^3} R_{j(li)} f_i \tag{6.124}$$

Si les coordonnées des projections des voxels v_i correspondaient aux points échantillonnés, les matrices de projection R_j seraient constituées de 0 ou de 1. Cette hypothèse n'étant pas vérifiée le problème revient alors à faire correspondre les v_i avec les p_l.

Une première solution est d'affecter la valeur du voxel v_i au point l de coordonnées $(p\triangle_p, q\triangle_p)$ qui minimise la distance $d(i,l)$ définie par:

$$d(i,l) = \sqrt{(u - p\triangle_p)^2 + (v - q\triangle_p)^2} \tag{6.125}$$

Une deuxième solution est de répartir la valeur f_i par une interpolation bilinéaire sur les quatre points de coordonnées, $(p\triangle_p, q\triangle_p)$ vérifiant:

$$|u - p\triangle_p| \leq 1 \tag{6.126}$$

$$|v - q\triangle_p| \leq 1 \tag{6.127}$$

Nous choisissons cette dernière solution car elle se rapproche plus du modèle physique de la projection.

La valeur des coefficients $R_{j(li)}$ est égale au produit des coefficients de l'interpolation bilinéaire.

$$R_{j(li)} = \begin{cases} (1-\lambda)(1-\mu) & \text{si } \lambda = |u - p\triangle_p| < 1 \quad \text{et } \mu = |v - q\triangle_p| < 1 \\ 0 & \text{sinon} \end{cases} \tag{6.128}$$

Si on examine l'algorithme itératif des méthodes par blocs, donné par l'équation 6.129, on peut remarquer que la matrice R_j intervient pour le calcul de l'image

projetée $R_j f_d$ et pour le calcul de la contribution à f_d, qui est fonction de la différence entre l'image mesurée et l'image projetée.

$$f_d^{(k+1)} = f_d^{(k)} + \lambda_k \, {}^t R_j \Omega_j (P_j - R_j f_d^{(k)}) \tag{6.129}$$

On exprime la valeur de l'image projetée $Imp_j = R_j f_d^{(k)}$ en chaque point l:

$$Imp_j(l) \quad = \quad \sum_{i=1}^{N^3} R_{j(li)} f_{d,(i)}^{(k)} \tag{6.130}$$

L'opération de projection $R_j f_d$ est donc implémentée en utilisant l'opération élémentaire de projection $\stackrel{\mathcal{P}}{\rightarrow}$.

De même, la contribution $f'_d(i)$ de l'*image de différence* ($Diff_j = \Omega_j(P_j - R_j f_d^{(k)})$) s'exprime par:

$$f'_d(i) = \sum_{l=1}^{M^2} {}^t R_{j(li)} Diff_j(l) \tag{6.131}$$

Le calcul de cette contribution revient à rétroprojeter chaque image $Diff_j$ à l'aide d'un opérateur de rétroprojection discret basé sur l'opération élémentaire $\stackrel{\mathcal{R}}{\leftarrow}$.

Nous utilisons donc les opérateurs discrets pour implémenter les méthodes par blocs.

6.2.2 Critères d'arrêt

Ces méthodes par blocs étant des méthodes itératives, elles sont stoppées lorsque l'écart calculé entre deux cycles de la reconstruction est inférieur à un seuil γ. Un cycle représente la résolution de tous les sous-systèmes $R_j f_d = P_j$ pour $j = 1...m$. Cet écart permet de mesurer la qualité de la reconstruction.
Différents écarts ont été proposés dans la littérature: l'écart quadratique moyen, l'écart quadratique moyen normalisé par la variance et le coefficient de corrélation. Nous utilisons le premier écart pour nos implémentations qui est défini par:

$$q(n) = \frac{\| f_d^{(n)} - f_d^{(n-m)} \|_2}{N^3} \tag{6.132}$$

Herman a montré que l'algorithme ART converge en 5-6 cycles, on se limitera alors à un petit nombre d'itérations pour tester nos algorithmes.

6.2.3 Méthode ART par blocs

A partir du schéma itératif (voir l'équation 6.133), nous explicitons l'algorithme de la méthode ART par blocs.

$$\begin{aligned} f^{(k+1)} &= f^{(k)} + \lambda_k \, {}^tR_j \Omega_j (P_j - R_j f^{(k)}) \\ \Omega_j &= [\mathrm{diag}(R_j \, {}^tR_j)]^{-1} \end{aligned} \tag{6.133}$$

L'algorithme déduit de ce schéma itératif peut se décomposer en trois étapes pour chaque itération.

- calcul de l'*image projetée* $Imp_j = R_j f^{(k)}$ effectué par l'opérateur discret de projection,
- calcul de l'*image de différence* $Diff_j = \Omega_j (P_j - Imp_j)$ nécessitant la connaissance de chaque matrice Ω_j,
- rétroprojection de l'*image de différence* $Diff_j$ effectuée par l'opérateur discret de rétroprojection.

Il est nécessaire, pour limiter les calculs, d'évaluer les matrices Ω_j lors d'une phase préalable à l'exécution de l'algorithme.

Evaluation des matrices Ω_j

On peut remarquer que le calcul des matrices Ω_j ne dépend que des matrices R_j. On exprime chaque terme d'indice (p,q) de cette matrice de taille $M^2 \times M^2$:

$$(R_j \, {}^tR_j)(p,q) = \sum_{i=1}^{N^3} R_{j(p,i)} R_{j(q,i)} \tag{6.134}$$

$$\begin{aligned} \Rightarrow \Omega_j(p,q) &= [\mathrm{diag}(R_j \, {}^tR_j)]^{-1} I \\ &= \begin{cases} \frac{1}{\sum_{i=1}^{N^3} R^2_{j(pi)}} & \mathrm{si}\, p = q \\ 0 & \mathrm{sinon} \end{cases} \end{aligned} \tag{6.135}$$

On extrait les éléments non nuls de la matrice Ω_j qui forment une image de poids W_j, de taille M^2, définie par:

$$W_{j(p,q)} = \Omega_j(p + Mq, p + Mq) = \frac{1}{\sum_{i=1}^{N^3} R^2_{j(p+Mq,i)}} \tag{6.136}$$

Le calcul des matrices W_j revient à effectuer une opération de projection discrète où les f_i sont égaux à 1 et où les coefficients $R_{j(li)}$ sont élevés au carré. Par analogie aux opérations élémentaires de projection $\xrightarrow{\mathcal{P}}$ et de rétroprojection $\xleftarrow{\mathcal{R}}$, on définit une opération élémentaire de pondération $\xrightarrow{\mathcal{W}}$ qui consiste à affecter au point l de la matrice W_j, le coefficient $R^2_{j(l,i)}$ correspondant au voxel v_i.

Algorithme séquentiel

Nous présentons les deux algorithmes séquentiels développés pour implémenter cette méthode. Le premier est l'algorithme de calcul des *images de poids* W_j. Le second présente l'algorithme itératif d'un cycle. Chaque cycle correspond à la rétroprojection de la différence entre les m images projetées et les m images mesurées.
Pour décrire ces algorithmes, nous introduisons les notations suivantes:

- Im_j est l'*image mesurée*, composée de M^2 pixels im_{jl},
- Imp_j est l'*image projetée*, définie à partir de l'équation 6.130, et composée de M^2 pixels imp_{jl},
- W_j est l'*image de poids* associée à chaque Im_j, composée de M^2 pixels w_{jl},
- $Diff_j$ est l'*image de différence* définie par la différence entre Im_j et Imp_j, et composée de M^2 pixels $diff_{jl}$.

Algorithme 14 *Calcul des* W_j

lire(V)
Pour $j = 1$ à m
  créer(W_j)
  Pour $i = 1$ à N^3
    $v_i \xrightarrow{\mathcal{W}} w_{ja}, w_{jb}, w_{jc}, w_{jd}$
  écrire(W_j)

Si on évalue le coût théorique du calcul des W_j, on s'aperçoit que celui-ci est égal en terme d'opérateurs élémentaires $\xrightarrow{\mathcal{W}}$ au coût d'une projection ou d'une rétroprojection discrète; soit $m.N^3.T(\xrightarrow{\mathcal{W}})$.
L'algorithme ART par blocs (algorithme 15) représente m itérations du schéma itératif, décrit par l'équation 6.133, qui définissent un cycle.

Algorithme 15 *ART par blocs*

Pour $j = 1$ à m
  créer(Imp_j)
  lire(Im_j)
  Pour $i = 1$ à N^3
    $v_i \xrightarrow{\mathcal{P}} imp_{ja}, imp_{jb}, imp_{jc}, imp_{jd}$
  Pour $l = 1$ à M^2
    $diff_{jl} \leftarrow \lambda_k w_{jl}(im_{jl} - imp_{jl})$
  Pour $i = 1$ à N^3
    $v_i \xleftarrow{\mathcal{R}} diff_{ja}, diff_{jb}, diff_{jc}, diff_{jd}$

Le coût d'un cycle en terme d'opérateurs élémentaires ($\xrightarrow{\mathcal{P}}, \xleftarrow{\mathcal{R}}$) est égal à:

$$T_{cycle} = m.N^3.T(\xrightarrow{\mathcal{P}}) + m.M^2 + m.N^3.T(\xleftarrow{\mathcal{R}}) \tag{6.137}$$

Cela permet d'évaluer le calcul du coût total de l'algorithme en fonction de N_c le nombre de cycles:

$$T_{ART} = m.N^3.T(\xrightarrow{\mathcal{W}}) + N_c(m.N^3.T(\xrightarrow{\mathcal{P}}) + m.M^2 + m.N^3.T(\xleftarrow{\mathcal{R}})) \tag{6.138}$$

Globalement cet algorithme a un coût en $N_c \times 2m.N^3$ que l'on peut comparer avec l'algorithme de Feldkamp, qui lui a un coût en $m.N^3$.

6.2.4 Méthode SIRT

Nous présentons une autre méthode qui calcule la contribution de toutes les *images de différence* en même temps. Son schéma itératif est le suivant:

$$\begin{aligned} f^{(k+1)} &= f^{(k)} + \lambda_k W \sum_{j=1}^{m} {}^t R_j (P_j - R_j f^{(k)}) \qquad (6.139) \\ W &= [\sum_{j=1}^{m} \mathrm{tr}(R_j \, {}^t R_j)]^{-1} \end{aligned}$$

Cette méthode est implémentée en utilisant les mêmes opérateurs discrets que la méthode précédente. Cependant ici, un cycle correspond à une itération de la méthode SIRT. L'algorithme séquentiel, basé sur ce schéma itératif, comprend trois étapes distinctes:

- Calcul des *images projetées* $Imp_j = R_j f^{(k)}$ pour $j = 1...m$,
- Calcul des *images de différences* $Diff_j = W(Im_j - Imp_j)$ pour $j = 1...m$,
- rétroprojection des *images de différence* $Diff_j$.

Nous évaluons la valeur de W avant d'exécuter l'algorithme itératif.

Calcul de W

W est égal à la somme des traces des matrices $R_j \, {}^t R_j$. La trace d'une matrice A est définie comme la somme des éléments diagonaux: $\mathrm{tr}(A) = \sum a_{ii}$. Nous avons vu que les éléments diagonaux des matrices $R_j \, {}^t R_j$ définissent une matrice de poids W_j. La trace de chaque matrice $R_j \, {}^t R_j$, notée Wt_j, peut se formuler par l'équation suivante:

$$\begin{aligned} Wt_j &= tr(R_j \, {}^t R_j) \qquad (6.140) \\ \Rightarrow W &= \sum_{j=1}^{m} Wt_j \qquad (6.141) \end{aligned}$$

Le calcul de W s'effectue donc en deux temps. On calcule tout d'abord les matrices W_j, puis on calcule la somme de tous les éléments de ces matrices.

Algorithme séquentiel

On décrit ci-dessous les deux algorithmes issus du schéma itératif décrit par l'équation 6.139, l'un permettant de calculer W et l'autre un cycle de la méthode SIRT.

Algorithme 16 *Calcul de W*

lire(V)
Pour $j = 1$ à m
  créer(W_j)
  Pour $i = 1$ à N^3
    $v_i \xrightarrow{\mathcal{W}} W_{ja}, w_{jb}, w_{jc}, w_{jd}$
$W \leftarrow 0$
Pour $j = 1$ à m
  Pour $l = 1$ à M^2
    $W \leftarrow W + w_{jl}$
  écrire(W)

Le coût du calcul de W, égal à: $m.N^3.T(\xrightarrow{\mathcal{W}}) + m.M^2$, est presque similaire au coût du calcul des matrices de poids de l'algorithme précédent.

Algorithme 17 *SIRT*

Pour $j = 1$ à m
  créer(Imp_j)
  Pour $i = 1$ à N^3
    $v_i \xrightarrow{\mathcal{P}} imp_{ja}, imp_{jb}, imp_{jc}, imp_{jd}$
  lire(Im_j)
Pour $j = 1$ à m
  Pour $l = 1$ à M^2
    $diff_{jl} \leftarrow \lambda_k W(im_{jl} - imp_{jl})$
Pour $j = 1$ à m
  Pour $i = 1$ à N^3
    $v_i \xleftarrow{\mathcal{R}} diff_{ja}, diff_{jb}, diff_{jc}, diff_{jd}$

On peut remarquer que cet algorithme est quasi identique à l'algorithme 15. On effectue le traitement sur toutes les images avant de passer au traitement suivant, tandis que l'algorithme précédent effectue tous les traitements de chaque image avant de passer à l'image suivante. Nous verrons dans la section suivante que cette petite différence algorithmique génère des techniques différentes de parallélisation. Le coût

théorique pour chaque cycle est identique au coût d'un cycle de la méthode ART par blocs. On peut remarquer que le nombre d'opérations effectuées par cycle pour les méthodes par blocs, est deux fois plus élevé qu'une reconstruction par l'algorithme de Feldkamp. Le coût total de la méthode SIRT est donc égal à:

$$T_{SIRT} = m.N^3.T(\overset{\mathcal{W}}{\rightarrow}) + m.M^2 + N_c(m.N^3.T(\overset{\mathcal{P}}{\rightarrow}) + m.M^2 + m.N^3.T(\overset{\mathcal{R}}{\leftarrow})) \quad (6.142)$$

6.2.5 Expérimentations

Nous présentons dans le tableau 14 les temps de reconstruction des algorithmes ART et SIRT implémentés sur notre machine de référence. Les temps correspondent au calcul d'un cycle et au calcul des matrices de poids. Nous avons testé ces algorithmes pour des petits jeux de données. Nous reconstruisons une sphère incluse dans un volume de taille N^3 à partir de N images, de taille N^2, de cette sphère projetée. Nous présentons des images de ces sphères dans un paragraphe ultérieur. Nous nous sommes limités à $N \leq 64$ en raison du coût de ces algorithmes séquentiels. Du fait de la complexité de ces algorithmes, en $N_c N^4(T(\overset{\mathcal{P}}{\rightarrow}) + T(\overset{\mathcal{R}}{\leftarrow}))$ lorsque $m = M = N$, on peut supposer que le temps est multiplié par 16 lorsque N double. D'après cette supposition, on peut estimer que le temps pour $N = 128$ est de l'ordre de 5 heures de calculs.

Algorithme	$N = 32$	$N = 64$
Art	76	1213
Sirt	113	1133

TAB. 14 - *Temps des implémentations séquentielles de Art et Sirt (sec)*

6.3 Parallélisation des méthodes par blocs

Nous avons montré, lors de l'implémentation de la méthode de Feldkamp, que les implémentations sur les machines MIMD permettaient d'obtenir des performances supérieures à celles obtenues sur machine SIMD. Nous parallélisons donc les méthodes par blocs uniquement sur des machines MIMD. Cela nous permet de supposer que l'ensemble des données est stocké dans la mémoire de tous les processeurs. La parallélisation est basée sur l'utilisation des opérateurs parallèles. Nous identifions pour chaque méthode, la meilleure approche parallèle: locale ou globale.

6.3.1 Choix des opérateurs parallèles

Nous examinons schématiquement l'ordonnancement des tâches des deux algorithmes, en faisant apparaître les résultats des opérations de projection ($\stackrel{\mathcal{P}}{\rightarrow}$), de calcul d'*image de différence* ($\stackrel{-}{\rightarrow}$) et de rétroprojection ($\stackrel{\mathcal{R}}{\leftarrow}$):

$$ART: \quad f_d^{(k)} \stackrel{\mathcal{P}}{\rightarrow} Imp_0 \stackrel{-}{\rightarrow} Diff_0 \stackrel{\mathcal{R}}{\leftarrow} f_d^{(k+1)} \cdots f_d^{(k+j)} \stackrel{\mathcal{P}}{\rightarrow} Imp_j \stackrel{-}{\rightarrow} Diff_j \stackrel{\mathcal{R}}{\leftarrow} f_d^{(k+j+1)} \cdots f_d^{(k+m-1)} \stackrel{\mathcal{P}}{\rightarrow} Imp_m \stackrel{-}{\rightarrow} Diff_m \stackrel{\mathcal{R}}{\leftarrow} f_d^{(k+m)} \tag{6.143}$$

$$SIRT: \quad f_d^{(k)} \stackrel{\mathcal{P}}{\rightarrow} Imp_0, \cdots, Imp_j, \cdots, Imp_m \stackrel{-}{\rightarrow} Diff_0 \cdots Diff_j, \cdots, Diff_m \stackrel{\mathcal{R}}{\leftarrow} f_d^{(k+1)} \tag{6.144}$$

On peut remarquer pour l'algorithme ART que chaque résultat est fonction des résultats précédents tandis que, pour l'algorithme SIRT, les images Imp_j comme les images $Diff_j$ peuvent être calculées indépendamment les unes des autres. En se basant sur la parallélisation des opérateurs de base, nous avons donc choisi pour paralléliser l'algorithme itératif de la méthode ART une approche globale, et pour l'algorithme itératif de la méthode SIRT une approche locale.

Nous nous intéressons aussi à la parallélisation du calcul des matrices W_j et de W en fonction des opérations de calcul du poids ($\stackrel{\mathcal{W}}{\rightarrow}$) et de sommation des éléments d'une matrice ($\stackrel{\sum}{\Rightarrow}$):

$$ART: \quad f_d^{(k)} \stackrel{\mathcal{W}}{\rightarrow} W_0 \cdots W_j \cdots W_m \tag{6.145}$$

$$SIRT: \quad f_d^{(k)} \stackrel{\mathcal{W}}{\rightarrow} W_0 \cdots W_j \cdots W_m \stackrel{\sum}{\Rightarrow} Wt_0 \cdots Wt_j \cdots Wt_m \rightarrow W \tag{6.146}$$

Dans les deux cas, le $j$ème résultat peut être calculé indépendamment des autres. Nous parallélisons donc le calcul de ces matrices suivant une approche locale, suivie, pour la méthode SIRT, d'une opération de réduction-diffusion.

6.3.2 Parallélisation de ART

La parallélisation est effectuée sur notre machine parallèle où les données initiales (Im_j, V_i) sont réparties sur les différents processeurs. Le calcul de l'*image de différence* $Diff_j$ nécessite que les *images de projection* Imp_j et que les *images de poids* W_j soient localisées sur le processeur possédant l'image Im_j. L'implémentation parallèle de l'algorithme de calcul des W_j et celle de l'algorithme itératif tiennent compte de cette remarque préalable.

Le calcul parallèle des W_j est comparable à une projection parallèle effectuée par une approche locale. Nous remplaçons les opérations élémentaires de projection $(\xrightarrow{\mathcal{P}})$ par des opérations élémentaires de poids $(\xrightarrow{\mathcal{W}})$ pour obtenir l'algorithme parallèle de calcul des W_j. Le schéma de communication des W_j est un anneau reliant l'ensemble des processeurs.

Algorithme 18 *Calcul Parallèle des* W_j

lire(V_i)
Pour $j = 1$ à $\frac{m}{PE}$
  créer(W_j)
  Pour $i = 1$ à PE
⌈ Pour $t = 1$ à N^3
⌊ $v_t \xrightarrow{\mathcal{W}} w_{ja}, w_{jb}, w_{jc}, w_{jd}$
⇑ send(PE_{i+1}, W_j)
⇓ recv(PE_{i-1}, W_j)
  écrire(W_j)

Il est nécessaire de communiquer l'*image de poids* finale W_j au processeur qui contient l'image Im_j. Cela se traduit par une communication supplémentaire des images W_j qui effectuent alors un tour complet de l'anneau, contrairement à une opération de projection parallèle. Le coût des communication est donc égal à:

$$T_{com(i)} = PE.\frac{m}{PE}(\beta + M^2.\tau) = m(\beta + M^2.\tau) \tag{6.147}$$

Le coût de calcul étant égal au nombre d'opérations élémentaires $\xrightarrow{\mathcal{W}}$ effectuées sur chaque processeur, il s'exprime par:

$$T_{cal(i)} = \frac{m.N^3}{PE}.T(\xrightarrow{\mathcal{W}}) \tag{6.148}$$

L'algorithme itératif est implémenté suivant une approche globale. Chaque processeur calcule une *image de projection* partielle $Imp'_{j,i}$. Ces images sont envoyées au processeur possédant l'*image mesurée* Im_j par une opération de réduction sommation: $Imp_j = \sum_{i=1}^{PE} Imp'_{j,i}$. Ce dernier calcule l'*image de différence* $Diff_j$ qui est ensuite diffusée à l'ensemble des processeurs. L'algorithme utilise des communications globales de tous les processeurs vers un processeur (*reduc-som*) et d'un processeur vers tous les autres (*diffusion*).

Algorithme 19 *ART parallèle*

Pour $j = 1$ à m
  si $(Im_j \in PE_i)$
⌈ Pour $t = 1$ à N^3
⌊ $v_t \xrightarrow{\mathcal{P}} imp'_{ja}, imp'_{jb}, imp'_{jc}, imp'_{jd}$
⇕ reduc-som(PE_i, Imp'_j)
⌈ Pour $l = 1$ à M^2
⌊ $diff_{jl} \leftarrow \lambda_k w_{jl}(im_{jl} - imp_{jl})$
⇕ diffusion$(Diff_j)$
⌈ Pour $t = 1$ à N^3
⌊ $v_t \xleftarrow{\mathcal{R}} diff_{ja}, diff_{jb}, diff_{jc}, diff_{jd}$
  sinon $\{Im_j \in PE_k\}$
⌈ Pour $t = 1$ à N^3
⌊ $v_t \xrightarrow{\mathcal{P}} im'_{ja}, im'_{jb}, im'_{jc}, im'_{jd}$
⇑ reduc-som(PE_k, Im'_j)
⇓ recv$(PE_k, Diff_j)$
⌈ Pour $t = 1$ à N^3
⌊ $v_t \xleftarrow{\mathcal{R}} diff_{ja}, diff_{jb}, diff_{jc}, diff_{jd}$
  finsi

On examine pour le calcul du coût de cet algorithme, le comportement de deux processeurs: PE_k et PE_i, le premier possédant l'image Im_j. Le coût d'une itération j s'exprime alors par:

$$T_j = \max(T_{PE_k}, \max_i(T_{PE_i})) \tag{6.149}$$

On considère que les temps d'émission sont négligeables. Pour le processeur PE_k, le coût de l'opération reduc-som est égal au temps de réception et de sommation des $PE - 1$ images Im'_j. Pour les processeurs PE_i, le coût de l'opération de diffusion est égal au temps de réception d'une image $Diff_j$. On supposera aussi que le calcul de la somme de deux images, comme le calcul de l'*image de différence*, sont égaux chacun à M^2 opérations. On évalue le temps total sur chaque processeur:

$$T_{PE_k} = \frac{N^3}{PE}.T(\xrightarrow{\mathcal{P}}) + \sum_{i=1}^{PE-1}(\beta + M^2.\tau + M^2) + M^2 + \frac{N^3}{PE}.T(\xleftarrow{\mathcal{R}}) \tag{6.150}$$

$$T_{PE_i} = \frac{N^3}{PE}.T(\xrightarrow{\mathcal{P}}) + T_{attente} + (\beta + M^2.\tau) + \frac{N^3}{PE}.T(\xleftarrow{\mathcal{R}}) \tag{6.151}$$

où $T_{attente}$ est le temps d'inactivité du processeur PE_i pendant que le processeur PE_k effectue l'opération de réduction-sommation et calcule l'*image de différence*.

Le temps d'attente est donc égal au temps de l'opération reduc-som et à celui du calcul de l'image $Diff_j$. On peut remarquer que le temps d'inactivité est comptabilisé comme un temps de communication car le processeur PE_i est en phase de réception de l'image $Diff_j$.

Le coût de m itérations, qui constituent un cycle (T_{cycle}), permet d'évaluer le coût total de l'algorithme ART en fonction du calcul du poids et du nombre de cycles N_c:

$$T_{cycle} = \frac{m.N^3}{PE}(T(\overset{\mathcal{P}}{\rightarrow}) + T(\overset{\mathcal{R}}{\leftarrow})) + m.PE.M^2 + m.PE(\beta + M^2.\tau) \quad (6.152)$$

$$T_{ART} = \frac{m.N^3}{PE}.T(\overset{\mathcal{W}}{\rightarrow}) + m(\beta + M^2.\tau) + N_c T_{cycle} \quad (6.153)$$

6.3.3 Parallélisation de SIRT

Nous présentons l'algorithme parallèle de calcul de W et l'algorithme itératif parallèle qui sont basés sur des opérateurs parallèles développés suivant une approche locale.

Le calcul de la matrice W est effectué, comme pour les matrices W_j, par une approche locale. Il utilise une opération de réduction-diffusion pour communiquer à tous les processeurs la valeur de W.

Algorithme 20 *Calcul de W*

```
lire(V_i)
Pour j = 1 à m/PE
    créer(W_j)
    Pour i = 1 à PE
⌈       Pour t = 1 à N^3
⌈         v_t -W-> w_ja, w_jb, w_jc, w_jd
⇑       send(PE_{i+1}, W_j)
⇓       recv(PE_{i-1}, W_j)

W_j ← 0
Pour j = 1 à m/PE
⌈       Pour l = 1 à M^2
⌊           Wt_j ← Wt_j + w_jl
⇑   reduc-som(PE_1, Wt_j)
‖   si PE_i = PE_1
‖       diffusion(W)
‖   sinon
⇓       recv(PE_1, W)
```

Pour le calcul de W, on néglige le coût de la réduction-diffusion des W_j car celui-ci n'est pas significatif en comparaison du coût des W_j. Il s'exprime par:

$$T_W = \frac{m.N^3}{PE}.T(\overset{w}{\rightarrow}) + \frac{m}{PE}M^2 + m(\beta + M^2.\tau) \tag{6.154}$$

Contrairement à la méthode ART, la méthode SIRT calcule, à chaque itération qui est assimilable à un cycle, la rétroprojection de l'ensemble des *images de différence* Les données étant réparties sur l'ensemble des processeurs, l'algorithme itératif calcule, dans un premier temps, les *images de projection* Imp_j et les *images de différence* par lots de PE images. Puis il effectue la rétroprojection de toutes les $Diff_j$. On peut remarquer que le calcul de l'*image de différence* est effectué lorsque l'*image de projection* Imp_j est réceptionnée par le processeur possédant l'image Im_j.

Algorithme 21 *SIRT parallèle*

Pour $j = 1$ à $\frac{m}{PE}$
  Pour $i = 1$ à PE
⌈ Pour $t = 1$ à N^3
⌊ $v_t \xrightarrow{\mathcal{P}} imp_{ja}, imp_{jb}, imp_{jc}, imp_{jd}$
⇑ send(PE_{i+1}, Imp_j)
⇓ recv(PE_{i-1}, Imp_j)
⌈ Pour $l = 1$ à M^2
⌊ $diff_{jl} \leftarrow \lambda_k W(im_{jl} - imp_{jl})$

Pour $j = 1$ à $\frac{m}{PE}$
  Pour $i = 1$ à PE
⌈ Pour $t = 1$ à N^3
⌊ $v_t \xleftarrow{\mathcal{R}} diff_{ja}, diff_{jb}, diff_{jc}, diff_{jd}$
⇑ send$(PE_{i+1}, Diff_j)$
⇓ recv$(PE_{i-1}, Diff_j)$

Lors de chaque étape de communication, on comptabilise seulement le temps de réception. Pour le calcul de PE images, chaque processeur reçoit PE images, donc le coût de communication pour un cycle est égal à:

$$T_{com(i)} = 2m(\beta + M^2.\tau) \tag{6.155}$$

Sur chaque processeur, on calcule la projection et la rétroprojection des V_i et $\frac{m}{PE}$ *images de différence* que l'on évalue par l'équation suivante:

$$T_{cal(i)} = \frac{m.N^3}{PE}(T(\xrightarrow{\mathcal{P}}) + T(\xleftarrow{\mathcal{R}})) + \frac{m}{PE}M^2 \tag{6.156}$$

Si on compare le coût d'un cycle de l'algorithme SIRT par rapport au coût d'un cycle de l'algorithme ART, on s'aperçoit que la réduction-sommation et le calcul de $Diff_j$ entraînent pour l'algorithme ART un sur-coût. celui-ci est causé par les communications multipliées par $\frac{PE}{2}$ et par le temps d'inactivité des processeurs égal à $m.PE.M^2$ opérations. L'efficacité de l'algorithme SIRT peut s'expliquer par un schéma de communication plus performant et par le calcul parallèle des *images de différences* qui est effectué en séquentiel par l'algorithme ART.

On peut aussi remarquer que, globalement, un cycle de l'algorithme SIRT est deux fois plus coûteux qu'une reconstruction par l'algorithme de Feldkamp. Ce qui rejoint le rapport des coûts des algorithmes séquentiels. Le coût total de l'algorithme SIRT parallèle, pour N_c cycles, est de:

$$\begin{aligned} T_{SIRT} &= \frac{m.N^3}{PE}.T(\xrightarrow{\mathcal{W}}) + \frac{m}{PE}M^2 + m(\beta + M^2.\tau) \\ &\quad + N_c(\frac{m.N^3}{PE}(T(\xrightarrow{\mathcal{P}}) + T(\xleftarrow{\mathcal{R}})) + \frac{m}{PE}M^2 + 2m(\beta + M^2.\tau)) \end{aligned} \tag{6.157}$$

6.3.4 Minimisation des temps de communication

Comme pour l'algorithme parallèle de Feldkamp, nous minimisons les temps de communication des méthodes ART et SIRT, d'après des premiers résultats de performance décrits dans la section suivante. Ces résultats montrent la part importante des communications dans le temps total d'exécution.

Deux stratégies complémentaires sont mises en œuvre suivant le schéma de communication des images. Lorsque les images sont transférées sur un anneau, nous recouvrons les communications par des calculs à l'aide de communications non bloquantes. Et nous modifions le schéma de communication des opérations réductions identifiable à un schéma en étoile, en un schéma utilisant un arbre binaire (figure 55)

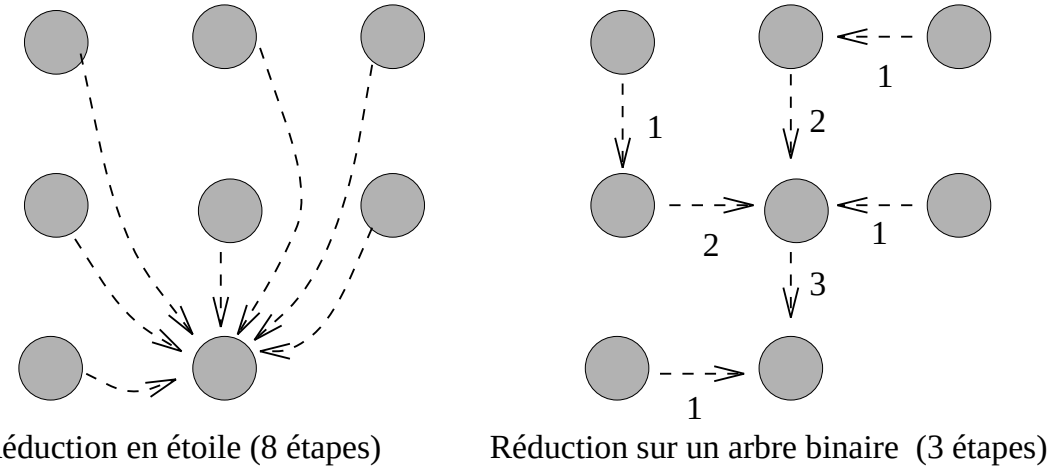


FIG. 55 - *Modification du schéma de communication de l'opération de réduction*

ART optimisé

L'algorithme ART optimisé est développé à partir de l'algorithme de réduction-sommation efficace présenté dans le chapitre 4. La réécriture de l'opération de réduction sommation sur un arbre binaire est basée sur une parallélisation des sommes des *images de projection* partielles Imp'_j.

Nous examinons, pour chaque itération, le comportement du processeur PE_k, possédant l'image Im_j, et celui des autres processeurs PE_i. On définit une nouvelle numérotation des processeurs. Le processeur PE_k devient le processeur PE_0

et les autres processeurs sont numérotés de 1 à $PE-1$, d'après le calcul suivant: $\tilde{i} \leftarrow (i-k+PE) \bmod PE$.

L'objectif de cet algorithme est d'obtenir, en un minimum d'étapes, l'*image de projection* Imp_j sur le processeur PE_0, après le calcul des *images de projection* partielles Imp'_j sur chaque processeur. En fait, si on suppose que le nombre de processeur peut s'exprimer comme une puissance de 2:$PE = 2^n$, on effectue cette réduction-sommation en n étapes. On simule l'arbre binaire qui est constitué de n niveaux. Si processeur $PE_{\tilde{i}}$ appartient à un nœud de l'arbre au niveau h, il appartiendra au niveau $h+1$, si il vérifie $\tilde{i} \bmod 2^{h+1} = 0$. Cette appartenance entraîne pour chaque processeur un traitement différent:

$\underline{PE_{\tilde{i}} \notin h+1}$ Il envoie sa somme partielle des images Imp'_j au processeur $PE_{\tilde{i}-(\tilde{i} \bmod 2^{h+1})}$,

$\underline{PE_{\tilde{i}} \in h+1}$ Il reçoit la somme partielle $Imp"_j$ du processeur $PE_{\tilde{i}+2^h-1}$, et calcule la somme des sommes partielles: $Imp'_j = Imp'_j + Imp"_j$.

On peut remarquer que le processeur PE_0 appartient à tous les niveaux de l'arbre, il réceptionne toutes les sommes des images Imp'_j.

On calcule la complexité de cet algorithme pour évaluer le gain de performances par rapport à l'algorithme 19. Nous calculons le coût de chaque itération pour le processeur PE_0 et pour un processeur $PE_{\tilde{i}}$. Si le processeur $PE_{\tilde{i}}$ appartient au niveau h de l'arbre binaire, il calcule h sommes partielles des images Imp'_j après réception des images $Imp"_j$, et envoie son résultat à un autre processeur. Son coût s'exprime alors en fonction de h:

$$T_{PE_{\tilde{i}}} = \frac{N^3}{PE}.T(\overset{\mathcal{P}}{\rightarrow}) + h(M^2 + (\beta + M^2.\tau)) + T_{attente} + (\beta + M^2.\tau) + \frac{N^3}{PE}.T(\overset{\mathcal{R}}{\leftarrow}) \quad (6.158)$$

Le temps d'attente $T_{attente}$ du processeur $PE_{\tilde{i}}$ est égal au temps des calculs supplémentaires effectués par le processeur PE_0. Ce dernier appartient à tous les niveaux de l'arbre binaire. Il effectue donc $n-h$ étapes en plus du processeur $PE_{\tilde{i}}$ et calcule l'*image de différence* $Diff_j$. Son temps est alors égal à:

$$T_{PE_0} = \frac{N^3}{PE}.T(\overset{\mathcal{P}}{\rightarrow}) + n(M^2 + (\beta + M^2.\tau)) + \frac{N^3}{PE}.T(\overset{\mathcal{R}}{\leftarrow}) \quad (6.159)$$

On en déduit le coût total d'une itération (T_j), définie par l'équation 6.149, qui sert à évaluer le coût d'un cycle de l'algorithme ART (T_{cycle}).

$$\begin{aligned} T_j &= \frac{N^3}{PE}.T(\overset{\mathcal{P}}{\rightarrow}) + n(M^2 + (\beta + M^2.\tau)) \\ &\quad + M^2 + (\beta + M^2.\tau) + \frac{N^3}{PE}.T(\overset{\mathcal{R}}{\leftarrow}) \qquad (6.160) \\ T_{cycle} &= m\frac{N^3}{PE}(T(\overset{\mathcal{P}}{\rightarrow}) + T(\overset{\mathcal{R}}{\leftarrow})) + m.(n+1)(M^2 + (\beta + M^2.\tau)) \qquad (6.161) \end{aligned}$$

Cette optimisation a permis de minimiser le coût total d'un cycle en divisant par $\frac{PE-1}{n}$ le coût de l'opération de réduction-sommation, où $n = \frac{\log PE}{\log 2}$.

Algorithme 22 *ART optimisé*

Pour $j = 1$ à m
  si ($Im_j \in PE_i$)
⌈ Pour $t = 1$ à N^3
⌊ $v_t \overset{\mathcal{P}}{\rightarrow} imp'_{ja}, imp'_{jb}, imp'_{jc}, imp'_{jd}$
  $h \leftarrow 0$
  Tant que $2^h < PE + 1$
⇕ recv($PE_{2^{h-1}}, Imp_j$")
⌈ Pour $l = 1$ à M^2
⌊ $imp'_{jl} \leftarrow imp'_{jl} + imp$"$_{jl}$
  $h \leftarrow h + 1$
⌈ Pour $l = 1$ à M^2
⌊ $diff_{jl} \leftarrow \lambda_k w_{jl}(im_{jl} - imp_{jl})$
⇕ diffusion($Diff_j$)
⌈ Pour $t = 1$ à N^3
⌊ $v_t \overset{\mathcal{R}}{\leftarrow} diff_{ja}, diff_{jb}, diff_{jc}, diff_{jd}$
  sinon {$Im_j \in PE_k$}
⌈ Pour $t = 1$ à N^3
⌊ $v_t \overset{\mathcal{P}}{\rightarrow} im'_{ja}, im'_{jb}, im'_{jc}, im'_{jd}$
  $\tilde{i} \leftarrow (i - k + PE) \bmod PE$
  $h \leftarrow 0$
  Tant que $2^h < PE + 1$
    $reste \leftarrow \tilde{i} \bmod 2^h$
    si $reste = 0$
⇕ recv($PE_{\tilde{i}+2^{h-1}}, Imp_j$")
⌈ Pour $l = 1$ à M^2
⌊ $imp'_{jl} \leftarrow imp'_{jl} + imp$"$_{jl}$
    $h \leftarrow h + 1$
    sinon
⇕ send($PE_{\tilde{i}-reste}, Imp'_j$)
    $h \leftarrow PE + 1$
    finsi
⇕ recv($PE_k, Diff_j$)
⌈ Pour $t = 1$ à N^3
⌊ $v_t \overset{\mathcal{R}}{\leftarrow} diff_{ja}, diff_{jb}, diff_{jc}, diff_{jd}$
  finsi

Pour minimiser toutes les communications de l'algorithme ART, nous optimisons aussi le calcul des *images de poids* W_j. Le calcul parallèle est effectué par une approche locale. Cet algorithme étant quasi identique à l'algorithme de la projection parallèle développé suivant une approche locale, sa version optimisée est semblable

à l'algorithme de projection en mode asynchrone (algorithme 5). Il est alors basé sur un recouvrement des communications par les calculs des W_j.

Pour évaluer le coût de cet algorithme, nous utilisons les calculs du coût de l'algorithme 5. Nous avons montré que le pourcentage de recouvrement des communications par des calculs était fonction des caractéristiques de chaque machine MIMD. Nous exprimons le coût théorique du calcul des W_j en désignant par ϵ le nombre de communications non-recouvertes:

$$T_{W_j} = \frac{m.N^3}{PE}.T(\overset{\mathcal{W}}{\rightarrow}) + \epsilon(\beta + M^2.\tau) \tag{6.162}$$

Le coût total de l'algorithme pour N_c s'écrit alors:

$$\begin{aligned} T_{ART\,optimise} &= \frac{m.N^3}{PE}(T(\overset{\mathcal{W}}{\rightarrow}) + N_c(T(\overset{\mathcal{P}}{\rightarrow}) + T(\overset{\mathcal{R}}{\leftarrow}))) + (N_c.m.(n+1)+1)M^2 \\ &\quad + (N_c.m.(n+1)+\epsilon)(\beta + M^2.\tau) \end{aligned} \tag{6.163}$$

SIRT optimisé

Les opérateurs parallèles, que nous avons utilisés pour paralléliser la méthode SIRT, sont implémentés suivant une approche locale. De ce fait, la version optimisée de l'algorithme SIRT est développée à partir de la version optimisée de ces opérateurs. Les communications alors utilisées sont des communications non-bloquantes qui permettent la minimisation du temps total d'exécution. Nous décrivons l'algorithme effectué sur chaque processeur. Il est constitué de deux étapes: le calcul des Imp_j et des $Diff_j$ et la rétroprojection des $Diff_j$.

Dans la première étape, un processeur PE_i calcule la projection du sous-volume V_i, soit sur l'image Imp_k reçue du processeur PE_{i-1}, soit sur une image Imp_j stockée dans sa mémoire. Cette image Imp_j est alors envoyée au processeur PE_{i+1}. Si l'image reçue Imp_k provient du processeur PE_i ($NVol_k = PE$), alors on calcule l'*image de différence* $Diff_k$ qui est stockée dans la mémoire du processeur.

Une fois que le processeur a projeté son sous-volume V_i sur l'ensemble des *images de projection* ($NIma_i = m$) et calculé toutes ses *images de différence*, il commence la rétroprojection des *images de différence*. Il calcule la rétroprojection soit de l'image $Diff_k$ reçue du processeur PE_{i-1}, soit celle stockée dans sa mémoire ($Diff_j$). Il

envoie au processeur PE_{i+1} les images n'ayant pas été rétroprojetées sur l'ensemble des sous-volumes ($NVol_k \leq PE$). Le cycle est terminé lorsque chaque sous-volume V_i a reçu la contribution de toutes les images $Diff_j$ ($NIma_i = m$).

Algorithme 23 *SIRT asynchrone*

```
   NIma_i ← 0                                        NIma_i ← 0
   Tant que NIma_i ≤ m                               Tant que NIma_i ≤ m
⇕     si (n-recv(PE_i, Imp_k)=faux)                ⇕     si (n-recv(PE_i, Diff_k)=faux)
        créer(Imp_j), NVol_j ← 0                          lire(Diff_j), NVol_j ← 0
      sinon                                             sinon
        Imp_j ← Imp_k                                     Diff_j ← Diff_k
      finsi        si NVol_j ≤ PE                       finsi        si NVol_j ≤ PE
⌈       Pour v_t ∈ V_i                                  Pour t = 1 à N^3
|         v_t --P--> imp_ja, imp_jb, imp_jc, imp_jd      v_t <--R-- diff_ja, diff_jb, diff_jc, diff_jd
⌊     NIma_i ← Nima + 1, NVol_j ← NVol_j + 1            NIma_i ← Nima + 1, NVol_j ← NVol_j + 1
⇕       send(PE_{i+1}, Imp_j)                        ⇕       send(PE_{i+1}, Diff_j)
      sinon                                             finsi
⌈       Pour l = 1 à M^2
|         diff_jl ← λ_k W(im_jl − imp_jl)
        écrire(Diff_j)
      finsi
```

L'algorithme étant composé de deux étapes différentes, on pourrait supposer que le déroulement de l'algorithme se divise en deux phases de calcul-communication séparées par une phase de synchronisation des processeurs. Celle-ci est implicite dans la version parallèle de l'algorithme SIRT (algorithme 21). Ici on peut considérer qu'il y a une seule phase de calcul-communication.

Par exemple, on peut supposer que le processeur PE_i a déjà calculé l'ensemble de ses *images de différence*, et qu'il lui reste à projeter son sous-volume V_i sur une image Imp_j, appartenant à un processeur $PE_{i+i'}$ (figure 56). Ce processeur PE_i calcule alors la projection sur Imp_j et l'envoie au processeur PE_{i+1}. Pendant que ce dernier la réceptionne, le processeur PE_i commence la rétroprojection des *images de différence*, à partir, soit d'une de ses images $Diff_j$ stockée dans sa mémoire, soit d'une image reçue du processeur PE_{i-1} (dans le cas où PE_{i-1} a déjà calculé la rétroprojection de $Diff_j$ et la lui a envoyée).

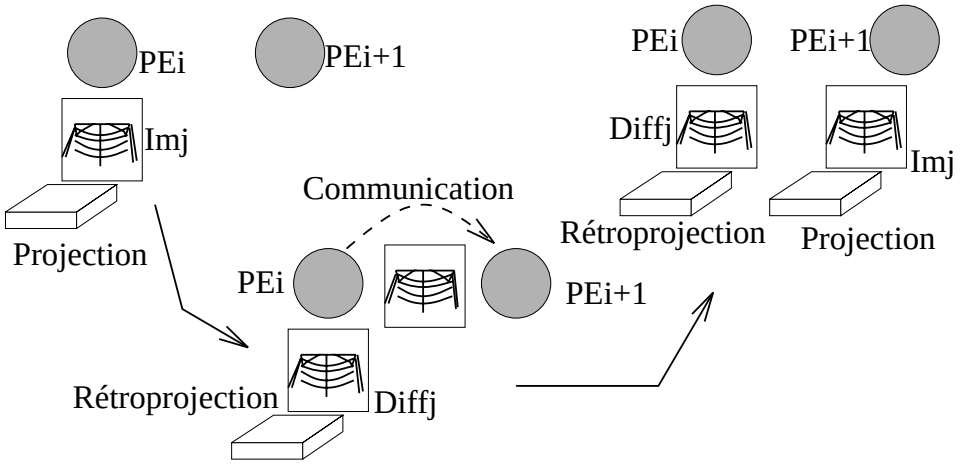


FIG. 56 - *Recouvrement des communications de l'algorithme SIRT optimisé*

On peut donc considérer que la plupart des communications de l'étape de calcul des Imp_j et des $Diff_j$ sont recouvertes. On exprime le coût d'un cycle de SIRT asynchrone en désignant par ϵ_1 les communications non-recouvertes lors des deux étapes.

$$T_{cycle} = \frac{m.N^3}{PE}(T(\overset{\mathcal{P}}{\rightarrow}) + T(\overset{\mathcal{R}}{\leftarrow})) + \frac{m}{PE}M^2 + \epsilon_1(\beta + M^2.\tau) \tag{6.164}$$

De la même manière que pour le calcul des W_j, on optimise l'algorithme de calcul de W en utilisant des communications asynchrones pour recouvrir les communications par des calculs. Le coût de cet algorithme est fonction du pourcentage ϵ_2 des communications non-recouvertes:

$$T_W = \frac{m.N^3}{PE}T(\overset{\mathcal{W}}{\rightarrow}) + \frac{m}{PE}M^2 + \epsilon_2(\beta + M^2.\tau) \tag{6.165}$$

Après chaque cycle, on évalue le critère d'arrêt qui correspond à une réduction-diffusion des écarts entre V_i^k et V_i^{k+1}. Du point de vue du parallélisme, cette réduction-diffusion permet une resynchronisation des processeurs. Si on suppose que le pourcentage des communications non-recouvertes ϵ_1 est constant, le coût de la version optimisée de SIRT s'écrit en fonction du nombre de cycles N_c:

$$\begin{aligned} T_{SIRT\,asynchrone} &= \frac{m.N^3}{PE}(T(\overset{\mathcal{W}}{\rightarrow}) + N_c(T(\overset{\mathcal{P}}{\rightarrow}) + T(\overset{\mathcal{R}}{\leftarrow}))) + \frac{m}{PE}(N_c+1)M^2 \\ &\quad + (N_c\epsilon_1 + \epsilon_2)(\beta + M^2.\tau) \end{aligned} \tag{6.166}$$

6.3.5 Expérimentations et discussion

Nous implémentons les algorithmes ART et SIRT dans leurs deux versions:

- Pour ART, nous testons la version basée sur une opération de réduction non-optimisée, le "pvm-reduce", et la version optimisée (Opt), basée sur une opération de réduction sur un arbre binaire.

- Pour SIRT, nous testons une version découpant le programme en phases de calcul et de communication synchrone (Syn) et, une version mettant en œuvre le recouvrement calcul/communication à l'aide de communications asynchrones (Asyn).

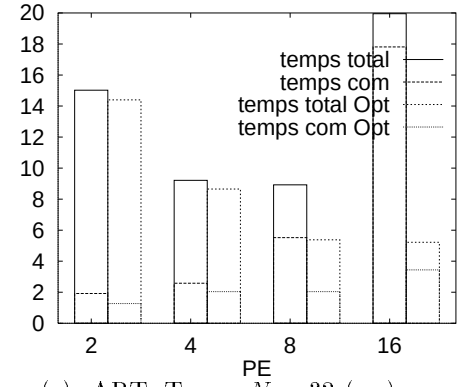
Nous utilisons les mêmes jeux d'essai que ceux employés pour tester l'algorithme de Feldkamp. Chaque jeu d'essai permet de reconstruire une boule incluse dans un volume V de taille N^3 à partir de N *images de projection* Im_j de taille N^2. Pour évaluer les performances de nos algorithmes, nous nous intéressons au calcul de la matrice de poids (W_j pour ART et W pour SIRT) et au calcul d'un cycle de chaque algorithme. Le problème à résoudre a un coût séquentiel de N^4 pour le calcul du poids, et de $2N^4$ pour le calcul d'un cycle, donc un coût séquentiel initial de l'ordre de $3N^4$. Dans la section suivante, nous présenterons les temps nécessaires à la reconstruction d'une image 3D pour un nombre de cycles donné N_c. Nous avons choisi, comme machine cible, la ferme de processeurs, le SP1 d'IBM et le Cray T3D, au vu de leurs performances obtenues avec l'algorithme de Feldkamp et parce qu'elles utilisent la même librairie de communication: PVM.

Ferme de Processeurs

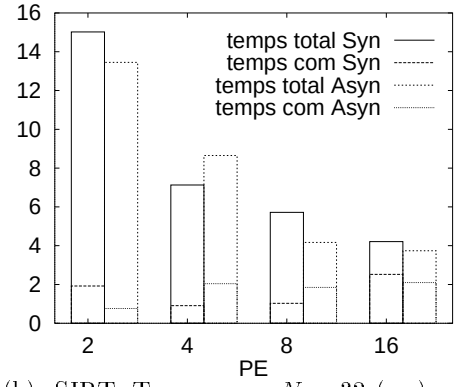
La figure 57 présente les résultats de performances et d'efficacités des implémentations de ART (colonne de droite) et de SIRT (colonne de gauche). Nous testons ces algorithmes pour différentes tailles de problèmes ($N = 32, 64, 128$). Pour des problèmes de grande taille ($N = 128$), nous présentons les temps de calcul sur seulement 8 et 16 processeurs en raison du coût de ces implémentations qui est de l'ordre de 800 à 900 secondes sur 8 processeurs. On peut supposer, au vu des résultats expérimentaux que le temps séquentiel est donc de l'ordre de 1600 à 1700 secondes.

Les figures 57(a),57(c) et 57(e) montrent bien que la version de ART optimisée permet de réduire le temps de communications. La part des communications étant plus importante pour des problèmes de petite taille, les résultats de la figure 57(a) mettent en évidence le gain de performances obtenu avec notre version optimisée qui est de l'ordre de 20% d'efficacité pour 8 et 16 processeurs (figure 57(g)). Les courbes d'efficacité (figure 57(g)) nous permettent de constater que l'efficacité des implémentations se dégrade lorsque le nombre de processeurs augmente en raison des nombreuses communications globales. Cependant, on peut souligner que nous obtenons de bons résultats d'efficacité (87% au minimum) avec notre version optimisée pour $N = 64$.

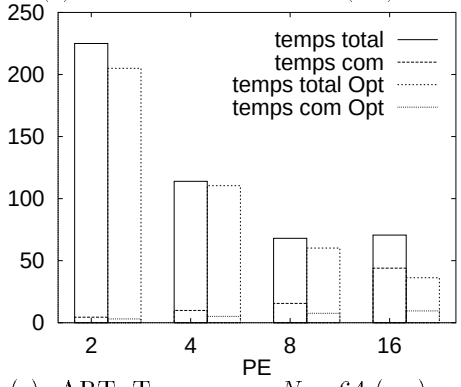
La méthode SIRT est parallélisée suivant une approche locale. Nous avons montré que cette approche était très efficace sur cette machine lors de l'implémentation de la méthode de Feldkamp (figure 45). Nous retrouvons ici des résultats identiques pour $N = 64$ (figure 57(h)). Le gain de performances de la version asynchrone (Asyn) que l'on peut observer (figures 57(b),57(d) et 57(f)) est de l'ordre de 1 à 2% en raison de la faible part des communications dans le temps total d'exécution. On peut noter que pour certaines expérimentations la version synchrone est plus rapide que la version asynchrone. Cette dernière est tributaire du rapport entre le temps de calcul et le temps de communication des images, qui peut générer des temps d'attente lorsque le nombre de processeurs n'est pas adapté à la taille du problème. Nous pouvons constater que l'efficacité de ces implémentations augmente en fonction de la taille du problème (figure 57(h)).



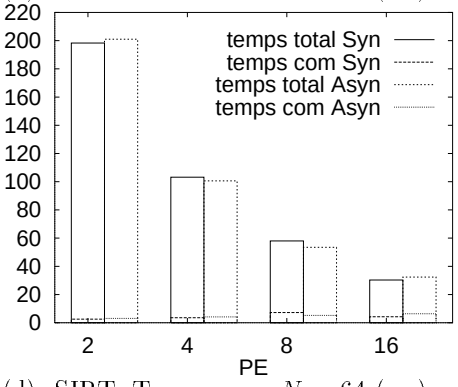
(a) ART: Temps $N = 32$ (sec)



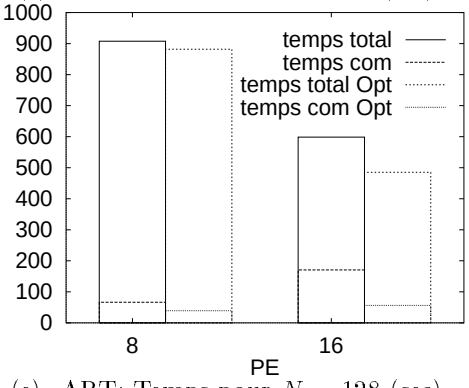
(b) SIRT: Temps pour $N = 32$ (sec)



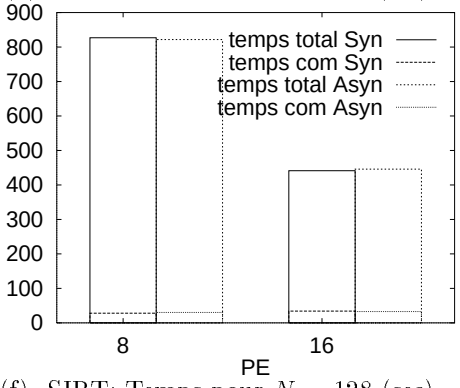
(c) ART: Temps pour $N = 64$ (sec)



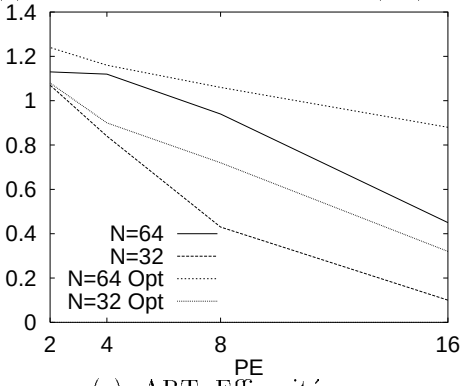
(d) SIRT: Temps pour $N = 64$ (sec)



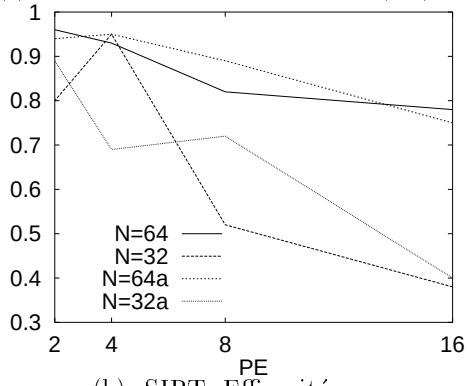
(e) ART: Temps pour $N = 128$ (sec)



(f) SIRT: Temps pour $N = 128$ (sec)



(g) ART: Efficacités



(h) SIRT: Efficacités

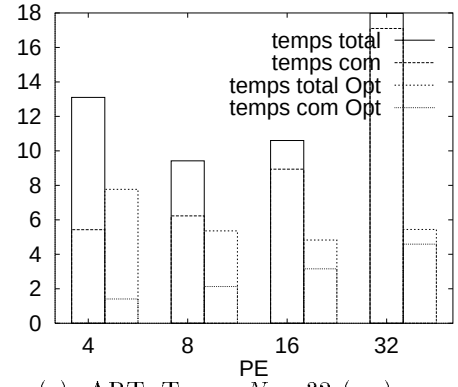
FIG. 57 - *Temps et efficacités de ART et SIRT sur la ferme de processeurs*

SP1

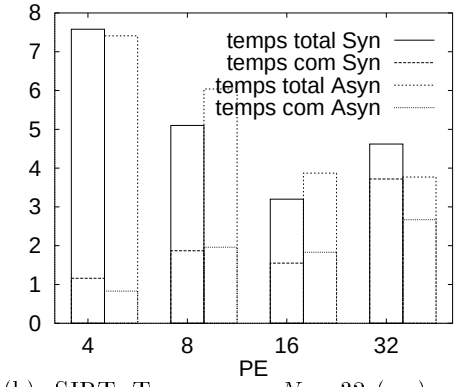
Nous présentons les résultats de nos expérimentations pour des implémentations de ART (figure 58 colonne gauche) et de SIRT (figure 58 colonne droite). Sur cette machine composée de 32 processeurs, nous comparons les performances obtenues pour des tailles de problèmes allant de $N = 32$ à $N = 128$, pour un nombre de processeurs allant de 4 à 32 processeurs.

Nous obtenons pour l'algorithme ART des résultats comparables à ceux obtenus sur la ferme de stations. Nous pouvons constater que l'opération de réduction fournie avec la librairie PVM n'est pas efficace sur 32 processeurs pour toutes les tailles de problèmes (figures 58(a), 58(c) et 58(e)). On peut observer que le gain de performances de notre algorithme optimisé est dû à la réduction très importante des temps de communication. Cependant la version standard comme la version optimisée sont des méthodes peu efficaces sur cette machine (figure 58(g)). On peut noter ici aussi que l'efficacité augmente avec la taille du problème. Si on se base sur le temps de la version optimisée sur 8 processeurs pour définir le temps séquentiel, on obtiendrait une efficacité de 72% sur 32 processeurs, ce qui nous amène à penser que cet algorithme est plus efficace pour des grandes tailles de problème.

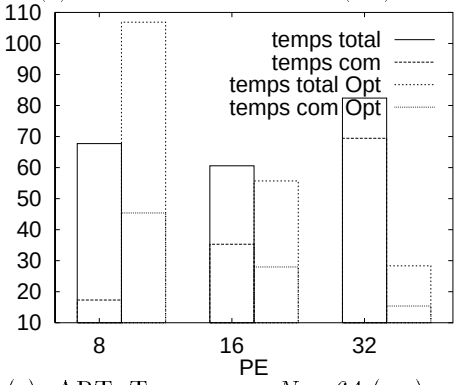
Les performances obtenues avec l'algorithme SIRT (figures 58(b), 58(d) et 58(f)) sont similaires à celles obtenues pour la méthode de Feldkamp. Lorsque l'on reconstruit un volume d'une petite taille ($N = 32$), la part du temps de communication dans le temps total d'exécution est très importante. On peut remarquer que la version asynchrone génère des temps d'attente qui pénalisent les performances. La figure 58(h) montre que pour de tels problèmes, les implémentations de SIRT ne sont pas efficaces. A l'opposé, pour des plus grandes tailles de problème, la part du temps de communication se réduit à 10%-20% du temps total d'exécution. Le gain de performance de la version asynchrone est très faible en raison de l'efficacité de la version synchrone (figure 58(h)). Comme cette machine est une machine ouverte, les processeurs ne sont pas attribués de manière exclusive. Ainsi la version synchrone est tributaire des processeurs qui sont occupés par d'autres processus PVM et, pour chaque processeur, on a des temps de communication et de calcul très différents. Nous avons constaté que la version asynchrone homogénéise les temps de calcul et communication.



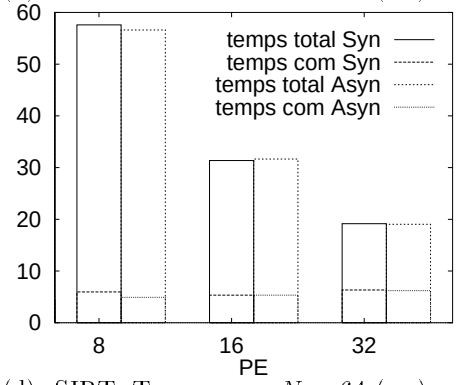
(a) ART: Temps $N = 32$ (sec)



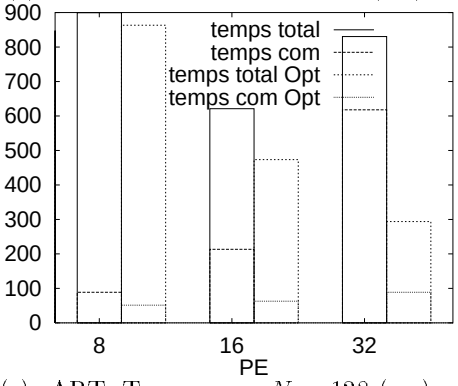
(b) SIRT: Temps pour $N = 32$ (sec)



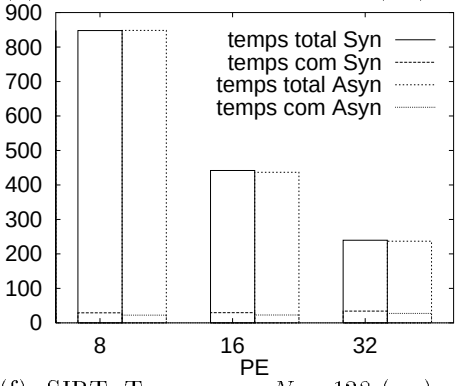
(c) ART: Temps pour $N = 64$ (sec)



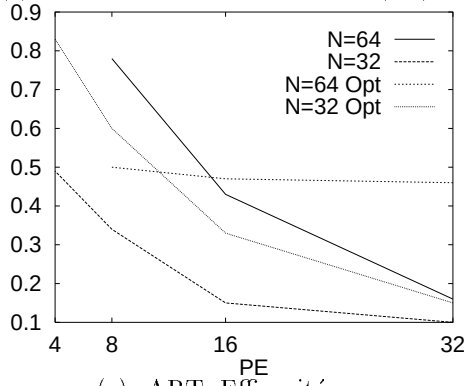
(d) SIRT: Temps pour $N = 64$ (sec)



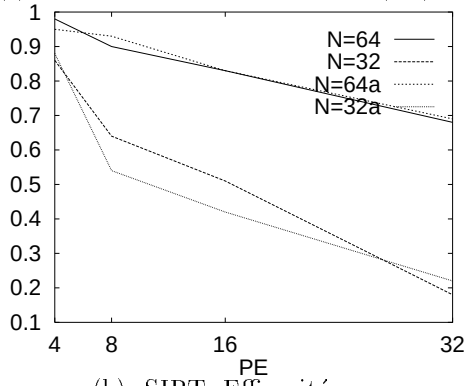
(e) ART: Temps pour $N = 128$ (sec)



(f) SIRT: Temps pour $N = 128$ (sec)



(g) ART: Efficacités



(h) SIRT: Efficacités

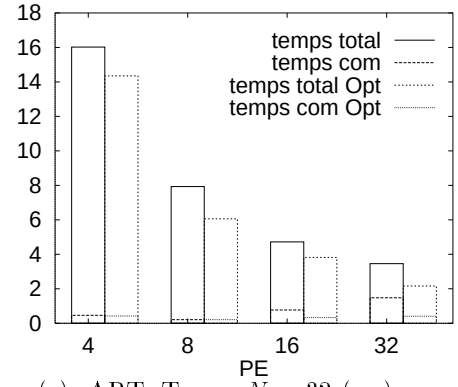
FIG. 58 - *Temps et efficacités de ART et SIRT sur le SP1*

T3D

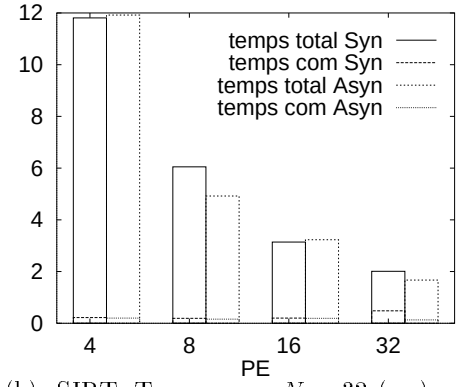
Nous avons pu constater avec l'implémentation de la méthode de Feldkamp sur cette machine, que la part des communications était infime devant la part des calculs grâce notamment à la rapidité du réseau d'interconnexion. Nous évaluons les performances de nos algorithmes sur les jeux de données présentés précédemment en faisant varier le nombre de processeurs en fonction de la taille du problème.

Les figures 59(a), 59(c) et 59(e) présentent les expérimentations de l'algorithme ART. Le T3D est un tore de dimension $8 \times 4 \times 4$ divisé. Lorsque l'on effectue des communications sur la troisième dimension ($PE = 64$), le temps des communications double ce qui montre que les opérations globales sur cette machine ne sont pas très efficaces. Notre version optimisée permet de diminuer sensiblement le coût des communications. Si on prenait le temps sur 16 processeurs pour définir le temps séquentiel, on obtiendrait une efficacité de seulement 79% pour notre version optimisée sur 128 processeurs. La figure 59(g) montre que pour 8 processeurs, on obtient un pic des courbes d'efficacité. On explique ces pics de performance par le faible nombre de communications et par la taille des données qui est inférieure à la taille de la mémoire cache du processeur.

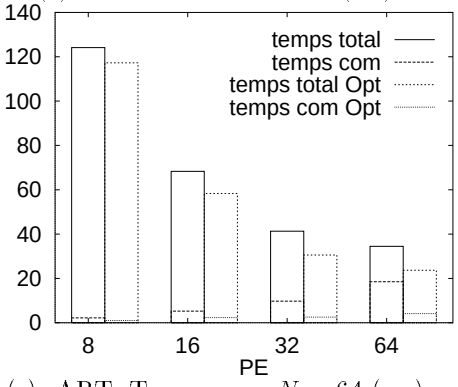
Pour les implémentations de SIRT (figures 59(b), 59(d) et 59(f)), on retrouve les performances des implémentations de la méthode de Feldkamp. La part des communications est quasi-inexistante sauf pour la version synchrone lorsque $N = 32$ et $PE = 32$. Dans ce cas, la taille du problème n'est pas adaptée au nombre de processeurs. La figure 59(h) présente les efficacités des implémentations. Globalement on peut dire que cette méthode est efficace sur le T3D. On peut remarquer que la version asynchrone est parfois moins rapide que la version synchrone en raison du très faible coût des communications. Pour $PE = 8$, on retrouve le même pic d'efficacité similaire à celui de l'implémentation de ART.



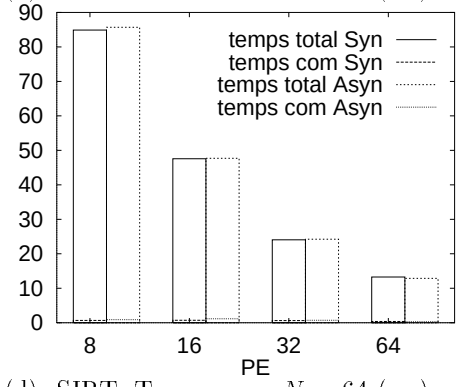
(a) ART: Temps $N = 32$ (sec)



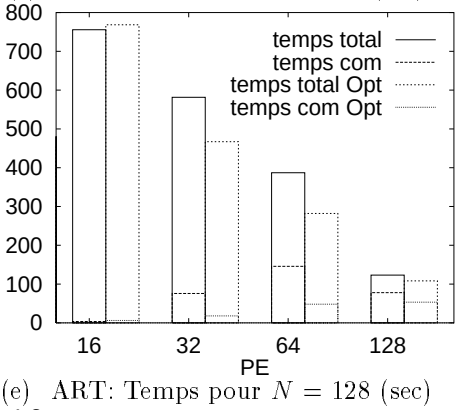
(b) SIRT: Temps pour $N = 32$ (sec)



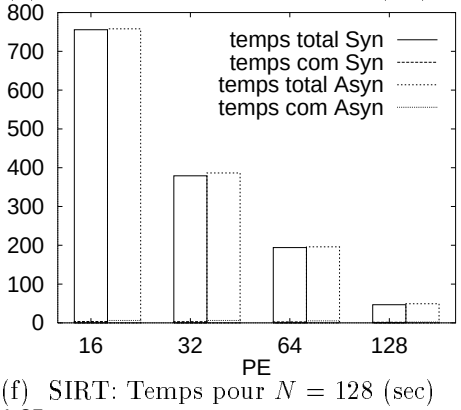
(c) ART: Temps pour $N = 64$ (sec)



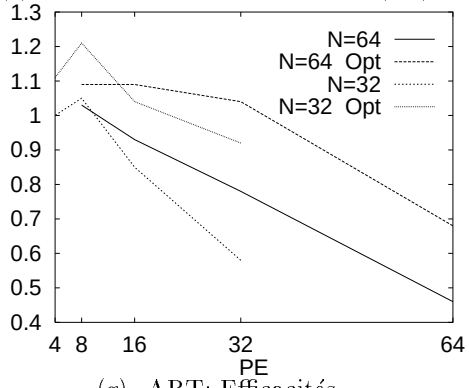
(d) SIRT: Temps pour $N = 64$ (sec)



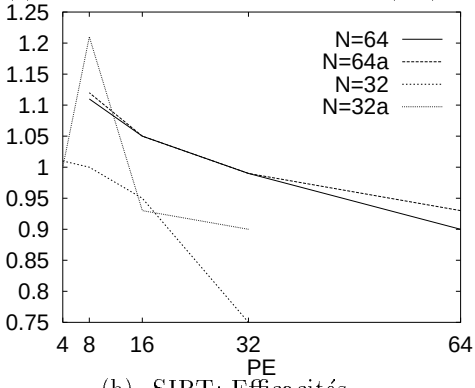
(e) ART: Temps pour $N = 128$ (sec)



(f) SIRT: Temps pour $N = 128$ (sec)



(g) ART: Efficacités



(h) SIRT: Efficacités

FIG. 59 - *Temps et efficacités de ART et SIRT sur le T3D*

6.4 Reconstruction d'image 3D

Nous reconstruisons par les méthodes de ART et SIRT les images 3D présentées dans le chapitre précédent. Pour valider nos reconstructions, nous calculons pour chaque reconstruction le coefficient de corrélation. Nous nous intéressons à l'évolution de ces coefficients en fonction du nombre de cycles et du paramètre de relaxation λ_k.

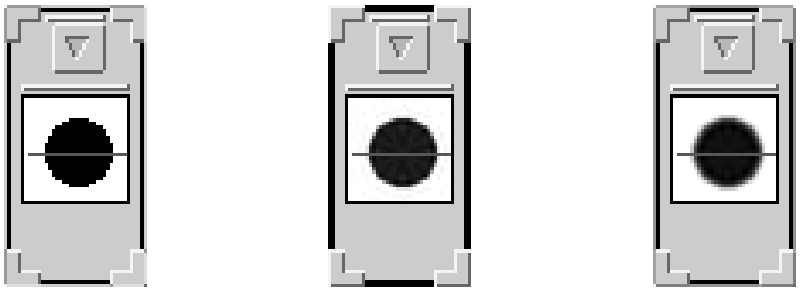
Sphère comprise dans un volume de taille 32^3

Nous reconstruisons ici la sphère comprise dans un volume de taille 32^3. Les figures 60(b) et 60(c) présentent une coupe de la sphère reconstruite par la méthode ART en 20 cycles et par la méthode SIRT en 30 cycles. Les figures 60(d) et 60(f) permettent de visualiser l'évolution des profils de la sphère reconstruite en fonction du nombre de cycles (a1,...,a20 pour le nombre de cycles de ART et s1,...,s30 pour SIRT et de les comparer au profil de la sphère initiale.

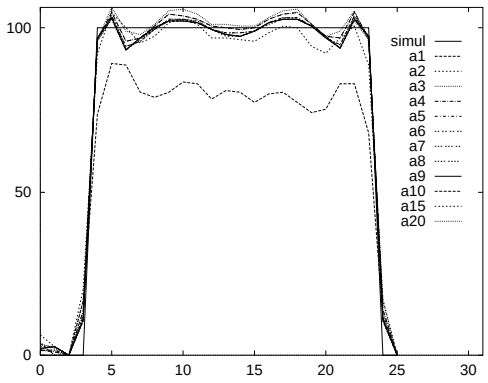
Après quelques cycles de l'algorithme ART, le profil se stabilise. On retrouve ce résultat sur la figure 60(e) qui montre qu'après 5-6 cycles les coefficients de corrélation évoluent très lentement. D'après les courbes de cette figure, nous pouvons constater que le paramètre de relaxation λ_k permet d'accélérer la convergence de la méthode ART lors des premières itérations.

Pour l'algorithme SIRT, le profil évolue en fonction du nombre de cycles. Cet exemple montre que l'algorithme SIRT converge plus lentement que ART d'après les courbes du coefficient de corrélation (figure 60(g)). Le paramètre de relaxation permet d'accélérer la convergence de cette méthode, mais le gain obtenu est minime au vu de l'echelle utilisée pour représenter les coefficients de corrélation.

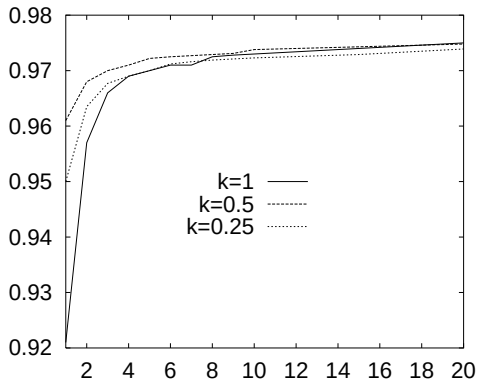
Si on compare ces deux méthodes de reconstruction, on s'aperçoit que la méthode ART obtient des meilleurs résultats de reconstruction que la méthode SIRT, au niveau du coefficent de corrélation (0.97 pour ART et 0.79 pour SIRT). La méthode de Feldkamp obtenait pour cette reconstruction, à partir de projections calculées, un coefficent de corrélation de 0.968, ce qui montre que la reconstruction par la méthode ART est meilleure pour cet exemple.



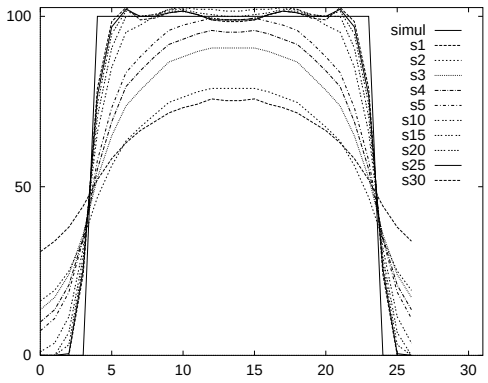
(a) (b) (c)



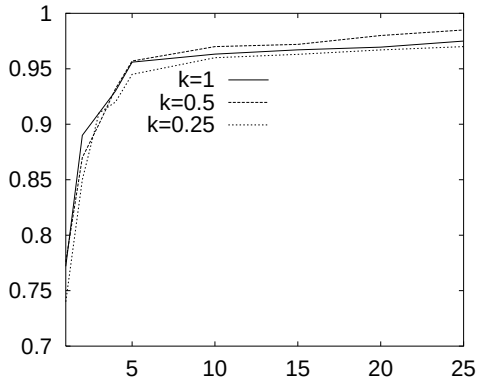
(d) Profil suivant le nombre de cycles pour la méthode ART



(e) Evolution du coefficient de corrélation pour la méthode ART



(f) Profil suivant le nombre de cycles pour la méthode SIRT



(g) Evolution du coefficient de corrélation pour la méthode SIRT

FIG. 60 - *Coupe médiane de la sphère initiale (a) et des sphères reconstruites par la méthode ART (b) et SIRT (c), Profils suivant le nombre de cycles de ART(d) et SIRT(f), Etude de la convergence de ART(d) et SIRT(g) d'après le coefficent de corrélation suivant le paramètre λ_k (k)*

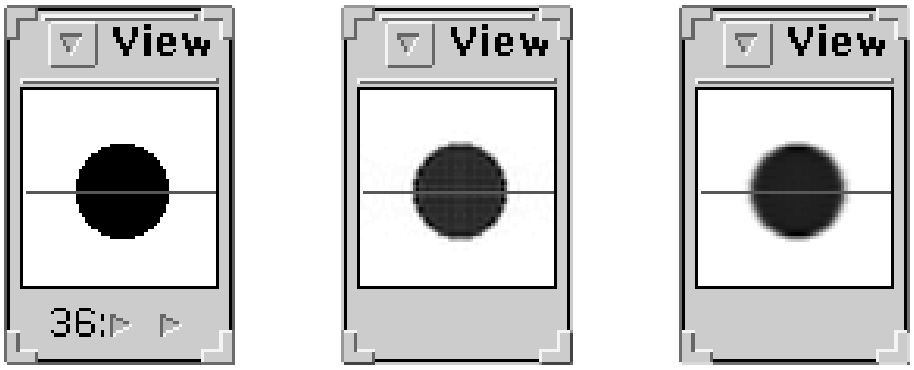
Sphère comprise dans un volume de taille 64^3

Les figures 61(b) et 61(c) présentent des coupes de la sphère comprise dans un volume de taille 64^3 reconstuites par les méthodes ART et SIRT. Nous avons fixé le paramètre de relaxation λ_k à 0.5 au vu des résultats obtenus avec la reconstruction de la sphère précédente. La convergence des algorithmes pour cette taille de problème est similaire à celle observée pour la reconstruction d'un volume 32^3.
En comparant les profils, on s'aperçoit que la méthode SIRT a un comportement relativement stable (figure 61(e)), et qu'elle converge lentement vers la sphère initiale. L'algorithme ART converge en 5-6 cycles pour ensuite commencer à osciller, si on regarde le comportement de son coefficient de corrélation (figure 61(d)) . La figure 61(e) montre bien que les profils se chevauchent lorsque le nombre de cycles est supérieur à 5. L'évolution des coefficients de corrélation en fonction du nombre de cycles (figure 61(f)) permet de mesurer la différence de reconstruction entre les méthodes. Si on compare les profils des trois méthodes (figure 61(g)), on s'aperçoit que l'on obtient des résultats similaires avec l'une des trois méthodes. La comparaison des coefficients de corrélation montre que la méthode de Feldkamp est reconstruction plus approximative (coefficient de corrélation de 0.976) ce qui est en accord avec la théorie.

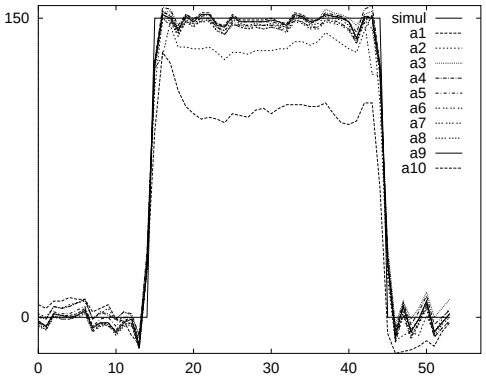
	Cycles	1	2	3	4	5	6	7	8	9	10
Art	Temps Total	27	50	72	93	114	138	159	182	206	220
	Temps Com	15	28	41	52	65	78	91	105	119	124
	Cycles	1	2	3	4	5	10	15	20	25	
Sirt	Temps Total	19	32	45	58	71	134	198	260	338	
	Temps Com	6	11	14	18	22	40	59	77	108	

TAB. 15 - *Temps de reconstruction de cette sphère pour ART et SIRT (sec)*

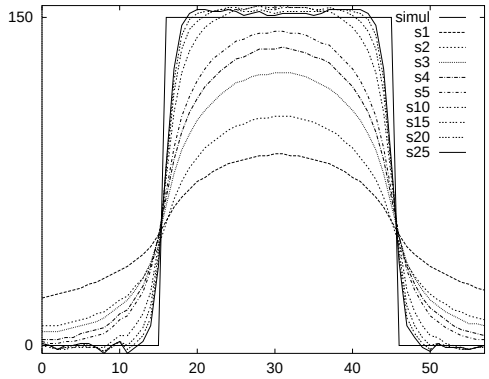
Le tableau 15 donne les temps de reconstruction de cette sphère par les trois méthodes de reconstruction mesurées sur le SP1 (32 processeurs). On peut comparer ces temps avec les 8 secondes (dont 2 de communications) de la méthode de Feldkamp. Il est intéressant de constater que pour obtenir une bonne reconstruction de cette sphère avec l'algorithme ART, il faut 17 fois plus de temps qu'avec la méthode de Feldkamp. Pour chaque méthode algébrique, on peut calculer les temps moyens de reconstruction et de communication en divisant les temps totaux par le nombre d'opérateurs utilisés (égal à $2N_c + 1$). On obtient des temps moyens constants pour chaque méthode quelque soit le nombre de cycles. Cela permet d'évaluer le temps de reconstruction des méthodes algébriques sur chaque machine pour N_c cycles, en multipliant le temps obtenu pour un cycle par le rapport $\frac{2N_c+1}{3}$.



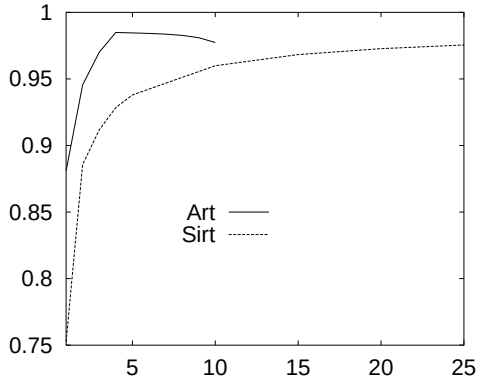
(a) (b) (c)



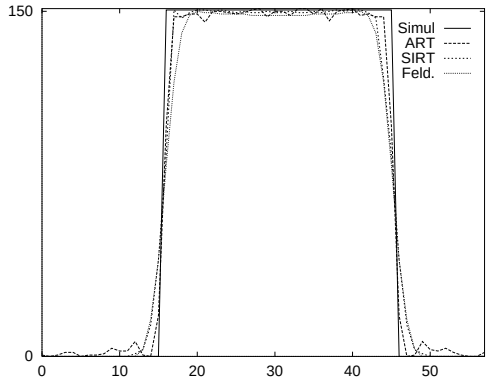
(d) Profil suivant le nombre de cycles pour la méthode ART



(e) Profil suivant le nombre de cycles pour la méthode SIRT



(f) Evolution du coefficient de corrélation pour les méthode ART et SIRT



(g) Comparaison des profils des trois méthodes

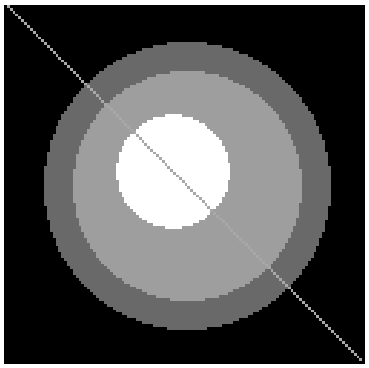
FIG. 61 - *Coupe médiane de la sphère initiale (a) et des sphères reconstruites par la méthode ART (b) et SIRT (c), Profils suivant le nombre de cycles de ART(d) et SIRT(e), Etude de la convergence de ART et SIRT (f), Comparaison des profils des trois méthodes(g)*

Sphères imbriquées comprises dans un volume de taille 128^3

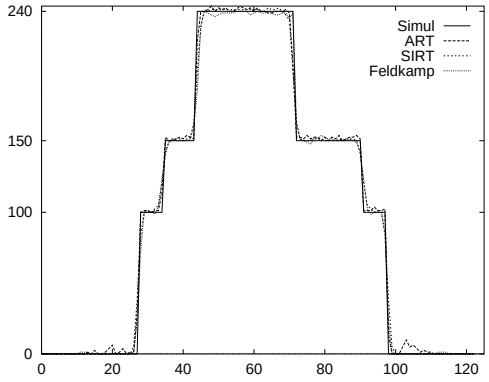
Nous présentons ici la reconstruction par les méthodes ART et SIRT des sphères imbriquées à partir du même jeu de données que celui reconstruit par la méthode de Feldkamp. Les deux séries de coupes (Z=0 et Z=-13) permettent de visualiser l'ensemble des sphères reconstruites par les deux méthodes (figures 62 et 63). Nous visualisons les profils associés à chaque coupe pour la méthode ART (profils 62(e) et 63(e)) et pour la méthode SIRT (profils 62(f) et 63(f)).

On peut remarquer que l'algorithme ART reconstruit avec une bonne qualité ces sphères imbriquées en comparaison de la méthode SIRT. Les comparaisons des profils des trois méthodes (figures 62(b) et 63(b)) montrent bien que la méthode ART est meilleure que la méthode de Feldkamp. En évaluant les critères de reconstruction (figure 64), nous retrouvons ces résultats. On peut noter que pour cet exemple l'algorithme ART converge toujours après 5-6 cycles contrairement à l'exemple précédent. Pour l'algorithme SIRT, on converge moins vite après cinq itérations.

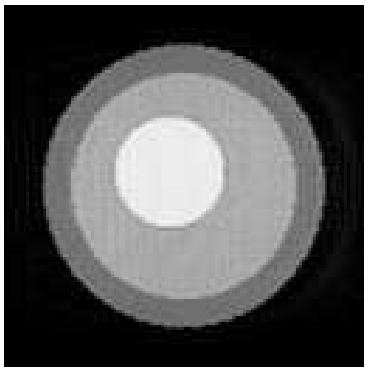
Le tableau 16 présente les temps de reconstruction de ces sphères imbriquées en fonction de la méthode utilisée. Nous avons effectué ces mesures sur les versions optimisées de nos algorithmes. Bien que l'algorithme ART converge plus vite en 9 cycles, il met deux fois plus de temps que l'algorithme SIRT. L'algorithme de Feldkamp, obtient une bonne qualité de reconstruction en un temps minimal. Le problème se pose alors de choisir une méthode en fonction des résultats qu'elle obtient au niveau des temps de reconstruction comme au niveau de la qualité de l'image 3D reconstruite.



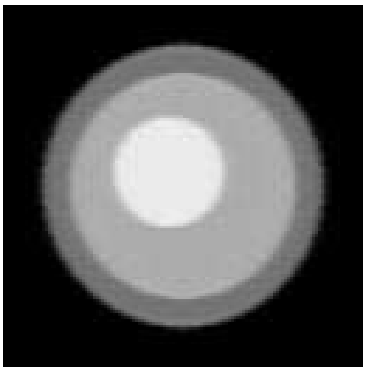
(a)



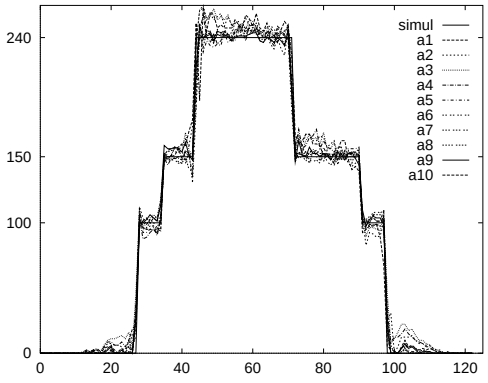
(b)



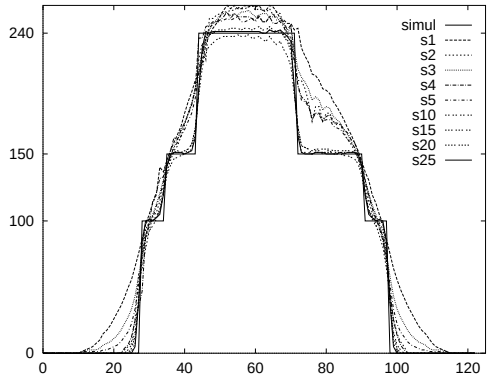
(c)



(d)

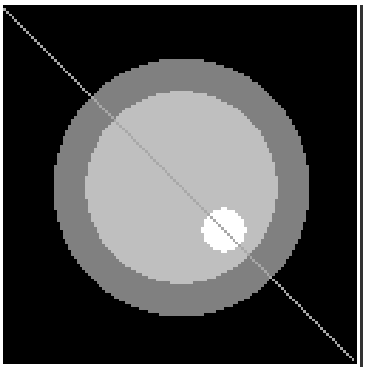


(e)

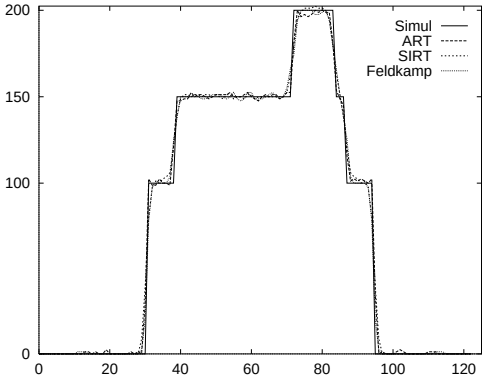


(f)

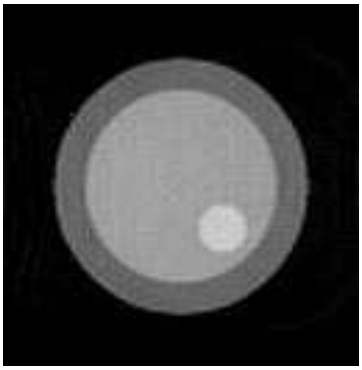
FIG. 62 - *Coupes dans le plan Z=0 de l'image initiale (a) et des images reconstruites par ART(c) et SIRT(d), profils de ART (e), de SIRT (f) et des trois méthodes(b)*



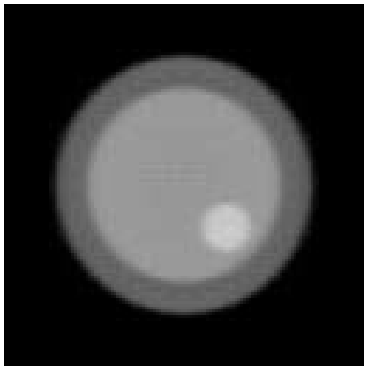
(a)



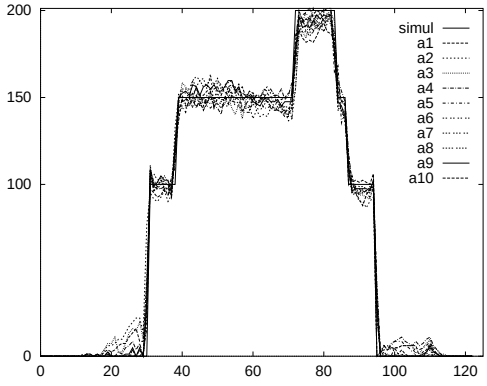
(b)



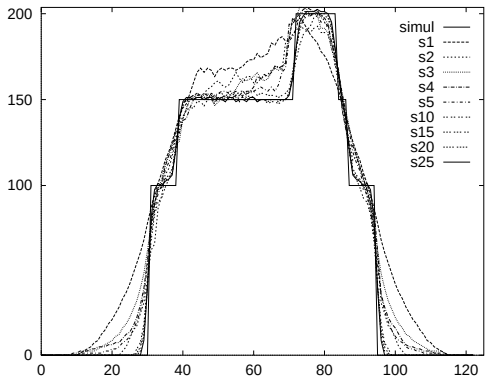
(c)



(d)



(e)



(f)

FIG. 63 - *Coupes dans le plan Z=-13 de l'image initiale (a) et des images reconstruites par ART(c) et SIRT(d), profils de ART (e), de SIRT (f) et des trois méthodes(b)*

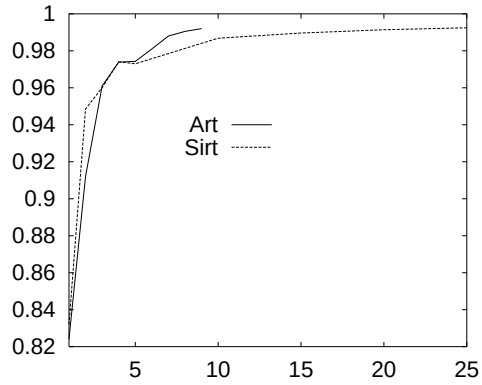


FIG. 64 - *Coefficient de corrélation de ART et SIRT pour les boules imbriquées*

	Cycles	1	2	3	4	5	6	7	8	9
Art	Temps Total	317	529	735	957	1165	1391	1609	1215	2061
	Temps Com	104	175	242	318	383	465	539	631	701
	Cycles	1	2	3	4	5	10	15	20	25
Sirt	Temps Total	237	407	574	733	916	1736	2567	3383	4220
	Temps Com	27	48	68	88	111	212	311	385	479
Feldkamp	Temps Total	110								
	Temps Com	22								

TAB. 16 - *Comparaison des temps de reconstruction des sphères imbriquées par les trois méthodes (sec)*

6.5 Conclusion

Dans ce chapitre, nous nous sommes intéressés à la parallélisation des méthodes algébriques. Nous avons choisi deux méthodes: la méthode Art par blocs et la méthode SIRT. En effet elles ont l'avantage de pouvoir s'exprimer comme un ensemble de sous-systèmes qui peuvent être résolus séparément. Grâce à cette structure par blocs, nous avons pu les formuler à partir des opérateurs de base, la projection et la rétroprojection, qui ont été définis dans le chapitre 4.

La parallélisation de ces méthodes est basée sur l'utilisation des opérateurs de base parallèle. Nous avons montré que les schémas algorithmiques de ces méthodes étant différents, leur parallélisation met en œuvre des schémas de communication différents:

- Les communications intrinsèques de la méthode ART sont des communications globales. Le schéma de communication est donc un schéma en étoile. La méthode ART est développée à partir d'opérateurs parallélisés suivant une approche globale. Le calcul des matrices de poids est cependant parallélisé suivant une approche locale.

- Les communications de la méthode SIRT sont essentiellement des commucations entre deux processeurs. Le schéma de communication choisi est l'anneau. La méthode SIRT est parallélisée à partir d'opérateurs parallélisés avec des opérateurs suivant une approche locale. Le calcul du poids W est parallélisé comme le calcul des matrices de l'algorithme ART.

Les implémentations de ces méthodes ont été optimisées en remplacant les opérateurs de base parallèles par des versions optimisées des opérateurs parallèles. La méthode ART optimisée met en œuvre des opérations de réduction sur un arbre binaire. La méthode SIRT optimisée utilise des techniques de recouvrement des communications par des calculs.

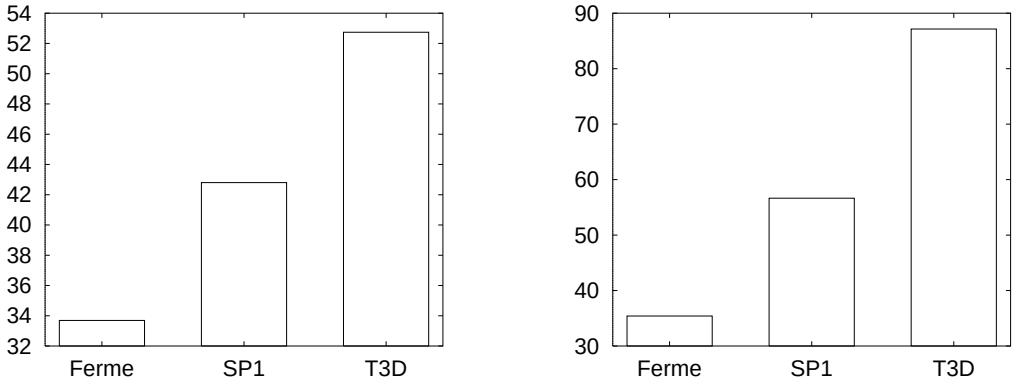
Ces algorithmes ont été implémentés dans leur version initiale et dans leur version optimisée sur trois machines parallèles: la ferme de 16 processeurs Alpha, le SP1 d'IBM composé de 32 processeurs RS6000 et le Cray T3D composé de 128 processeurs Alpha. Le choix de ces machines a été déterminé par leurs performances obtenues avec la méthode de Feldkamp et par leur bibliothèque de communication commune: PVM.

Nous testons nos algorithmes sur un cycle car nous avons montré que le temps de reconstruction et de communication sur plusieurs cycles était constant si on divise chaque temps par le nombre d'opérateurs parallèles mis en œuvre $(2N_c + 1)$. L'implémentation optimisée de ART permet de réduire sensiblement les communications en raison de l'implémentation du "pvm-reduce" qui n'est pas très efficace. Pour des problèmes de grande taille, on obtient des bonnes efficacités car la part du temps de communication est très petite. Le gain de performances obtenu par la version optimisée de SIRT est très faible en raison de l'efficacité de la version initiale (95%).

On retrouve ici des résultats de performances similaires à ceux des implémentations de Feldkamp. Il est intéressant de noter qu'en moyenne la méthode SIRT parallèle est plus rapide que la méthode ART pour un même nombre de cycles.

Au niveau des reconstructions des images tests, l'algorithme ART converge plus rapidement que l'algorithme SIRT. En comparant les résultats des trois méthodes, on obtient une meilleure qualité d'image avec la méthode ART suivant le critère utilisé. Cependant cet algorithme nécessite un temps de calcul très important pour obtenir un tel résultat.

Pour conclure, nous présentons les accélérations relatives calculées d'après les performances obtenues sur notre machine de référence (figure 65). On peut remarquer que l'on obtient les meilleures accélérations relatives avec les implémentations parallèles de SIRT, et ce gain de performances augmente en fonction du nombre de processeurs: les implémentation sur le T3D (avec seulement 64 processeurs) sont les plus rapides. Cependant, si on compare ces résultats avec ceux de la méthode de Feldkamp, on s'apercoit que cette dernière méthode est la plus efficace. Toutefois, ces résultats ont été calculés sur une reconstruction d'un image de taille 64^3 pour ART et SIRT et sur une image de taille 128^3 pour Feldkamp, en raison du temps rédhibitoire estimé de ART et SIRT sur notre machine de référence.



(a) Accélération relative de ART (b) Accélération relative de SIRT

FIG. 65 - *Accélération relative des implémentations de ART et SIRT*

Chapitre 7

Application à des données réelles et discussion

7.1 Introduction

Nous abordons avec ce chapitre l'application de nos méthodes de recontruction à des données expérimentales. Nous présenterons le système d'acquisition en géométrie conique: le Morphomètre. Les données expérimentales sont des acquisitions d'une main. Nous montrerons des vues de cette main reconstruite par nos méthodes implémentées sur le Cray T3D.

Nous discuterons ensuite des différents aspects de cette étude. Nous montrerons l'apport du parallélisme aux problèmes de reconstruction. Nous comparerons notre méthodologie avec celles présentées dans la littérature. Enfin d'un point de vue théorique, nous établirons de quelle manière l'adéquation des méthodes de reconstructions 3D au parallélisme doit prendre en compte la qualité de l'image, les temps de reconstruction et le prix à payer pour obtenir un tel résultat.

7.2 Reconstruction d'une main

7.2.1 Principe du Morphomètre

Cadre du projet de recherche

Le Morphomètre est issu d'un projet de recherche [Mor94], qui associe deux régions (Bretagne et Rhônes-Alpes) et de nombreux partenaires du public et du privé: Ministère de la recherche, Ministère de la santé, CHUR de Rennes, Hospices civils de Lyon, CEA-LETI de Grenoble, CNRS, Université de Lyon 1, Université Joseph Fourier de Grenoble, CERIUM et General Electric Medical Systems-Europe.

Ce projet d'un coût total de 76 MF, a deux objectifs principaux:

- L'étude, le développement et la réalisation de deux maquettes d'un nouvel équipement permettant d'acquérir et de visualiser des données volumiques. Elles ont été réalisées par le CEA-LETI en association avec General Electric Medical Systems-Europe.

- L'évaluation et l'optimisation dans le cadre clinique (CHU de Rennes et Hospices civils de Lyon). Les deux maquettes ont été installées dans les années 93-94.

Présentation du Morphomètre

Ce dispositif est construit autour d'un appareil d'angiographie, sur lequel sont embarquées deux chaînes d'acquisition d'image (figure 66). Chaque chaîne est composée d'un ensemble de détection et d'un ensemble d'émission:

- L'ensemble de détection est un détecteur de 40 cm formé d'un écran luminescent, d'un tube intensificateur d'images (amplificateur de brillance) et d'une caméra vidéo.

- L'ensemble d'émission est formé d'un tube à rayons X qui émet un faisceau conique traversant le volume examiné.

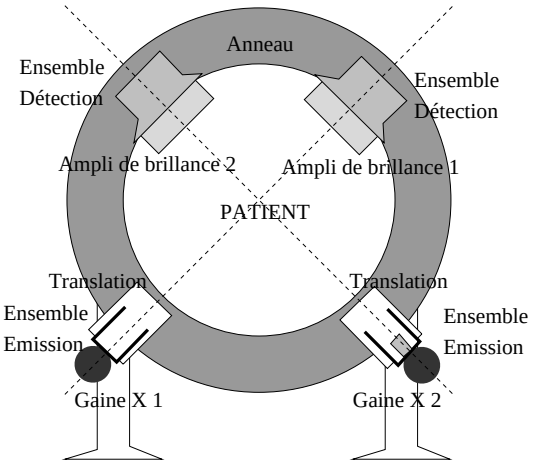


FIG. 66 - *Description du Morphomètre*

Le patient s'allonge sur un support transparent aux rayons X. Autour de lui, les acquisitions sont effectuées au rythme de 25 images par seconde au cours de la rotation, sur l'anneau, des deux chaînes d'acquisition (figure 67).

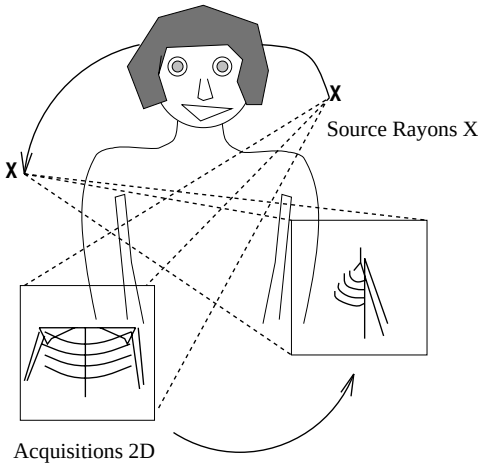


FIG. 67 - *Rotation d'une chaîne d'acquisition autour du patient*

La vitesse de rotation peut varier de 4 sec/tour à 20 sec/tour. Le tableau 17 donne les principales caractéristiques du Morphomètre. Le champ 2D définit la taille du détecteur suivant la résolution recherchée. Le champ 3D définit la taille du volume reconstruit et donc la taille des voxels.

Champs 2D (cm)	35.5	30	23
Pixel 512 (mm)	0.71	0.59	0.43
Champs 3D (cm)	27	23	17
Voxel 512 (mm)	0.53	0.44	0.33
Voxel 256 (mm)	1.06	0.88	0.66
Nombre de projections	64	128	256
Durée de rotation/tour (sec)	5.12	10.24	20.48

TAB. 17 - *Caractéristiques du Morphomètre*

Les applications cliniques du Morphomètre sont définies en fonction de trois protocoles d'acquisitions:

Angiographie standard Ce protocole regroupe les acquisitions de neurologie, les acquisitions vasculaires abdominales et thoraciques. On injecte un produit de contraste pour permettre la visualisation des organes (carotides, aorte, rein, foie, artère pulmonaire).

Cardiaque On effectue des acquisitions du cœur ou des grosses artères en réglant les prises de vue sur le rythme cardiaque.

Contrastes naturels Ce protocole permet de visualiser des parties du squelette (membres, crâne...). Il a été développé pour la traumatologie, pour l'orthodontie, et pour le positionnement des prothèses. Nos images de la main sont issues de ce protocole.

Pour chaque protocole, on effectue des acquisitions sur chaque chaîne. Pour le protocole d'angiographie, les acquisitions se répartissent en deux séquences d'acquisitions: la séquence objet et la séquence référence qui permet de visualiser le patient sans la dose injectée. Par prétraitement de ces deux séquences, on obtient une séquence soustraite contenant les images d'atténuation qui sont ensuite traitées (figure 68).

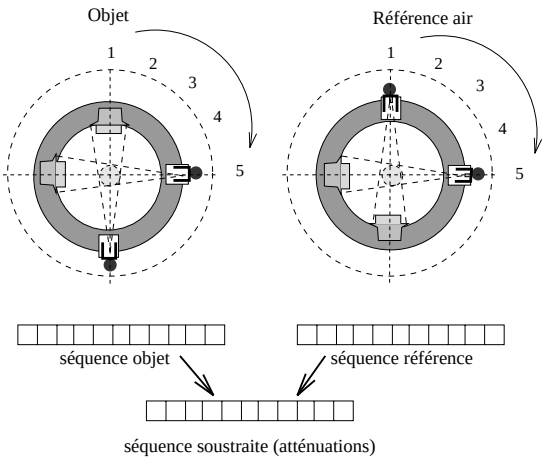


FIG. 68 - *Protocole d'acquisition*

Un système de traitement est associé à ce système d'acquisition: il est constitué d'un système de pilotage du Morphomètre, de machines de stockage et de gestion des données acquises et d'un système de reconstruction basé sur une machine parallèle une Alliant à 8 processeurs.

7.2.2 Acquisitions

Nos données initiales sont 256 acquisitions de taille 512^2 d'une main. En raison du temps de reconstruction d'un volume de taille 512^3 à partir de 256 acquisitions (3019 secondes sur le T3D) et l'espace mémoire nécessaire pour stocker le volume reconstruit (128 Mbytes en 256 niveaux de gris), nous avons créé des jeux de données pour la reconstruction de volumes plus petits. Nous construisons à partir d'un jeu de données de N acquisitions de taille N^2, un jeu de données de $\frac{N}{2}$ acquisitions de taille $\frac{N^2}{2^2}$ pour reconstruire un volume de taille $\frac{N^3}{2^3}$. Dans ce but, on prend une acquisition sur deux (sauf pour passer de $N = 512$ à $N = 256$), et on réduit les images. La réduction des images peut être effectuée par deux méthodes:

- Réduction par moyenne: chaque pixel des images de taille $\frac{N^2}{2^2}$ est calculé à partir de la moyenne de quatre pixels des images de taille N^2.

- Réduction Gaussienne: on calcule la valeur du pixel à partir d'une grille de 25 pixels à laquelle est associée une pyramide de réduction gaussienne 5×5 (figure 69).

Nous utilisons la deuxième méthode pour créer des jeux de données de 256 acquisitions de taille 256^2 et de 128 acquisitions de taille 128^2. La figure 70 présente des images d'acquisition de la main de taille 256^2.

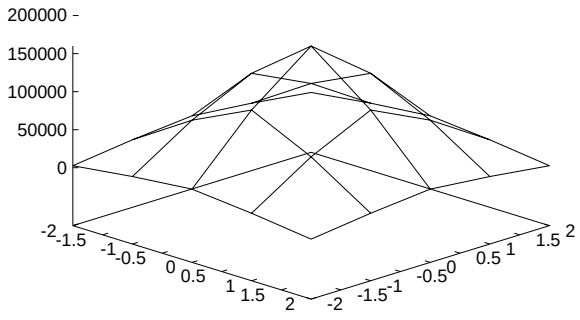
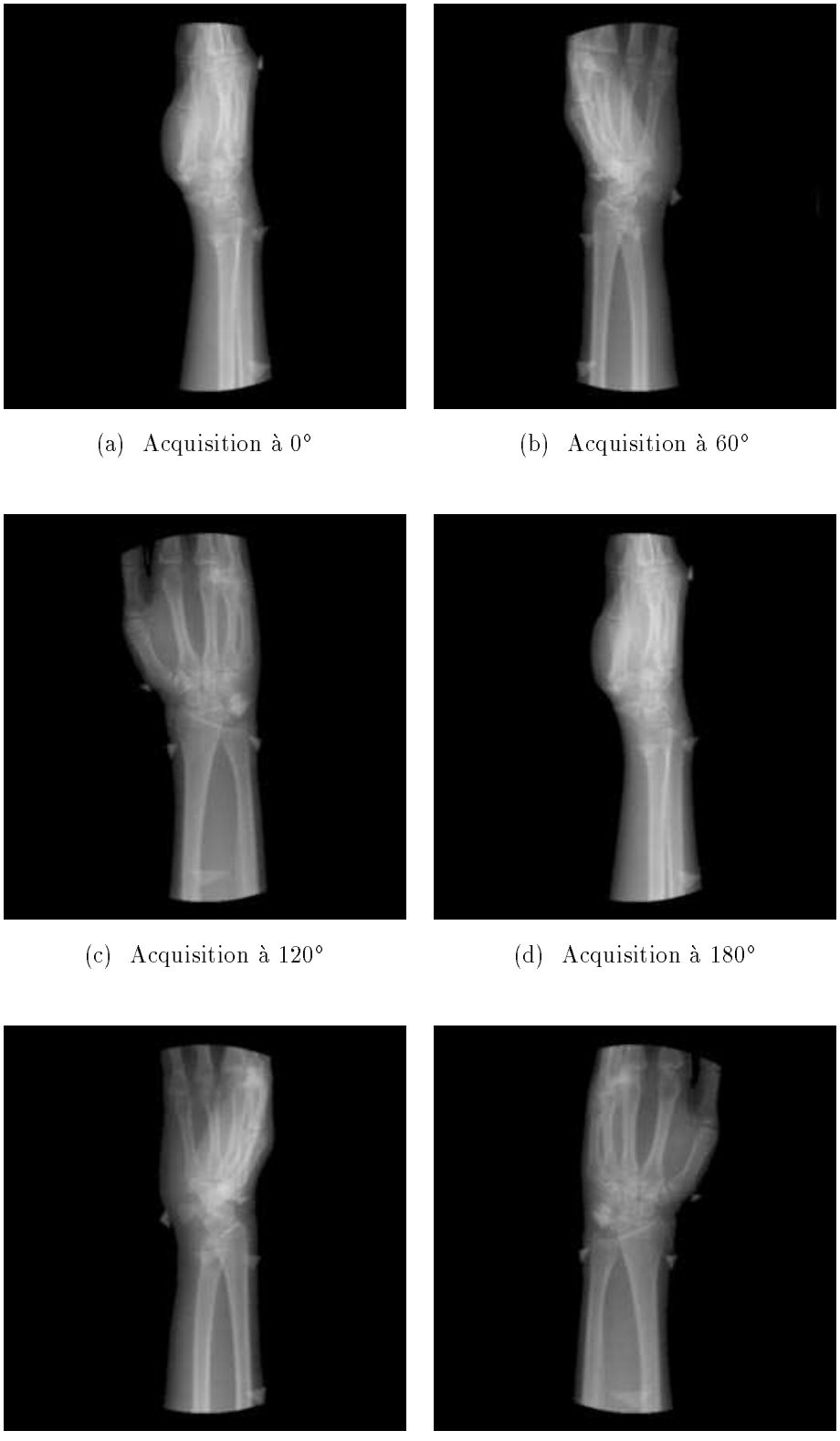


FIG. 69 - *Réduction Gaussienne*

7.2.3 Mise en oeuvre des méthodes de reconstruction

Nous avons dû adapter nos programmes à la géométrie d'acquisition. En fait, l'adaptation se situe au niveau du calcul de l'adresse de projection de chaque voxel sur chaque plan d'acquisition. Si on considère le couple (U, V) comme étant les coordonnées de la projection de chaque voxel, la nouvelle adresse de projection a pour coordonnées le couple $(-V, U)$.

Nous avons reconstruit la main par les trois méthodes (Felkamp, ART et SIRT) pour un volume de taille 128^3 (figure 71). Pour la méthode de Felkamp, on utilise un filtre de Shepp et Logan. Pour les méthodes algébriques, on reconstruit le volume en un cycle car le bruit dégrade fortement les images pour un nombre de cycles plus grand. Nous présentons deux coupes du volume reconstruit par la méthode de Feldkamp et la méthode SIRT. La première est un plan de coupe XZ pour Y=110 et la deuxième est un plan de coupe YZ pour X=75. Nous obtenons de très bons résultats avec l'algorithme de Feldkamp. La figure 72 présente des vues de la main reconstruite pour un volume 256^3. Nous avons utilisé le logiciel edit3D qui calcule les vues d'un volume 3D par "lancer de rayon".

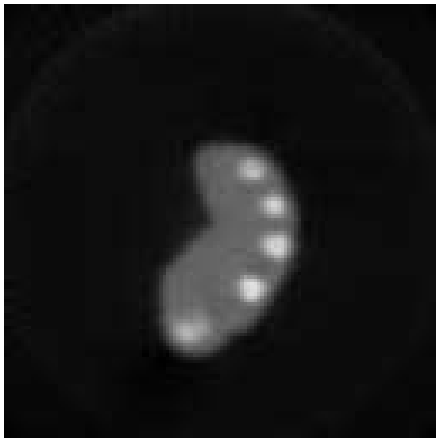


(a) Acquisition à 0° (b) Acquisition à 60°

(c) Acquisition à 120° (d) Acquisition à 180°

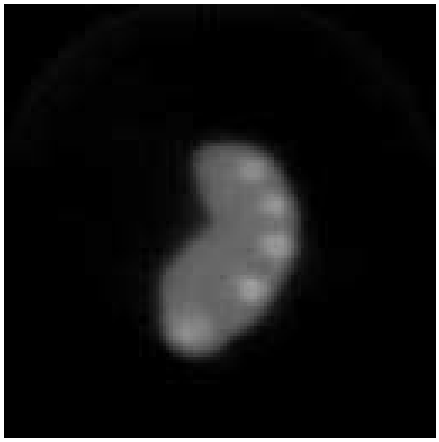
(e) Acquisition à 240° (f) Acquisition à 300°

FIG. 70 - *Images d'acquisition de la main* (256^2)

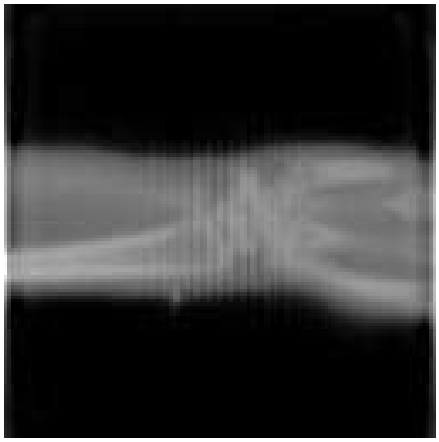


(a) Feldkamp: Coupe XZ (Y=110)

(b) Feldkamp: Coupe YZ (X=75)

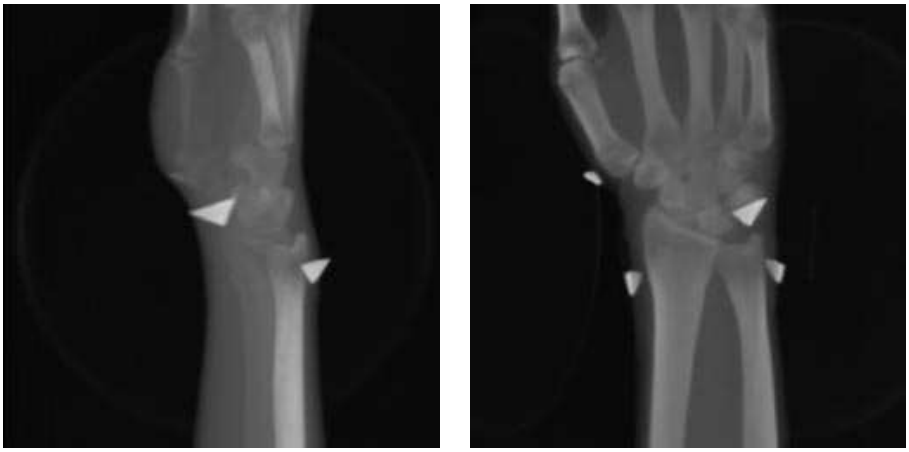


(c) SIRT: Coupe XZ (Y=110)

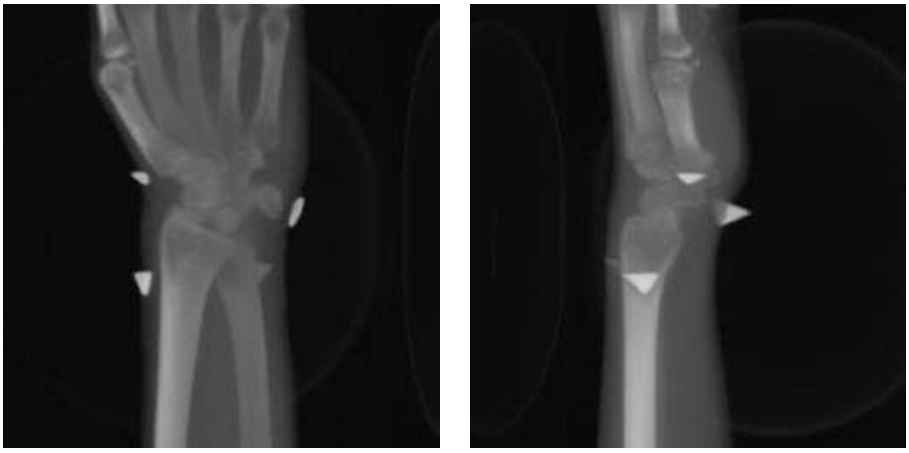


(d) SIRT: Coupe YZ (X=75)

FIG. 71 - *Coupes de la main reconstruite (volume* 128^3*)*



(a) Vue à 0° (b) Vue à 60°



(c) Vue à 120° (d) Vue à 180°

(e) Vue à 240° (f) Vue à 300°

FIG. 72 - *Vues de la main reconstruite par Feldkamp (volume 256^3)*

7.2.4 Conclusion sur la reconstruction de données expérimentales

Nous avons montré que nos algorithmes permettent de reconstruire des images 3D à partir de données expérimentales. On peut noter que la méthode de Felkamp obtient les meilleurs résultats au niveau de la qualité de l'image 3D reconstruite.

Au niveau des temps de reconstruction, le Morphomètre utilise une machine MIMD: une Alliant à 8 processeurs pour ces reconstructions d'images 3D. Le tableau 18 permet de comparer les temps de reconstruction obtenus sur cette machine (méthode ART, méthode de RADON [Gra87]), et les temps obtenus sur le Cray T3D pour nos trois méthodes. Les temps présentés sur le T3D correspondent à des exécutions sur 128 processeurs. Ce tableau montre bien le gain de performance que nous obtenons sur le Cray T3D.

Résolution		Alliant		Cray T3D		
Image 3D	Détecteurs	ART	RADON	ART	SIRT	Feldkamp
256	256	180	40	23.8	15.33	5
512	256		420			50

TAB. 18 - *Comparaison des performances du Morphomètre [Mor94] et du Cray T3D (minutes)*

7.3 Discussion

Notre étude a permis de définir une méthodologie pour la parallélisation des méthodes de reconstruction. Nous resituons ce travail pour constater l'apport du parallélisme au problème de reconstruction 3D et pour le comparer aux autres travaux. Nous essayons de dégager les points importants de cette étude qu'elle apporte, au vu des résultats obtenus au niveau de la reconstruction d'images 3D (données simulées et expérimentales), et au niveau des performances de nos implémentations.

7.3.1 Apport du parallélisme

En considérant les résultats obtenus sur les machines parallèles, nous pouvons constater que les approches parallèles permettent de diminuer les temps de reconstruction de façon considérable. Nous rappelons que la complexité de l'algorithme de

Feldkamp est égale à $m.N^3$ où m est le nombre d'acquisitions et N^3 la taille du volume reconstruit. En comparaison avec l'étude de Jacquet[Jac88] qui reconstruit un volume de taille 256^3 à partir de 512 acquisitions (256^2) en 44 heures sur un Sun4-260, nous mettons 5 minutes pour la reconstruction d'un volume de taille identique avec 256 acquisitions. Nous avons reconstruit sur le T3D une image 3D de taille 512^3 à partir de 256 acquisitions de taille 512^2. Pour le calcul des reconstructions, les pixels (respectivement les voxels) composant les acquisitions 2D (respectivement le volume 3D) sont codés en "float" sur 4 bytes, ce qui représente un volume total de données (acquisitions 2D et volume 3D) de 807 Mbytes. En raison de la taille mémoire requise, ce type de problème ne peut être résolu sur une machine classique.

L'apport du parallélisme se situe donc au niveau des performances obtenues et offre la possibilité de résoudre des problèmes de grande taille grâce notamment à la puissance et la mémoire de ces machines. Nous avons mis aussi en évidence que l'efficacité de nos implémentations augmente avec la taille du problème à résoudre en raison de la diminution de la part des communications dans le temps total d'exécution. Cependant, il faut adapter le nombre de processeurs à la taille du problème pour obtenir une bonne efficacité.

Au niveau des machines parallèles utilisées, cette étude nous amène à faire deux remarques. Nous avons montré que les machines SIMD à grain fin ne semblaient pas bien adaptées aux problèmes de reconstruction 3D, en raison de la taille mémoire associée à chaque processeur. De plus, le nombre de processeurs étant important, la répartition homogène des données multiplie les communications, ce qui pénalise les performances. Les machines MIMD obtiennent de bonnes performances. Toutefois, si on compare leur ratio $\frac{\text{Performances}}{\text{Coût}}$, on s'aperçoit que le réseau de stations et la ferme de processeurs permettent des solutions parallèles à faible coût. L'utilisation de machines massivement parallèles (Paragon, SP1 et T3D) n'est pas encore vulgarisée en raison de leurs coûts.

7.3.2 Méthode utilisée

Nous nous sommes basés sur une analyse qu'on peut qualifier de descendante pour paralléliser les méthodes de reconstruction. Cette analyse à permis de décrire

les méthodes en terme d'opérateurs de base. L'effort de parallélisation se situe alors au niveau de la parallélisation de ces opérateurs de base. Nous avons décrit nos algorithmes pour une implémentation sur une machine parallèle abstraite. Ce paradigme nous permet de ne pas prendre en compte l'architecture de nos machines cibles. De plus, la plupart de nos implémentations utilisent la librairie de communication PVM qui s'identifie à ce paradigme. Cette librairie nous assure la portabilité de nos algorithmes, et nous permet de comparer les performances obtenues pour chaque méthode sur nos machines cibles.

Les approches, décrites dans le chapitre 4, sont pour la plupart dépendantes de l'architecture de la machine, notamment celles utilisant des machines dédiées ou des machines vectorielles. Mc Carty [MM91] et Miller [MB93] proposent des approches efficaces sur les machines SIMD car ils reconstruisent des images 2D et 3D en géométrie parallèle. Dans ce cas, les opérateurs de base tirent profit de l'architecture SIMD. Sur les machines MIMD, de nombreux travaux mettent en œuvre des techniques de minimisation des communications et de régulation de charge. Nous avons utilisé les mêmes concepts pour optimiser nos implémentations.

L'originalité de notre méthodologie est de construire les méthodes de reconstruction à partir d'une palette d'opérateurs de base parallélisés suivant deux approches (globale, locale). Chaque opérateur est décliné suivant plusieurs versions:

- Pour l'approche locale, nous avons développé des versions synchrone, asynchone et asynchrone avec partitionnement adaptatif.
- Pour l'approche globale, nous avons développé des versions utilisant une opération de réduction basée sur un schéma de communication en étoile et en anneau.

Nous avons ainsi créé une bibliothèque d'opérateurs de base parallélisés.

7.3.3 Choix de la méthode de reconstruction

Face à un problème de reconstruction 3D, le problème du choix de la méthode parallélisée reste un problème ouvert. Nos expérimentations nous ont permis d'évaluer les performances des implémentations et la qualité des images reconstruites. Ainsi, nous avons montré que pour un cas idéal, la méthode ART obtenait une meilleure

reconstruction. Toutefois, le temps de reconstruction pour obtenir un tel résultat est prohibitif, si on compare ces résultats avec ceux de la méthode de Feldkamp. De plus, la méthode ART est une méthode peu efficace en comparaison avec les performances des méthodes de Feldkamp et SIRT. La question se pose donc d'évaluer la qualité de reconstruction et le temps de reconstruction que l'on désire obtenir, pour définir la méthode la plus adaptée. Notre étude donne quelques éléments de réponse à cette question.

Cependant, lors de la reconstruction à partir de données expérimentales, nous avons montré que la méthode de Feldkamp obtenait des meilleurs résultats par rapport aux méthode algébriques. Cela semble indiquer qu'il est difficile de définir la meilleure méthode. Toutefois, il est important de préciser que les résultats obtenus par les méthodes algébriques peuvent être améliorés en choisissant un paramètre de relaxation optimal. Celui-ci permettrait d'accélérer la convergence de ces méthodes vers une bonne solution.

7.4 Conclusion

Nous avons présenté dans ce chapitre des reconstructions d'images 3D à partir de données expérimentales. Les résultats au niveau de la qualité de la reconstruction comme au niveau des performances valident nos approches parallèles.

L'apport du parallélisme pour les problèmes de reconstruction a été montré. Notre méthode nous a conduit à développer une bibliothèque d'opérateurs parallèles qui sont à la base d'un grand nombre de méthodes de reconstruction. Nous avons évoqué le choix d'une bonne méthode adaptée aux données et à la machine parallèle cible.

Chapitre 8

Conclusion générale et perspectives

Cette étude se situe dans le contexte de l'évolution des technologies liées à l'imagerie médicale 3D. Nous avons démontré à travers cette étude l'adéquation du parallélisme aux problèmes de reconstruction 3D en tomographie X.

Pour en fixer le cadre, nous avons rappelé les fondements de la tomographie assistée par ordinateur et présenté les machines parallèles. Afin de situer notre méthodologie, nous nous sommes intéressés aux différents travaux proposant une approche parallèle pour résoudre un problème de reconstruction.

Notre méthodologie définit une analyse descendante des problèmes de reconstructions 3D. Elle est basée sur la parallélisation des opérateurs de base (projection et rétroprojection), suivant deux approches (locale, globale) sur une machine parallèle abstraite. Pour chaque approche, nous avons mis en place des techniques de minimisation des communications (recouvrement calcul/communication, communication sur un arbre binaire) et de régulation de charge (partitionnement adaptatif). L'ensemble de ces opérateurs parallèles forme une bibliothèque permettant une conception plus rapide des méthodes parallèles de reconstruction.

A partir de cette bibliothèque, trois méthodes parallèles de reconstruction ont été élaborées: la méthode de Felkamp, la méthode ART par blocs et la méthode SIRT.

Les nombreuses expérimentations réalisées sur différentes machines parallèles (Maspar, Réseau de stations Sun, Ferme de processeurs Alpha, Paragon d'Intel, IBM SP1 et Cray T3D) ont permis d'établir l'adéquation du parallélisme aux problèmes de reconstruction.

Pour améliorer davantage les performances et la qualité de nos reconstructions, on pourra s'intéresser à de nouvelles techniques complémentaires:

- La réduction du volume reconstruit dans deux cas de figures:
  - l'utilisation d'un à priori sur le volume permettrait de limiter le volume de reconstruction,
  - il serait possible d'effectuer après chaque cycle des méthodes algébriques un seuillage pour cerner au mieux la zone de reconstruction.

  Le volume ainsi réduit n'étant pas réparti de façon homogène, on pourrait réguler la charge des processeurs soit avec un partitionnement adaptatif, soit avec une redistribution élastique des données [MR91].

- L'implémentation de méthodes plus rapides. Il existe un bon nombre de méthodes algébriques qui convergent plus rapidement grâce à des techniques de préconditionnement. Le problème est de savoir si ces méthodes peuvent être parallélisées suivant notre méthodologie.

- La mise en place de nouvelles techniques de recouvrement des communications par les calculs. Elles sont basées sur les caractéristiques de chaque machine en établissant le rapport entre le temps de calcul élémentaire et le temps de communication pour trouver la taille optimale des envois.

En outre, il peut être souhaitable d'établir une liste non exhaustive des méthodes implémentées en parallèle explicitant la taille du problème, la machine utilisée (nombre de processeurs et type de processeur) ainsi que les performances obtenues. A partir d'un certain nombre d'expérimentations, cette liste permettra à un utilisateur de choisir la machine parallèle la mieux adaptée à sa méthode, et réciproquement de choisir une méthode en fonction de la machine cible.

Notre étude est basée sur une géométrie d'acquisition conique directement applicable à la reconstruction d'images 3D à partir d'acquisitions du Morphomètre. Les méthodes algébriques sont adaptables à toutes sortes de géométries. Il est donc envisageable de définir de nouveaux champs d'application de nos méthodes de reconstruction 3D.

Pour conclure sur notre étude, les temps de reconstruction pour résoudre des problèmes de grande taille (512^3), que nous avons présentés, sont très acceptables (50 minutes). Ces performances obtenues sont donc très encourageantes, et au vu de la qualité de la reconstruction de nos données expérimentales (une main), on peut promettre un bel avenir aux méthodes parallèles de reconstruction.

Annexe A

Mise en oeuvre des Méthodes

Pour comparer nos approches lors de l'implémentation de nos méthodes de reconstruction parallèle, nous avons développé un ensemble de programmes similaire sur chaque machine utilisée. Les machines que nous avons choisies pour notre étude sont des machines parallèles appartenant pour la plupart à des laboratoires ou des organismes de la région Rhône-Alpes. Pour créer des fichiers de tests et visualiser les résultats, nous avons créé une interface graphique appelée TomoTool.

A.1 Organisation modulaire des programmes

Pour tester ces algorithmes et les méthodes développées à partir de ces opérateurs de base, il a été nécessaire de définir la géométrie, de créer des données et de visualiser les résultats. Dans cette optique, nous avons conçu un ensemble de programmes permettant d'implémenter nos méthodes. Notre stratégie, pour nos implémentations, a été de définir une librairie de fonctions communes à nos programmes et une interface permettant de créer les fichiers de paramètres, de visualiser les résultats et d'exécuter certaines applications. Les programmes ayant une fonctionnalité commune ont été regroupés dans un module. Chacun des programmes a d'abord été testé en séquentiel puis a été parallélisé sur les machines cibles. Nous ne détaillerons pas l'implémentation des programmes parallèles permettant de créer des jeux de données et de visualiser les résultats.

Création de jeux de données Nous avons développé deux programmes permettant de créer des jeux de tests pour nos algorithmes. Ces programmes utilisent un fichier de paramètres permettant de définir des fantômes constitués

d'objets simples (sphère, ellipsoide, parallélépipède). Le programme *CreerCube* définit le fantôme d'une fonction discrétisée sur un volume. Ce volume est utilisé pour tester le programme de projection et permet de mesurer la qualité d'une rétroprojection. Le programme *ProjCalc* calcule un ensemble de projections d'un fantôme. Ces projections sont calculées à partir des formules analytiques de projection d'objets simples. Ce programme génère des projections non bruitées ce qui permet de tester les algorithmes de rétroprojection.

Test des opérateurs Nous avons mis au point des programmes servant uniquement à tester les opérateurs. Le programme *Project* permet de projeter un volume sur un ensemble de plans de détection Il est basé sur l'algorithme 1 de projection en séquentiel et a été développé en parallèle par une approche locale et par une approche globale. Le programme *Retro* effectue l'opération de rétroprojection d'un ensemble de projections sur un volume suivant l'algorithme 2 de rétroprojection. Nous l'avons aussi testé en parallèle en utilisant une approche locale et une approche globale. Pour tester les différents filtres utilisés par notre méthode analytique, nous avons défini un programme de filtrage: *Filtrage*.

Méthodes implémentées Nous avons implémenté trois méthodes à partir des opérateurs de projection et de rétroprojection. Le programme *BackProject* est basé sur la méthode de Feldkamp, le programme *ART* est basé sur la méthode ART par blocs et le programme *SIRT* est basé sur la méthode SIRT. Nous détaillerons leur implémentation séquentielle et parallèle dans les chapitres 5 et 6.

Visualisation des résultats Pour visualiser les résultats obtenus par les programmes de test et ceux des méthodes implémentées, il est nécessaire de transformer les données codées en réel, en niveaux de gris. Le programme *TransCub* transforme un fichier de points réels contenant une image 3D, en une image 3D en niveaux de gris. Le programme *VisuProj* transforme un ensemble d'images de projection codées en réels en un ensemble d'images en niveaux de gris. Des logiciels standards comme XV ou AVS nous servent à visualiser ces images en niveaux de gris. Nous avons conçu des programmes pour comparer nos résultats dans notre logiciel TomoTool.

A.2 TomoTool

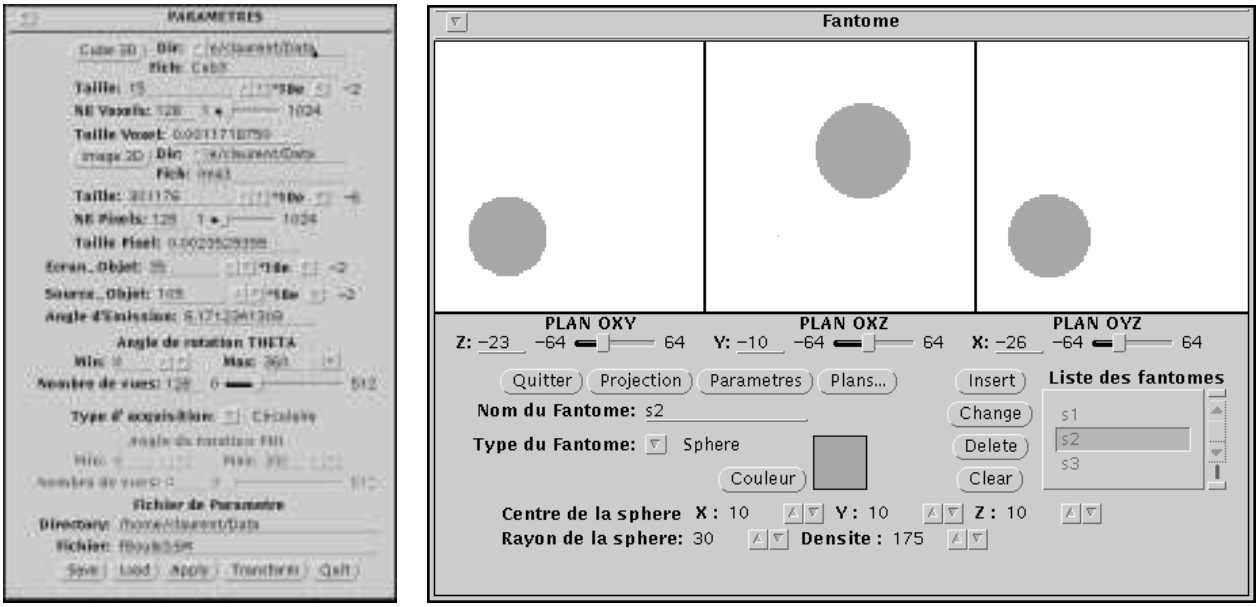
Pour tester nos algorithmes, nous avons défini un logiciel TomoTool composé de trois modules: un module de saisie des paramètres, un module d'exécution et un module de visualisation. Ces modules nous permettent de créer des données communes à chaque programme sur chaque machine, d'exécuter les programmes séquentiels et parallèles sur notre réseau de stations et de visualiser les résultats. Ce logiciel est écrit avec la librairie graphique Xview sous Openwindow.

A.2.1 Module de saisie des paramètres

Ce module sert à définir les paramètres de la géométrie d'acquisition (figure 73(a)). Il permet aussi de créer des fantômes composés de sphères, de parallélogrammes, et d'ellipsoïdes(figure 73(b)). Un fantôme est défini par les caractéristiques de chaque objet le composant (centre de la sphère, rayon, densité...). Les caractéristiques de chaque objet sont fournies aux programmes qui créent les jeux de données: *Creer-Cube* et *ProjCalc*. Les fichiers de paramètres sont adaptés en fonction de chaque machine utilisée car les normes de codage des entiers et des réels diffèrent souvent entre les machines. Cependant les caractères ASCII étant codés avec la même norme sur la plupart des machines, c'est pourquoi nous utilisons des images en niveaux de gris codés de 0 à 255 pour visualiser nos données.

A.2.2 Module d'exécution

Le module d'exécution est l'interface des différents programmes implémentés sur la machine séquentielle de référence: un SUN4 (figure 74). Il permet aussi d'exécuter les programmes parallèles écrits en PVM sur un réseau de stations de travail grâce à une gestion des machines formant le réseau. Chaque programme est exécuté avec un certain nombre de paramètres liés à la géométrie d'acquisition: fichier de paramètres, ou à l'exécution: nombre de processeurs, types de filtres utilisés, modes de communication, calculs des performances ... .



(a) Saisie des paramètres (b) Définition des fantômes

FIG. 73 - *Module de saisie*

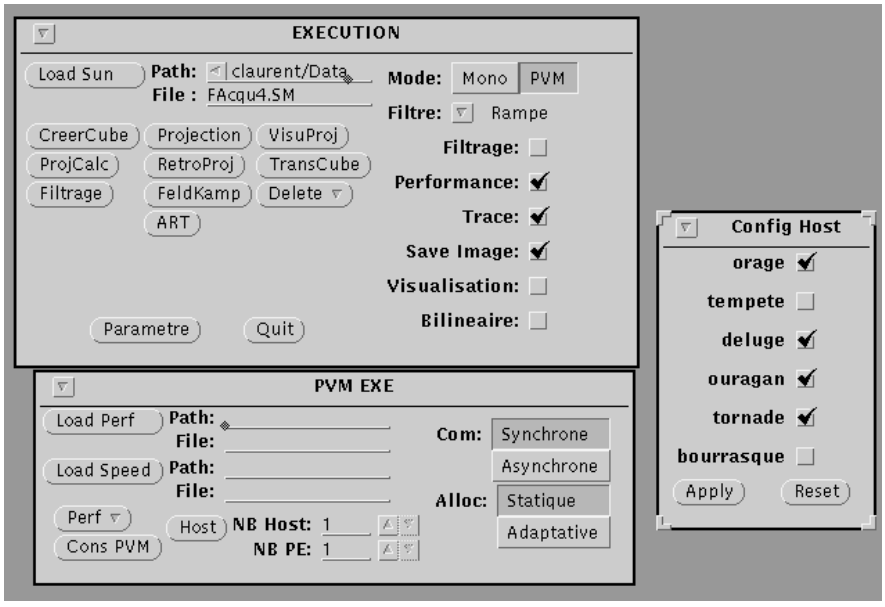


FIG. 74 - *Module d'exécution*

A.2.3 Module de visualisation

Pour visualiser nos résultats et les comparer, nous avons conçu une interface permettant de visualiser les images de projection et les coupes d'image 3D (figure75). Ce module permet d'effectuer des opérations sur ces images (somme, différence,...) et de visualiser des profils (2D, 3D) et des histogrammes par l'intermédiaire du logiciel Gnuplot. Ainsi l'image "view1" et l'image "view2" sont des coupes d'image 3D. L'image "view2"correspond à une coupe de deux boules simulées et l'image "view1" correspond à une coupe de l'image reconstruite par la méthode de Feldkamp. L'image "view3" correspond à l'image de différence quadratique des deux images précédentes. La fenêtre Gnuplot permet de visualiser le profil des deux images suivant la ligne tracée sur chaque image. Pour visualiser les images 3D, nous utilisons le logiciel Edit3D mis au point au laboratoire par Parazza et Usson [PHU93].

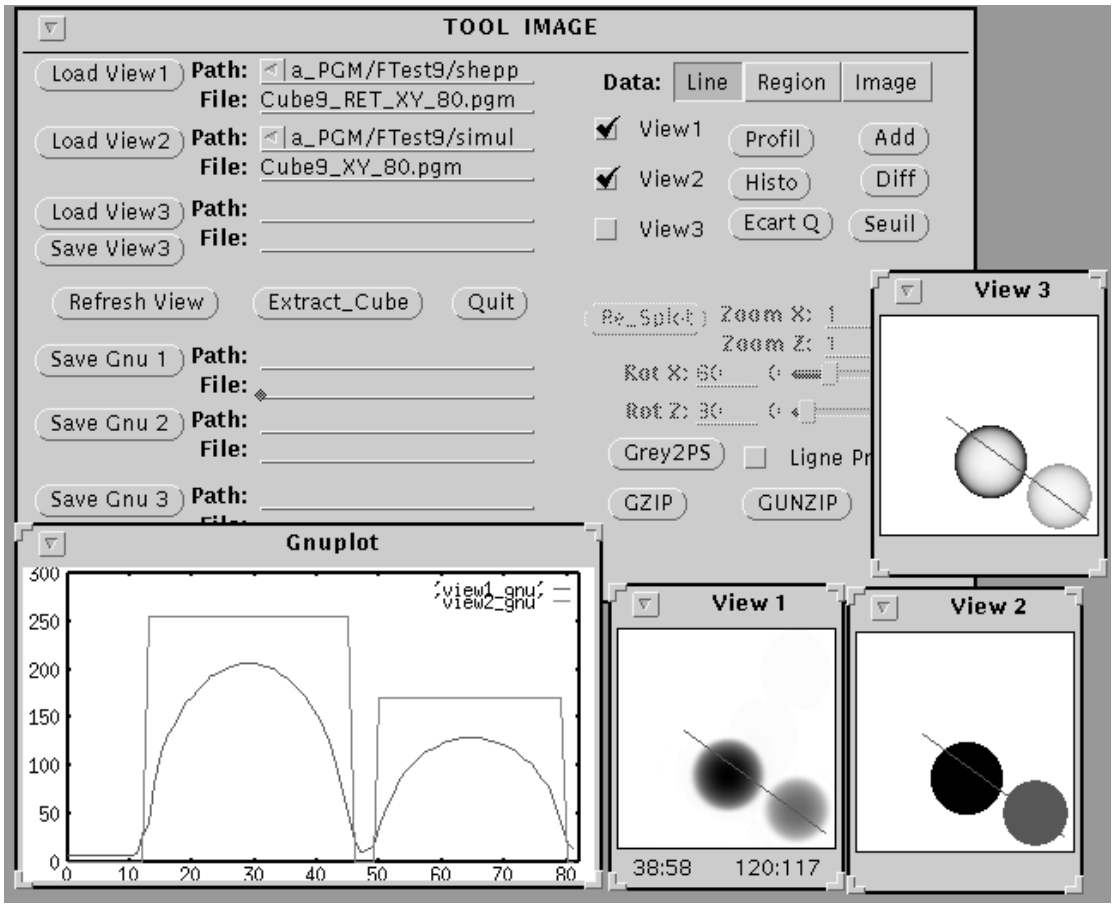


FIG. 75 - *Module de visualisation*

Bibliographie

[AMH93] M. S. Atkins, D. Murray, and R. Harrop. Use of transputers in a 3-D Positron Emission Tomograph. *IEEE Transactions on Medical Imaging*, 12(2):173–181, 1993.

[AP88] F. André and J.-L. Pazat. Le placement de tâches sur des architectures parallèles. *Technique et Science Informatiques*, 7(4):387–400, 1988.

[APA94] LMC-IMAG and LGI-IMAG. *Rapport APACHE*, 1994.

[BBG90] S. Barresi, D. Bollini, and A. Del Guerra. Use of a transputer for fast 3-D image reconstruction in 3-D PET. *IEEE Transactions on Nuclear Science*, 37(2):812–816, 1990.

[BCD95] V. Bouchard, P. Cinquin, and L. Desbat. First Compton scatter correction in SPECT using PVM. In *Fully Three-Dimensional Image Reconstruction in Radiology and Nuclear Medecine*, pages 109–112, 1995.

[BdA89] A. R. Borges and A. M. B. Ferarri de Almeida. A multimicroprocessor architecture for image reconstruction in CT. In P. M. Dew, editor, *Parallel Processing for Computer Vision and Display*, pages 489–496, 1989.

[BDJ+94] A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam. PVM3 User's Guide and Reference Manual. Technical report, Oak Ridge National Laboratory, 1994.

[BGH79] T. F. Budinger, G. T. Gullberg, and R. H. Huesman. Emission computed tomography. *Image reconstruction from projection: Implementation and Applications*, pages 147–246, 1979.

[BKTS95] Th. Beyer, P. Kinahan, D. Townsend, and D. Sashin. Attenuation correction for a combined 3D PET/CT scanner. In *Fully Three-Dimensional Image Reconstruction in Radiology and Nuclear Medecine*, pages 55–60, 1995.

[BL92] R. Butler and E. Lusk. User's guide to the P4 programming system. Technical report, Argonne National Laboratory, 1992.

[Bla90] T. Blank. The MASPAR MP-1 Architecture. In *Proceedings of the IEEE*, pages 20–24, 1990.

[BLPvdG93] M. Barnett, R. Littlefield, D.G. Payne, and R. van de Geijn. On the efficiency of global combine algorithms for 2-D meshes with wormhole routing. 1993.

[CA89] C. Caquineau and J.-L. Amans. Propostion d'architectures pour la reconstruction d'images et de volumes à partir de projections. In *vide*, pages 00–00, 1989.

[CD92] J. Cook and F. Dickin. Real-Time Quantitative Tomographic Image Reconstruction. Technical report, University of Bergen, Norway, 1992.

[CD93] C. Calvin and F. Desprez. Minimizing communication overhead using pipeling for multi-dimensional FFT on distributed memory machine. In *PARCO*, 1993.

[CDPW92] J. Choi, J. Dongarra, R. Pozo, and D. Walker. ScaLAPACK: A Scalable Linera Algebra Library for Distributed Memory Concurrent Computers. Technical Report LAPACK WN 55, University of Tennessee, 1992.

[Cen83] Y. Censor. Finite Series-Expansion Reconstruction Methods. *Proceedings of the IEEE*, 71(3):409–419, 1983.

[CL94] C. M. Chen and S.-Y. Lee. On Parallelizing the EM Algorithm for 3-D PET Image Reconstruction. *IEEE Transactions on Parallel and Distributed Sytems*, 5(8):860–873, 1994.

[CLC90] C. M. Chen, S.-Y. Lee, and Z. H. Cho. A Parallel Implementation of 3-D CT Image Reconstruction on Hypercube Multiprocessor. *IEEE Transactions on Nuclear Science*, 37(3):1333–1346, 1990.

[CLC91] C. M. Chen, S.-Y. Lee, and Z. H. Cho. Parallelisation of EM Algorithm for 3-D PET Image Reconstruction. *IEEE Transactions on Medical Imaging*, 10(4):513–522, 1991.

[CLM93] H.-P. Charles, J.-J. Li, and S. Miguet. 3D image processing on distributed memory parallel computers. In *SPIE*, volume 1905, pages 379–390, 1993.

[CM-87] Thinking Machine Corporation. *CM-2, Technical summary*, 1987.

[CM-93] Thinking Machine Corporation. *CM-5, Technical summary*, 1993.

[Col94] L. Colombet. *Parallélisation d'applications pour des réseaux de processeurs homogènes ou hétérogènes.* PhD thesis, Institut National Polytechnique de Grenoble, 1994.

[Cra94] Inc. Cray Research. *Cray T3D, Technical summary*, 1994.

[CS-93] Meiko. *The CS-2 system, product description*, 1993.

[CT65] C.W. Cooley and J.W. Tuckey. An algorithm for the machine calculation of complex fourier series. *Math. Comput.*, 19:297–301, 1965.

[DC94] M. Defrise and R. Clack. A cone-beam reconstruction algorithm using shift-variant filtering and cone-beam backprojection. *IEEE Transactions on Medical Imaging*, 13:186–195, 1994.

[Des95] L. Desbat. Efficient sampling in 3D tomography: parallel schemes. In *Fully Three-Dimensional Image Reconstruction in Radiology and Nuclear Medecine*, pages 281–287, 1995.

[DFR79] R. V. Denton, B. Friedlander, and A. J. Rockmore. Direct three dimensional image reconstrcution from divergent rays. *IEEE Transactions on Nuclear Science*, 26:4695–4703, 1979.

[DKT95] M. Defrise, P. Kinahan, and D. Townsend. A new rebinning algorithm for 3D-PET: principle, implementation and performance. In *Fully Three-Dimensional Image Reconstruction in Radiology and Nuclear Medecine*, pages 235–240, 1995.

[DLR95] L. Desbat, C. Laurent, and S. Rouault. Parallel reconstruction in tomography: work in progress. In *International Workshop Parallel Imaging and Application*, pages 147–156, 1995.

[DMBS88] J.J Dongarra, C.B Moler, J.R. Bunch, and G.W. Stewart. Linpack users guide. *SIAM Review*, 1988.

[DT93] F. Desprez and B. Tourancheau. LOCCS low overhead communication and computation subroutines. In *High Performance Computing and Networking conference*, 1993.

[ED95] J. Eriksson and E. Danielsson. Helical scan 3D reconstruction using linogram method. In *Fully Three-Dimensional Image Reconstruction in Radiology and Nuclear Medecine*, pages 287–290, 1995.

[EH81] P.P.B Eggermont and G.T. Herman. Iterative algorithms for large partitioned linear systems with applications to image reconstruction. *Linear algebra and its application*, 40:37–67, 1981.

[Elf80] T. Elfing. Block-iterative methods for consistent and inconsistent linear equations. *Numer. Math.*, 35:1–12, 1980.

[Eti94] D. Etiemble. L'évolution des machines parallèles et massivement parallèles. *La Lettre du Transputer*, 6(4):55–60, 1994.

[Exe94] Convex Exemplar Salable Parallel Processing. *Exemplar, System Overview*, 1994.

[FDK84] L. A. Feldkamp, L.C. Davis, and J.W. Kress. Practical Cone-Beam Algorithm. *J. Opt. Soc. Am.*, 1(6):612–619, 1984.

[FJT95] E.C. Frey, Z.W. JU, and B.M.W. Tsui. Reducing reconstruction time for 3D iterative reconstruction scatter compensation. In *Fully Three-Dimensional Image Reconstruction in Radiology and Nuclear Medecine*, pages 201–206, 1995.

[Fly79] M. J. Flynn. Some computer organizationn and their effectiveness. *IEEE Transactions on Computer*, pages 948–960, 1979.

[GBH70] R. Gordon, R. Bender, and G. T. Herman. Algebraic reconstruction technique (ART) for three dimensional electron microscopy and X-ray photography. *IEEE Transactions on Nuclear Science*, 29:471–481, 1970.

[GHPW90] G. A. Geist, M.T. Hetah, B. W. Peyton, and P. H. Worley. PICL: a protable instrumented communication library. Technical report, Oak Ridge National Laboratory, 1990.

[GP93] L. Garnero and F. Peyrin. Methodes de Reconstruction 3D en Tomographie X. Technical report, GDR TDSI CNRS, France, May 1993. Rapport de synthèse 93-01.

[Gra87] P. Grangeat. *Analyse d'un système d'imagerie 3D par reconstruction à partir de radiographies X en géométrie conique.* PhD thesis, ENST Paris, 1987.

[GS89] C. Guerrini and G. Spaletta. An image reconstruction algorithm in tomography: a version for the CRAY X-MP vector computer. *Computers and Graphics*, 13:367–372, 1989.

[Gus88] J. L. Gustafson. Reevaluting amdahl's law. *Communication of the ACM*, 31, 1988.

[GZ95] G. T. Gullber and G. L. Zeng. Three-dimensional SPECT reconstruction of combined cone-beam and fan-beam data acquired using a three-detector SPECT system. In *Fully Three-Dimensional Image Reconstruction in Radiology and Nuclear Medecine*, pages 329–332, 1995.

[HBM85] R. Hartz, D. Bristow, and N. Mullani. A real-time tofpet slice-backproject engine employing dual Am29116 microprocessors. *IEEE Transactions on Nuclear Science*, 32:839–841, 1985.

[Her80] G. T. Herman. *Image Reconstruction from Projections.* Academic Press, New York, 1980.

[Hou72] G. M. Hounsfield. A method and apparatus for examination of a body radiation such as x or gamma. *Pat. Spec.*, 128(3915), 1972.

[HPF93] HPF Forum. *High Performance Fortran language specification, version 1.0*, 1993.

[Jac88] Jacquet. Reconstruction d'image 3D par l'algorithme éventail généralisé. Mémoire pour l'obtention du diplome d'ingénieur, CNAM, Grenoble, 1988.

[JBC88] W.F. Jones, L.G. Byars, and M.E. Casey. Positron emission tomographic images and expectation maximization: a VLSI architecture for multiple iterations per second. *IEEE Transactions on Nuclear Science*, 35:620–624, 1988.

[JBC90] W.F. Jones, L.G. Byars, and M.E. Casey. Design of a super fast three-dimensional projection system for postron emission tomography. *IEEE Transactions on Nuclear Science*, 37:800–804, 1990.

[Kau87] L. Kaufman. Implementing and accelerating the EM algorithm for Positron Emission Tomography. *IEEE Transactions on Medical Imaging*, 1:37–51, 1987.

[KSR92] Kendall Square Research Corporation. *KSR, Technical summary*, 1992.

[Lat88] D. Lattard. *Architecture massivement parallèle: un réseau de cellules intégré pour la reconstruction d'images*. PhD thesis, Institut National Polytechnique de Grenoble, 1988.

[LCPC95] C. Laurent, C. Calvin, F. Peyrin, and J.-M. Chassery. Efficient Implementation of Parallel Image Reconstruction Algorithms for 3D X-RAY Tomography. In *PARCO 95*, Gent(Belgique), September 1995.

[Lew83] R. M. Lewitt. Reconstruction Algorithms: Transform Methods. *Proceedings of the IEEE*, 71(3):390–408, 1983.

[LPC94a] C. Laurent, F. Peyrin, and J.-M. Chassery. Calculs Parallèles de Projection en Tomographie 3D. In *Actes des 6ièmes rencontres sur le parallélisme(RenPar6)*, page 324, Lyon(France), Juin 1994.

[LPC94b] C. Laurent, F. Peyrin, and J.-M. Chassery. PVM Implementations of 3D Reconstruction Algorithms. In *First European PVM Users Group Meeting*, pages 220–224, Roma(Italia), October 1994.

[LPC95a] C. Laurent, F. Peyrin, and J.-M. Chassery. Comparaison of Parallel Reconstruction Algorithms for 3D X-RAY Tomography on MIMD computers. In *High Performance Computing Symposium 95-HPCS 95*, pages 255–266, Montréal(Canada), July 1995.

[LPC95b] C. Laurent, F. Peyrin, and J.-M. Chassery. Evaluation of Parallel Approaches for 3D Cone-Beam Reconstruction. In *Fully Three-Dimensional Image Reconstruction in Radiology and Nuclear Medecine*, pages 311–316, Aix-Les-Bains(France), July 1995.

[LPC95c] C. Laurent, F. Peyrin, and J.-M. Chassery. Méthodologie pour la Parallélisation d'Algorithmes de Reconstruction 3D: Application à la Tomogaphie X. In *GRETSI 95*, pages 965–968, Juan-les-Pins(France), Septembre 1995.

[LPC95d] C. Laurent, F. Peyrin, and J.-M. Chassery. Parallélisation de Méthodes de Reconstruction 3D par des Approches Locales et Globales. In *Actes des 7ièmes rencontres sur le parallélisme-RenPar7*, pages 158–162, Mons(Belgique), Mai 1995.

[MB93] M. I. Miller and C. S. Butler. 3-D maximun *a posteriori* Estimation for Single Photon Emission Computed Tomography on Massively-Parallel Computers. *IEEE Transactions on Medical Imaging*, 12(3):560–565, 1993.

[MB95] S. Matej and J.A. Browne. Performance of a fast maximun likehood algorithm for fully 3D PET reconstruction. In *Fully Three-Dimensional Image Reconstruction in Radiology and Nuclear Medecine*, pages 247–254, 1995.

[MM91] A. W. McCarty and M. I. Miller. Maximun Likelihood SPECT in Clinical Computation Times Using Mesh-Connected Parallel Computers. *IEEE Transactions on Medical Imaging*, 10(3):426–436, 1991.

[Mor94] Hospices Civils de Lyon. *Le projet Morphomètre 3D*, 1994.

[MPI93] MPI Forum. *Message Passing Interface,Document for a standard message-passing interface*, 1993.

[MR91] S. Miguet and Y. Robert. Elastic load-balancing for image processing algorithm. In *Parallel computation, First ACPC conference*, pages 438–451, 1991.

[NC78] O. Nalcioglu and Z. H. Cho. reconstruction of 3D objects from cone beam projections. *Proceedings of the IEEE*, 66(11):1584–1585, 1978.

[Oa95] R.J Ott and all. 3D scatter characterictics for large area PET cameras. In *Fully Three-Dimensional Image Reconstruction in Radiology and Nuclear Medecine*, pages 121–126, 1995.

[OEM95] G.J. O'Keefe, G.F. Egan, and I. Morisson. 3D PET image reconstruction using an intel i860-array supercomputer. In *Fully Three-Dimensional Image Reconstruction in Radiology and Nuclear Medecine*, pages 359–360, 1995.

[Par91] Intel Corporation. *Paragon, Product Overview*, 1991.

[Pey82] F. Peyrin. *Méthodes de reconstruction et de visualisation d'images tridimensionnelles en tomographie num rique par rayon X*. PhD thesis, Université Claude Bernard LYON 1, 1982.

[Pey90] F. Peyrin. *Méthodes de Reconstruction d'Images 3D à partir de Projections Coniques de Rayons X*. PhD thesis, Université Claude Bernard LYON 1, 1990.

[P.G72] P.Gilbert. Iterative methods for the three dimensional reconstruction of an object from projections. *J. Theor. Biol*, 36:105–117, 1972.

[PHU93] F. Parazza, C. Humbert, and Y. Usson. Method for 3D volumetric analysys of intranuclear fluorescence distribution in confocal microscopy. *Computerized Medical Imaging and Graphics*, 17:189–200, 1993.

[PRA88] F. Peyrin, R.Goutte, and M. Amiel. Description d'un algorithme de reconstruction de volume à partir de projections coniques par déconvolution tri-dimensionnelle. *Traitement du Signal*, 5(4):223–233, 1988.

[Rad17] J. Radon. Über die bestimmung von funktionen durch ihre integralwerte langs gewisser mannigfaltigkeiten. *Ber. Verb. Saechs. Akad. Wiss., Leipzig, Math. Phys. Kl.*, 69:262–277, 1917.

[RKR+80] E.L. Ritman, J.H. Kinsey, R.A. Robb, L.D. Harris, and B.K. Gilbert. Physics and technical considerations in the design of the DSR. *J. of Roenhgenology*, 134:369–374, 1980.

[RL71] G.N. Ramachadran and A.V. Lakshminnarayanan. Three dimensional reconstrcution from radiographs and electron micrographs: application of convolutions instead of fourier transforms. *Proc. Nat. Acad. Sci. USA*, 68:2236–2240, 1971.

[RPR94] K. Rajan, L. M. Patnaik, and J. Ramakrishna. High-speed computation of the EM algorithm for PET image reconstruction. *IEEE Transactions on Nuclear Science*, 41:1721–1728, 1994.

[RT94] J. L. Roch and D. Trystram. Méthodologie pour la Programmation Efficace d'Applications Parallèles. *La Lettre du Transputer*, 6(4):133–138, 1994.

[SFa92] D. Saint-Félix and all. A New system for 3D computerized X-RAY angiography: first in vivo result. In *Annual International Conference of the IEEE Engineering in Medecine and Biology Society*, pages 2051–2052, 1992.

[SL74] L. A. Shepp and B. F. Logan. The fourier reconstruction of a head section. *IEEE Transactions on Nuclear Science*, 21:904–909, 1974.

[Smi95] M. F. Smith. An accelerated algorithm for 3D SPECT image reconstruction with generalized matrix inverses. In *Fully Three-Dimensional Image Reconstruction in Radiology and Nuclear Medecine*, pages 187–194, 1995.

[SP193] IBM. *IBM SP1,product description*, 1993.

[SS89] Y. Saad and M. H. Schultz. Data communication in parallel architecture. *Parallel Computing*, 11:131–150, 1989.

[TP81] C. J. Thompson and T. M. Peters. A fractional address accumumator for fast backprojection. *IEEE Transactions on Nuclear Science*, 4:3648–3650, 1981.

[Tuy83] H. K. Tuy. An inversion formula for cone beam reconstruction. *SIAM Journal of Applied Mathematics*, 43(3):546–552, 1983.

[WGB95] D. H. Wilson, H. Gifford, and H. Barret. Image reconstruction for the FASTSPECT imaging system using low-resolution and high-resolution detector elements. In *Fully Three-Dimensional Image Reconstruction in Radiology and Nuclear Medecine*, pages 291–298, 1995.

[WOC95] C. Wu, C. Ordonez., and C. T. Chen. FIPI: Fast 3D PET reconstrcution by fourier inversion of rebinned plane integrals. In *Fully Three-Dimensional Image Reconstruction in Radiology and Nuclear Medecine*, pages 240–246, 1995.

[Woo79] E. Wood. Applications of high temporal resolution computerized tomography to physiology and medecine. *Image reconstruction from projection: Implementation and application*, pages 247–279, 1979.

[ZBR+90] E. L. Zapata, I. Benavides, F. F. Rivera, J. D. Bruguera, and J. M. Carazo. Image Reconstruction on Hypercube Computers. In *The 3rd Symposium on the Frontiers of Massively Parallel Computation*, pages 127–133, 1990.