Iterative Methods in regularized tomographic reconstruction

Pierre Paleo

Communauté
UNIVERSITÉ Grenoble Alpes

THÈSE

Pour obtenir le grade de

DOCTEUR DE LA COMMUNAUTÉ UNIVERSITÉ GRENOBLE ALPES

Spécialité : **Signal, Image, Parole et Télécommunications**

Arrêté ministériel : 25 mai 2016

Présentée par

Pierre PALEO

Thèse dirigée par **Michel Desvignes**
et codirigée par **Alessandro Mirone**

préparée au sein du **ESRF - The European Synchrotron**
et de **l'école doctorale Électronique, Électrotechnique, Automatique et Traitement du Signal (EEATS)**

Iterative methods in regularized tomographic reconstruction

Méthodes itératives pour la reconstruction tomographique régularisée

Thèse soutenue publiquement le **13 Novembre 2017**,
devant le jury composé de :

Monsieur Michel Desvignes
Professeur, Grenoble INP, Directeur de thèse
Monsieur Alessandro Mirone
Ingénieur de Recherche, ESRF, Co-Directeur de thèse
Madame Federica Marone
Chargée de recherche, Swiss Light Source - Suisse, Rapporteur
Ali Mohammad-Djafari
Directeur de Recherche, CNRS Délégation Ile de France Sud, Rapporteur
Joost Batenburg
Professeur, Université de Leiden - Pays-bas, Président
Emmanuel Brun
Chargé de recherche, INSERM Délégation Alpes, Examinateur

Contents

Acronyms

ADMM Alternating Direction Method of Multipliers. 61, 62, 71, 73, 74, 76–78, 117, 120, 125, 127, 210, 211

ART Algebraic Reconstruction Technique. 38, 39, 41

C-P Chambolle-Pock. 62, 63, 69–71, 73, 76–80, 82–84, 109, 127, 137, 138, 210

CG Conjugate Gradient. 40, 112, 113, 122, 124–127

CS Compressive Sensing. 43–45, 47, 69, 85, 88

CSG Conjugate Subgradient. 112, 117, 118, 120, 127, 187

CT Computed Tomography. 9, 10, 38, 43, 47, 105, 135

DFT Discrete Fourier Transform. 22, 24, 120, 121, 203

DWT Discrete Wavelet Transform. 51, 52, 85, 87, 88, 90, 91, 103–105, 110, 138, 200, 206, 210

EM Expectation Maximization. 40, 41, 43, 208

ESRF European Synchrotron Radiation Facility. 10, 69, 87

FBP Filtered Back-Projection. 26–30, 33, 48, 78, 80, 82–84, 106–108, 111, 121, 163, 176, 210, 212

FFT Fast Fourier Transform. 21, 25, 78

FISTA Fast Iterative Shrinkage-Thresholding Algorithm. 60, 61, 71, 73, 76–78, 80, 82, 104, 111, 117, 118, 127, 138, 206, 210, 211

FT Fourier Transform. 25, 26, 120, 121, 197, 198, 203, 204

FW Fourier-Wavelets. 134, 139

GPU Graphical Processing Unit. 69, 79, 103, 127, 210

ISTA Iterative Shrinkage-Thresholding Algorithm. 60, 205, 206

LIP Linear Inverse Problem. 34, 35, 44, 49, 58, 120

MAP Maximum A Posteriori. 38, 41, 42, 63, 199

ML Maximum Likelihood. 36, 37, 41, 63

MSE Mean Squared Error. 74, 76, 77, 80, 109, 205

PCT Phase Contrast Tomography. 13, 14, 106, 108, 110

PDF Probability Density Function. 35–37, 198

PET positron emission tomography. 10

POCS Projection Onto Convex Sets. 38

PSF Point Spread Function. 20, 120, 184, 185

RCP Rings Correction in Polar Coordinates. 134, 139

SART Simultaneous Algebraic reconstruction Technique. 39

SIR Standard Iterative Reconstruction. 162, 174, 177

SIRT Simultaneous Iterative Reconstruction Technique. 40, 41, 43, 82, 83, 110, 122, 124, 208

SNR Signal to Noise Ratio. 33, 50, 77, 82, 106

SVD Singular Value Decomposition. 18, 19, 29, 50

SWT Stationary Wavelet Transform. 52, 103–105, 138

TV Total Variation. 48, 49, 61, 69–73, 78–82, 84, 88, 105, 109, 111, 120, 127, 137, 138, 150, 162, 210

Remerciements

Ce travail a été financé par un contrat CFR à l'ESRF, le Synchrotron européen. L'ESRF est environnement très stimulant à tous les égards, et j'ai pris plaisir à y réaliser ma thèse.

Je tiens d'abord à remercier mes deux directeurs de thèse, Michel Desvignes et Alessandro Mirone. Michel, pour ses conseils et retours sur toutes les parties de mon travail, réussissait toujours à me faire prendre du recul. J'ai aussi apprécié l'expérience dans l'enseignement qu'il a rendue possible. Travailler avec lui était agréable tant sur le plan technique qu'humain. Alessandro, qui a fait bien plus qu'apporter un soutien essentiel pendant la thèse : beaucoup des grandes idées de ce travail venaient de lui, et il arrivait à avoir une grande acuité dans toutes les phases de réalisation. Mark Kac écrivait qu'il existe deux types de génie : l'ordinaire, compréhensible par les gens remarquablement intelligents, et le "magicien" dont le mode de pensée reste hermétique pour le commun des mortels - ceux qui connaissant Alessandro savent à quelle catégorie il appartient !

Je remercie Ali Mohammad-Djafari, Federica Marone, Joost Batenburg et Emmanuel Brun d'avoir accepté d'être membres du jury de thèse. Merci à Joost, Federica et Ali pour les échanges et invitations à des séminaires. Merci à Emmanuel Brun pour avoir rendu possibles les expériences sur ID17 avec beaucoup de patience et de bienveillance.

Je remercie également les organisateurs de l'action européenne EXTREMA (COST) d'avoir mis en place des séminaires très intéressants sur tous les aspects de la tomographie. Les échanges et les discussions de points de vue variés sur un même problème ont été très enrichissants.

Je remercie particulièrement mes collègues de la *Data Analysis Unit*. Travailler parmi eux était très plaisant et j'espère retrouver un jour la même ambiance de travail. Je n'ai pas besoin de chercher un motif particulier pour les remercier, j'ai simplement beaucoup apprécié d'être avec eux tant au travail que dans les occasions extra-professionnelles. Les ~~trolldi~~ discussions philosophiques du vendredi vont me manquer. Merci à Jérôme Kieffer pour m'avoir initié au calcul GPU, à l'auto-hébergement, à l'admin-sys et tant d'autres choses. Merci à Pierre Knobel d'avoir partagé les expériences de son projet de permaculture, en particulier les "chroniques des canards". Merci à Henri Payno pour avoir initié l'idée des soirées jeux, j'espère que nous pourrons en prochainement. Merci à Claudio Ferrero qui a toujours été là pour me conseiller et me guider à l'ESRF. Merci à Valentin Valls, Thomas Vincent, Armando Sole, Christian Nemoz et Andy Götz.

Merci à Audrey, mon épouse, pour sa gentillesse et sa patience - il est vrai que je passe "trop" de temps devant l'écran, mais cette fois le serveur de mails fonctionne ! Merci à ma famille pour m'avoir accompagné et encouragé jusqu'ici.

Résumé

Au cours des dernières années, les techniques d'imagerie par tomographie se sont diversifiées pour de nombreuses applications. Cependant, des contraintes expérimentales conduisent souvent à une acquisition de données limitées, par exemple les scans rapides ou l'imagerie médicale pour laquelle la dose de rayonnement est une préoccupation majeure. L'insuffisance de données peut prendre forme d'un faible rapport signal à bruit, peu de vues, ou une gamme angulaire manquante. D'autre part, les artefacts nuisent à la qualité de reconstruction. Dans ces contextes, les techniques standard montrent leurs limitations.

Dans ce travail, nous explorons comment les méthodes de reconstruction régularisée peuvent répondre à ces défis. Ces méthodes traitent la reconstruction comme un problème inverse, et la solution est généralement calculée par une procédure d'optimisation. L'implémentation de méthodes de reconstruction régularisée implique à la fois de concevoir une régularisation appropriée, et de choisir le meilleur algorithme d'optimisation pour le problème résultant.

Du point de vue de la modélisation, nous considérons trois types de régularisations dans un cadre mathématique unifié, ainsi que leur implémentation efficace : la variation totale, les ondelettes et la reconstruction basée sur un dictionnaire. Du point de vue algorithmique, nous étudions quels algorithmes d'optimisation de l'état de l'art sont les mieux adaptés pour le problème et l'architecture parallèle cible (GPU), et nous proposons un nouvel algorithme d'optimisation avec une vitesse de convergence accrue.

Nous montrons ensuite comment les modèles régularisés de reconstruction peuvent être étendus pour prendre en compte les artefacts usuels : les artefacts en anneau et les artefacts de tomographie locale. Nous proposons notamment un nouvel algorithme quasi-exact de reconstruction en tomographie locale.

Abstract

In the last years, there have been a diversification of the tomography imaging technique for many applications. However, experimental constraints often lead to limited data - for example fast scans, or medical imaging where the radiation dose is a primary concern. The data limitation may come as a low signal to noise ratio, scarce views or a missing angle wedge. On the other hand, artefacts are detrimental to reconstruction quality. In these contexts, the standard techniques show their limitations.

In this work, we explore how regularized tomographic reconstruction methods can handle these challenges. These methods treat the problem as an inverse problem, and the solution is generally found by the means of an optimization procedure. Implementing regularized reconstruction methods entails to both designing an appropriate regularization, and choosing the best optimization algorithm for the resulting problem.

On the modelling part, we focus on three types of regularizers in an unified mathematical framework, along with their efficient implementation : Total Variation, Wavelets and dictionary-based reconstruction. On the algorithmic part, we study which state-of-the-art convex optimization algorithms are best fitted for the problem and parallel architectures (GPU), and propose a new algorithm for an increased convergence speed.

We then show how the standard regularization models can be extended to take the usual artefacts into account, namely rings and local tomography artefacts. Notably, a novel quasi-exact local tomography reconstruction method is proposed.

Definitions and notations

Typographic conventions

This manuscript follows the following typographic convention regarding the mathematical entities.

Mathematical entity	Typographic convention	Example
Scalar	Lower case Latin and Greek	for $i \in \{0, 1, 2, \ldots\}$ and $\alpha > 0$
Vector	Bold lower case	The signal $\boldsymbol{x} = [x_1, \ldots, x_n]$
Discrete operator (matrix)	Bold upper case	The inverse problem $\boldsymbol{Ax} = \boldsymbol{b}$
Continuous operator	Upper case calligraphic	The Fourier Transform $\mathcal{F}$.
Standard sets	Upper case Roman	$\mathbb{N}, \mathbb{Z}, \mathbb{R}, \mathbb{C}$
Other sets	Upper case Greek	$\Omega = \{\boldsymbol{x} \mid \quad \boldsymbol{Ax} = \boldsymbol{b}\}$

The terminology "continuous", in this context, is understood as opposed to discrete in the sense that values are indexed by a "continuum", generally a subset of $\mathbb{R}$. It does not mean that the involved functions/operators are continuous in the "regularity'" sense of the term.

Discrete functions, signals and images are represented by vectors, as elements of vector spaces of finite dimension. Continuous functions, signal and images are also represented by vectors, as elements of vector spaces of infinite dimension. Continuous functions may be denoted by bold lowercase Greek letters, for example $t \mapsto \boldsymbol{\varphi}(t)$. For clarity, the following abuse of notation is used : the variable dependency is denoted between parenthesis. For example, a continuous time-dependent signal can be denoted $\boldsymbol{x}(t)$ instead of $t \mapsto \boldsymbol{x}(t)$.

The only exceptions are functions from a vector space to a scalar field, for example *objective functions*. These functions take a vector as an input and return a scalar value. For readability, bold will be omitted. For example $f(\boldsymbol{x})$ will be written, where $\boldsymbol{x}$ is a signal, and f can be a norm for example.

Adjoint of linear operators are denoted with a star : if $\mathcal{M}$ is a linear continuous operator, $\mathcal{M}^*$ is its adjoint. In the discrete, real case (matrices), the adjoint of $\boldsymbol{M}$ is its transpose, denoted $\boldsymbol{M}^T$ for real matrices.

Transforms

Transforms are operators taking a function as an input, and returning another function as an output. The transformation of a function $\boldsymbol{f}$ to a function $\boldsymbol{g}$ by a transform $\mathcal{T}$, evaluated in the variable t_0, is denoted

$$\mathcal{T}[\boldsymbol{f}](t_0) = \boldsymbol{g}(t_0)$$

where the brackets stand for the evaluation of an operator in a function, and the parenthesis stand for the evaluation of a function in a variable.

The *one dimensional* Fourier transform of a time-dependent signal $\boldsymbol{f}(t)$ is defined as

$$\mathcal{F}_1[\boldsymbol{f}](\nu) = \int_{-\infty}^{\infty} \boldsymbol{x}(t) e^{-\boldsymbol{\jmath} 2\pi\nu t} \, \mathrm{d}t$$

where $\boldsymbol{j}$ is the imaginary unit. Generalization to superior dimensions is straightforward In this manuscript, the *two dimensional* Fourier Transform of a signal $\boldsymbol{g}(x, y)$ is of special interest :

$$\mathcal{F}_2[\boldsymbol{g}](\nu_x, \nu_y) = \iint_{-\infty}^{\infty} \boldsymbol{g}(x, y) e^{-\boldsymbol{j} 2\pi(\nu_x x + \nu_y y)} \, \mathrm{d}x \, \mathrm{d}y$$

Inner products and norms

Vector norms

The inner product between two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ is denoted $\langle \boldsymbol{x} \, , \, \boldsymbol{y} \rangle$. The inner product used in general is the Euclidean inner product, equal to $\boldsymbol{x}^T \boldsymbol{y} = \sum_i \boldsymbol{x}_i \boldsymbol{y}_i$ in the discrete case or $\int_t \boldsymbol{x}(t) \boldsymbol{y}(t) \, \mathrm{d}t$ in the continuous case.
The Euclidean norm (or ℓ_2 norm), associated to the standard inner product $\langle \cdot \, , \, \cdot \rangle$, is denoted $\|\boldsymbol{x}\|_2 = \sqrt{\langle \boldsymbol{x} \, , \, \boldsymbol{x} \rangle}$.
The norm of $\boldsymbol{x}$ with metric $\boldsymbol{R}$, where $\boldsymbol{R}$ is a positive semi-definite matrix, is defined as $\|\boldsymbol{x}\|_R = \sqrt{\langle \boldsymbol{R}\boldsymbol{x} \, , \, \boldsymbol{x} \rangle}$.
The ℓ_1 norm of $\boldsymbol{x}$ is defined as $\|\boldsymbol{x}\|_1 = \sum_i |\boldsymbol{x}_i|$.
The ℓ_∞ norm of $\boldsymbol{x}$ is defined as $\|\boldsymbol{x}\|_\infty = \max_i |x_i|$.
The $\ell_{p,q}$ norm of doubly indexed vector $\boldsymbol{x}$, is defined as $\|\boldsymbol{x}\|_{p,q} = \left[\sum_i \left(\sum_j |\boldsymbol{x}_{i,j}|^p \right)^{q/p} \right]^{1/q}$ where $\frac{1}{p} + \frac{1}{q} = 1$.
Given a norm $\|\cdot\|$, the unit ball of $\|\cdot\|$ is the set of all vectors having norm less that one:

$$B^1 = \{\boldsymbol{x}, \ \|\boldsymbol{x}\| \leq 1\}$$

Similarly, the λ ball is defined as

$$B^\lambda = \{\boldsymbol{x}, \ \|\boldsymbol{x}\| \leq \lambda\}$$

Operator norms

A norm $\|\cdot\|$ on vector space E "induces" a norm on the operators acting on elements of E. In general, the induced norm of the operator $\mathcal{M}$ is defined as

$$\|\mathcal{M}\| = \sup_{\boldsymbol{x} \neq \boldsymbol{0}} \left\{ \frac{\|\mathcal{M}\boldsymbol{x}\|}{\|\boldsymbol{x}\|} \right\}$$

The spectral norm, which is the norm induced by the ℓ_2 norm, is of special interest. It is equal to

$$\|\mathcal{M}\|_2 = \sqrt{\lambda_{\max}(\mathcal{M}^*\mathcal{M})} = \sigma_{\max}(\mathcal{M})$$

where $\lambda_{\max}$ and $\sigma_{\max}$ denote the largest eigenvalue and singular value, respectively.

Chapter 0

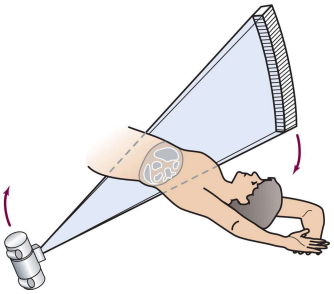
Introduction

This chapter gives a brief introduction on X-Ray tomography and gives the context and motivation of this work. It also provides a reading guide of the chapters, and a list of the main contributions.

0.1 X-Ray tomography

From their discovery of by Wilhelm Röntgen in 1895, X-Rays proved to have applications in a wide variety of scientific fields. Simply put, X-ray radiation is a light of relatively high frequency/energy with respect to the visible light. Due to its high energy, X-ray radiation has a notable penetrating property, which makes it particularly interesting for imaging applications (Figure 0.1.1).

(a)



(b)

Figure 0.1.1: (a) First radiograph realized by Wilhelm Röntgen in 1895 on the hand of his wife Anna Ludwig. Image in public domain. (b) Principle of computed tomography. In this case, moving X-ray source and detector are rotating around a patient. From the collected data, it is possible to reconstruct an image of the patient's cross-section. Image from [Mos12].

X-Ray Computed Tomography (CT) is an imaging technique which can be viewed as an extension of X-ray radiography (Figure 0.1.1.a). Instead of acquiring a single radiograph, several *projection images* are acquired at different angles around the scanned object (Figure 0.1.1.b). By combining these projections and using a numerical reconstruction method, the *interior of the scanned volume* can be rendered. The word *tomography* comes from the Greek *tomos graphein*, literally "writing slices". Instead of cutting a volume slice by slice to examine its interior – which is of limited interest in medical imaging – computed tomography aims at *numerically reconstructing* the interior of a scanned volume.

This non-destructive imaging technique finds broad applications from industrial characterization to medical imaging. The same imaging principle can be used with other

modalities (i.e replacing the X-rays photons with other particles) like positron emission tomography (PET).

0.2 Context and motivations

This work has been funded by a CFR[1] at the European Synchrotron Radiation Facility (ESRF). A synchrotron (Figure 0.2.1) is a large instrument which generates intense X-Rays for various scientific applications.

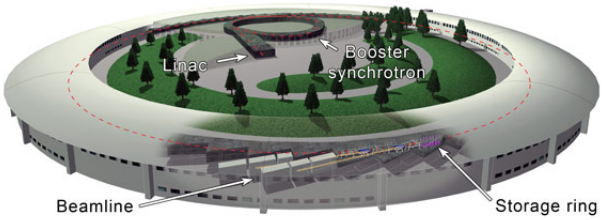


Figure 0.2.1: Principle of a synchrotron, here the ESRF. Electron are first produced by an "electron gun", and are accelerated in a linear accelerator (Linac). Then, they are accelerated to in the booster (6 GeV at ESRF), before being sent to the storage ring where they are kept to the same average energy. By passing through successive magnetic elements such as bending magnets, the electrons are radially accelerated and emit a radiation. This X-Ray radiation is collected at Beamlines built around the storage ring. Image credits: esrf.eu

At ESRF, tomography beamlines can use "hard X-rays" in an energy range from 10 keV (0.124 nm wavelength) to 250 keV (4.96 pm wavelength). In contrast to X-rays produced at laboratories, synchrotron radiation has many attractive qualities.

Regardless of the X-ray source, the traditional computed tomography process can be roughly divided in three steps: data acquisition, reconstruction and data analysis. With a simple model of the data acquisition, efficient reconstruction algorithms were derived in traditional CT. However, despite the constant progresses made in detectors and recent development of more advanced reconstruction algorithms, modern X-ray tomography comes with several challenges:

- Micro and nano-imaging requires an increase in spatial resolution, sometimes with "not enough data"
- When radiation dose is a concern, a major challenge is being able to reconstruct from highly limited data: small number of projection angles or weak signal-to-noise ratio.
- Fast tomography constraints (*in-situ* scan) often entail limited data
- Artefacts are detrimental to the reconstruction quality.
- The huge amount of data, produced by new generations of detectors, always overwhelms the computing power.

[1] Contrat Formation-Recherche

Particularly, in the case of limited data, classical reconstruction method do not yield satisfactory results.

In this work, we explore how advanced reconstruction methods, called regularized methods, can handle these challenges. Indeed, in many cases, the experimental set-up cannot be modified to acquire a better data quality; or the data is already acquired and has to be analysed/further processed. In these cases, the "numerical" reconstruction step has to catch up for the "experimental" step, in order to get the most out of the data.

This works aims at keeping the best compromise between the modelling part and the computational part, i.e between the modelling accuracy and practical usability. More precisely, the reconstruction is treated as an inverse problem and involves regularization; but ultimately the computational cost is always kept in sight. Importantly, all the methods are developed on GPU for a high speed data processing, in order to cope with the always increasing amount of data outputed by modern detectors.

0.3 Reading guide

This dissertation is split in four chapters.

- In the first chapter, we briefly introduce the continuous and numerical tomography. An emphasis is put on the distinction between the "continuous domain" – the Radon transform and its mathematical properties – and the "discrete domain" – suitable for numerical implementations. The usual tomography operators are put in the form of discrete operators, which brings the convenience of numerical linear algebra, keeping in mind that some mathematical properties – like the ability to reconstruct the infinite spectrum of an image – does not hold in the discrete case.

- In the second chapter, classical iterative reconstruction methods are presented in an unified Bayseian framework. The regularized methods are then naturally introduced both in the framework of Bayseian reconstruction and compressed sensing. Three sparsifying frames are considered: the gradient representation (Total Variation), over-complete learned dictionaries and wavelets. Finally, convex optimization tools necessary for this work are presented.

- The third chapter deals with the efficient implementation of regularized reconstruction methods for parallel-beam geometry. An extensive comparison of state-of-the-art convex optimization algorithms is carried on for the Total Variation reconstruction; The most promising one is implemented on GPU for high throughput reconstruction. Then, the parallel implementation of a discrete wavelet transform library is presented. This library aims at being used for regularized tomographic reconstruction, and in other regularized inverse problems. Finally, we present a new optimization algorithm for ℓ_2-ℓ_1 minimization, primarily dedicated to a fast convergence of dictionary-based reconstruction with combined rings artefacts removal.

- In the fourth chapter, we show how the standard regularization models can be extended to take the usual artefacts into account, namely rings and local tomography artefacts. Notably, a novel quasi-exact local tomography reconstruction method is proposed.

0.4 Main contributions

The main contribution of this work can be outlined according to the chapters.

- **Chapter 1.** An overview of the properties of the Radon Transform (in parallel geometry) is carried on with an operator formalism. This approach, seldom adopted in the literature, allows for much more concise statements and derivations of the usual properties. We also highlighting the difference between the continuum and discrete settings, and link the properties to practical implementation issues.

- **Chapter 2.** The usual reconstruction algorithms are described in a unified mathematical framework, explicitly separating the modelling from the optimization. We also review and classify sparse representation for regularized tomographic reconstruction and state-of-the-art optimization algorithms.

- **Chapter 3.** On the algorithmic side, we benchmark the state-of-the-art optimization algorithms for Total Variation regularized reconstruction (3.1). The parallel geometry is fully exploited to improve the convergence rate (3.1.4 and 3.4). We also propose a new optimization algorithm suited for the LASSO problem (3.3) and show that it outperforms the current best algorithms. On the implementation side, we implement all the algorithms and the involved operators on GPU; notably, we propose a library for a the Discrete Wavelet Transform and all its variant for regularized inverse problems (3.2).

- **Chapter 4.** We show that the proposed unified formalism can be further extended to take artefacts into account. Rings artefacts (4.1) appear as a structured noise in the sinogram and can be accounted with an additional variable in the optimization problem, leading to an iterative method which simultaneously reconstructs and corrects artefacts. We then propose a local tomography reconstruction method (4.2) aiming at correcting the low frequencies artefacts by modifying the forward model. This method is able to recover quantitativeness if a known zone constraint is satisfied.

Chapter 1

Continuous and numerical tomography

In this chapter, we review the essential properties of the "X-ray transform" modelling the acquisition of projection data. These properties are the building blocks of tomographic reconstruction methods. We adopt an operator approach, which makes the statements and derivations much more concise.

1.1 Absorption and phase-contrast tomography

1.1.1 Absorption tomography

Absorption tomography aims at measuring how an object *absorbs* (or, in the complementary way, transmits) an X-ray beam by measuring both the incoming and transmitted beam.

The transmission process can be described by the Beer's law (or Bouguer-Lambert-Beer's law) which can be derived as follows. Let us consider a *monochromatic* X-ray beam propagating in one direction measured by the coordinate x, having an intensity $I(x)$. The beam hits an object of thickness $\mathrm{d}x$ at the coordinate x. After leaving the object at coordinate $x + \mathrm{d}x$, the intensity has been attenuated in a proportion that is characteristic of the object. This attenuation is described by the *linear absorption coefficient* $\mu(x)$. The Beer's law states that the infinitesimal intensity decay is linearly linked to the incoming intensity, the object thickness and the linear attenuation coefficient: $I(x + \mathrm{d}x) = I(x) + \mathrm{d}I(x)$, where $\mathrm{d}I(x)$ is given by Equation (1.1.1).

$$\mathrm{d}I(x) = -\mu(x)I(x)\,\mathrm{d}x \tag{1.1.1}$$

Ordinary Differential Equation (1.1.1) enables to know the intensity profile $I(x)$ from arbitrary position x_0. Letting $I_0 = I(x = x_0)$, this leads to Equation (1.1.2).

$$\begin{aligned} I(x) &= I_0 \exp\left(-\int_{x_0}^{x} \mu(l)\,\mathrm{d}l\right) \\ -\ln\left(\frac{I(x)}{I_0}\right) &= \int_{x_0}^{x} \mu(l)\,\mathrm{d}l \end{aligned} \tag{1.1.2}$$

The tomographic reconstruction problem aims at reconstructing a map $\mu(x, y)$ (or $\mu(x, y, z)$ in 3D) of the attenuation coefficient from a set of measurements $-\ln\left(\frac{I(x)}{I_0}\right)$ – called *sinogram* – at various angles. These measurements are modeled, in the Beer's law setting, by integrals along lines (Equation (1.1.2)).

1.1.2 Phase contrast tomography

Instead of measuring how an object absorbs the X-rays, Phase Contrast Tomography (PCT) aims at measuring how an object slows down the X-rays. In the absorption tomography setting, the absorption property was characterized by the attenuation coefficient

$\mu(x, y)$. In PCT, the rays deviation is measured by the refractive index. Formally, let $n(x, y, z) = 1 - \delta(x, y, z) + i\beta(x, y, z)$ be the map of the complex refractive index of an object. The real part and imaginary part act on the amplitude attenuation and phase shift of the wave field, respectively. The latter effect is dominant by orders of magnitude for hard X-rays (with energy > 10 keV); thus, a better contrast is actually obtained from the phase than from the absorption at these energies.

In this work, we mainly consider *propagation-based* PCT, which considers a coherent plane wave. In this setting, the wave in the outgoing plane of the object is given by [Zan+13]

$$T(x, y, z) = \exp\left(-B(x, y) + j\varphi(x, y)\right) \tag{1.1.3}$$

where z is the beam propagation direction, and

$$\begin{aligned} B(x, y) &= \frac{2\pi}{\lambda} \int \beta(x, y, z)\, \mathrm{d}z = \frac{1}{2} \int \mu(x, y, z)\, \mathrm{d}z \\ \varphi(x, y) &= \frac{-2\pi}{\lambda} \int \delta(x, y, z)\, \mathrm{d}z \end{aligned} \tag{1.1.4}$$

The term $B(x, y)$, linked to the imaginary part of the refractive index, characterizes the amplitude attenuation of the beam. The term $\varphi(x, y)$, linked to the (deviation from one of the) real part of the refractive index, characterizes the phase shift of the beam. Absorption tomography aims at reconstructing μ from $2B(x, y) = \int \mu(x, y, z)\, \mathrm{d}z$ and phase contrast tomography aims at reconstructing δ from $\varphi(x, y) = \frac{-2\pi}{\lambda} \int \delta(x, y, z)\, \mathrm{d}z$. The phase φ, however, is not measured directly; and a procedure is needed to retrieve the phase from the projection images. The phase retrieval is out of the scope of this work, which primarily focuses on reconstruction. When working in the PCT setting, it is assumed that the phase is already available.

In any case, it can be seen that the problem boils down to reconstructing a quantity of interest from a set of line integrals. This motivates the mathematical formalization of tomography using the Radon Transform.

1.2 The Radon Transform

The ground of computed tomography is a model of the acquisition process. This process can be characterized by a linear operator called the *Radon Transform*. On the other hand, implementing any reconstruction method on processing units entails to perform a discretization. This chapter addresses two main questions: what is the mathematical model of the tomography acquisition process and its properties ? What are the numerical properties of the discretized operators ?

Knowing the mathematical properties of the involved continuous operators enables to have an insight on the reconstruction problem, its solutions, and the discrete implementation of the operators. Understanding the numerical properties of the discrete operators will be useful when it comes to designing reconstruction algorithms.

1.2.1 Definition of the 2D Radon Transform

The Radon Transform, or *tomography projector*, is the basic operator of tomography. Here the Radon Transform is defined in two dimensions - generalization to n dimensions can be found for example in [NW01].

Definition 1 (Radon Transform)
Let $\boldsymbol{f} \in \ell_1(\mathbb{R}^2)$ be a function. The Radon Transform of f is a function $\boldsymbol{p} : [0, 2\pi] \times \mathbb{R} \to \mathbb{R}$ defined by

$$\begin{aligned} \boldsymbol{p}(\theta, s) = \mathcal{R}\left[\boldsymbol{f}\right](\theta, s) &= \int_{-\infty}^{\infty} \boldsymbol{f}(s\boldsymbol{\theta} + t\boldsymbol{\theta}^{\perp})\, \mathrm{d}t \\ &= \int_{-\infty}^{\infty} \boldsymbol{f}(s\cos\theta - t\sin\theta\,,\; s\sin\theta + t\cos\theta)\, \mathrm{d}t \end{aligned} \tag{1.2.1}$$

The quantity $\boldsymbol{p}(\theta, s)$ is called *sinogram*.
The Radon Transform of $\boldsymbol{f}$ for angle $\theta_0 \in [0, 2\pi]$, or "projection for angle θ_0" $\in [0, 2\pi]$, is the one-dimensional function equal to the Radon transform evaluated at a fixed angle:

$$\mathcal{R}_{\theta_0}[\boldsymbol{f}](s) = s \mapsto \mathcal{R}\left[\boldsymbol{f}\right](\theta_0, s) \tag{1.2.2}$$

In Definition 1, $\boldsymbol{\theta}$ and $\boldsymbol{\theta}^{\perp}$ are the unit vectors of $\mathbb{R}^2$ of respective coordinates $(\cos\theta, \sin\theta)$ and $(-\sin\theta, \cos\theta)$ in the $\mathbb{R}^2$ canonical frame. The summation occurs on the support of $\boldsymbol{f}$.
Figure 1.2.1 illustrates the definition (1.2.1).

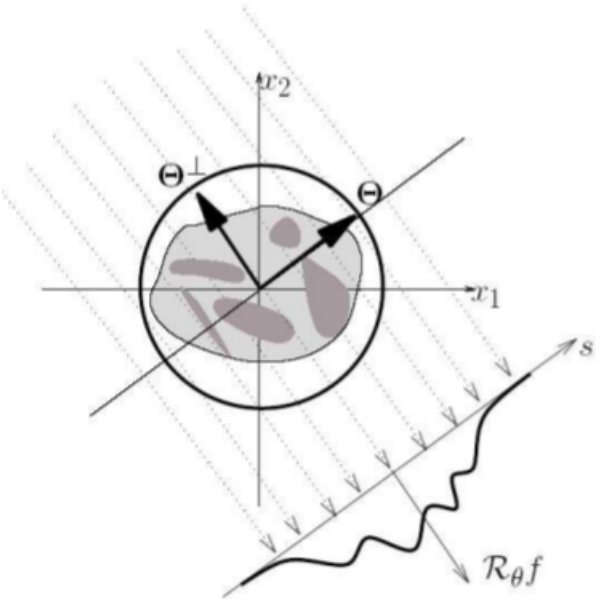


Figure 1.2.1: Schematic of the Radon Transform, where $\boldsymbol{\theta}$ and $\boldsymbol{\theta}^{\perp}$ were defined previously. Image from [Bil07].

The coordinates (θ, s) of the Radon domain $[0, 2\pi] \times \mathbb{R}$ have the following geometrical meaning: s measures the position on the detector, while θ characterizes the rotation of the object (or, equivalently, the X-ray source) with respect to the reference frame. In the real world, the function $\boldsymbol{f}$ represents a quantity of interest which can be the map of an object attenuation coefficient $\mu(x, y)$ (in the case of absorption tomography) or refractive index[1] $\delta(x, y)$ (in the case of phase contrast tomography).

[1] more precisely, the deviation from unity of the real part of the refractive index

In simple words, the Radon Transform consists in computing the integral along lines crossing $\boldsymbol{f}$ in a given direction. The process is repeated for all the angles in $[0, 2\pi]$. Illustrations are provided in section 1.3.2.

The 2D Radon Transform is a special case of the X-ray transform describing a tomographic setup. This setup has two main components:

- Geometry. The lines along which $\boldsymbol{f}$ is integrated are not always parallel; for example, they can form a cone whose origin is the X-ray source (*cone-beam geometry* in 3D, or *fan-beam geometry* in 2D).
- Trajectory. Here it is assumed that the object (or X-ray source) is simply rotating around the origin of the reference frame; but arbitrary trajectories can be considered [CD10]. For example, translation can be considered at the same time as the rotation: a vertical translation in the case of a helical scan, or a longitudinal translation in the case of the conveyor belt setup [Aar+16].

These components make the definition of the X-ray transform – and the associated reconstruction problem – more complicated. Fortunately, the X-ray light produced by a synchrotron is highly collimated, enabling to work in the setup of a parallel geometry. On the other hand, most tomography scans performed on the beamlines use a standard circular geometry. This provides a rare and convenient case where the 2D Radon Transform can be used as the X-ray transform. In the parallel setting, the "sinogram space" is the transpose of the "projections space" (Figure 1.2.2). In this manuscript, the operator $\mathcal{R}$ defined in (1.2.1) will therefore denote the 2D Radon Transform.

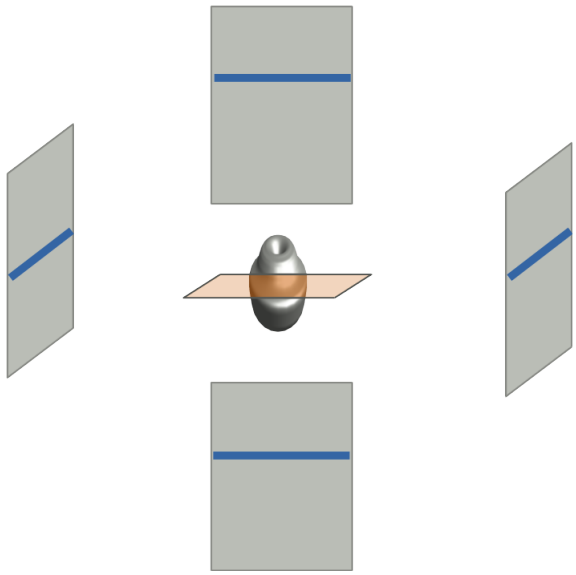


Figure 1.2.2: Illustration of the circular parallel geometry. The orange slice of the scanned object defines a plan perpendicular to the rotation axis. An X-ray beam contained in this plan goes through the object and reaches the detector at locations indicated by blue lines. The four grey rectangles illustrate projections at different angles. To reconstruct the orange slice, all the blue lines are concatenated to form a sinogram. Thus, to reconstruct a slice at altitude z, a sinogram has to be created by extracting the line at location z in all the projections. This enables to reconstruct each slice independently.
If the X-ray beam is divergent (cone beam geometry), each volume slice requires several detector lines to be reconstructed.

1.2.2 Adjoint operator

It can be shown [Bil07] that the Radon Transform admits an adjoint operator $\mathcal{R}^*$, called *backprojection*, defined In Definition 2.

Definition 2 (Backprojection)
The backprojection operator, adjoint of the Radon Transform, is defined[2] by Equation (1.2.3).

$$\begin{aligned}\mathcal{R}^*[\boldsymbol{p}](x,y) &= \int_0^{2\pi} \boldsymbol{p}(\theta, \begin{pmatrix} x \\ y \end{pmatrix} \cdot \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix}) \,\mathrm{d}\theta \\ &= \int_0^{2\pi} \boldsymbol{p}\left(\theta,\, x\cos\theta + y\sin\theta\right) \mathrm{d}\theta \end{aligned} \tag{1.2.3}$$

Simply put, the backprojection of $\boldsymbol{p}$ consists in smearing the curves $(\boldsymbol{p}(\theta, s))_\theta$ on an infinite support and accumulating them. Illustration is provided in section 1.3.2.

1.2.3 Essential properties

From its definition, it is clear that the Radon Transform is a linear operator. This section states main properties of the Radon Transform which are used in the reconstruction algorithms. For a comprehensive list of the Radon Transform properties, which is out of the scope of this manuscript, the reader can refer to [NW01], [Far+03], [Bil07].

Proposition 1.2.1 (Fourier-Slice theorem)
Let $\boldsymbol{f} \in \ell_1(\mathbb{R}^2)$, and $\theta_0 \in [0, 2\pi]$. Let $\mathcal{R}_{\theta_0}[\boldsymbol{f}]$ be the Radon transform of $\boldsymbol{f}$ at angle θ_0. The one dimensional Fourier Transform of $\mathcal{R}_{\theta_0}[\boldsymbol{f}]$ is equal to the two dimensional Fourier transform of $\boldsymbol{f}$ evaluated along a line forming an angle θ_0 with respect to the origin:

$$\mathcal{F}_1[\mathcal{R}_{\theta_0}[\boldsymbol{f}]](\nu) = \mathcal{F}_2[\boldsymbol{f}](\nu\cos\theta_0,\, \nu\sin\theta_0) \tag{1.2.4}$$

The proof can be found in Appendix 6.1.1. Proposition 1.2.1 is useful to prove many properties of the Radon Transform. It can also be used to design an efficient reconstruction algorithm described in 1.4.1. The following Proposition is a direct application of the Fourier-Slice theorem.

Proposition 1.2.2 (Convolution in the image space and in the Radon space)
Let $\boldsymbol{f}$, $\boldsymbol{g}$ be two functions of $\ell_1(\mathbb{R}^2)$. Then, Equation (1.2.5) holds

$$\mathcal{R}[\boldsymbol{f} * \boldsymbol{g}] = \mathcal{R}[\boldsymbol{f}] *_1 \mathcal{R}[\boldsymbol{g}] \tag{1.2.5}$$

meaning that the two dimensional convolution of functions $\boldsymbol{f}$ and $\boldsymbol{g}$, $\boldsymbol{f} * \boldsymbol{g}$, consists in the one dimensional convolution $(*_1)$ in the Radon domain.

The proof can be found in Appendix 6.1.1. Proposition 1.2.2 is used in computed tomography for the efficient computation of the convolution, notably in the filtered backprojection algorithm (see 1.4.2).

[2] In some textbooks, the backprojection is normalized by $\frac{1}{2\pi}$

It can also be easily seen that the Radon Transform satisfies a symmetric rule given by Proposition 1.2.3.

Proposition 1.2.3 (Symmetry of the Radon Transform)
Let $\boldsymbol{f} \in \ell_1(\mathbb{R}^2)$. Then Equation (1.2.6) holds

$$\mathcal{R}\left[\boldsymbol{f}\right](s, \theta + \pi) = \mathcal{R}\left[\boldsymbol{f}\right](-s, \theta) \tag{1.2.6}$$

A practical consequence of proposition 1.2.3 is that the angular range is generally $[0, \pi[$ instead of $[0, 2\pi]$ when working in parallel geometry. Additional properties which are directly used in reconstruction are detailed in section 1.4.

1.2.4 Singular Value Decomposition

Knowing the analytical form of the Radon Transform and its adjoint, it is possible to calculate the singular elements of $\mathcal{R}$. In the finite dimension setting, the Singular Value Decomposition (SVD). of a matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ is a factorization of $\boldsymbol{A}$ in the form

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^* \tag{1.2.7}$$

where $\boldsymbol{U}$, $\boldsymbol{V}^*$ are unitary and $\boldsymbol{\Sigma}$ is diagonal, containing the singular values of $\boldsymbol{A}$. The operator $\boldsymbol{\Sigma}$ thus consists in multiplying each component of a vector with the singular values of $\boldsymbol{A}$. Knowing the SVD of $\boldsymbol{A}$ can be helpful to evaluate the difficulty of inverting $\boldsymbol{A}$ – which will be of interest for tomographic reconstruction – and can lead to a class of inverses for $\boldsymbol{A}$.

The SVD of the (continuous) Radon Transform was derived in [Lou86], [Maa87]. It is shown that the singular functions of the Radon Transform – or, equivalently, the eigenfunctions of $\mathcal{R}^*\mathcal{R}$ – involve the Jacobi and Chebyshev polynomials. Explicit formulas are given in [Bil07], [Ker+10]. Interestingly, the singular value number k (in descending order) is given by

$$\sigma_k^{\mathcal{R}} = 2\sqrt{\frac{\pi}{k+1}} \tag{1.2.8}$$

and notably decay to zero when $k \to +\infty$.

The considered SVD holds for functions of bounded support, for example on the unit disk $\{\boldsymbol{x}, \|\boldsymbol{x}\| \leq 1\}$. As the backprojection $\mathcal{R}^*$ "smears" a function on an infinite domain ($\mathbb{R}^2$), it is not surprising that the singular elements $\boldsymbol{\psi}_k$ of $\mathcal{R}$ are relatively complex, as they should satisfy the property "$\mathcal{R}^*\mathcal{R}\boldsymbol{\psi}_k$ has a finite support" – the functions $\boldsymbol{\psi}_k$ are built in such a way that the backprojections compensate outside the support of $\boldsymbol{\psi}_k$. The SVD is also built for a particular norm of $\ell_2([-1,1])$: $\|\boldsymbol{f}\|^2 = \displaystyle\int_{-1}^{1} |\boldsymbol{f}(s)|^2 \sqrt{1-s^2}\,\mathrm{d}s$ making the operator $\mathcal{R}$ compact in the Hilbert space $\ell_2([-1,1])$.

However, an eigenvalue decomposition of $\mathcal{R}^*\mathcal{R}$ can be trivially derived for functions of unbounded support. If the integral of the Radon Transform are extended to the distributions, it can be shown (see 1.4.3) that $\mathcal{R}$ is actually diagonal in the Fourier domain, i.e the complex exponentials are the singular elements of $\mathcal{R}$.

1.2.5 Discrete Singular Value Decomposition and condition number

The discretized reconstruction problem amounts to finding an image $\boldsymbol{f}_0$ such that, given the discrete projection data $\boldsymbol{p}_0$, $\boldsymbol{P}\boldsymbol{f}_0 = \boldsymbol{p}_0$. In its discrete form, the reconstruction problem

therefore amounts to solving a set of linear equations. Generally speaking, the problem of finding a vector $\boldsymbol{x}$ given the observation $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ for some operator $\boldsymbol{A}$ is called *linear inverse problem*, this will be discussed in more details in 2.1.2. The difficulty of solving a set of linear equations can be measured by a scalar called *condition number*, defined as follows.

Definition 3 (Condition number of a matrix)
Let $\boldsymbol{A} \in \mathbb{R}^{N\times M}$. Let $\{\sigma_1, \ldots, \sigma_k\}$ be the set of nonzero singular values of $\boldsymbol{A}$. The condition number of $\boldsymbol{A}$, denoted by $\kappa_{\boldsymbol{A}}$, is defined by Equation (1.2.9) [Mar73]

$$\kappa_{\boldsymbol{A}} = \|\boldsymbol{A}\| \cdot \left\|\boldsymbol{A}^{+}\right\| \tag{1.2.9}$$

Where $\boldsymbol{A}^{+}$ is the Moore-Penrose pseudoinverse of $\boldsymbol{A}$. For the norm $\|\cdot\| = \|\cdot\|_2$, equality (1.2.10) holds

$$\kappa_{\boldsymbol{A}} = \frac{\sigma_{\max}(\boldsymbol{A})}{\sigma_{\min}(\boldsymbol{A})} \tag{1.2.10}$$

where $\sigma_{\min}(\boldsymbol{A}) = \min\limits_{i\in[1,k]} \{\sigma_i\}$ and $\sigma_{\max}(\boldsymbol{A}) = \max\limits_{i\in[1,k]} \{\sigma_i\}$.

The condition number of an operator $\boldsymbol{A}$ is related to the sensitivity of the output given some input: if there is some error in the input, this error might be amplified after evaluating the operator. For example, an isometry does not amplify the input error, and has a condition number of one. A linear inverse problem involving an operator with a large condition number is said *ill-conditioned*. Solutions of ill-conditioned problems are subject to errors whose magnitude is characterized by the condition number.

As seen in 1.2.4, the singular vectors of continuous operator $\mathcal{R}$ are relatively complex and involves modulated Jacobi polynomials. The singular values, however follow a simple decay as $1/\sqrt{k}$. Notably, the condition number would be infinite in the case of an infinitely fine discretization $k \to +\infty$. Therefore, in practical purposes, only the *truncated* SVD is considered, in the sense that only the first k singular elements corresponding to the largest singular values are kept.

The project "tomo-tv" [Gou12] provides an implementation of the discretized tomography projector as a sparse matrix (see 1.3.3), which is convenient for computing the singular elements. The scipy Python module [J+01] provides tools for numerical linear algebra on sparse matrices. As in section 1.3.3, let $\boldsymbol{P}_{N,Q}$ be the projection operator mapping $N \times N$ images to $Q \times N$ sinograms.

From table 1.1, it is clear that decreasing the number of projections for the same slice width does increase the condition number. In other words, the SVD quantifies how difficult it is to numerically reconstruct an image/volume from fewer projections.

1.3 Discretization of the space and tomography operators

1.3.1 Space sampling and discretization

Although the mathematics of tomography involve continuous spaces and operators, the processing units where computed tomography is performed are fundamentally discrete (digital) entities. On the other hand, the detectors where the projection signal is acquired have a finite number of bins (pixels). Therefore, a discretization work has to be carried out.

Shape	$\sigma_{\max}$	$\sigma_{\min}$	κ
(64, 16)	2.98e+01	4.24e-15	7.02e+15
(64, 32)	4.21e+01	7.50e-15	5.61e+15
(64, 64)	5.95e+01	3.98e-04	1.49e+05
(64, 96)	7.29e+01	1.74e-03	4.19e+04
(64, 128)	8.42e+01	2.91e-03	2.89e+04
(128, 32)	5.95e+01	3.57e-15	1.67e+16
(128, 64)	8.46e+01	1.73e-14	4.86e+15
(128, 128)	1.19e+02	2.40e-04	4.95e+05
(128, 192)	1.46e+02	8.88e-04	1.64e+05
(128, 256)	1.68e+02	1.31e-03	1.29e+05

Table 1.1: Evolution of the condition number of $\boldsymbol{P}$ as a function of the number of projections. The operator $\boldsymbol{P}$ is represented by its *shape* (N, N_p) where N denotes the number of pixels of the image in one dimension (so the total number of pixels is N^2), and N_p denotes the number of projections. Thus, the operator $\boldsymbol{P}$ has N^2 lines and $N_p \times N$ columns.

The discretization of the tomography operators depend on the way signals themselves are discretized. The most natural discretization is the standard Cartesian grid ("spikes basis"), as it is directly connected to an ideal sampling of the continuous world. Let $\boldsymbol{p}(\theta, s)$ be the (continuous) projection data. This function is not measured in practice; instead, a sampling of $\boldsymbol{p}(\theta, s)$ is measured[3]. The ideal sampling corresponds to the multiplication of the continuous function $\boldsymbol{p}(\theta, s)$ by a Dirac comb (Equation (1.3.1))

$$\text{III}_{\Delta_\theta, \Delta_s}(\theta, s) = \sum_{(k,l) \in \mathbb{Z}^2} \delta(\theta - k\Delta_\theta)\delta(s - l\Delta_s) \tag{1.3.1}$$

where Δ_θ and Δ_s are the sampling periods in θ (projection angle) and s (detector width coordinate) respectively. Let $\boldsymbol{p}_0(k, l)$ denote the *discrete* projection data resulting from an ideal sampling: $\boldsymbol{p}_0(k, l) = \boldsymbol{p}(\theta, s) \cdot \text{III}_{\Delta_\theta, \Delta_s}(\theta, s)$ [4] . This sampling is not encountered in practice, as any detector has a Point Spread Function (PSF) characterizing the signal spread over other detection elements (pixels). The data actually measured consists in the convolution of $\boldsymbol{p}_0(k, l)$ with the point spread function. This aspect is discussed in section 4.2.9. Until then, we only consider the ideal sampling.

If the ideal sampling is assumed, the values read from the detector bins directly map to a Cartesian grid: each Dirac delta function is placed at the center of one detector pixel. In this context, the projection and backprojection operators are implemented by replacing integrals with finite sums in Definition 1. However, a particular care has to be taken for modeling the rays passing through the elements of the discretized volume (voxels). Popular schemes involve splitting the voxels into sub-voxels (supersampling) (for example in the case of the Matlab® software, Figure 1.3.1), or linear interpolation (see for example [Jos82]).

[3] more precisely, the sinogram $\boldsymbol{p}(\theta, s)$ is not directly measured ; instead, radiographies are measured and the sinogram corresponds to the negative logarithm of the intensity ratio

[4] There is an abuse of notation: $\boldsymbol{p}(\theta, s)$ is a function of $\mathbb{R}^2$ and $\text{III}_{\Delta_\theta, \Delta_s}(\theta, s)$ is a distribution of $\mathbb{R}^2$. The multiplication is a distribution having non-zero values only at locations which are integer multiples of Δ_θ and Δ_s. This distribution of $\mathbb{R}^2$ is therefore assimilated to a "discrete function" of $\mathbb{Z}^2$ keeping only the nonzero values.

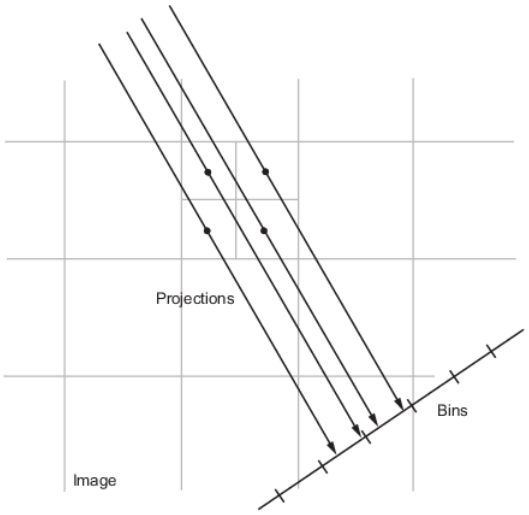


Figure 1.3.1: Illustration of the implementation of the projector with 4-supersampling. Source: Matworks

The detailed implementation of the projection and backprojection operators is out of the scope of this manuscript. In this work, these operators are viewed as building blocks of reconstruction methods. Many available softwares provide high-performance projectors, for example the general-purpose ASTRA Toolbox [Aar+16] and OpenRTK [Rit+14], and the ESRF-tailored PyHST2 program [Mir+14]. A list of reconstruction softwares used at various institutes is provided in Appendix 6.4.3.

Other space discretization schemes, and therefore operators implementations, can be considered. The space can be discretized in a Gaussian blob basis [Lew92], spline functions [Fes93], wavelets [DB95] and finite elements [BYW04]. The Gaussian blob basis can be of interest for its mathematical properties making it appealing for computed tomography [Wan11]. In this work, speed is the critical aspect of the tomography operator, especially when it comes to iterative reconstruction algorithms. For this reason, the standard discretization was adopted in order to benefit from the speed of the available operators.

1.3.2 Operators discretization

The continuous operators and their properties are a first step to implement reconstruction algorithms. Although understanding their properties in the continuous world is important, it is even more crucial to determine the properties of discrete operators which are actually used in algorithms. Schematically, designing reconstruction algorithms follows these steps:

- Determine what continuous operators are involved and establish their properties
- Discretize the space and operators
- Implement the operators

Each step is subject to some "approximation error" with respect to the previous step. For example, the Fast Fourier Transform (FFT) is an implementation of a discretization of the continuous Fourier Transform; it does not verify all the properties of the continuous Fourier Transform.

In this part, we use the operators discretization to introduce the linear algebra formalism which is used in this manuscript. Discrete linear operators can be represented with

matrices, even for two dimensional signals. This matrix representation is seldom used in practical implementations, but it is useful for deriving properties of the discrete operators. Conveniently, the operator analysis is easier with numerical linear algebra than with continuous integrals, and it can be check with numerical simulations.

Let $\boldsymbol{f}$ and $\boldsymbol{g}$ be two functions of a continuous space, for example $\ell_1(\mathbb{R})$, with a finite support. Let $\boldsymbol{f}_0$ and $\boldsymbol{g}_0$ be their discrete counterpart, in the standard "spikes" basis. As $\boldsymbol{f}$ and $\boldsymbol{g}$ have a finite support, $\boldsymbol{f}_0$ and $\boldsymbol{g}_0$ can be represented as element of a vector space of finite dimension. If $\boldsymbol{f}_0$ (resp. $\boldsymbol{g}_0$) has N (resp. M) samples, the vector space is $\mathbb{R}^N$ (resp. $\mathbb{R}^M$).

The convolution operation has a straightforward discretization, boiling down to replacing the (possibly) infinite integral with a sum computed on the support of involved vectors. The integral (1.3.2)

$$(\boldsymbol{f} * \boldsymbol{g})(\tau) = \langle \boldsymbol{f}\,,\, \tau \mapsto \boldsymbol{g}(\tau - t) \rangle\,(\tau) = \int_{-\infty}^{\infty} \boldsymbol{f}(t)\boldsymbol{g}(\tau - t)\,\mathrm{d}t \tag{1.3.2}$$

becomes the finite sum (1.3.3)

$$(\boldsymbol{f}_0 * \boldsymbol{g}_0)(k) = \langle \boldsymbol{f}_0\,,\, k \mapsto g(k - i) \rangle\,(k) = \sum_i \boldsymbol{f}_0(i)\boldsymbol{g}_0(k - i) \tag{1.3.3}$$

where the summation is computed on the support of $\boldsymbol{f}_0$ and $\boldsymbol{g}_0$ with boundaries conditions.

The Fourier Transform operator $\mathcal{F}$ has also a straightforward discretization known as the Discrete Fourier Transform (DFT) (Equation (1.3.4))

$$\mathcal{F}[\boldsymbol{f}_0](k) = \sum_{i_0}^{N-1} \boldsymbol{f}(k)\omega_N^{i \cdot k} = \begin{bmatrix} \omega_N^{0 \cdot 0} & \dots & \omega_N^{0 \cdot (N-1)} \\ \vdots & \vdots & \vdots \\ \omega_N^{(N-1) \cdot 0} & \dots & \omega_N^{(N-1)(N-1)} \end{bmatrix} \boldsymbol{f}_0 \tag{1.3.4}$$

where ω_N is a primitive N-root of the unity: $\omega_N = e^{-\boldsymbol{\imath} 2\pi / N}$. The DFT has an efficient implementation known as the Fast Fourier Transform, but it makes the implicit assumption that $\boldsymbol{f}_0$ is periodic outside of its support.

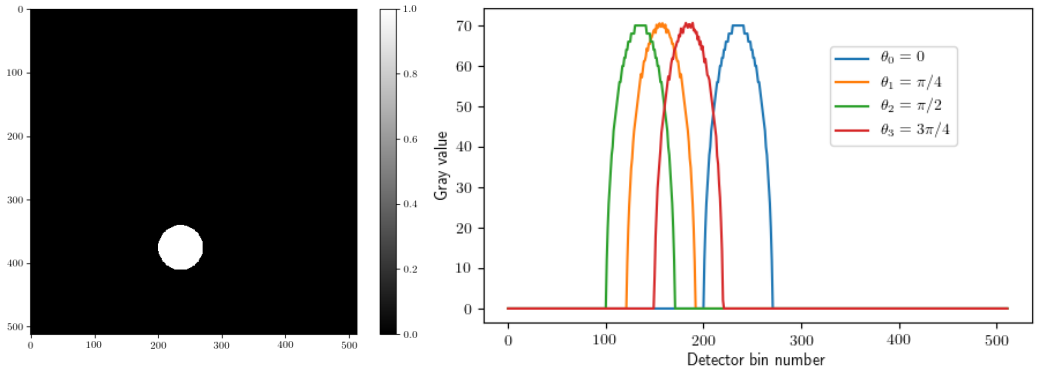
Replacing the integral with a finite sum actually causes a "scaling" effect in the operator. To invert the DFT, one first needs to scale the Fourier data by dividing with N; that is, $\boldsymbol{F}^{-1} = \frac{1}{N}\boldsymbol{F}^*$. However, the continuous Fourier Transform $\mathcal{F}$ defined in the "Definitions and notations" section benefits from the unitary property: $\mathcal{F}^{-1} = \mathcal{F}^*$, *i.e* the inverse transform is simply the adjoint. In order to keep the unitary property in the discrete case, the DFT can be normalized with the square root of the number of samples; thus, the operator $\frac{1}{\sqrt{N}}\boldsymbol{F}$ defines an unitary DFT.

Linear operators acting on two dimensional signals can also be represented with a matrix by representing 2D signals (images) as *stacked one-dimensional* vectors. For example, an operator $\boldsymbol{A}$ mapping $\boldsymbol{f}_0 \in \mathbb{R}^{N_1 \times N_2}$ to $\boldsymbol{g}_0 \in \mathbb{R}^{M_1 \times M_2}$ is a matrix with $M_1 \times M_2$ lines and $N_1 \times N_2$ columns.

The discrete Radon Transform, or *projection operator*, is defined by summing the image pixels values along straight lines, at different angles. The summation may involve linear interpolation, which keeps the operator linearity. Let $\boldsymbol{f}_0$ a discrete 2D vector (image). The projection of $\boldsymbol{f}_0$ on angle θ_k is defined by Equation (1.3.5)

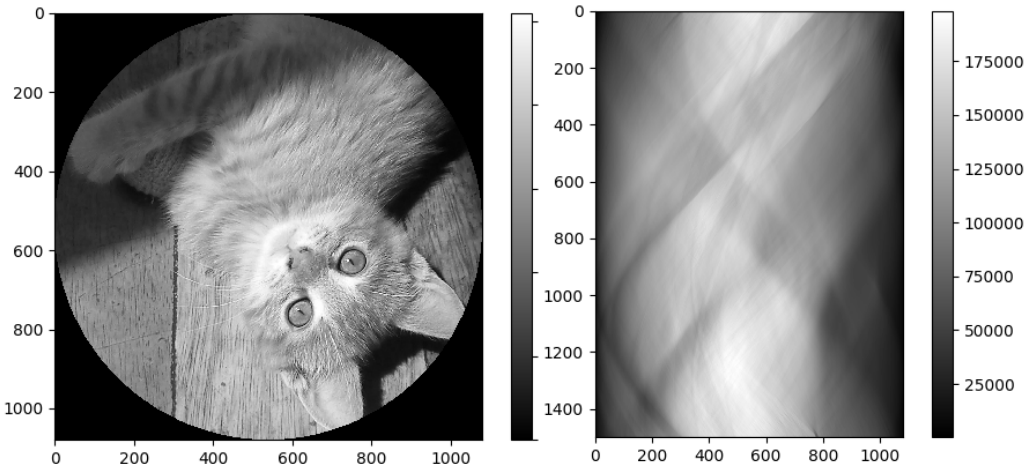
$$\boldsymbol{P}[\boldsymbol{f}_0](\theta_k, s) = \sum_u \boldsymbol{f}_0(s\cos\theta_k - u\sin\theta_k\,,\, s\sin\theta_k + u\cos\theta_k) \tag{1.3.5}$$

where s denotes a discrete index. The fact that neither $s\cos\theta_k - u\sin\theta_k$ nor $s\sin\theta_k + u\cos\theta_k$ are not integer in general, even for integer (s, u), justifies the need for interpolation. Figures 1.3.2 and 1.3.3 illustrate the effect of the projector. In this manuscript, the projection operator is denoted by $\boldsymbol{P}$.



(a) (b)

Figure 1.3.2: (a) Image of 512×512 pixels containing a uniform disk or radius 35 pixels, centered in $(x, y) = (235, 375)$ (origin on top left). (b) Four projection lines corresponding to four projection angles.



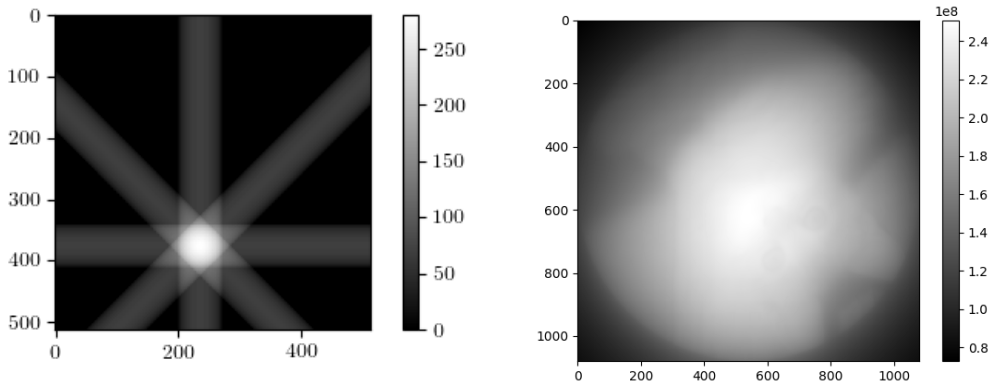
(a) (b)

Figure 1.3.3: (a) Sample test image *Cookie*, 1080×1080 pixels. (b) Sinogram of *Cookie* with 1500 projection angles. The vertical axis denotes the angle number, and the horizontal axis denotes the detector bin number.

The *backprojection operator*, adjoint of the discrete Radon Transform, can be defined by Equation (1.3.6)

$$\boldsymbol{P}^T[\boldsymbol{p}_0](i,j) = \sum_{k=0}^{N_p-1} \boldsymbol{p}_0(\theta_k,\, i\cos\theta_k + j\sin\theta_k) \tag{1.3.6}$$

where $\boldsymbol{p}_0$ is the discretization of $\boldsymbol{p}(\theta, s)$ with N_p projection angles. In this manuscript, the backprojection operator is denoted by $\boldsymbol{P}^T$.



(a) (b)

Figure 1.3.4: (a) Backprojection of the disk of Figure 1.3.2. Each projection line of Figure 1.3.2a is smeared on all the image support according to the corresponding angle. The result is the sum of all the projections contributions. (b) Backprojection of the sinogram in Figure 1.3.3b.

The projection-backprojection pair suffer from the same "scaling problem" as the DFT. Although the continuous Radon Transform do not verify any unitary property like the Fourier Transform, a *scaled backprojection* can be defined by multiplying with $\frac{\pi}{N_p}$ where N_p denotes the number of projections. This scaled backprojection is notably used in the Filtered Back-Projection algorithm (section 1.4.2). The scaling step comes from the sampling of the angular range: the interval $[0, \pi]$ sampled with N_p projections gives a step $\frac{\pi}{N_p}$. In the case of an angular range $[0, 2\pi]$ sampled with N_p projections, the angular step is $\frac{2\pi}{N_p}$. In this case, the backprojection can still be multiplied with $\frac{\pi}{N_p}$ due to the symmetry of the Radon Transform, as each projection is accounted twice (the projections at angles $\theta + \pi$ provide the same information as the projections at angles θ). More details can be found in [NW01], Theorem 4.4 and Section 5.1.1.

1.3.3 Sparse representation

Let $\boldsymbol{P}_{N,Q}$ be the projection operator mapping $N \times N$ pixels images to a sinogram with Q angles. In the matrix representation, the images are $(N^2, 1)$ vectors (N^2 lines uni-column vector), the sinograms are $(Q \times N, 1)$ vectors and $\boldsymbol{P}_{N,Q}$ is a $(Q \times N, N^2)$ matrix. This matrix quickly becomes impossible to store when N (and, consequently, Q) gets large. For example, 1.0 GigaBytes of memory is required for $N = Q = 128$ pixels, and 70 TeraBytes for $N = Q = 2048$ pixels – which is a routinely used image size.

Fortunately, many entries of the matrix $\boldsymbol{P}_{N,Q}$ are actually zero. Indeed, a ray passing through a given pixel will hit the detector at a certain position (θ, s). The contribution of this pixel is usually smeared on a small neighborhood around (θ, s), as this position does not have integer coordinates in general. Let $\rho > 0$ be the average neighborhood size for an image pixel contribution to the sinogram. Then, in average, each pixel in an image is projected on ρ detector bins for each projection. This means that the matrix $\boldsymbol{P}_{N,Q}$ has at most $\rho \cdot N^2 \times Q$ nonzero entries. $\boldsymbol{P}_{N,Q}$ is said to be *sparse*; efficient representations and numerical methods exists for such matrices.

1.4 Analytical reconstruction

The *reconstruction problem* consists in reconstructing $\boldsymbol{f} \in \ell_1(\mathbb{R}^2)$ from its Radon Transform $\boldsymbol{p}(\theta, s) = \mathcal{R}[\boldsymbol{f}](\theta, s)$. In this section, methods hereby called "analytical reconstruction methods" are considered. The name comes from the fact that these methods come from an analytical formula. Three main reconstruction methods are considered: Fourier-slice theorem, Filtered Back-Projection and Lambda tomography.

1.4.1 Fourier reconstruction

The Fourier-Slice theorem (Proposition 1.2.1) provides directly the reconstruction algorithm 1.4.1.

Algorithm 1.4.1 Fourier Reconstruction

$\boldsymbol{p}(\theta, s)$: Radon Transform of function $\boldsymbol{f}$ to reconstruct

1: $\hat{\boldsymbol{p}}(\theta, \nu) = \mathcal{F}_1[\boldsymbol{p}(\theta, s)]$ ▷ Compute the 1D Fourier Transform of each projection θ
2: Place $(\hat{\boldsymbol{p}}(\theta, \nu))_\theta$ on a Cartesian Grid
3: $\boldsymbol{f}(x, y) = \mathcal{F}_2^{-1}[\hat{\boldsymbol{p}}(\theta, \nu)]$ ▷ Compute the inverse 2D transform

Algorithm 1.4.1 seems appealing, as it only involves 1D and 2D Fourier Transforms of the projection data $\boldsymbol{p}(\theta, s)$, which can be efficiently computed with the FFT. However, this method suffers from a major drawback: once the 1D Fourier Transform (FT) of each projection angle is performed, the Fourier data has to be placed from a polar grid (series of 1D FT) to a Cartesian grid (2D FT, Figure 1.4.1). This entails to interpolate between polar to Cartesian in the Fourier space, which is known to be cumbersome, as any interpolation error in the Fourier space results in an error in *all* the real space after the inverse transform.

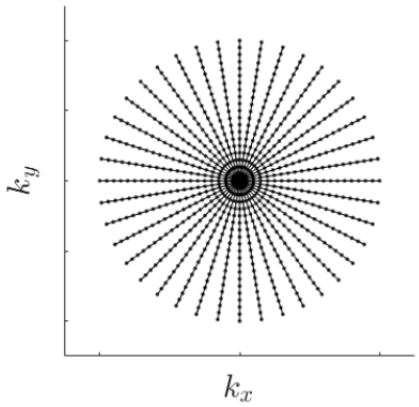


Figure 1.4.1: The series of 1D FT of each projection angle samples the plan according to a polar grid. To perform the reconstruction, this set of 1D FT have to be placed on a Cartesian grid. Unfortunately, low frequencies components (near the center) are over-represented with respect the high frequency components (far from the center), which makes the interpolation difficult.

Reconstruction algorithms using the Fourier-Slice theorem and interpolating between polar grid to a Cartesian grid are called *regridding* (or "gridrec") algorithms. The in-

terpolation can be done with truncated cardinal sine function [Shk+15] with minimal reconstruction artifacts if carefully performed. However, the interpolation is still costly as the sinc function has a very slow decay: many neighbouring Fourier points have to be involved in order to compute one interpolated point. Other interpolating function such as the prolate spheroidal wavefunctions can be used [MS12]. These functions form the eigenfamily of the *windowed Fourier Transform* operator, and notably converge to the Gaussian function when iterating infinitely many windowed Fourier Transforms. Such interpolating functions are interesting because they are optimally localized in the Fourier domain – meaning they involve relatively few neighbouring points in the interpolation process – and in the spatial domain – meaning that the interpolation kernel is limitedly spread in the spatial space.

Interestingly, Fourier-based reconstruction used to be deserted in favour of the Filtered Back-Projection method 1.4.2, as the delicate interpolation step was difficult to perform both accurately and efficiently. Recently, the increase of computational power and the work on Fourier interpolation methods revived the interest for Fourier-based techniques. Fourier-based methods are particularly interesting if GPU devices cannot be used (i.e only CPU are available) as they are intrinsically faster. Notably, gridrec is the default reconstruction algorithm of the TomoPy package [Gür+14].

1.4.2 Filtered Back-Projection

The Filtered Back-Projection (FBP) is the standard algorithm used for 2D tomographic reconstruction. It involves two steps: a filtering of the projection data, and a backprojection step, i.e the application of the adjoint operator of the Radon Transform.

Proposition 1.4.1 (Filtered Back-Projection)
Let $\boldsymbol{p}(\theta, s)$ be the projection data (Radon Transform) of $\boldsymbol{f} \in \ell_1(\mathbb{R}^2)$. Letting $\boldsymbol{v}(s)$ be a function of one variable verifying $\mathcal{F}_1[\boldsymbol{v}](\nu) = \hat{\boldsymbol{v}}(\nu) = |\nu|$, Equation (1.4.1) holds.

$$\begin{aligned} f(x,y) &= \mathcal{R}^*[\boldsymbol{p}(\theta,s) *_1 \boldsymbol{v}(s)] \\ &= \int_0^{2\pi} \boldsymbol{p'}(\theta,\, x\cos\theta + y\sin\theta)\,\mathrm{d}\theta \\ \text{where} \quad \boldsymbol{p'}(\theta,s) &= \int_{-\infty}^{\infty} \mathcal{F}_1[\boldsymbol{p}](\theta,\nu)|\nu| e^{+\boldsymbol{\jmath} 2\pi\nu s}\,\mathrm{d}\nu \end{aligned} \tag{1.4.1}$$

The proof can be found in Appendix 6.1.1.

Proposition 1.4.1 highlights that $\boldsymbol{f} \in \ell_1(\mathbb{R}^2)$ can be *exactly* reconstructed from its Radon Transform. Thus, the FBP can be seen as the *inverse* of the continuous Radon Transform[5]. However, in practice, the functions to be reconstructed have a compact support in the spatial domain, since they correspond to a quantity of interest in a scanned object which has a finite size. Thus, the Fourier Transform of $\boldsymbol{f}$ has a theoretically infinite support. This means that the multiplication with $|\nu|$ of the 1D FT of the projections has to be performed far in the high frequencies, which is practically inconvenient.

In [NW01], the FBP is derived by natively taking into account the *essential bandwidth* of $\boldsymbol{f}$. A function $\boldsymbol{f}$ has an essential bandwidth Ω if most of its energy in the Fourier domain is contained in $\Omega \subset \mathbb{C}^2$. Instead of being multiplied with $\hat{\boldsymbol{v}}(\nu) = |\nu|$, the 1D FT of the projection $\hat{\boldsymbol{p}}(\theta,\nu)$ are multiplied with $|\nu|\hat{\boldsymbol{w}}_{\boldsymbol{\Omega}}(\nu)$ where $\hat{\boldsymbol{w}}_{\boldsymbol{\Omega}}$ vanishes outside Ω.

[5] some software packages like Matlab or scikit-image use the term iradon for "inverse Radon"

Thus, the multiplication with $|\nu|$ is not carried out on an infinitely supported domain, but only on a domain containing most of the energy of $\boldsymbol{f}$. This is equivalent, in the spatial domain, to reconstructing $\boldsymbol{q_\Omega} * \boldsymbol{f}$ instead of $\boldsymbol{f}$, where $\boldsymbol{q_\Omega}$ is the spatial domain counterpart of $|\nu|\hat{\boldsymbol{w}}_{\boldsymbol{\Omega}}(\nu)$ in the "1D-Fourier-Radon" domain. Indeed, Proposition 1.4.1 is a particular case of Theorem 2.3 in [NW01] which is stated in Equation (1.4.2)

$$\boldsymbol{f} * \mathcal{R}^*[\boldsymbol{h}] = \mathcal{R}^* \left[\mathcal{R}\boldsymbol{f} *_1 \boldsymbol{h}\right] \tag{1.4.2}$$

In our case, $\mathcal{R}\boldsymbol{f} = \boldsymbol{p}$ is the projection data, and $\boldsymbol{h} = \boldsymbol{v} *_1 \boldsymbol{w_\Omega}$ is the filter, meaning that $\boldsymbol{q_\Omega} = \mathcal{R}^*[\boldsymbol{h}]$ acts as a "point spread function" hampering the reconstruction of $\boldsymbol{f}$. In practical applications, the two dimensional function $\boldsymbol{q_\Omega}$ should be as close as possible to a 2D Dirac in order to faithfully approximate $\boldsymbol{f}$.

Equation (1.4.3) sums up the setting in our case. The left column is the image space, the right column is the Radon space, and the bottom line is the Fourier space.

$$\begin{array}{ccc} \boldsymbol{f} * \boldsymbol{q_\Omega} & \xrightarrow{\quad\mathcal{R}\quad} & \boldsymbol{p}(\theta, s) *_1 \boldsymbol{v}(s) *_1 \boldsymbol{w_\Omega}(s) \\ & & \downarrow \mathcal{F}_1 \\ & & \hat{\boldsymbol{p}}(\theta, \nu) \cdot |\nu| \cdot \hat{\boldsymbol{w}}_{\boldsymbol{\Omega}}(\nu) \end{array} \tag{1.4.3}$$

In the case where $\hat{\boldsymbol{w}}_{\boldsymbol{\Omega}} = 1$, meaning that $\boldsymbol{w_\Omega}$ is a 1D Dirac delta function $\boldsymbol{\delta}_1$, the continuous FBP is recovered (it is easily seen that applying Proposition 1.4.1 with $\boldsymbol{p}(\theta, s) = \boldsymbol{\delta}_1(s)$ yields $\boldsymbol{q_\Omega} = \mathcal{R}^*[\boldsymbol{v}] = \boldsymbol{\delta}_2$, a 2D Dirac). As previously mentioned, this perfect reconstruction is not possible in practice due to the infinite support of the Fourier transform of $\boldsymbol{f}$; practical implementations of the FBP reconstruct $\boldsymbol{f} * \boldsymbol{q_\Omega}$ instead of $\boldsymbol{f}$. The function $\boldsymbol{q_\Omega}$ is determined by the filter applied to the 1D Fourier Transform of the projections. A popular choice is the Ramachandran-Lakshminarayanan (Ram-Lak) filter corresponding to $\hat{\boldsymbol{w}}_{\boldsymbol{\Omega}}(\nu) = 1$ for $\nu \in \Omega$ and $\hat{\boldsymbol{w}}_{\boldsymbol{\Omega}}(\nu) = 0$ for $\nu \notin \Omega$. The discretized form of the corresponding filter is given by Equation (1.4.4)

$$(\boldsymbol{v} *_1 \boldsymbol{w_\Omega})(l) = \frac{1}{\Delta_s^2} \begin{cases} 1/4 & l = 0 \\ 0 & l \neq 0,\ l \text{ is even} \\ -1/(\pi^2 l^2) & l \text{ is odd} \end{cases} \tag{1.4.4}$$

where Δ_s is the spatial sampling step (see [Hsi03], Chapt. 3). Other popular filters use a smoother apodization function instead of a rectangle, for example a cosine or a Hamming window. The high frequencies are therefore dampened, which can be of interest when noise hinders the reconstruction quality of $\boldsymbol{f}$.

The FBP algorithm can be summarized by Algorithm 1.4.2.

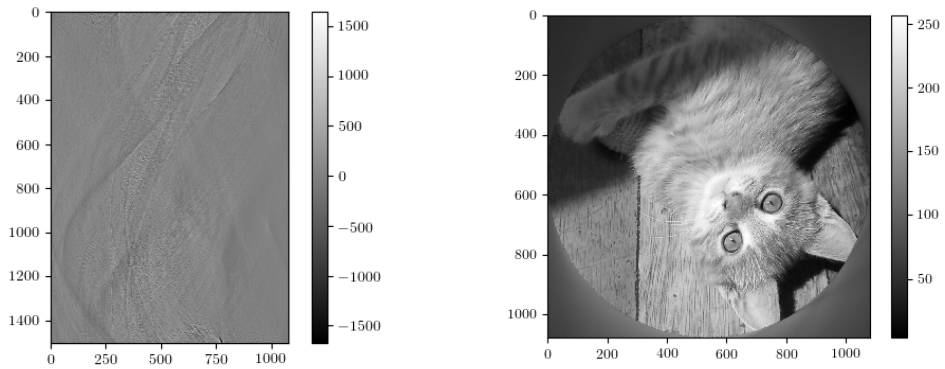
Algorithm 1.4.2 Filtered Back-Projection

$\boldsymbol{p}(\theta, s)$: Radon Transform of function $\boldsymbol{f}$ to reconstruct

1: $\boldsymbol{q}(\theta, s) = \boldsymbol{p}(\theta, s) *_1 \boldsymbol{h}(s)$ ▷ 1D filtering of each projection
2: $\boldsymbol{f} \simeq \mathcal{R}^*(\boldsymbol{q})$ ▷ Apply the backprojection operator to obtain an approximate of $\boldsymbol{f}$

The filtering step can be performed after the backprojection ("backproject-then-filter") rather than before the projection ("filter-then-backproject"). However, this approach has two drawbacks: the two dimensional filtering is computationally more costly, and the filtering has to be performed on a large grid as the backprojection operator has a theoretically infinite support.

Figure 1.4.2 shows an example of the FBP with the Ram-Lak filter (1.4.4).



(a) (b)

Figure 1.4.2: (a) Sinogram of figure 1.3.3 after filtering. Each line undergoes a high-pass filtering with the Ram-Lak filter (1.4.4). (b) Filtered backprojection after scaling the backprojection with $\frac{\pi}{1500}$.

1.4.3 Lambda tomography

Lambda tomography consists in reconstructing an approximation of $\boldsymbol{f} \in \ell_1(\mathbb{R}^2)$ instead of $\boldsymbol{f}$. In the case of the FBP, the approximation Lambda tomography traditionally aims at reconstructing the high frequencies of $\boldsymbol{f}$ in the context of *local tomography* [Far+03]. The topic of local tomography is discussed in a greater extent in 4.2. The Calderón's operator, defined in Definition 4 plays a central role in Lambda tomography. The definitions and formula proposed in this work hold in the context of parallel geometry. Generalization to more general 3D geometries like cone beam is not straightforward, although works in this direction are proposed for example in [YYW06], although assuming a nonstandard trajectory.

Definition 4 (Calderón's operator)
The Calderón's operator Λ is defined in the Fourier domain by Equation (1.4.5)

$$\mathcal{F}_2[\Lambda \boldsymbol{f}](\nu_x, \nu_y) = \sqrt{\nu_x^2 + \nu_y^2} \cdot \mathcal{F}_2[f](\nu_x, \nu_y) \tag{1.4.5}$$

where it is assumed that $\boldsymbol{f}$ is a function of two variables. In other words, the Calderón's operator is a multiplication by the frequency magnitude $\left|\begin{pmatrix}\nu_x \\ \nu_y\end{pmatrix}\right|$ in the frequency domain.

In ambiguous statements, the Calderón's operator will be denoted by Λ_2 (resp. Λ_1) when it acts on two variables (resp. one variable) of a function.

From basic Fourier properties, it is clear that $\Lambda^2 = -\Delta$, i.e applying twice the Calderón's operator yields the opposite of the Laplacian. A general reconstruction formula [Far+03], which is a generalization of 1.4.1 (and Theorem 2.3 in [NW01]) is given by Equation (1.4.6)

$$\boldsymbol{q} * \Lambda_2^m \boldsymbol{f} = \mathcal{R}^* \left[(\Lambda_1^{m+1} \mathcal{R}\boldsymbol{q}) *_1 \mathcal{R}\boldsymbol{f} \right] \qquad m \geq -1 \tag{1.4.6}$$

where Λ_1 is only applied on the second component when used in the Radon domain.

Taking $\boldsymbol{q} = \delta_2$ in (1.4.6), the case $m = 0$ yields the Filtered Back-Projection. The case $m = 1$ boils down to backprojecting the (opposite) Laplacian of the sinogram. The case $m = -1$ boils down to backprojecting the sinogram. Lambda tomography consists in approximating $\boldsymbol{f}$ by a linear combination of $\Lambda\boldsymbol{f}$ and $\Lambda^{-1}\boldsymbol{f}$ in order to mix the high frequencies and low frequencies of $\boldsymbol{f}$, respectively.

Lambda tomography was not investigated further in this work. We however give an essential property derived from Lambda tomography, stated by Equation (1.4.7), which is essential in section 3.4.

$$\mathcal{R}^*\mathcal{R} = \Lambda^{-1} \tag{1.4.7}$$

meaning that in the frequency domain, $\mathcal{R}^*\mathcal{R}$ consists in dividing by the frequency magnitude $\left|\begin{pmatrix}\nu_x\\ \nu_y\end{pmatrix}\right|$ – hence the "blurring" effect of the plain backprojection rather than the filtered backprojection. This can be put into the form of Equation (1.4.8)

$$\mathcal{R}^*\mathcal{R} = \mathcal{F}_2^*\mathcal{D}_{\Lambda_2^{-1}}\mathcal{F}_2 \tag{1.4.8}$$

where $\mathcal{D}_{\Lambda_2^{-1}}$ is the *diagonal* operator consisting in multiplying element-wise by the inverse of the frequency magnitude. Equation (1.4.8) means that $\mathcal{R}^*\mathcal{R}$ is diagonal in the Fourier domain, providing immediately a SVD of the operator $\mathcal{R}$. However, this SVD holds for functions with *infinite* support, as the complex exponentials of the basis $\mathcal{F}_2$ have an infinite support. The SVD mentioned in 1.2.4 holds for functions with finite support. It can be noted that the singular values, which are given by the square root of the values of the diagonal operator $\mathcal{D}_{\Lambda_2^{-1}}$, are proportional to $1/\sqrt{\nu_x^2 + \nu_y^2}$, in accordance with the decay as $1/\sqrt{k}$ in 1.2.4.

Equations (1.4.7) and (1.4.8) still provide a useful ground for understanding the link between FBP and the minimum norm solution. In the discrete setting, the minimum norm solution is given by the following proposition.

Proposition 1.4.2 (Minimum norm solution of a linear inverse problem)
Let $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ be an inverse problem. The *minimum norm solution* of a linear inverse problem is a solution $\boldsymbol{x}_m$ given by Equation (1.4.9)

$$\boldsymbol{x}_m = \boldsymbol{A}^T\left(\boldsymbol{A}\boldsymbol{A}^T\right)^{-1}\boldsymbol{y} \tag{1.4.9}$$

It is the solution of Problem (1.4.10)

$$\begin{aligned} &\underset{\boldsymbol{x}}{\operatorname{argmin}}\ \{\|\boldsymbol{x}\|_2\} \\ &\text{s. t. } \boldsymbol{A}\boldsymbol{x} = \boldsymbol{y} \end{aligned} \tag{1.4.10}$$

The proof can be found in Appendix 6.1.1. Now in the continuous setting, the operator $\boldsymbol{P}\boldsymbol{P}^T$ becomes $\mathcal{R}\mathcal{R}^*$. On the other hand, Equation (1.4.7) states that $\mathcal{R}^*\mathcal{R} = \Lambda_2^{-1}$, i.e a low-pass frequency filter in the image domain. It can then be shown that $\mathcal{R}\mathcal{R}^* = \Lambda_1^{-1}$, a 1D low-pass filter in the Radon domain. Indeed, from Equation (1.4.7) and the the FBP theorem, we have

$$\begin{cases} \mathcal{R}^*\mathcal{R} = \Lambda_2^{-1} \\ \mathcal{R}^{-1} = \mathcal{R}^*\Lambda_1 \end{cases} \tag{1.4.11}$$

Using the first equality (i.e Equation (1.4.7)) yields

$$\Lambda_2 \mathcal{R}^* \mathcal{R} = \boldsymbol{I} \quad \Rightarrow \quad \mathcal{R}^{-1} = \Lambda_2 \mathcal{R}^* \tag{1.4.12}$$

as $\mathcal{R}^{-1} = \mathcal{R}^* \Lambda_1$ from the FBP theorem, we then have

$$\mathcal{R}^* \Lambda_1 = \Lambda_2 \mathcal{R}^* \tag{1.4.13}$$

which justifies the equivalence between the "backproject-then-filter" and "filter-then-backproject" approaches. Multiplying the previous equality with Λ_1^{-1} yields

$$\begin{aligned} \mathcal{R}^* &= \underbrace{\Lambda_2 \mathcal{R}^*}_{\mathcal{R}^{-1}} \Lambda_1^{-1} \\ \mathcal{R}\mathcal{R}^* &= \Lambda_1^{-1} \end{aligned} \tag{1.4.14}$$

Proposition 1.4.2 applied to the operator $\mathcal{R}$ exactly means that the FBP is also the minimum norm solution of $\mathcal{R}\boldsymbol{f} = \boldsymbol{p}$. In the discrete case, using the ramp filter is equivalent to approximate $(\boldsymbol{P}\boldsymbol{P}^T)^{-1}$ with its continuous version $(\mathcal{R}\mathcal{R}^*)^{-1}$.

The equalities $\mathcal{R}^*\mathcal{R} = \Lambda_2^{-1}$ and $\mathcal{R}\mathcal{R}^* = \Lambda_1^{-1}$ also mean that $\mathcal{R}^*\mathcal{R}$ (or $\mathcal{R}\mathcal{R}^*$) is a shift-invariant operator, as it is diagonalized by the Fourier basis (i.e it is a simple filter in the frequency domain). In the discrete case, it might not be the case anymore, but still be a good approximation. Thus, modifying the "ramp filter" Λ_1^{-1} to approximate $(\boldsymbol{P}\boldsymbol{P}^T)^{-1}$ (or, equivalently, $(\boldsymbol{P}^T\boldsymbol{P})^{-1}$) for the "subsampled" version $\boldsymbol{P}$ of $\mathcal{R}$ leads to a computationally efficient minimum norm solution computation (an iterative process is approximated by a filtering process). An extensive work in this direction can be found in [Pel16].

Bibliography

[Aar+16] Wim van Aarle et al. "Fast and flexible X-ray tomography using the ASTRA toolbox". In: *Opt. Express* 24.22 (Oct. 2016), pp. 25129–25147. DOI: 10.1364/OE.24.025129. URL: http://www.opticsexpress.org/abstract.cfm?URI=oe-24-22-25129.

[Bil07] Anne Bilgot. "Méthodes locales d'identification de surfaces de discontinuité à partir de projections tronquées pour l'imagerie interventionnelle". PhD thesis. Université Joseph-Fourier-Grenoble I, 2007.

[BYW04] Jovan G Brankov, Yongyi Yang, and Miles N Wernick. "Tomographic image reconstruction based on a content-adaptive mesh model". In: *IEEE Transactions on medical imaging* 23.2 (2004), pp. 202–212.

[CD10] R. Clackdoyle and M. Defrise. "Tomographic Reconstruction in the 21st Century". In: *Signal Processing Magazine, IEEE* 27.4 (July 2010), pp. 60–80. ISSN: 1053-5888. DOI: 10.1109/MSP.2010.936743.

[DB95] Alexander H Delaney and Yoram Bresler. "Multiresolution tomographic reconstruction using wavelets". In: *IEEE Transactions on image processing* 4.6 (1995), pp. 799–813.

[Far+03] Adel Faridani et al. "Introduction to the mathematics of computed tomography". In: *Inside Out: Inverse Problems and Applications* 47 (2003), pp. 1–46.

[Fes93] Jeffrey A Fessler. "Tomographic reconstruction using information-weighted spline smoothing". In: *Biennial International Conference on Information Processing in Medical Imaging*. Springer. 1993, pp. 372–386.

[Gou12] Emmanuelle Gouillart. *tomo-tv*. https://github.com/emmanuelle/tomo-tv. 2012.

[Gür+14] Doga Gürsoy et al. "TomoPy: a framework for the analysis of synchrotron tomographic data". In: *Journal of synchrotron radiation* 21.5 (2014), pp. 1188–1193.

[Hsi03] Jiang Hsieh. *Computed tomography: principles, design, artifacts, and recent advances*. Vol. 114. SPIE press, 2003.

[J+01] Eric Jones, Travis Oliphant, Pearu Peterson, et al. *SciPy: Open source scientific tools for Python*. [Online; accessed 2017-03-21]. 2001–. URL: http://www.scipy.org/.

[Jos82] Peter M Joseph. "An improved algorithm for reprojecting rays through pixel images". In: *IEEE transactions on medical imaging* 1.3 (1982), pp. 192–196.

[Ker+10] Gérard Kerkyacharian et al. "Inversion of noisy Radon transform by SVD based needlets". In: *Applied and Computational Harmonic Analysis* 28.1 (2010), pp. 24–45.

[Lew92] Robert M Lewitt. "Alternatives to voxels for image representation in iterative reconstruction algorithms". In: *Physics in Medicine and Biology* 37.3 (1992), p. 705.

[Lou86] Alfred K. Louis. "Incomplete data problems in x-ray computerized tomography". In: *Numerische Mathematik* 48.3 (1986), pp. 251–262. ISSN: 0945-3245. DOI: 10.1007/BF01389474. URL: http://dx.doi.org/10.1007/BF01389474.

[Maa87] P Maass. "The x-ray transform: singular value decomposition and resolution". In: *Inverse problems* 3.4 (1987), p. 729.

[Mar73] Albert W. Marshall. "Norms and inequalities for condition numbers, III". In: *Linear Algebra and its Applications* 7.4 (1973), pp. 291–300. ISSN: 0024-3795. DOI: http://dx.doi.org/10.1016/S0024-3795(73)80002-3. URL: http://www.sciencedirect.com/science/article/pii/S0024379573800023.

[Mir+14] Alessandro Mirone et al. "The PyHST2 hybrid distributed code for high speed tomographic reconstruction with iterative reconstruction and a priori knowledge capabilities". In: *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms* 324.0 (2014). 1st International Conference on Tomography of Materials and Structures, pp. 41–48. ISSN: 0168-583X. DOI: http://dx.doi.org/10.1016/j.nimb.2013.09.030. URL: http://www.sciencedirect.com/science/article/pii/S0168583X14000251.

[MS12] F Marone and M Stampanoni. "Regridding reconstruction algorithm for real-time tomographic imaging". In: *Journal of synchrotron radiation* 19.6 (2012), pp. 1029–1037.

[NW01] Frank Natterer and Frank Wübbeling. *Mathematical methods in image reconstruction*. SIAM, 2001.

[Pel16] D.M. Pelt. "Filter-based reconstruction methods for tomography". PhD thesis. University of Leiden, 2016.

[Rit+14] S Rit et al. "The Reconstruction Toolkit (RTK), an open-source cone-beam CT reconstruction toolkit based on the Insight Toolkit (ITK)". In: *Journal of Physics: Conference Series* 489.1 (2014), p. 012079. URL: http://stacks.iop.org/1742-6596/489/i=1/a=012079.

[Shk+15] Roman Shkarin et al. "GPU-optimized direct Fourier method for on-line tomography". In: *Fundamenta Informaticae* 141.2-3 (2015), pp. 245–258.

[Wan11] Han Wang Wang. "X-ray CT Image Reconstruction from Few Projections". Theses. Université de Grenoble, Oct. 2011. URL: https://tel.archives-ouvertes.fr/tel-00680100.

[YYW06] Hengyong Yu, Yangbo Ye, and Ge Wang. "Practical cone-beam lambda tomography". In: *Medical physics* 33.10 (2006), pp. 3640–3646.

[Zan+13] Irene Zanette et al. "Looking Inside Marine Organisms with Magnetic Resonance and X-ray Imaging". In: *Imaging Marine Life*. Wiley-VCH Verlag GmbH & Co. KGaA, 2013, pp. 122–184. ISBN: 9783527675418. DOI: 10.1002/9783527675418.ch7. URL: http://dx.doi.org/10.1002/9783527675418.ch7.

Chapter 2

Regularized iterative reconstruction methods

In this chapter, we briefly review the long-lived reconstruction algorithms and present the regularized methods. We use the Bayesian formalism to regroup all the reconstruction methods in a common framework. This approach makes explicit the assumptions made on the noise/volume properties, and decouples the model from the algorithm needed to solve the optimization problem.

We then review more recent reconstruction methods based on the compressed sensing framework. As they lead to problems which are more difficult to solve than usual methods, we also review state-of-the-art convex optimization algorithms.

2.1 From analytical to iterative reconstruction

In this section, we present the motivation to use iterative reconstruction algorithms rather than the direct methods presented in 1.4.

2.1.1 Limits of analytical reconstruction

Reconstruction with direct method (FBP, direct Fourier reconstruction) requires that the discrete operator $\boldsymbol{P}$ accurately represents the continuous operator $\mathcal{R}$. More precisely, to accurately reconstruct the frequency support of an image of N pixels width, at least N_p projections are needed, where [KS88] [NW01]

$$N_p \geq \frac{\pi}{2} N \tag{2.1.1}$$

which can be seen as a consequence of the Shannon-Nyquist sampling criterion. In practice, $N_p \simeq N$ is sufficient in most cases without causing subsampling artefacts. In fact, the number of projections N_p essentially depends on the frequency range of the object of interest to reconstruct. For example, an object of 300×300 pixels support contained in an image of 1024×1024 pixels (filled with zeros otherwise) will not need $\frac{\pi}{2} \cdot 1024$ projections to be accurately reconstructed, but rather $\frac{\pi}{2} \cdot 300$.

However, practical cases can involve highly limited data where $N_p < N$ significantly due to various experimental constraints (fast tomography, radiation dose concern). In these cases, the standard analytical methods do not yield a satisfactory reconstruction. Another such case is where the projection images have a low Signal to Noise Ratio (SNR). In this case, more advanced methods are required to improve the SNR of the reconstructed volume. Lastly, a direct reconstruction method might be lacking in some geometries/trajectories.

2.1.2 Tomographic reconstruction as an inverse problem

In signal processing, an inverse problem consists in finding a latent signal $\boldsymbol{x}$ giving the observed data $\boldsymbol{y}$ after the effect of some process $\mathcal{H}$. The operator $\mathcal{H}$ and the noise model

fully characterize the inverse problem. The acquisition model is formally

$$\boldsymbol{y} = \mathcal{H}(\boldsymbol{x}) \tag{2.1.2}$$

up to the acquisition noise.

For example, a measure with a detector involves some spill of the signal over the detector imaging elements (pixels), i.e a feature smaller than a pixel might yield a signal across several neighbouring pixels. This spill can be modelled as a point spread function; in this case, operator $\mathcal{H}$ in Equation (2.1.2) is a convolution. The problem of finding a "sharp" signal $\boldsymbol{x}$ from a "blurred" observed data $\boldsymbol{y}$ is called *deconvolution*.

An inverse problem can be described by its *well-posedness* properties. More precisely, an inverse problem is well-posed in the Hadamard sense [Han10] if

- A solution exists
- The solution is unique
- The solution depends continuously on the initial conditions

An interesting subset of problem (2.1.2) is given when the operator $\mathcal{H}$ is linear, this corresponds to *linear inverse problems*. In this case, $\mathcal{H}$ can be modelled as a matrix $\boldsymbol{H}$, so that (2.1.2) becomes

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x} \tag{2.1.3}$$

up to the acquisition noise. Linear Inverse problems are convenient with respect to non-linear inverse problems: powerful mathematical frameworks and numerical solvers can be used. Fortunately, a large class of signal processing problems can be modelled by Linear Inverse Problem (LIP)s; for example deconvolution, denoising, inpainting, motion estimation, segmentation, source separation and tomography [CP11] [SF09].

Formally, solving a LIP amounts to solving a set of linear equations. Thus, from the matrix $\boldsymbol{H}$, the Hadamard conditions are easy to determine:

- A solution exists if and only if the set of equations formed by $\boldsymbol{H}$ is consistent
- The solution is unique if and only if $\boldsymbol{H}$ has full rank (there are exactly as many equations as unknowns)
- The solution always depends continuously on the initial conditions, as $\boldsymbol{H}$ is a linear operator in a finite vector space

The third condition is actually of little help in practice. Indeed, despite the continuous behaviour of the solution(s) with respect to the initial conditions, the solution may vary with a great extent given the data. This behaviour is given by the *condition number* of $\boldsymbol{H}$ (see Definition 3). In general, LIP are both *ill-posed* (there are more or fewer measurements than needed, i.e no solution or infinitely many solutions) and *ill-conditioned*.

With the proper assumptions [1], tomographic reconstruction is a special instance of LIP. In absorption tomography, if the X-rays are "sufficiently monochromatic" (i.e the relative energy range $\Delta E/E$ is thin), the measurement model can be given by the Beer's law. The logarithm of the incoming/outgoing intensities ratio yields the sinogram, which is linearly linked to the linear absorption coefficient map of the scanned volume. The same

[1] Beer's law modelling the X-ray transmission process, leading to a linear forward projector

holds for phase contrast tomography, replacing the absorption coefficient map with the refractive index map[2].

The tomography reconstruction problem is then modelled as the LIP

$$\boldsymbol{d} = \boldsymbol{P}\boldsymbol{x} \tag{2.1.4}$$

up to the acquisition noise, where $\boldsymbol{d}$ is the acquired sinogram, $\boldsymbol{x}$ is the latent volume to reconstruct, and $\boldsymbol{P}$ is the projection operator (in our working case, the discretized Radon Transform).

As problem (2.1.4) is ill-posed, a common strategy is to solve a related *surrogate problem*. In general, the surrogate problem is designed to satisfy the following requirements:

- Well-posed
- Consistent with the original problem: the solution should belong to the solution space of the original problem
- Computationally tractable to solve

The surrogate problem associated with a LIP often have the form

$$\underset{\boldsymbol{x}}{\operatorname{argmin}} \ \{f(\boldsymbol{x}, \boldsymbol{y}) + g(\boldsymbol{x})\} \tag{2.1.5}$$

where $f(\boldsymbol{x}, \boldsymbol{y})$ is a *data fidelity term*, measuring the distance between the (projection of the) solution and the acquired data; and $g(\boldsymbol{x})$ is a function encoding some prior knowledge on the solution, if any. More details on what f, g can be will be given in the next section. On the other hand, the well-posedness of Problem (2.1.5) follows from the choice of f, g. First, the solution exists and is unique for *convex functions* f, g defined on a *convex space*. Second, the solution continuously depend on the input, as in general f, g are Lipschitz functions in a finite dimensional space, hence continuous functions. More details are given in the section 2.6.

2.2 Iterative reconstruction in a Bayesian framework

2.2.1 The Maximum Likelihood approach

Let $\boldsymbol{x}$ be a latent signal to recover (here, a digital volume or image). The vector $\boldsymbol{x}$ is not directly observed in practice ; what is observed is

$$\boldsymbol{y} = \boldsymbol{P}\boldsymbol{x} \tag{2.2.1}$$

in a noiseless setting. As any observation is corrupted with noise, let $\boldsymbol{n}$ be a random variable modelling the acquisition noise. In a first approach, the noise is modelled as an additive noise – this assumption is further discussed in 2.4.4. The observed vector is then

$$\boldsymbol{y} = \boldsymbol{P}\boldsymbol{x} + \boldsymbol{n} \tag{2.2.2}$$

As $\boldsymbol{n}$ is a random variable, the observed data $\boldsymbol{y}$ is also a random variable which depends on the noise $\boldsymbol{n}$. Let $p_{\boldsymbol{\theta}}(\boldsymbol{n})$ denote the Probability Density Function (PDF) of the random variable $\boldsymbol{n}$; this function depends on several parameters $\boldsymbol{\theta}$. In the case of a normal distribution, only two parameters fully characterize the PDF: $\boldsymbol{\theta} = \begin{pmatrix} \mu \\ \sigma \end{pmatrix}$. By a slight abuse of

[2] more precisely, the deviation from unity of the real part of the complex refractive index

notation, the fact that the random variable $\boldsymbol{n}$ has the PDF $p_{\boldsymbol{\theta}}(\boldsymbol{n})$ will be simply denoted by $\boldsymbol{n} \sim p_{\boldsymbol{n}}(\boldsymbol{n})$.

As $\boldsymbol{n} \sim p_{\boldsymbol{n}}(\boldsymbol{n})$, it is easily shown[3] that $\boldsymbol{y} \sim p_{\boldsymbol{y}}(\boldsymbol{y})$ where $p_{\boldsymbol{y}}(\boldsymbol{y}) = p_{\boldsymbol{n}}(\boldsymbol{y} - \boldsymbol{P}\boldsymbol{x})$ if $\boldsymbol{x}$ is assumed to be known. The assumption that $\boldsymbol{x}$ is known obviously does not holds in practice ; the previous equality is then denoted by Equation (2.2.3)

$$\boldsymbol{y} \sim p_{\boldsymbol{n}}(\boldsymbol{y} - \boldsymbol{P}\boldsymbol{x} \mid \boldsymbol{x}) \tag{2.2.3}$$

which is to be read "probability of observing $\boldsymbol{y}$ given $\boldsymbol{x}$".

The quantity $p_{\boldsymbol{n}}(\boldsymbol{y} - \boldsymbol{P}\boldsymbol{x} \mid \boldsymbol{x})$ is called *likelihood* of observing $\boldsymbol{y}$. The idea of the Maximum Likelihood (ML) algorithm is to *maximize* this quantity with respect to the unknown $\boldsymbol{x}$, so that the acquired data $\boldsymbol{y}$ corresponds to the *most probable* observed signal. The rather natural underlying hypothesis of this framework is the likelihood of the observation, i.e the assumption that the data does not correspond to a rare event.

Let us now assume that the noise is a zero-mean white Gaussian noise independent from the data. This hypothesis is discussed further in 2.4.4. Then:

$$p_{\boldsymbol{n}}(\boldsymbol{n}) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2\sigma^2} \|\boldsymbol{n}\|_2^2\right) \tag{2.2.4}$$

where $\sigma = \sigma_{\boldsymbol{n}}$ is the noise standard deviation (here a scalar, as the noise is assumed uncorrelated). The likelihood can then be written

$$p_{\boldsymbol{y}}(\boldsymbol{y}) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2\sigma^2} \|\boldsymbol{y} - \boldsymbol{P}\boldsymbol{x}\|_2^2\right) \tag{2.2.5}$$

The ML algorithm boils down to

$$\underset{\boldsymbol{x}}{\operatorname{argmax}} \ \{p_{\boldsymbol{y}}(\boldsymbol{y})\} = \underset{\boldsymbol{x}}{\operatorname{argmin}} \ \left\{\|\boldsymbol{y} - \boldsymbol{P}\boldsymbol{x}\|_2^2\right\} \tag{2.2.6}$$

where the negative logarithm was applied in Equation (2.2.6), so that the ML algorithm is equivalent to minimizing the negative log-likelihood. It can be noted that all the multiplicative factors involving σ were dropped, as they do not depend on $\boldsymbol{x}$.

To sum up, the ML algorithm applied with a white Gaussian noise assumption to recover $\boldsymbol{x}$ is nothing but a least-squares solution. As the observation model $\boldsymbol{y} = \boldsymbol{P}\boldsymbol{x}$ is linear, a closed-form formula can be given[4]:

$$\boldsymbol{x}_{\mathrm{ML}} = (\boldsymbol{P}^T\boldsymbol{P})^{-1}\boldsymbol{P}^T\boldsymbol{y} \tag{2.2.7}$$

which corresponds to the (left) Moore-Penrose pseudoinverse of $\boldsymbol{P}$. In a statistics perspective, the ML solution $\boldsymbol{x}_{\mathrm{ML}}$ is a linear combination of the data $\boldsymbol{y}$, as the operator $(\boldsymbol{P}^T\boldsymbol{P})^{-1}\boldsymbol{P}^T$ is clearly linear. Thus, $\boldsymbol{x}_{\mathrm{ML}}$ is a *linear estimator* of $\boldsymbol{y}$. The fact that $\boldsymbol{x}_{\mathrm{ML}}$ is obtained through a least squares minimization can then be viewed as a consequence of the Gauss-Markov theorem.

In a signal processing perspective, $\boldsymbol{x}_{\mathrm{ML}}$ is obtained by *filtering* the backprojected data $\boldsymbol{P}^T\boldsymbol{y}$, the filter being $(\boldsymbol{P}^T\boldsymbol{P})^{-1}$. Here, the notion of filter is to be understood in the general setting of the application of a linear operator. In the continuous case and in parallel geometry, $(\boldsymbol{P}^T\boldsymbol{P})^{-1}$ is nothing but the ramp filter in the spatial domain (see

[3] the random variable $\boldsymbol{y} \sim p_{\boldsymbol{y}}(\boldsymbol{y})$ depends on the random variable $\boldsymbol{n} \sim p_{\boldsymbol{n}}(\boldsymbol{n})$ through a deterministic relation $\boldsymbol{y} = g(\boldsymbol{n}) = \boldsymbol{a} + \boldsymbol{n}$ where $\boldsymbol{a} = \boldsymbol{P}\boldsymbol{x}$ is deterministic. Applying the change of variable theorem to $g(\boldsymbol{n}) = \boldsymbol{a} + \boldsymbol{n}$ yields $p_{\boldsymbol{y}}(\boldsymbol{y}) = p_{\boldsymbol{n}}(\boldsymbol{y} - \boldsymbol{a})$.

[4] provided that $\boldsymbol{P}$ has linearly independent columns

1.4.3), so that $(\boldsymbol{P}^T\boldsymbol{P})^{-1}\boldsymbol{P}^T\boldsymbol{y}$ exactly corresponds to the filtered backprojection method. Thus, the ML estimator with a white Gaussian noise assumption converges to the inverse of $\boldsymbol{P}$ (or $\mathcal{R}$) in the continuous setting and parallel geometry.

In the case of a correlated (non-white) Gaussian noise, the correlation between the noise samples can be modelled with a covariance matrix $\boldsymbol{\Sigma}$. The entry (i, j) of $\boldsymbol{\Sigma}$ is the covariance between n_i and n_j. The PDF of $\boldsymbol{n}$ becomes

$$p_{\boldsymbol{\Sigma}}(\boldsymbol{x}) = \frac{1}{\sqrt{|\det 2\pi\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}\boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{x}\right) \tag{2.2.8}$$

where it is still assumed that the noise is zero-mean. It is easily seen that Equation (2.2.8) leads to the same optimization problem as (2.2.6) replacing the Euclidean distance $\|\boldsymbol{x}\|_2^2 = \boldsymbol{x}^T\boldsymbol{x}$ with a *weighted* Euclidean distance $\|\boldsymbol{x}\|_{\boldsymbol{\Sigma}^{-1}}^2 = \boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{x}$.

Now, let us assume that the noise is actually data-dependent. In imaging applications, fluctuation of photon counts on the detector creates a "shot noise". As the data is itself related to the photon count, the noise is data dependent. The Poisson statistics is mostly used for modelling this noise, especially for low number of detector counts (for example in electron tomography). For a Gaussian noise, the probability $p_{\boldsymbol{y}}(\boldsymbol{y} \mid \boldsymbol{x})$ was a normal distribution of mean $\boldsymbol{P}\boldsymbol{x}$. Now, it is a Poisson distribution of mean $\boldsymbol{P}\boldsymbol{x}$:

$$p_{\boldsymbol{y}}(\boldsymbol{y} \mid \boldsymbol{x}) = \prod_i \frac{(Px)_i^{y_i} e^{-(Px)_i}}{y_i!} \tag{2.2.9}$$

where it is assumed that all the samples follow the same distribution Poisson$(\boldsymbol{P}\boldsymbol{x})$. Minimizing the negative log-likelihood leads to

$$\operatorname*{argmin}_x \left\{ \sum_i \left[(Px)_i - y_i \log(Px)_i\right] \right\} \tag{2.2.10}$$

In the signal processing context, the objective function in Equation (2.2.10) is called the Kullback-Leibler "distance"[5] between $\boldsymbol{P}\boldsymbol{x}$ and $\boldsymbol{y}$. This model is of course computationally more complicated since it involves the computation of $\log((Px)_i)$: for example, positivity constraints on $\boldsymbol{x}$ have to be added provided that $\boldsymbol{P}$ is a positive operator.

2.2.2 The Maximum A Posteriori approach

From Equation (2.2.3), the ML approach was derived assuming that $\boldsymbol{x}$ is a deterministic quantity. However, this quantity is not known, or at least only some statistical properties of $\boldsymbol{x}$ are known. The vector $\boldsymbol{x}$ is then modelled as another random process having some probability density function $p_{\boldsymbol{x}}(\boldsymbol{x})$. This function is the prior knowledge (or *Gibbs prior*) available on $\boldsymbol{x}$.

As $\boldsymbol{x}$ in unknown and $\boldsymbol{y}$ is observed, the aim is to recover $\boldsymbol{x}$ from $\boldsymbol{y}$. In a probabilistic point of view, one would like to know which values $\boldsymbol{x}$ are the most likely given observed data $\boldsymbol{y}$, that is, $p_{\boldsymbol{x}}(\boldsymbol{x} \mid \boldsymbol{y})$. The Bayes formula gives $p_{\boldsymbol{x}}(\boldsymbol{x} \mid \boldsymbol{y})$ (the quantity of interest) as a function of $p_{\boldsymbol{y}}(\boldsymbol{y} \mid \boldsymbol{x})$ (which can be computed from the noise model):

$$p_{\boldsymbol{x}}(\boldsymbol{x} \mid \boldsymbol{y}) = \frac{p_{\boldsymbol{y}}(\boldsymbol{y} \mid \boldsymbol{x}) p_{\boldsymbol{x}}(\boldsymbol{x})}{p_{\boldsymbol{y}}(\boldsymbol{y})} \tag{2.2.11}$$

[5] The Kullback-Leibler function is not a distance strictly speaking, as the triangle inequality does not hold. It is nevertheless used to measure a similarity between two distributions.

The quantity $p_{\boldsymbol{x}}(\boldsymbol{x} \mid \boldsymbol{y})$ is called *posterior probability* while $p_{\boldsymbol{x}}(\boldsymbol{x})$ is called *prior probability* corresponding to the prior knowledge on $\boldsymbol{x}$. Maximum A Posteriori (MAP) methods aim at finding $\boldsymbol{x}$ which maximizes the quantity (2.2.11): given the observed data $\boldsymbol{y}$, which latent vector $\boldsymbol{x}$ has the higher probability of giving $\boldsymbol{y}$ through the forward model $\boldsymbol{y} = \boldsymbol{P}\boldsymbol{x}$ (up to the noise). Again, as it is equivalent but simpler, the negative log-likelihood is minimized. The quantity $p_{\boldsymbol{y}}(\boldsymbol{y})$ does not depend on $\boldsymbol{x}$ and can be omitted of the optimization procedure.

If the latent signal $\boldsymbol{x}$ is assumed to be normally distributed with a standard deviation $\sigma_{\boldsymbol{x}}$, i.e $\boldsymbol{x} \sim p_{\boldsymbol{x}}(\boldsymbol{x}) = (\sigma_{\boldsymbol{x}}\sqrt{2\pi})^{-1} \exp\left(-(2\sigma_{\boldsymbol{x}}^2)^{-1} \|\boldsymbol{x}\|_2^2\right)$, then it is easily shown that maximizing (2.2.11) amounts to solving

$$\underset{\boldsymbol{x}}{\operatorname{argmin}} \left\{ \frac{1}{2\sigma_{\boldsymbol{n}}^2} \|\boldsymbol{y} - \boldsymbol{P}\boldsymbol{x}\|_2^2 + \frac{1}{2\sigma_{\boldsymbol{x}}^2} \|\boldsymbol{x}\|_2^2 \right\} \tag{2.2.12}$$

In the same fashion, if the data $\boldsymbol{x}$ is known to follow a Laplace distribution with parameter $\lambda > 0$, the MAP estimator is given by

$$\underset{\boldsymbol{x}}{\operatorname{argmin}} \left\{ \frac{1}{2\sigma_{\boldsymbol{n}}^2} \|\boldsymbol{y} - \boldsymbol{P}\boldsymbol{x}\|_2^2 + \lambda \|\boldsymbol{x}\|_1 \right\} \tag{2.2.13}$$

The case where the noise model $p_{\boldsymbol{n}}(\boldsymbol{n})$ follows a Poisson distribution can be formalized by replacing the ℓ_2 norm with a Kullback-Leibler divergence, as previously shown.

2.3 Classical iterative reconstruction methods

In this section, several classical iterative reconstruction algorithms are briefly reviewed. These algorithms were proposed relatively early in CT and are well understood. However, these are often described simply by their update rule, which can seem artificial. Instead, we describe these method from a statistical perspective, for example making explicit the assumptions on the noise.

2.3.1 Algebraic Reconstruction Technique

The Algebraic Reconstruction Technique (ART) is the only method which is not directly related to a noise model, although it converges to a minimum norm solution of (2.2.1). ART is based on the Projection Onto Convex Sets (POCS) method based on the fact that the inverse problem $\boldsymbol{d} = \boldsymbol{P}\boldsymbol{x}$ consists in a set of linear equations. More precisely, if the latent signal $\boldsymbol{x}$ has N samples (the total number of pixels of an image, or the total number of voxels of a volume), and if the acquired data $\boldsymbol{d}$ has M samples, then $\boldsymbol{d} = \boldsymbol{P}\boldsymbol{x}$ is a set of M linear equations of N unknowns. Geometrically, each equation defines a hyperplane of $\mathbb{R}^N$ characterized by the coefficients of the corresponding line of the matrix $\boldsymbol{P}$. The POCS method applied to the particular case of solving the set of linear equations $\boldsymbol{d} = \boldsymbol{P}\boldsymbol{x}$ is called the Kaczmarz method.

This method iterates by successively *projecting* (in the mathematical sense of the term) an estimate solution $\boldsymbol{x}_n$ on each hyperplane. For example, $\boldsymbol{x}_1$ is obtained by projecting $\boldsymbol{x}_0$ on the hyperplane formed by the first line of $\boldsymbol{d} = \boldsymbol{P}\boldsymbol{x}$; $\boldsymbol{x}_2$ is obtained by projecting $\boldsymbol{x}_1$ on the second hyperplane, and so on. It can be shown [GPR67] that the method converges to the intersection of the hyperplanes, which is nothing but the solution of $\boldsymbol{d} = \boldsymbol{P}\boldsymbol{x}$. Theoretically, the intersection is a point for a full rank system. However, in practice, the method is used when there are fewer measurements than needed, i.e $M < N$. In this case,

the intersection of the hyperplanes is not a single point, but a vector subspace of $\mathbb{R}^N$ of dimension at least $N-M$, containing infinitely many points (solutions). The point of this set reached by the ART depends on a *relaxation factor* $\lambda \in [0,1]$ which is a parameter of this method [KS88]. From an initial estimate $\boldsymbol{x}_0$ of the reconstructed volume, the update rule if ART is given by Equation (2.3.1)

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i - \lambda \frac{\boldsymbol{P_i}^T \boldsymbol{x}_i - d_i}{\|\boldsymbol{P_i}\|_2^2} \boldsymbol{P_i} \tag{2.3.1}$$

where $\boldsymbol{P_i}$ is the vector equal to the line number i of the projector.

As the projection matrix $\boldsymbol{P}$ often model a smooth trajectory, the successive lines define equations that are "very close". For example, the first line models the acquisition at angle 0, and the second line models the acquisition at angle Δ_θ; if Δ_θ is relatively small, the two equations are highly correlated. Thus, a faster convergence rate is obtained by randomly selecting the hyperplanes where the solution is projected onto.

A refinement of the ART is the Simultaneous Algebraic reconstruction Technique (SART) which notably updates the estimate with all the rays in a projection at each iteration [KS88]. It therefore yields a better convergence rate, at the expense of a higher cost per iteration.

As it can be highlighted, the (S)ART methods converge to *one among many* solutions of $\boldsymbol{d} = \boldsymbol{P}\boldsymbol{x}$, depending on a damping parameter of the algorithm. In the following sections, algorithms based on the minimization of an objective function are presented. The considered objective functions have exactly one global minimum, meaning that there is exactly one solution. This can be seen as an advantage of these methods.

2.3.2 Least Squares Reconstruction Technique

The least squares reconstruction, as seen previously, corresponds to the maximum likelihood solution of the acquisition problem $\boldsymbol{d} = \boldsymbol{P}\boldsymbol{x}$ with a white Gaussian noise (see 2.2.1). Although a closed-form solution is given by $\boldsymbol{x}_{\text{ML}} = (\boldsymbol{P}^T\boldsymbol{P})^{-1}\boldsymbol{P}^T\boldsymbol{d}$, this solution is never computed directly. The reason is that the matrix $\boldsymbol{P}^T\boldsymbol{P}$ is too huge to be stored in memory – even if a sparse representation of $\boldsymbol{P}^T\boldsymbol{P}$ is considered, its inverse is not sparse. For this reason, the maximum likelihood solution is computed iteratively.

Fortunately, efficient optimization algorithms for solving $\underset{\boldsymbol{x}}{\text{argmin}} \left\{ \|\boldsymbol{P}\boldsymbol{x} - \boldsymbol{d}\|_2^2 \right\}$ are available. For example, the ASTRA Toolbox [Aar+15] provides an efficient implementation of the least squares reconstruction "CGLS" based on the Conjugate Gradient optimization method (more details are given in section 3.3.2).

This algorithm converges extremely fast to the (least squares) solution. However, as mentioned in the introduction of section 2.4.1, the least-squares solution (which is obtained numerically by running the optimization algorithm until its convergence, i.e no decreasing of the objective function) is likely to be noisy due to the problem being ill-conditioned. In other words, although the least squares reconstruction works well on noiseless data, it does amplify the noise that might be present in the input data, even in tiny amounts. To prevent this, the user would often tune the number of iterations as a parameter of the reconstruction algorithm. On the other hand, the least squares solution obtained by the conjugate gradient does not allow for constraints like the positivity constraint: the reconstructed volume is a map of a quantity of interest which is often positive (linear attenuation coefficient, refraction index decrement).

2.3.3 Simultaneous Iterative Reconstruction Technique

The Simultaneous Iterative Reconstruction Technique (SIRT) is often defined by the means of its update rule (Equation (2.3.2))

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{C}\boldsymbol{P}^T\boldsymbol{R}(\boldsymbol{d} - \boldsymbol{P}\boldsymbol{x}_k) \tag{2.3.2}$$

where $\boldsymbol{C}$ and $\boldsymbol{R}$ are the diagonal matrices[6] given by Equation (2.3.3) [Aar+15]

$$\begin{aligned} 1/C_{j,j} &= \sum_i P_{i,j} \\ 1/R_{i,i} &= \sum_j P_{i,j} \end{aligned} \tag{2.3.3}$$

It can be shown [GB08] that this method corresponds to solving

$$\operatorname*{argmin}_{\boldsymbol{x}} \left\{ \|\boldsymbol{P}\boldsymbol{x} - \boldsymbol{d}\|_{\boldsymbol{R}}^2 \right\} \tag{2.3.4}$$

Thus, SIRT is a method for finding a *weighted* least-squares solution. The standard Euclidean distance has been replaced with a metric $\|\boldsymbol{d}\|_{\boldsymbol{R}}^2 = \boldsymbol{d}^T\boldsymbol{R}^T\boldsymbol{d}$, which corresponds to a correlated Gaussian noise model of covariance matrix $\boldsymbol{\Sigma} = \boldsymbol{R}^{-1}$ (see 2.2.1).

In practice, a volume reconstructed with SIRT bears minor differences with a volume reconstructed with "CGLS", although it is more stable when used with many iterations. Unlike Conjugate Gradient (CG), the SIRT algorithm allows for constraints, but converges slower as it is an instance of Landweber iteration (see section 2.7.1).

2.3.4 Expectation Maximization algorithm

Generally speaking, the Expectation Maximization (EM) method aims at computing the maximum likelihood estimate from incomplete data [DLR77]. In the context of a linear inverse problem with a Poisson noise model, EM simply boils down to the maximization of the log-likelihood. From 2.2.1, the objective function to maximize is

$$\log p_{\boldsymbol{n}}(\boldsymbol{d} \mid \boldsymbol{x}) = -\sum_i \left[(Px)_i - d_i \log(Px)_i\right] \tag{2.3.5}$$

Applying the Kuhn-Tucker condition of Equation (2.3.5) with the non-negativity constraint $\boldsymbol{x} \geq \boldsymbol{0}$ yields

$$\boldsymbol{0} = \boldsymbol{x}\boldsymbol{P}^T\left(\boldsymbol{1} - \frac{\boldsymbol{d}}{\boldsymbol{P}\boldsymbol{x}}\right) \tag{2.3.6}$$

where the vector operations are understood component-wise. Equation (2.3.6) suggests for an iterative fixed-point scheme

$$\boldsymbol{x}_{k+1} = \frac{1}{\boldsymbol{P}^T\boldsymbol{1}}\boldsymbol{x}_k\boldsymbol{P}^T\frac{\boldsymbol{d}}{\boldsymbol{P}\boldsymbol{x}_k} \tag{2.3.7}$$

(here $\boldsymbol{1}$ is a vector filled with ones) which indeed converges to the maximum likelihood solution $\boldsymbol{x}_{\mathrm{ML}}$ [NW01]. In the signal processing community, when $\boldsymbol{P}$ is replaced by a convolution operator, iteration (2.3.7) is called the Richardson-Lucy deconvolution algorithm. Contrarily to previous methods, the update is multiplicative; notably, the initial estimate should be filled with ones rather than with zeros.

As a multiplicative algorithm, EM always provides positive solutions, at the price of a slower convergence.

[6] $\boldsymbol{R}$ and $\boldsymbol{C}$ are mistakenly called "inverse row sum" and "inverse column sum" in [GB08], and the summation indices i and j in [Aar+15] should be inverted

2.4 Regularized reconstruction methods

The classical reconstruction methods (ART, SIRT, EM) are linked to the maximum likelihood (ML) estimate given a noise model. Notably, they do not account for any prior knowledge on the latent signal. This section presents regularized reconstruction methods, which are linked to the maximum a posteriori (MAP) estimate given a probability distribution on the latent signal.

In sections 2.4.1, 2.4.2, 2.4.3, we explain what is regularization and *how* it works. Section 2.4.5 explains *why* ℓ_1 regularization works well for signal retrieval in linear inverse problems, introducing the compressive sensing framework.

2.4.1 Motivation of regularization

The inverse problem $\boldsymbol{P}\boldsymbol{x} = \boldsymbol{d}$ is ill-posed in general; notably, it can have infinitely many solutions. The previous section described methods providing one unique solution of a surrogate problem, for example $\underset{\boldsymbol{x}}{\operatorname{argmin}} \left\{ \|\boldsymbol{P}\boldsymbol{x} - \boldsymbol{d}\|_2^2 \right\}$. The probabilistic framework provides a way to make explicit the assumptions on the noise model, and to know what kind of solution is reached.

However, although the surrogate problem is *well-posed* (the solution exists, is unique, and depends continuously on the input data), the operator $\boldsymbol{P}$ makes the solution computation sensitive to deviation in the input data: the problem is *ill-conditioned*. For example, it is well known that applying the Landweber iteration (for example SIRT reconstruction) until convergence can yield an amplification of noise [VRU08].

For this reason, a *regularization* is considered. Regularizing a problem consists in incorporating prior knowledge on the solution, that is, computing a MAP solution rather than the ML solution. Considering Problem (2.1.5) and section 2.2.2, the data-fidelity distance f is based on the noise model, and the function g is based on the prior knowledge available on the latent signal.

2.4.2 Tikhonov regularization

The simplest regularization method is known as the Tikhonov regularization. Problem (2.1.5) becomes

$$\underset{\boldsymbol{x}}{\operatorname{argmin}} \left\{ f(\boldsymbol{x}, \boldsymbol{d}) + \|\boldsymbol{D}\boldsymbol{x}\|_{\boldsymbol{\Gamma}}^2 \right\} \tag{2.4.1}$$

i.e the prior knowledge is encoded as $g(\boldsymbol{x}) = \|\boldsymbol{D}\boldsymbol{x}\|_{\boldsymbol{\Gamma}}^2$. From a Bayesian perspective, it means that for a certain transform $\boldsymbol{D}$, the transformed signal $\boldsymbol{D}\boldsymbol{x}$ is normally distributed with a covariance matrix $\boldsymbol{\Gamma}^{-1}$. In the case where the noise is also assumed normally distributed with a covariance matrix $\boldsymbol{\Sigma}$, Equation (2.4.1) becomes

$$\underset{\boldsymbol{x}}{\operatorname{argmin}} \left\{ \|\boldsymbol{P}\boldsymbol{x} - \boldsymbol{d}\|_{\boldsymbol{\Sigma}^{-1}}^2 + \|\boldsymbol{D}\boldsymbol{x}\|_{\boldsymbol{\Gamma}}^2 \right\} \tag{2.4.2}$$

A complete proof of Equation (2.4.2) can be found in Appendix 6.1.2. For the sake of simplicity, the covariance matrix $\boldsymbol{\Gamma}$ is often diagonal, i.e $\Gamma = \lambda I$ for some $\gamma > 0$, so that the optimization problem is $\underset{\boldsymbol{x}}{\operatorname{argmin}} \left\{ \|\boldsymbol{P}\boldsymbol{x} - \boldsymbol{d}\|_{\boldsymbol{\Sigma}^{-1}}^2 + \lambda \|\boldsymbol{D}\boldsymbol{x}\|_2^2 \right\}$

Intuitively, the Tikhonov regularization enforces some *smoothness* in the solution. Indeed, the non-regularized least-squares problem amounts to minimizing the distance between the projection of the solution and the acquired data; while the regularized approach entails to minimize two terms. If $\boldsymbol{D} = \boldsymbol{I}$, the second term is proportional to the sum of the

squared magnitudes of the solution; thus, large component values will have a significant contribution. Therefore, this prior will penalize *outliers* in the solution components.

A closed-form solution of problem (2.4.2) can be found by applying the first-order optimality condition. Setting to zero the gradient of the total objective function gives

$$\begin{aligned} \mathbf{0} &= \boldsymbol{P}^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{P}\boldsymbol{x}_{\text{MAP}} - \boldsymbol{d}) + \boldsymbol{D}^T\boldsymbol{\Gamma}\boldsymbol{D}\boldsymbol{x}_{\text{MAP}} \\ \boldsymbol{x}_{\text{MAP}} &= \left(\boldsymbol{P}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{P} + \boldsymbol{D}\boldsymbol{\Gamma}\boldsymbol{D}\right)^{-1}\left(\boldsymbol{P}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{d}\right) \end{aligned} \tag{2.4.3}$$

which is a *regularized* pseudo-inverse of $\boldsymbol{P}$. The solution of a Tikhonov-regularized least-squares problem is therefore obtained by a filtering (linear) process.

On the statistical point of view, Tikhonov regularization is equivalent to assuming that $\boldsymbol{D}\boldsymbol{x}$ is normally distributed (possibly with a covariance matrix). The operator $\boldsymbol{D}$ is often chosen as the spatial gradient ∇ (Defined in Appendix 6.2.1). In this case, the appropriate assumption would be a Rayleigh distribution (when using $\sqrt{g_x^2 + g_y^2}$ to compute the magnitude of the gradient components (g_x, g_y)) or the sum of half-normal distributions (when using $|g_x| + |g_y|$ to compute the gradient magnitude).

2.4.3 ℓ_1 regularization

Another regularization type consists in replacing the ℓ_2 norm with the ℓ_1 norm. Problem (2.4.2) becomes

$$\underset{\boldsymbol{x}}{\operatorname{argmin}} \left\{ \|\boldsymbol{P}\boldsymbol{x} - \boldsymbol{d}\|^2_{\boldsymbol{\Sigma}^{-1}} + \lambda \|\boldsymbol{D}\boldsymbol{x}\|_1 \right\} \tag{2.4.4}$$

which corresponds, in the Bayesian framework, to the MAP estimator when the latent signal is known to follow a Laplace distribution with parameter $\lambda > 0$ in the basis $\boldsymbol{D}$[7]. Problem (2.4.4), sometimes called *basis pursuit*, admits no closed-form solution because of the non-differentiable ℓ_1 term. Compared to the Tikhonov regularization, ℓ_1 regularization is less sensitive to large components values in the solution, as the regularization is proportional to the absolute value of the components rather than their squares.

If $\boldsymbol{D} = \nabla$ is the gradient operator, mapping one image to two images (the derivative along both axes), the regularization $\|\nabla\boldsymbol{x}\|_1$ is known as the Total Variation: it is the sum of the derivatives magnitudes. Using the ℓ_1 norm has found many successful applications for inverse problems, which can partly be explained with the *compressive sensing* framework. Simply put, using the ℓ_1 regularization both brings stability in the reconstruction process[8] and promotes *sparsity* of the solution in some basis.

2.4.4 The case for using ℓ_2 norm for the data fidelity term

As seen in section 2.2.1, using a squared ℓ_2 norm for the data fidelity term makes the underlying assumption that the noise is Gaussian. When acquiring projection data, however, the statistical process of "photon counting" is known to follow a Poisson distribution. Therefore, one might think that a Kullback-Leibler divergence should be used to measure the distance between the data and the projection of the estimate. However, using the Gaussian law has two crucial advantages:

- Mathematical advantage: derivations are easier, and in fact, the Gaussian distribution is almost the only one for which all the results can be derived analytically.

[7] this can easily be seen from Proof (2.4.2) by replacing the Gaussian prior on $\boldsymbol{x}$ with a Laplace prior

[8] in the sense that running an optimization algorithm until convergence will not amplify the noise present in the input data

The quadratic term resulting of the (negative) log-likelihood differentiates nicely to a successive evaluation of linear operators.

- Computational advantage: optimization algorithms can be very efficient to minimize a quadratic term rather than, for example, the sum of logarithmic terms. Besides, the quadratic term does not have pitfalls like non-differentiability or divergence behaviour near zero.

Having this in mind, there are also reasons supporting the fact that it is not statistically valid to use a Poisson distribution as an underlying process for $\boldsymbol{P}\boldsymbol{x} - \boldsymbol{d}$.

The first reason is that the sinogram $\boldsymbol{d}$ is defined as $-\log(\boldsymbol{p}/\boldsymbol{p}_0)$ (elementwise) where $\boldsymbol{p}$ is the projection data[9] and $\boldsymbol{p}_0$ is a measure of the incoming beam ("flat-field"). The projection data $\boldsymbol{p}$ might follow a Poisson distribution, but it does not need to be true for the negative logarithm of $\boldsymbol{p}/\boldsymbol{p}_0$. If the photon count of the projection data $\boldsymbol{p}$ is similar to the photon count of the incoming beam $\boldsymbol{p}_0$ (which is usually true), then $\boldsymbol{p}/\boldsymbol{p}_0 \sim 1$, so $-\log(\boldsymbol{p}/\boldsymbol{p}_0) \sim 1 - \boldsymbol{p}/\boldsymbol{p}_0$. Therefore, what matters is the distribution of the ratio $\boldsymbol{p}/\boldsymbol{p}_0$ rather than $\boldsymbol{p}$ itself. Assuming that both $\boldsymbol{p}$ and $\boldsymbol{p}_0$ follow a Poisson distribution, the ratio is more complicated.

The second reason is that the projection data itself might not follow a Poisson distribution. Although direct photon counting detectors yield a Poisson statistics, detectors used with scintillators may follow another distribution. Measurements of the Fano factor[10] on various scintillators found that the statistical process is actually sub-Poisson [Bou+10] [Bor+16].

The third reason is that the noise model ($f(\boldsymbol{x})$ in Equation 2.1.5) is much less important, in practice, than the choice of the prior ($g(\boldsymbol{x})$). For example, the SIRT method often provides results similar to those of a (non-weighted) least-squares reconstruction, and using the EM method does not bring significant differences (except for the positivity constraint). A thorough study of the noise factors for CT [Nuy+13] suggest that noise models more complicated than Gaussian-Poisson are intractable and do not bring significant improvement in iterative reconstructions. By contrast, choosing a Tikhonov regularization or a total variation regularization (see section 2.5.1) yields in very different results. Advanced works on the modelling of iterative regularized reconstruction (see for example [WMG17] and references therein) focus on the estimating the regularization parameter of the prior, rather than on non-Gaussian noise priors, which gives confidence in our approach.

For these reasons, the ℓ_2 norm will always be used in this work in the data fidelity term, which is consistent with goal of having the best compromise between the modelling accuracy and the computational tractability.

2.4.5 The Compressed Sensing framework

Compressive Sensing (CS) is a framework dealing with both the acquisition and the reconstruction of a signal with few measurements. CS was formalized from 2004 mainly by Donoho, Candes, Romberg and Tao [CRT06b] [CRT06a] [Don06]. The starting point of this formalization was a "puzzling numerical experiment" where it was observed that an image could be *exactly* reconstructed from very few of its tomographic projections using the Total Variation regularization; that is, solving 2.4.4 with $\boldsymbol{D} = \nabla$ (Figures 2.6.1, 2.4.2).

[9] more precisely, the line k_0 of the sinogram $\boldsymbol{d}$ corresponds to the (negative logarithm of) concatenation the line k_0 of all projections. In parallel geometry, the 3D sinogram is the transpose of the projection data.

[10] the ratio between the variance and the mean, being 1 for a Poisson process

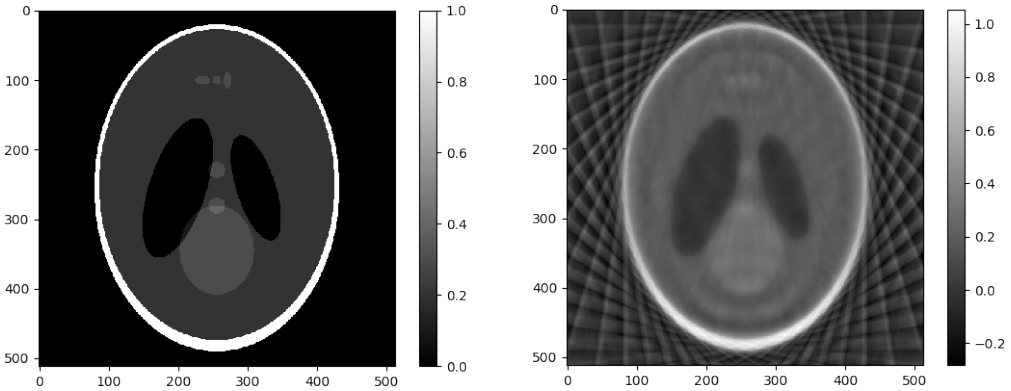


Figure 2.4.1: (a) Shepp-Logan phantom, 512×512 pixels (b) Least-squares reconstruction of noiseless sinogram $\boldsymbol{d} = \boldsymbol{P}\boldsymbol{x}$ acquired with 22 projection angles

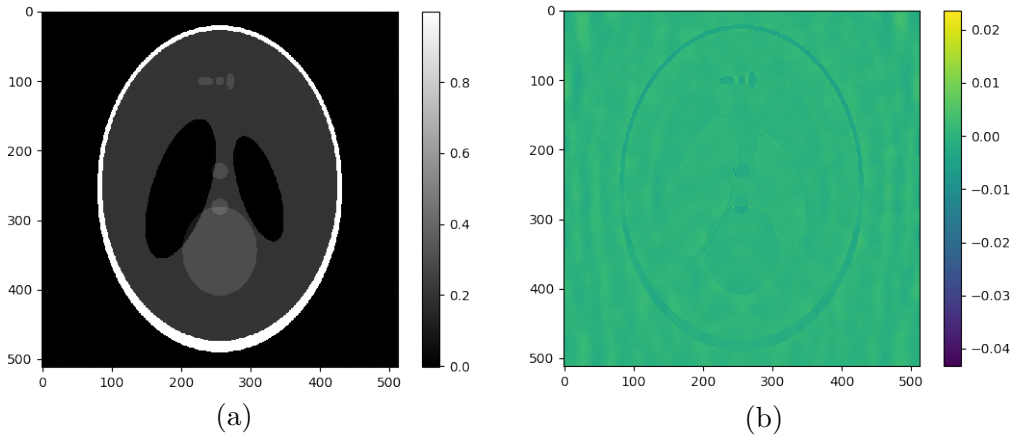


Figure 2.4.2: (a) Reconstruction of noiseless sinogram $\boldsymbol{d} = \boldsymbol{P}\boldsymbol{x}$ by solving 2.4.4 with $\boldsymbol{D} = \nabla$ and $\lambda = 8 \cdot 10^{-3}$ (b) Difference between the TV reconstruction and the Shepp-Logan phantom

The main contribution of the CS framework was to understand why such exact recoveries are possible, despite being in contradiction with the Nyquist sampling criterion, and to provide theoretical guarantees and conditions. Let $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ be a LIP where $\boldsymbol{y}$ is the observed data, $\boldsymbol{A}$ is the acquisition matrix and $\boldsymbol{x}$ is the latent signal to recover. Loosely, the CS theory states that if both the sensing matrix $\boldsymbol{A}$ and the latent signal $\boldsymbol{x}$ have particular properties, then $\boldsymbol{x}$ can be accurately recovered with a convex optimization program involving the ℓ_1 regularization.

More precisely, a sufficient condition on the recovery guarantee of $\boldsymbol{x}$ is given by considering the *sparsity* of $\boldsymbol{x}$ and the *restricted isometry constraint* of $\boldsymbol{A}$. A vector $\boldsymbol{c}$ is said to be s-sparse with $s \geq 0$ if it has at most s nonzero components. A matrix $\boldsymbol{A}$ is said to verify the *Restricted Isometry Constraint* with constant $\delta_s > 0$ if

$$\forall\ T \leq s,\ \forall\ \boldsymbol{c}, \quad (1 - \delta_s)\,\|\boldsymbol{c}\|_2^2 \leq \|\boldsymbol{A}_T\boldsymbol{c}\|_2^2 \leq (1 + \delta_s)\,\|\boldsymbol{c}\|_2^2 \tag{2.4.5}$$

where $\boldsymbol{A}_T$ denotes the sub-matrix formed from $\boldsymbol{A}$ by keeping T columns. Equation (2.4.5)

characterizes how the matrix $\boldsymbol{A}$ behaves like an isometry, i.e how it shrinks or expands the distances.

The *sparsity* of a signal simply the measure of its number of non-zero components. A signal is said to be s-sparse if it has at most s non-zero components. It was shown that in general, minimizing the ℓ_0 norm is a hard combinatorial problem. Fortunately, this "norm" can be replaced by a *convex relaxation*, the ℓ_1 norm. Theoretical guarantees were stated that for large underdetermined systems of linear equations, the minimal ℓ_1 norm solution is also the sparsest solution [Don06]. Other sparsity-inducing norms (ℓ_p norms with $p > 1$ and a kink at the origin) can also be used [Bac+12].

Then, given the s-sparsity of a signal $\boldsymbol{x}$, Equation (2.4.5) states that a sufficient δ_s exist for $\boldsymbol{A}$ in order to recover $\boldsymbol{x}$ with a convex optimization program [CT06] [CR07]. This guarantee is however difficult to check in practice. On the one hand, signal of interest are seldom sparse, even in a basis, as the numerical equality to zero is hard to reach. On the other hand, the restricted isometry constraint is hard to compute, as it involves a combinatorial search. For this reason, CS guarantees are refined by involving *compressible* signals rather than sparse signals, and by providing another sufficient condition on the matrix $\boldsymbol{A}$ called *coherence*. These two notions will be discussed in more details.

Definition 5 (Compressibility of a signal)

Let $\boldsymbol{c}$ denote a signal, or the coefficients of a signal in some representation. Without loss of generality, we will suppose that the components of $\boldsymbol{c}$ are sorted by decreasing order of magnitude. The *weak* ℓ_p ball of radius $R > 0$ is defined as

$$\left\{\boldsymbol{c},\ |c|_k \leq R \cdot k^{-1/p}\right\} \tag{2.4.6}$$

Then, $\boldsymbol{c}$ is said to be compressible if it belongs to a weak ℓ_p ball for some radius $R > 0$ and $0 < p < +\infty$.

The notion of compressibility measures how the entries of a signal (or signal representation) decay following a power law. For example, it is well-known that given a one-dimensional n-differentiable signal, the Fourier Transform samples magnitudes decay as $1/k^n$. The same holds for wavelet coefficients of piecewise-smooth signals [CT06].

The compressibility can be characterized by the *approximation error* of a signal representation) reconstructed from $K > 0$ largest coefficients in terms of magnitude. More precisely, letting $\boldsymbol{c}_K$ the vector containing the first K (sorted) coefficients and zeros otherwise, the ℓ_2 difference between $\boldsymbol{c}$ and $\boldsymbol{c}_K$ is bounded by

$$\|\boldsymbol{c} - \boldsymbol{c}_K\|_2 \leq C_p \cdot R \cdot K^{1/2-1/p} \tag{2.4.7}$$

where C_p is a constant depending on p. If the coefficients $\boldsymbol{c}$ are obtained from a signal $\boldsymbol{x}$ through an orthonormal transform or a tight frame (see 2.5.2), then the approximation error of the signal is bounded by

$$\|\boldsymbol{x} - \boldsymbol{x}_K\|_2 \leq \tilde{C}_p \cdot R \cdot K^{1/2-1/p} \tag{2.4.8}$$

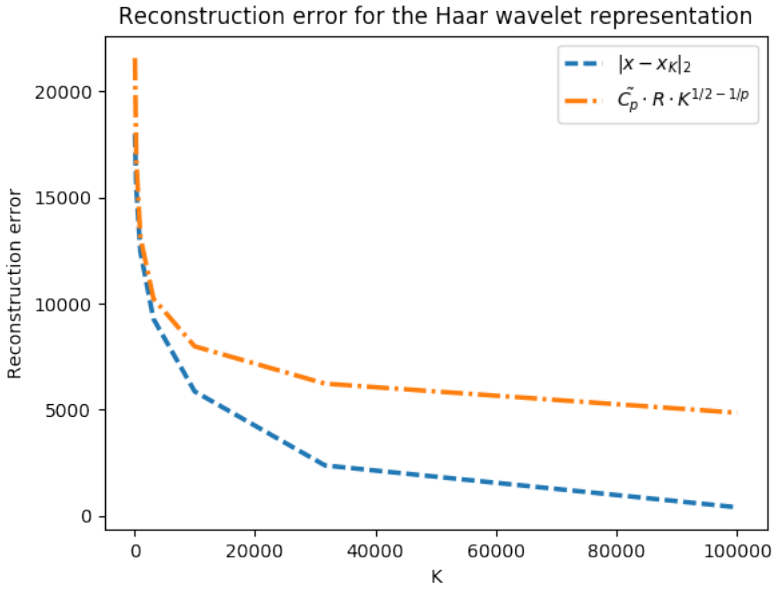


Figure 2.4.3: Comparison between the reconstruction error $\|\boldsymbol{x} - \boldsymbol{x}_K\|_2$ and the bound $\tilde{C}_p \cdot R \cdot K^{1/2-1/p}$ for the "ascent" test image in the Haar wavelet representation. For this image and this representation, $p = 1.397$, $R = 44.8 \cdot 10^3$ and $\tilde{C}_p = 1.3$. This figure can be reproduced with the jupyter notebook in [Pal17].

To summarize, compressible signals are signal that can be accurately approximated from few coefficient of a transform/frame, the approximation error decay following a power law as a function of the number of coefficients. Thus, compressible signals are a faithful approximation of sparse signals: setting to zero all except K coefficients results in an approximation error decaying very fast.

The coherence between two matrices, according to Definition 6, is the largest correlation coefficient between the lines [11] of two matrices.

Definition 6 (Coherence between two matrices)

Let $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{D} \in \mathbb{R}^{p \times n}$ be two matrices. The coherence between $\boldsymbol{A}$ and $\boldsymbol{D}$ is defined by

$$\mu(\boldsymbol{A}, \boldsymbol{D}) = \max_{\substack{1 \leq j \leq p \\ 1 \leq i \leq m}} \left\{ \frac{|\langle \boldsymbol{A}_i \, , \, \boldsymbol{D}_j \rangle|}{\|\boldsymbol{A}_i\|_2 \, \|\boldsymbol{D}_j\|_2} \right\} \quad \in [1, \sqrt{n}] \tag{2.4.9}$$

where $\boldsymbol{A}_i$ and $\boldsymbol{D}_j$ are the lines i and j of $\boldsymbol{A}$ and $\boldsymbol{D}$, respectively.

A theoretical recovery guarantee for sparse signals is then given by Theorem 2.4.1.

Theorem 2.4.1 ([CR07])

Let $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}^\sharp$ be a linear inverse problem. If the coefficient vector $\boldsymbol{w}$ of the latent

[11] the coherence is usually defined with columns of basis. The fact that the signal $\boldsymbol{x}$ is represented by the coefficients $\boldsymbol{c}$ in a basis Ψ is often denoted in a synthesis formulation $\boldsymbol{x} = \Psi\boldsymbol{c}$. In this manuscript, we adopt a *analysis* formulation, meaning that matrices denote *transforms* rather than *bases*. What is written is therefore $\boldsymbol{c} = \boldsymbol{D}\boldsymbol{x}$. For orthonormal transforms, it means that $\boldsymbol{D} = \boldsymbol{\Phi}^T$. Working with the lines of a transform is thus equivalent to working with the columns of the corresponding basis/frame.

signal $\boldsymbol{x}^\sharp \in \mathbb{R}^n$ is s-sparse in the basis $\boldsymbol{D}^T$, the solution of

$$\underset{\boldsymbol{w}}{\operatorname{argmin}} \left\{ \left\| \boldsymbol{A}\boldsymbol{D}^T\boldsymbol{w} - \boldsymbol{y} \right\|_2^2 + \lambda \left\| \boldsymbol{w} \right\|_1 \right\} \tag{2.4.10}$$

enables the unique and exact recover of $\boldsymbol{x}^\sharp$ with high probability, provided that $m \geq m_{\min}$ measurements are acquired uniformly at random, where

$$m_{\min} = C \cdot s \cdot \mu^2(\boldsymbol{A}, \boldsymbol{D}) \cdot \log(n) \tag{2.4.11}$$

The coherence enables to avoid the computation of the restricted isometry constraint. Practically, a low coherence pair $(\boldsymbol{A}, \boldsymbol{D})$ is desirable for reconstructing a signal from few measurements. Theorem 2.4.1 somehow extends the Nyquist sampling criterion: instead of reconstructing a signal from all its Fourier coefficients by the means of a linear process, CS aims at reconstructing a signal from *few* of its coefficients (in some basis) by the means of a nonlinear process.

More recent refinements of CS involve *over-complete dictionaries* (frames) rather than basis for the representation of the latent signal [Can+11], and do not rely on coherence. However, CS guarantees still rely on some randomness in the acquisition process, which can be impracticable in some contexts, especially in CT. Yet, good results are obtained in practice, suggesting that the theory needs to be completed.

2.4.6 How to choose the regularization parameter λ ?

Taking in consideration sections 2.4.4 and 2.4.5, the regularized inverse problem $\boldsymbol{P}\boldsymbol{x}^\sharp = \boldsymbol{d}$ (up to noise) is solved using the optimization program given by Equation (2.4.4). The use of a ℓ_2 norm in the data fidelity term has been discussed in section 2.4.4. The use of a ℓ_1 norm in the regularization term finds its justification in the compressed sensing framework (section 2.4.5), as the ℓ_1 norm acts as a proxy of the ℓ_0 "norm" for finding a sparse solution in a representation basis (section 2.5). Now, a question often raised in the context of regularized tomographic reconstruction is: *how to choose the regularization parameter* λ in Equation (2.4.4) ? Intuitively, choosing a too high λ results in a solution with too many regularization (for example a piecewise-constant solution in the case of total variation), while choosing a λ too small results in the solution not being regularized properly.

Several strategies were proposed to automatically find the optimal regularization parameter (henceforth denoted λ). The "L-curve" method [HO93] [Yan+15] consists in plotting the ℓ_1 norm of a solution as a function of its residual, parametrized with λ. More precisely, given a solution $\boldsymbol{x}_{\lambda_1}$ of Equation (2.4.4) with $\lambda = \lambda_1$, the quantities $\left\| \boldsymbol{P}\boldsymbol{x}_{\lambda_1} - \boldsymbol{d} \right\|_2^2$ (residual) and $\left\| \boldsymbol{x}_{\lambda_1} \right\|_1$ (regularization norm) are computed. Then, another solution $\boldsymbol{x}_{\lambda_2}$ is computed with $\lambda = \lambda_2$, and the two aforementioned quantities are computed. The same is done for several values of λ, and a curve of the ℓ_1 regularization as a function of the residual is plotted. Heuristically, this curve is shaped like a "L", and the inflexion point corresponds to the best value of λ. This method entails to perform several reconstruction in order to estimate λ, which is a drawback. Besides, the L-curve is only a heuristic, and we saw it fail in many cases. A similar criterion is given [Mir+14]: given a solution $\boldsymbol{x}_0$ computed with a *very small* regularization, the optimal λ is $\lambda^\sharp = \underset{\lambda}{\operatorname{argmin}} \{\cos(\boldsymbol{x}_\lambda - \boldsymbol{x}_0, \boldsymbol{x}_\lambda)\}$.

Another family of strategy is the use of Stein Unbiaised Risk Estimator (SURE). It consists of estimating the (unknown) mean square error $\mathbb{E}\left[\left\| \boldsymbol{x}_\lambda - \boldsymbol{x}^\sharp \right\|_2^2 \right]$ where $\mathbb{E}[\cdot]$ denotes

the mathematical expectation. This method is out of the scope of this work, a derivation in the case of the Iterative Shrinkage-Thresholding algorithm can be found in Appendix 6.4.1. We did not pursue this strategy as, like the L-curve method, it involves further computational burden.

Lastly, another approach consists in building a fully Bayesian model by considering λ as another random variable. A probability distribution function should therefore be assigned to λ. It can be shown that for the exponential family, the priors over λ such as the posterior probability also belongs to the exponential family (thus making the computations more convenient), called conjugate priors, is the Gamma distribution family. This approach is interesting as only one reconstruction is performed – instead of several reconstructions in the case of the L-curve, although the additional prior makes the objective function more complicated. An example of Joint Maximum A Posteriori (JMAP) technique solving the fully Bayesian inference is proposed in [WMG17]. A complete review of sparsity-enforcing priors and parameter selection in the Bayesian framework is proposed in [Moh12].

Ultimately, we did not pursue the automatic parameter selection goal for the following reasons:

- The regularization parameter value depends on the features the user wants to promote in the final reconstruction. As there is no universal automatic metric for this aim, the reconstruction is therefore user-tuned. For example, the regularization can be chosen to a unusually high value when the user knows that the reconstructed volume has to be segmented. One might argue that parameter selection is cumbersome, but the whole experiment (setup, scan, phase retrieval, reconstruction) involves user-tuned parameters.

- It entails more computational burden, when the iterative reconstruction is already in itself a costly process.

However, in order for the user to have an initial estimate of λ, we use the following empirical formula based on the Donoho universal threshold (in the context of wavelets regularization):

$$\hat{\lambda} = \operatorname{arcsinh}\left(\hat{\sigma}\sqrt{2\log N}\right) \tag{2.4.12}$$

where N is the total number of pixels in an image, and $\hat{\sigma}$ is an estimate of the standard deviation of the noise of an initial FBP reconstruction, using the estimator described in [Imm96]. Formula (2.4.12) does not have a rigorous mathematical justification, the rationale is to adapt λ as a function of the noise that might be present in a FBP reconstruction (the higher noise level, the higher λ should be). The purpose of the inverse hyperbolic sine is to shrink the Donoho threshold which is often overestimated in this context.

2.5 Sparse representations for tomography reconstruction

2.5.1 Gradient and Total variation

The gradient representation has been used for a long time in the context of signal and image denoising, based on the ROF model [ROF92]. Roughly, this model considers the Total Variation (TV) of a function $\boldsymbol{f}$, which is equal to the sum of the norms of its gradient: $\|\nabla \boldsymbol{f}\|_1$. Formally, the TV is defined for functions of *bounded variations*, which is out of the scope of this work. We first define the spatial gradient operator ∇ and its adjoint.

The definition of the discrete gradient and divergence operators can be found in Appendix 6.2.1. Notably, the spatial gradient is not *centered*, i.e the corresponding one-dimensional discrete convolution kernel is $[-1, 1]$ rather than $[-1, 0, 1]$. The reason is that the centered spatial gradient is less sensitive to the high frequencies (image edges); and edges are particularly important in total variation regularization.

Now that the gradient and divergence operators are defined, the TV operator can be defined.

Definition 7 (Total Variation)

Let $\boldsymbol{c} \in \mathbb{R}^n$, for example an image. The TV of $\boldsymbol{c}$ is defined by the norm of its gradient:

$$\mathrm{TV}(\boldsymbol{c}) = \|\nabla \boldsymbol{c}\|_1$$

For signal with two dimensions or more, the norm computation can be done in two ways. The *isotropic* total variation computes the $\ell_{2,1}$ norm of the gradient:

$$\mathrm{TV}_{\mathrm{iso}}(\boldsymbol{c}) = \sum_{i,j} \sqrt{(g^1_{i,j})^2 + (g^2_{i,j})^2}$$

where $(g_{i,j})^k$ denotes the component k of the gradient of $\boldsymbol{c}$ at the pixel location (i, j). The *anisotropic* total variation computes the ℓ_1 norm of the gradient:

$$\mathrm{TV}_{\mathrm{aniso}}(\boldsymbol{c}) = \sum_{i,j} |g^1_{i,j}| + |g^2_{i,j}|$$

The tomographic reconstruction problem with TV regularization is defined by

$$\underset{\boldsymbol{x}}{\operatorname{argmin}} \left\{ \frac{1}{2} \|\boldsymbol{P}\boldsymbol{x} - \boldsymbol{d}\|_2^2 + \lambda \|\nabla \boldsymbol{x}\|_1 \right\} \tag{2.5.1}$$

where $\boldsymbol{d}$ is the acquired sinogram and $\boldsymbol{P}$ is the projection operator. Problem (2.5.1) can be extended to any LIP by replacing $\boldsymbol{P}$ with any linear operator. The penalization term $\lambda \|\boldsymbol{x}\|_1$, as a convex relaxation of the ℓ_0 norm, promotes solutions such that $\nabla \boldsymbol{x}$ is sparse. Thus, solutions of Problem (2.5.1) are piecewise-constant signals. The higher regularization parameter $\lambda > 0$, the more piecewise-constant will be the solution.

The TV regularization has been successfully used for tomographic reconstruction for more than a decade, with a theoretical root given by the compressed sensing framework (see section 2.4.5).

2.5.2 Dictionary-based reconstruction

Regularized LIP, and particularly tomographic reconstruction, aim at solving problems of the form

$$\underset{\boldsymbol{x}}{\operatorname{argmin}} \left\{ \frac{1}{2} \|\boldsymbol{P}\boldsymbol{x} - \boldsymbol{d}\|_2^2 + \lambda \|\boldsymbol{D}\boldsymbol{x}\|_1 \right\} \tag{2.5.2}$$

where $\boldsymbol{D}$ is a linear operator. It promotes sparse coefficients of the transform $\boldsymbol{D}$ (or "in the $\boldsymbol{D}^T$ basis, see footnote 11). Problem (2.5.2) is in the form called *analysis formulation* of the reconstruction problem. Another form is the *synthesis formulation*

$$\underset{\boldsymbol{w}}{\operatorname{argmin}} \left\{ \frac{1}{2} \|\boldsymbol{P}\boldsymbol{D}^*\boldsymbol{w} - \boldsymbol{d}\|_2^2 + \lambda \|\boldsymbol{w}\|_1 \right\} \tag{2.5.3}$$

where the minimization holds on the *coefficients* $\boldsymbol{w}$ in the transform domain. It can be shown that formulations (2.5.2) and (2.5.3) are equivalent if and only if $\boldsymbol{D}$ is an orthonormal (or unitary) transform, i.e $\boldsymbol{D}^*\boldsymbol{D} = \boldsymbol{I}$. In general, however, these form are not equivalent.

Of particular interest is the case where the coefficients $\boldsymbol{w}$ have more component than the signal $\boldsymbol{x}$, meaning that $\boldsymbol{D}$ represented as a matrix has more rows than column. The transform $\boldsymbol{D}$ is said *over-complete*, as it maps $n \in \mathbb{N}$ components of the signal $\boldsymbol{x}$ to $n_2 > n$ components in the "$\boldsymbol{D}$ domain". In this case, the representation $\boldsymbol{D}^*$ is called a *dictionary* in the image processing terminology. Mathematically, $\boldsymbol{D}^*$ is called a *redundant frame*, defined as follows.

Definition 8 (Frame)

Let $\boldsymbol{D} \in \mathbb{R}^{n_2 \times n}$ be a linear operator mappings signals $\boldsymbol{x} \in \mathbb{R}^n$ to coefficients $\boldsymbol{w} \in \mathbb{R}^{n_2}$. Let $\boldsymbol{d}_k \in \mathbb{R}^n$ denote the n_2 lines of $\boldsymbol{D}$. The operator $\boldsymbol{D}$ represents a frame if there exists $0 < \mu_1 \leq \mu_2$ satisfying

$$\forall\, \boldsymbol{x} \in \mathbb{R}^n,\ \mu_1 \left\|\boldsymbol{x}\right\|_2^2 \leq \sum_k \left|\langle \boldsymbol{x}\,,\, \boldsymbol{d}_k \rangle\right|^2 \leq \mu_2 \left\|\boldsymbol{x}\right\|_2^2 \tag{2.5.4}$$

and this fact will be denoted by "$\boldsymbol{D}^T$ is a frame".
If $n_2 > n_1$, the frame $\boldsymbol{D}^T$ is said to be redundant.
If $\mu_1 = \mu_2$, the frame $\boldsymbol{D}^T$ is said to be tight. If $\mu_1 = \mu_2 = 1$, we recover a "Parseval's identity", i.e a signal has the same energy (ℓ_2 norm) in the identity basis as in the $\boldsymbol{D}^T$ frame.

The frames were defined over $\mathbb{R}^n$, but can be defined for a general Hilbert space replacing the transpose with the adjoint.

A frame can be seen as a generalization of a linear algebra basis, as the vectors $\boldsymbol{d}_k$ of $\boldsymbol{D}$ need not to be linearly independent. For example, in the redundant case, the columns of $\boldsymbol{D}^T$ are clearly linearly linked. In this setting, the output space of the operator $\boldsymbol{D}$ ($\mathbb{R}^{n_2}$) is bigger than its input space ($\mathbb{R}^n$). In particular, there can be vectors $\boldsymbol{w}$ which are not the transform $\boldsymbol{D}\boldsymbol{x}$ of any signal $\boldsymbol{x}$. This means that after solving Problem (2.5.3), the optimal $\hat{\boldsymbol{w}}$ might not correspond to any signal $\hat{\boldsymbol{x}}$!

Fortunately, for a tight frame, a signal can always be obtained from coefficients. Indeed, as the frame bounds are equal to $\mu > 0$, differentiating twice the Parseval's identity $\left\|\boldsymbol{D}\boldsymbol{x}\right\|_2^2 = \mu \left\|\boldsymbol{x}\right\|_2^2$ yields

$$\boldsymbol{D}^*\boldsymbol{D} = \mu \boldsymbol{I} \tag{2.5.5}$$

thus, the operator $\boldsymbol{D}$ is *semi-orthogonal*. The *non-square* matrix $\mu^{-1}\boldsymbol{D}^*$ can be viewed as a pseudo-inverse of $\boldsymbol{D}$ according to Equation (2.5.5)[12].

Dictionary-based reconstruction aims at solving Problem 2.5.3 where $\boldsymbol{D}^*$ is a *dictionary* (i.e redundant frame). The core idea is that a dictionary has been "learned" over a signal similar to the latent signal to reconstruct. Formally, if a good quality reconstructed volume $\boldsymbol{x}_0$ is available, for example after a scan with many views and a good SNR, a dictionary $\boldsymbol{D}^*$ is built based on $\boldsymbol{x}_0$ by the means of a *learning procedure*. This learning step can be viewed as a *sparse SVD* of the image (or volume) $\boldsymbol{x}_0$ in the sense that the most significant "atoms" of $\boldsymbol{x}_0$ are used to compute the dictionary $\boldsymbol{D}^*$.

[12] It is indeed easy to show that it corresponds to the minimum norm Moore-Penrose pseudo-inverse of $\boldsymbol{D}$, since $(\boldsymbol{D}^*\boldsymbol{D})^{-1}\boldsymbol{D}^* = \mu^{-1}\boldsymbol{I}\boldsymbol{D}^*$

The signal $\boldsymbol{x}$ is first split into subsets of smaller sizes. For example, if $\boldsymbol{x}$ is an image, it can be tiled as N (non-overlapping) *patches* of n pixels each. Let $\boldsymbol{X}$ denote the matrix obtained by stacking the N signals of size n. Each signal $\boldsymbol{x}_i$ has a representation as a coefficient vector $\boldsymbol{w}_i$ with M components. If $\boldsymbol{D}$ is the dictionary and g_i is the column i of $\boldsymbol{D}$, the representation reads

$$\underbrace{\begin{pmatrix} \boldsymbol{x}_1 & \dots & \boldsymbol{x}_N \end{pmatrix}}_{\begin{array}{c}\boldsymbol{X} \ : \ (n, N) \\ N \text{ signals} \\ \text{of } n \text{ components}\end{array}} = \underbrace{\begin{pmatrix} \boldsymbol{g}_1 & \dots & \boldsymbol{g}_M \end{pmatrix}}_{\begin{array}{c}\boldsymbol{D} \ : \ (n, M) \\ M \text{ atoms} \\ \text{of } n \text{ components}\end{array}} \cdot \underbrace{\begin{bmatrix} \boldsymbol{w}_1 & \dots & \boldsymbol{w}_N \end{bmatrix}}_{\begin{array}{c}\boldsymbol{W} \ : \ (M, N) \\ N \text{ coefficient vectors} \\ \text{of } M \text{ components}\end{array}} \tag{2.5.6}$$

Every signal $\boldsymbol{x}_i$ is a linear combination of the M dictionary atoms, as illustrated on Figure 2.5.1.

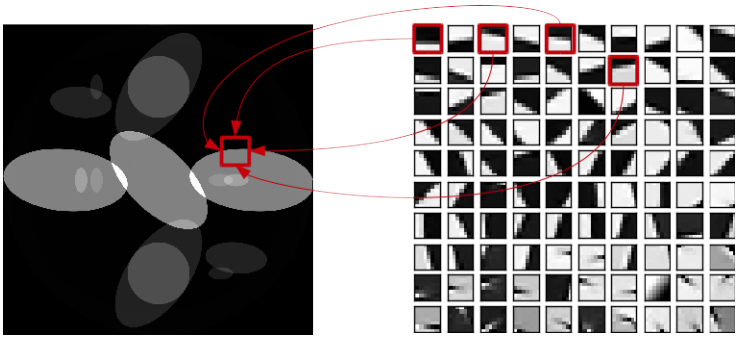


Figure 2.5.1: Left: image, Right: dictionary atoms. The image (signal) is split into a set of patches. Every image patch is a linear combination of the atoms. Image: [MBC14]

A commonly used dictionary learning algorithm is the K-SVD algorithm, or its "enhanced version" EK-SVD [MG08]. It consists in solving

$$\underset{\boldsymbol{D}, \boldsymbol{W}}{\text{argmin}} \left\{ \|\boldsymbol{X} - \boldsymbol{D}\boldsymbol{W}\|_2^2 \right\} \quad \text{s.t. } \|w_i\|_0 \leq T_0 \text{ for } i \in [1, N] \tag{2.5.7}$$

i.e finding a *sparse approximation* of the set of signals $\boldsymbol{X}$ in the dictionary $\boldsymbol{D}$ for the matrices $\boldsymbol{W}$ and $\boldsymbol{X}$ defined above. Problem 2.5.2 is non-convex and is usually solved with a greedy algorithm like the Matching Pursuit method.

Once the dictionary is built from a learning set of signals, the reconstruction problem (2.5.3) is solved with a standard convex optimization algorithm. A non-convex problem with the ℓ_0 norm can also be solved with a greedy algorithm [Xu+12].

In the work [MBC14], the formulation (2.5.3) is extended to *overlapping* image patches, with similar patches in the overlapping region, to yield a better reconstruction quality. This work is part of the PyHST reconstruction software [Mir+14].

2.5.3 Wavelets frames

A brief introduction to Wavelets in signal processing, with useful references, can be found in Appendix 6.2.2/ This work primarily focuses on two dimensional Wavelet transforms, i.e Discrete Wavelet Transform (DWT) of images. In this work, $\boldsymbol{W}$ denotes the forward

wavelet transform (wavelet analysis) [13] and $\boldsymbol{W}^*$ denotes its adjoint, which is also its inverse (possibly up to a multiplicative constant, see section 2.5.2). Wavelet transforms basically fall into two categories: the critically-sampled wavelet transform and undecimated wavelet transform. The former is often called DWT by default, whereas the latter is generally called Stationary Wavelet Transform (SWT) (or undecimated wavelet transform).

The DWT has many interesting properties:

- Most "natural" signals/images are compressible (in the sense of subsection 2.4.5) in a wavelet basis.
- It is a non-redundant transform: it maps a signal with n components (eg. image with n pixels) to a set of n coefficients. Therefore, this transform is not memory demanding.
- It can be implemented in a very efficient way, the "fast wavelets transform" (Mallat's multi-resolution method), a $O(n)$ algorithm.
- In certain conditions (see section 3.2), the transform $\boldsymbol{W}$ is orthonormal: $\boldsymbol{W}^*\boldsymbol{W} = \boldsymbol{I}$, which is of interest in optimization algorithms (see subsection 2.6.2).

The DWT, however, suffers from a drawback: the lack of shift invariance, meaning that the coefficients of a *shifted* version of a signal are not equal to the shifted version of the coefficients of the original signal. This lack of translation invariance can lead to artifacts when the wavelet coefficients are modified, i.e as soon as they are used in a regularized inverse problem. To address this issue, the redundant SWT can be used. It maps n components of a signal to $n_2 > n$ coefficients in the wavelet domain. In general, the SWT satisfies the tight frame condition

$$\boldsymbol{W}^*\boldsymbol{W} = \mu\boldsymbol{I} \tag{2.5.8}$$

i.e is a semi-orthogonal transform. This transform is however more computationally expensive and memory demanding. The implementation and capabilities will be discussed in section 3.2.

2.6 Notions of convex optimization

The previous sections described how the modelling of tomographic reconstruction as an inverse problem, with a model on the noise and the latent signal, leads to an optimization problem. The following sections will now deal with the *actual* solving of these optimization problems by considering state-of-the-art non-smooth minimization algorithms compatible with an efficient implementation.

This section 2.6 introduces the mathematical tools required for the optimization algorithms used in this work.

2.6.1 Convex functions and convex sets

Definition 9 (Convex function and convex set)

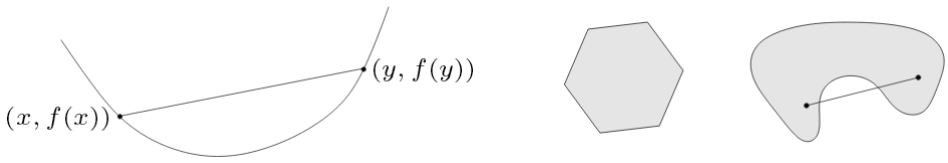
Let f be a function from some vector space $\boldsymbol{E}$ to $\mathbb{R}$. The function f is said to be convex if

$$\forall\, \boldsymbol{x}_1, \boldsymbol{x}_2 \in E,\ \forall t \in [0,1], \quad f(t\boldsymbol{x}_1 + (1-t)\boldsymbol{x}_2) \leq tf(\boldsymbol{x}_1) + (1-t)f(\boldsymbol{x}_2)$$

[13] in many other works, $\boldsymbol{W}$ rather denotes the *wavelet synthesis*, corresponding to $\boldsymbol{W}^T*$ in this context

The set $\boldsymbol{E}$ is said to be convex if

$$\forall \boldsymbol{x}_1, \boldsymbol{x}_2 \in \boldsymbol{E},\ \forall t \in \mathbb{R}, \quad t \in [0,1] \Longrightarrow t\boldsymbol{x}_1 + (1-t)\boldsymbol{x}_2 \in \boldsymbol{E}$$



(a) (b)

Figure 2.6.1: (a) 1D illustration of a convex function. For any couple of points x, y, the straight line between x and y is always above the curve of the function. (b) Example of a convex set (left) and a non-convex set (right). For a convex set $\boldsymbol{E}$, any straight line between $\boldsymbol{x}, \boldsymbol{y} \in \boldsymbol{E}$ is contained in $\boldsymbol{E}$. Images: [BV09]

The minimization of a convex function over a convex set is called a *convex problem*. Proposition 2.6.1 lists useful properties of convex functions. Proofs and further properties can be found in chapter 3 of [BV09].

Proposition 2.6.1 (Properties of a convex function)
Let f be a function defined on a convex set. The following properties hold:

- If f is differentiable, then f is convex if and only if it is greater than its affine approximation:
$$\forall \boldsymbol{x}, \boldsymbol{y}, \quad f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^T(\boldsymbol{y} - \boldsymbol{x}) \tag{2.6.1}$$
- If f is a twice differentiable, then f is convex if and only if its Hessian is positive semi-definite:
$$\forall \boldsymbol{x}, \quad \nabla^2 f(\boldsymbol{x}) \succeq 0 \tag{2.6.2}$$
- If f is convex, any local minimum of f is a global minimum.
- If f is convex, the set of global minima of f is convex.

Convex functions are of special interest in optimization, as converging to a local minimum is equivalent to converging to a global minimum. In many problems, a physical model entails the minimization of an *energy*, or objective function, which is generally convex. More specifically, linear inverse problems involve a data fidelity term which is the norm of an affine function (the difference $\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}$ between the observations and the projected estimate), which is convex for any norm.

The (unconstrained) linear Least Squares minimization

$$\underset{\boldsymbol{x}}{\operatorname{argmin}} \left\{ \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 \right\} \tag{2.6.3}$$

is a convex problem, as the *squared* ℓ_2 norm is convex[14]. Its solution, if any, is then unique. Problem (2.6.3) happens to have a solution (even when $\boldsymbol{A}$ does not have full rank), which makes the least squares approach appealing. The same holds for weighted ℓ_2 norms $\|\cdot\|_{\boldsymbol{\Gamma}}$ provided that $\boldsymbol{\Gamma}$ is positive semi-definite, which is the case for covariance matrices.

Linear Least Squares problems regularized with the ℓ_1 norm in some basis

$$\underset{\boldsymbol{x}}{\operatorname{argmin}} \left\{ \frac{1}{2} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_{\boldsymbol{\Gamma}}^2 + \lambda \|\boldsymbol{D}\boldsymbol{x}\|_1 \right\} \quad \lambda > 0 \tag{2.6.4}$$

are also convex, since the objective function is the sum of the convex (weighted) ℓ_2 norm and the convex ℓ_1 norm. Therefore, problem (2.6.4) also has an unique solution. More generally, the problem

$$\underset{\boldsymbol{x}}{\operatorname{argmin}} \left\{ \frac{1}{2} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_{\boldsymbol{\Gamma}}^2 + \sum_{k=0}^{n} \lambda_k \boldsymbol{\varphi}_k(\boldsymbol{x}) \right\} \quad \lambda_k > 0 \tag{2.6.5}$$

where $\boldsymbol{\varphi}_k$ are convex functions encoding some constraints, is convex.

Apart from being *regularized*, optimization problems can be *constrained*, meaning that the solution can be involved in an equality or inequality constraint. Such problems can be written under the form

$$\begin{aligned} &\underset{\boldsymbol{x}}{\operatorname{argmin}} \ \{f(\boldsymbol{x})\} \\ &\text{s. t. } \boldsymbol{x} \in \Omega \end{aligned} \tag{2.6.6}$$

where Ω is a convex set. The constraint $\boldsymbol{x} \in \Omega$ can be encoded by a function called *indicator function*.

Definition 10 (Indicator function of a set)

Let Ω be a set. The indicator function of Ω is defined by

$$i_\Omega(\boldsymbol{x}) = \begin{cases} 0 & \text{if } \boldsymbol{x} \in \Omega \\ +\infty & \text{otherwise} \end{cases} \tag{2.6.7}$$

It can be shown that indicator functions of convex sets are convex. Problem (2.6.6) can be rewritten

$$\underset{\boldsymbol{x}}{\operatorname{argmin}} \ \{f(\boldsymbol{x}) + i_\Omega(\boldsymbol{x})\} \tag{2.6.8}$$

Lastly, we give the definition of a Lipschitz function. Lipschitz functions are not directly linked to convex functions, but this property is frequently used in optimization algorithms.

Definition 11 (Lipschitz function)

Let $\boldsymbol{f}$ be a vectorial function. $\boldsymbol{f}$ is said to be β-Lipschitz if

$$\forall\, \boldsymbol{x}, \boldsymbol{y},\ \|\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{y})\| \leq \beta \|\boldsymbol{x} - \boldsymbol{y}\| \tag{2.6.9}$$

[14] he least squares problem $\|\boldsymbol{y} - \mathcal{A}\boldsymbol{x}\|_2^2$, where $\mathcal{A}$ is not linear, is *not* convex in general. In this case, this problem suffers from the non-convexity drawback: lack of convergence guarantees and local minima.

2.6.2 The proximity operator

Before defining the proximity operator, we introduce the Fenchel conjugate which is a tool commonly used in convex optimization.

Definition 12 (Fenchel conjugate)
The Fenchel conjugate of a function f is defined by

$$f^*(\boldsymbol{y}) = \sup_{\boldsymbol{x} \in \mathrm{dom} f} \left\{ \boldsymbol{y}^T \boldsymbol{x} - f(\boldsymbol{x}) \right\} \tag{2.6.10}$$

The function f^* is convex, even if f is not. The mapping $f \mapsto f^*$ for differentiable f is called Legendre Transform.

If f is a convex function then $f^{**} = (f^*)^* = f$. Other properties and examples of Fenchel conjugate are given in [BV09]. Of special interest are Fenchel conjugate of ℓ_p norms, which are given by Proposition (2.6.2).

Proposition 2.6.2 (Fenchel conjugate of power ℓ_p norms)
Let $1 < p < +\infty$ and q such that $\frac{1}{p} + \frac{1}{q} = 1$.
The Fenchel conjugate of $\boldsymbol{x} \mapsto \frac{1}{p} \|\boldsymbol{x}\|_p^p$ is $\boldsymbol{x} \mapsto \frac{1}{q} \|\boldsymbol{x}\|_q^q$. Notably, the half squared ℓ_2 norm is self-conjugate.
For $p = 1$, the conjugate of $\boldsymbol{x} \mapsto \|\boldsymbol{x}\|_1$ is the indicator of the ℓ_∞ unit ball.
For $p = +\infty$, the conjugate of $\boldsymbol{x} \mapsto \|\boldsymbol{x}\|_\infty$ is the indicator of the ℓ_1 unit ball.
For $1 < p < +\infty$, the conjugate of $\boldsymbol{x} \mapsto \|\boldsymbol{x}\|_p$ is the indicator of the ℓ_q unit ball.

The Fenchel conjugate is closely linked to the *dual problem*. Let the optimization problem

$$\min_{\boldsymbol{x}} \{ f(\boldsymbol{x}) + g(\boldsymbol{K}\boldsymbol{x}) \} \tag{2.6.11}$$

In many applications, both f and g are convex. Since g is convex, we have $(g^*)^* = g$, so we can write

$$\begin{aligned} g(\boldsymbol{K}\boldsymbol{x}) &= \max_{\boldsymbol{y}} \{ \langle \boldsymbol{K}\boldsymbol{x}\,,\, \boldsymbol{y} \rangle - g^*(\boldsymbol{y}) \} \\ &= \max_{\boldsymbol{y}} \{ \langle \boldsymbol{x}\,,\, \boldsymbol{K}^*\boldsymbol{y} \rangle - g^*(\boldsymbol{y}) \} \end{aligned}$$

The optimization problem is thus

$$\min_{\boldsymbol{x}} \left\{ \max_{\boldsymbol{y}} \{ \langle \boldsymbol{x}\,,\, \boldsymbol{K}^*\boldsymbol{y} \rangle - g^*(\boldsymbol{y}) + f(\boldsymbol{x}) \} \right\} \tag{2.6.12}$$

Problem (2.6.12) is called *saddle-point* formulation of Problem (2.6.11). Under assumptions on the original problem (2.6.11) (Slater's conditions [BV09]), the min and max operators can be exchanged. These conditions are usually met, for example if f is convex (which is the case for the squared ℓ_2 norm), the strong duality holds. The problem is then

$$\max_{\boldsymbol{y}} \left\{ \min_{\boldsymbol{x}} \{ \langle \boldsymbol{x}\,,\, \boldsymbol{K}^*\boldsymbol{y} \rangle - g^*(\boldsymbol{y}) + f(\boldsymbol{x}) \} \right\}$$

Now having a closer look on the term

$$\begin{aligned}\min_{\boldsymbol{x}} \{\langle \boldsymbol{x}\,,\, \boldsymbol{K}^*\boldsymbol{y}\rangle + f(\boldsymbol{x})\} &= \min_{\boldsymbol{x}} \{-\langle \boldsymbol{x}\,,\, -\boldsymbol{K}^*\boldsymbol{y}\rangle + f(\boldsymbol{x})\} \\ &= -\max_{\boldsymbol{x}} \{\langle \boldsymbol{x}\,,\, -\boldsymbol{K}^*\boldsymbol{y}\rangle - f(\boldsymbol{x})\} \\ &\triangleq -f^*(-\boldsymbol{K}^*\boldsymbol{y})\end{aligned}$$

the problem becomes $\max_{\boldsymbol{y}} \{-f^*(-\boldsymbol{K}^*\boldsymbol{y}) - g^*(\boldsymbol{y})\}$, also written

$$\min_{\boldsymbol{y}} \{f^*(-\boldsymbol{K}^*\boldsymbol{y}) + g^*(\boldsymbol{y})\} \tag{2.6.13}$$

which is the dual problem of (2.6.11).

The last concept we need to define the proximal operator is the subgradient, which is a generalization of the gradient of a differentiable function.

Definition 13 (subgradient)

Let $f : E \to \mathbb{R}$ be a convex function where E is a Hilbert space. The vector $\boldsymbol{g}_x \in E$ is a subgradient of f at $\boldsymbol{x}$ if

$$\forall\, \boldsymbol{y}, \quad f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \boldsymbol{g}_x^T(\boldsymbol{y} - \boldsymbol{x})$$

If the set of all subgradients of f is denoted ∂f, thus, ∂f is a *set-valued* function. If ∂f at $\boldsymbol{x}$ has only one vector $\boldsymbol{g}_x$, then f is differentiable at $\boldsymbol{x}$ and its gradient at $\boldsymbol{x}$ is $\boldsymbol{g}_x$.

Subgradient are set-valued at points where a function is not differentiable. A notable example is given the ℓ_1 norm:

$$(\partial \|\cdot\|_1)(\boldsymbol{x})_i = \begin{cases} \operatorname{sign}(x_i) & x_i \neq 0 \\ [-1, 1] & x_i = 0 \end{cases} \tag{2.6.14}$$

Intuitively, the subgradient generalizes the gradient of a differentiable function, according to the first point of Proposition (2.6.1). For a differentiable function f, the first order optimality condition (or Fermat rule) is $\nabla f = \mathbf{0}$. Now, if f is not differentiable, we have to consider its subgradient instead of its gradient. If $\mathbf{0} \in \boldsymbol{g}_x = \partial f$, then from the definition: $\forall\, \boldsymbol{y}, \quad f(\boldsymbol{y}) \geq f(\boldsymbol{x})$, i.e $\boldsymbol{x}$ is a *global minimizer* of f. Thus, the first order optimality condition is extended to the subgradient.

The subgradient and the following notion of proximal mapping of an operator are part of the modern theory of convex analysis and monotone operators theory. This theory is out of the scope of this manuscript, only notions of interest for this work will be given. An comprehensive presentation can be found in [BC11a]. Many theoretical guarantees hold for convex functions that are *lower semi-continuous* (l.s.c), i.e having a *closed epigraph*[15]. As it is the case in practice for many functions of interest (for example continuous functions and indicator of convex sets), **convex functions will also be assumed to be lower semi-continuous**, unless explicitly indicated.

The proximity operator can now be defined.

[15]The epigraph of a function f is defined by $\{(\boldsymbol{x}, \boldsymbol{y}) \in \mathrm{dom} f \times \mathbb{R},\ f(\boldsymbol{x}) \leq \boldsymbol{y}\}$

Definition 14 (Proximity operator)
Let f be a convex function. The proximity operator (or proximal mapping) of f is defined by

$$\operatorname{prox}_f(\tilde{\boldsymbol{x}}) = (\partial f + \boldsymbol{I})^{-1}(\tilde{\boldsymbol{x}})$$

For convex functions, the proximal is well-defined and unique, and equal to

$$\operatorname{prox}_f(\tilde{\boldsymbol{x}}) = \underset{\boldsymbol{x}}{\operatorname{argmin}} \left\{ \frac{1}{2}\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\| + f(\boldsymbol{x}) \right\}$$

Proof. Only the equivalence between the proximal formulas is proved. The first order optimal condition of $\underset{\boldsymbol{x}}{\operatorname{argmin}} \left\{ \frac{1}{2}\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\| + f(\boldsymbol{x}) \right\}$ is :

$$\boldsymbol{0} \in \boldsymbol{x} - \tilde{\boldsymbol{x}} + \partial f(\boldsymbol{x}) \iff \tilde{\boldsymbol{x}} \in (\boldsymbol{I} + \partial f)(\boldsymbol{x}) \iff \boldsymbol{x} = (\boldsymbol{I} + \partial f)^{-1}(\tilde{\boldsymbol{x}})$$

Where the last equivalence uses the fact that f is convex, so its subgradient defines an injective mapping, which is invertible on its proper domain. Hence the argument of $\min_{\boldsymbol{x}} \left\{ \frac{1}{2}\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\| + f(\boldsymbol{x}) \right\}$ is $\operatorname{prox}_f(\tilde{\boldsymbol{x}})$. □

Computation rules (like translation and scaling) exist for the Fenchel conjugate and the proximity operator, but they will only be stated when needed.

The alternative form of the proximal mapping of $f(\boldsymbol{x})$ as a minimization problem brings the following insight: *computing the proximal mapping amounts to solving a denoising problem* regularized with $f(\boldsymbol{x})$. If a function $f(\boldsymbol{x})$ is such that $\operatorname{prox}_f(\boldsymbol{x})$ is easily computed, then the denoising problem regularized with $f(\boldsymbol{x})$ can be solved efficiently (and conversely). A particular case is the combination of the last two lines of the following Proposition (2.6.3), which lists some useful proximity operators. A table of closed-form proximal mapping can be found in [CP09].

Proposition 2.6.3 (Table of common proximity operators)
This proposition gives a table of the proximal mappings of common functions.

$f(\boldsymbol{x})$	$\operatorname{prox}_f(\boldsymbol{x})$
$\frac{\gamma}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2$ $\gamma > 0$	$(\boldsymbol{I} + \gamma \boldsymbol{A}^*\boldsymbol{A})^{-1}(\boldsymbol{x} + \gamma \boldsymbol{A}^*\boldsymbol{y})$
$i_\Omega(\boldsymbol{x})$	$P_\Omega(\boldsymbol{x})$
$\varphi(\boldsymbol{A}\boldsymbol{x})$ $\boldsymbol{A}\boldsymbol{A}^* = \mu \boldsymbol{I}$	$\boldsymbol{x} + \mu^{-1}\boldsymbol{A}^*\left(\operatorname{prox}_{\mu\varphi}(\boldsymbol{A}\boldsymbol{x}) - \boldsymbol{A}\boldsymbol{x}\right)$
$\lambda\|\boldsymbol{x}\|_1$	$(S_\lambda(\boldsymbol{x}))_i = \max(\lvert x_i\rvert - \lambda, 0)\operatorname{sign}(x_i)$

where $P_\Omega(\boldsymbol{x})$ denotes the Euclidean projection onto Ω, and the operator $\mathcal{S}_\lambda$ is called *soft thresholding*.

An interesting prox is given by combining the last two lines of Proposition 2.6.3. Let $\boldsymbol{W}$ be an *orthogonal* (or unitary) transform, i.e $\boldsymbol{W}^*\boldsymbol{W} = \boldsymbol{I}$. Then, the prox of $\boldsymbol{x} \mapsto \|\boldsymbol{W}\boldsymbol{x}\|_1$ is

$$\operatorname{prox}_{\gamma\|\boldsymbol{W}\cdot\|_1}(\boldsymbol{x}) = \boldsymbol{W}^*\mathcal{S}_\gamma(\boldsymbol{W}\boldsymbol{x}) \tag{2.6.15}$$

In words, the prox consists in transforming $\boldsymbol{x}$ into the $\boldsymbol{W}$ domain, soft-thresholding the coefficients, and then transforming the thresholded coefficients back in the "$\boldsymbol{x}$ domain". This method has been used for a long time in the context of the Wavelets denoising. The *denoising problem* is an instance of a LIP

$$\underset{\boldsymbol{x}}{\operatorname{argmin}} \left\{ \frac{1}{2} \|\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda \|\boldsymbol{W}\boldsymbol{x}\|_1 \right\} \tag{2.6.16}$$

where $\boldsymbol{y}$ is the signal to denoise. Problem (2.6.16) is a least-squares approximation of $\boldsymbol{x}$ under the regularization of sparse coefficients in the $\boldsymbol{W}$ domain. The solution is, by definition, the proximal of $\boldsymbol{x} \mapsto \|\boldsymbol{W}\boldsymbol{x}\|_1$. Thus, the denoising problem regularized with an orthonormal Wavelets transform has a closed-form, one-step solution which is the proximal (2.6.15).

The proximal operator has numerous interesting properties. Appendix 6.2.3 gives properties which are used in this work; for a more comprehensive list, the reader might refer to [CP09].

The proximal operator is useful to generalize the Banach-Picard fixed point theorem: given a *contraction operator* $\mathcal{T}$ and two distances d, d'

$$\exists 0 \leq \nu < 1,\ \forall \boldsymbol{x}, \boldsymbol{y},\ d\left(\mathcal{T}(\boldsymbol{x}),\, \mathcal{T}(\boldsymbol{y})\right) \leq d'\nu\left(\boldsymbol{x},\, \boldsymbol{y}\right) \tag{2.6.17}$$

the operator $\mathcal{T}$ admits a fixed point $\tilde{\boldsymbol{x}}$ such that $\mathcal{T}(\tilde{\boldsymbol{x}}) = \tilde{\boldsymbol{x}}$, and the sequence defined by $\boldsymbol{x}_{n+1} = \mathcal{T}(\boldsymbol{x}_n)$ converge to $\tilde{\boldsymbol{x}}$[16]. Building such an operator in practice from a given problem is difficult, and raises the need to extend the principle. It can be shown that the proximal mapping of a convex function f is a *firmly nonexpansive* mapping in the sense

$$\left\|\operatorname{prox}_f\left(\boldsymbol{x}\right) - \operatorname{prox}_f\left(\boldsymbol{y}\right)\right\|_2^2 \leq \left\langle \boldsymbol{x} - \boldsymbol{y}\,,\, \operatorname{prox}_f\left(\boldsymbol{x}\right) - \operatorname{prox}_f\left(\boldsymbol{y}\right)\right\rangle \tag{2.6.18}$$

This property enables to define a *proximal point algorithm* defined by the iterates $\boldsymbol{x}_{n+1} = \operatorname{prox}_{\gamma_n f}\left(\boldsymbol{x}_n\right)$. If $\sum_{n=0}^{+\infty} \gamma_n = +\infty$, then these iterates converge (weakly) to a global minimizer of f, provided that f is convex ([BC11b], Theorem 27.1 p. 399).

Proximal point algorithms can be further extended to *proximal splitting algorithms* aiming at minimizing the sum of convex functions. Convergence guarantees of such methods is out of the scope of this manuscript, many such methods are described in [CP09].

2.7 Proximal optimization algorithms

Proximal algorithms aim at solving *non-smooth* optimization problems by using the proximal mapping. More precisely, let the optimization problem

$$\underset{\boldsymbol{x}}{\operatorname{argmin}}\ \{f(\boldsymbol{x}) + g(\boldsymbol{x})\} \tag{2.7.1}$$

where f and g are both convex. In the regularized inverse problem setting, f can be the data fidelity term and g can be the regularization term. Proximal method can mainly be split into three types, as summarized in Table 2.1

[16] the underlying space should be complete, which is the case for our usual setting of a finite dimensional vector space over $\mathbb{R}^n$

Name	Operators needed	Assumptions
Proximal gradient	∇f and $\text{prox}(g)$	f is differentiable, $\text{prox}(g)$ is simple
Douglas-Rachford	$\text{prox}(f)$ and $\text{prox}(g)$	Prox of f and g are simple
Primal-dual	$\text{prox}(F)$ and $\text{prox}(G^*)$ for some F and G	Problem is re-formulated

Table 2.1: Categories of optimization algorithms depending on the use of proximal

These categories, described in the following parts, have their own assumptions, advantages and drawbacks. Depending on the problem setting (the form of f and g), different algorithms should be used.

2.7.1 Gradient descent and first order methods

In optimization, first order methods are methods using only information on the gradient of the objective function to optimize. Before studying proximal methods, the gradient descent is briefly reviewed as an introduction. Gradient descent is one of the simplest optimization methods. Let f be a differentiable convex function to minimize, the gradient descent is defined by the iteration [17]

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \gamma_n \nabla f(\boldsymbol{x}_k) \tag{2.7.2}$$

where γ_n is the *step* in the (opposite) direction of the gradient. Intuitively, $\nabla f(\boldsymbol{x}_k)$ locally is the steepest direction of f at $\boldsymbol{x}$, pointing to an increasing of f, hence the negative sign for the minimization. Under assumptions on the sequence $(\gamma_k)_k$, iteration (2.7.2) converges to the global minimizer of f.

The step size is usually chosen with a fixed value $\gamma_k = \gamma$. It can be shown that if ∇f is β-Lipschitz (see Definition 11), then iteration (2.7.2) with constant step size $\gamma_k = \gamma$ converges if $0 < \gamma < 2/\beta$. If f is twice differentiable, the Lipschitz constant of ∇f is equal to the operator norm of its Hessian matrix $\nabla^2 f$ (i.e its largest eigenvalue for the standard Euclidean distance):

$$\beta = \sup_{\boldsymbol{x}} \left\{ \left\| \nabla^2 f \right\| \right\} \tag{2.7.3}$$

which can be efficiently computed with the *power method* given in Appendix 6.2.4.

An extension of the gradient descent called *projected gradient descent* is given by the following iteration

$$\boldsymbol{x}_{k+1} = P_\Omega(\boldsymbol{x}_k - \gamma_k \nabla f(\boldsymbol{x}_k)) \tag{2.7.4}$$

It aims at solving

$$\begin{aligned} &\underset{\boldsymbol{x}}{\text{argmin}} \ \{f(\boldsymbol{x})\} \\ &\text{s.t. } \boldsymbol{x} \in \Omega \end{aligned} \tag{2.7.5}$$

where Ω is a convex set.

Although simple, the gradient descent is notably slow. If $\boldsymbol{x}^\sharp$ denotes the minimizer of f, it can be shown [DT14] that at iteration n, the "inaccuracy" $f(\boldsymbol{x}_n) - f(\boldsymbol{x}^\sharp)$ is bounded by $\frac{\left\|\boldsymbol{x}_0 - \boldsymbol{x}^\sharp\right\|_2^2}{4n+2}$, i.e is $O(1/n)$. A Fast Gradient method was introduced by Nesterov with

[17] in this context, the notation $\boldsymbol{x}_k$ denotes the *iterate number* k of the vector $\boldsymbol{x}$, and not the *component number* k of $\boldsymbol{x}$. The ambiguity can be cleared by noticing that the estimate $\boldsymbol{x}_k$ is a vectorial entity (bold notation), while a component x_i is a scalar entity (non-bold notation).

iteration (2.7.6)

$$\begin{cases} \boldsymbol{x}_{k+1} = \boldsymbol{y}_k - \gamma \nabla f(\boldsymbol{y}_k) \\ t_{k+1} = \dfrac{1}{2}\left(1 + \sqrt{1 + 4t_k^2}\right) \\ \boldsymbol{y}_{k+1} = \boldsymbol{x}_{k+1} + \dfrac{t_k - 1}{t_{k+1}}(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k) \end{cases} \tag{2.7.6}$$

Intuitively, the Nesterov method adds a "momentum" term: the estimate $\boldsymbol{x}_k$ do not only follows the opposite gradient of f, it uses the information of the previous gradient. For example, the minimization of a quadratic form can be seen geometrically as rolling a ball in a "bowl" (ellipsoid). The gradient descent with a relaxation step is comparable to a "heavy momentum rolling ball" model.

This method has an inaccuracy decaying as $O(1/n^2)$, which can be shown to be optimal for any first-order method. In particular, any iteration using the information of the Q previous gradients $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \gamma \sum_{q=0}^{Q-1} h_{k+1,q} \nabla f(\boldsymbol{x}_k)$, for some factor $h_{k+1,q}$, will also have an inaccuracy bounded by $O(1/n^2)$ – which is a "good news", as storing many previous gradients would be expansive in terms of memory.

Further practical accelerations[18] of this method involve changing the relaxation step [CD15] or adding an new momentum term [KF16].

2.7.2 Proximal gradient algorithm

The proximal gradient method is directly linked to the gradient descent. While the gradient descent aims at minimizing a differentiable function f, the proximal gradient method aims at minimizing the sum of a differentiable function f and a convex (possibly non-differentiable) function g (Problem (2.7.1)).

One iteration of the basic proximal gradient method reads

$$\boldsymbol{x}_{k+1} = \operatorname{prox}_{\gamma g}\left(\boldsymbol{x}_k - \gamma \nabla f(\boldsymbol{x}_k)\right) \tag{2.7.7}$$

where $\gamma > 0$ is the descent step. It can be shown that this method converges when γ is chosen as the inverse of the Lipschitz constant of ∇f. When the function g encodes the constraint $\boldsymbol{x} \in \Omega$ for some convex set Ω, i.e $g(\boldsymbol{x}) = i_\Omega(\boldsymbol{x})$, then the proximal of g is $\operatorname{prox}_{\gamma g}(\boldsymbol{x}) = P_\Omega(\boldsymbol{x})$, the Euclidean projection onto Ω[19]. Thus, the proximal gradient algorithm can be seen as an extension of the projected gradient descent (2.7.4).

When the function g encodes a regularization about the sparsity of the coefficients of some *orthogonal* transform $\boldsymbol{W}$, i.e $g(\boldsymbol{x}) = \|\boldsymbol{W}\boldsymbol{x}\|_1$, then (see (2.6.15)) the prox consists in thresholding the coefficients of the $\boldsymbol{W}$ transform. As the gradient step $\boldsymbol{x}_k - \gamma \nabla f(\boldsymbol{x}_k)$ is a "shrinkage" step, iteration (2.7.7) is therefore called Iterative Shrinkage-Thresholding Algorithm (ISTA).

Like the gradient descent, iteration (2.7.7) has a slow convergence rate in the sense that the inaccuracy decays as $O(1/n)$. Based on the Nesterov idea to add a "momentum term", accelerations were proposed, leading to a "two steps ISTA" (TwIST) [BF07] or Fast Iterative Shrinkage-Thresholding Algorithm (FISTA) [BT09] which have a $O(1/n^2)$

[18] practical means by a multiplicative constant, as these constants are omitted by the big O notation
[19] notably, the prox does not depend on γ

convergence rate. One iteration of FISTA reads[20]

$$\begin{cases} \boldsymbol{x}_{k+1} = \text{prox}_{\gamma g}\left(\boldsymbol{y}_k - \gamma \nabla f(\boldsymbol{y}_k)\right) \\ t_{k+1} = \dfrac{1}{2}\left(1 + \sqrt{1 + 4t_k^2}\right) \\ \boldsymbol{y}_{k+1} = \boldsymbol{x}_{k+1} + \dfrac{t_k - 1}{t_{k+1}}(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k) \end{cases} \tag{2.7.8}$$

Algorithm (2.7.8) is especially interesting when the prox of g is simple to compute, as in Equation (2.6.15). Otherwise, the prox of g has to be computed iteratively, which is the case for example for the TV regularized inverse problem

$$\underset{\boldsymbol{x}}{\text{argmin}} \left\{ \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda \|\boldsymbol{D}\boldsymbol{x}\|_1 \right\} \tag{2.7.9}$$

2.7.3 Douglas-Rachford splitting algorithms

The proximal gradient algorithm uses the gradient of f and the proximal of g, i.e is adapted when g has a simple prox and f is differentiable. Douglas-Rachford splitting methods make use of the prox of both f and g, with f and g possibly non-differentiable. The idea is to minimize a function F which has not a simple proximal, but can be written as $F = f + g$ where both prox of f and g are simple. One Douglas-Rachford iteration reads [OV14]

$$\begin{cases} \boldsymbol{x}_{k+1} = \text{prox}_{\gamma f}\left(\boldsymbol{y}_k\right) \\ \boldsymbol{y}_{k+1} = \boldsymbol{y}_k + \rho\left(\text{prox}_{\gamma g}\left(2\boldsymbol{x}_{k+1} - \boldsymbol{y}_k\right) - \boldsymbol{x}_{k+1}\right) \end{cases} \tag{2.7.10}$$

where $0 < \rho < 2$ is a relaxation step. Again, iteration-dependent step sizes can be chosen, with similar convergence conditions (for example [BC11b], Corollary 27.4 p. 401).

A well-known instance of the Douglas-Rachford algorithm is the Alternating Direction Method of Multipliers (ADMM), which aims at solving the problem

$$\begin{aligned} &\min_{\boldsymbol{x}_1, \boldsymbol{x}_2} \{f(\boldsymbol{x}_1) + g(\boldsymbol{x}_2)\} \\ &\text{s.t. } \boldsymbol{A}_1\boldsymbol{x}_1 + \boldsymbol{A}_2\boldsymbol{x}_2 = \boldsymbol{c} \end{aligned} \tag{2.7.11}$$

by solving its dual version

$$\max_{\boldsymbol{z}} \left\{ -f^*(\boldsymbol{A}_1^T\boldsymbol{z}) - g^*(\boldsymbol{A}_2^T\boldsymbol{z}) + \boldsymbol{c}^T\boldsymbol{z} \right\} \tag{2.7.12}$$

It is mainly used to virtually decouple variables, for example when the function to minimize has the form $f(\boldsymbol{x}) + g(\boldsymbol{K}\boldsymbol{x})$. Without constraints, the generic ADMM iteration for minimizing $f + g$ is defined by [P+14]

$$\begin{cases} \boldsymbol{x}_{k+1} = \text{prox}_{\gamma f}\left(\boldsymbol{z}_k - \boldsymbol{u}_k\right) \\ \boldsymbol{z}_{k+1} = \text{prox}_{\gamma g}\left(\boldsymbol{x}_{k+1} + \boldsymbol{u}_k\right) \\ \boldsymbol{u}_{k+1} = \boldsymbol{u}_k + \boldsymbol{x}_{k+1} - \boldsymbol{z}_{k+1} \end{cases} \tag{2.7.13}$$

ADMM is also called the Split-Bregman method. It is based on the *augmented Lagrangian*, as the variables decoupling is made by adding an extra term in the Lagrangian to enforce the equality of the split variables. Interestingly, Douglas-Rachford/ADMM only access the functions f and g through their proximal.

[20] the accelerated version proposed in [KF16] uses $\boldsymbol{y}_{k+1} = \boldsymbol{x}_{k+1} + \frac{t_k - 1}{t_{k+1}}(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k) + \frac{t_k}{t_{k+1}}(\boldsymbol{x}_{k+1} - \boldsymbol{y}_k)$

When the problem has constraints, the ADMM algorithm for solving

$$\begin{aligned} \underset{\boldsymbol{x}_1,\boldsymbol{x}_2}{\operatorname{argmin}} \ & \{f(\boldsymbol{x}_1)+g(\boldsymbol{x}_2)\} \\ \text{s. t.} \ & \boldsymbol{G}\boldsymbol{x}_1=\boldsymbol{x}_2 \end{aligned} \tag{2.7.14}$$

is given by [ABF10]

$$\begin{cases} \boldsymbol{x}_{k+1}=\underset{\boldsymbol{x}}{\operatorname{argmin}} \left\{ f(\boldsymbol{x})+\dfrac{\mu}{2}\left\|\boldsymbol{G}\boldsymbol{x}-\boldsymbol{z}_k-\boldsymbol{u}_k\right\|_2^2 \right\} \\ \boldsymbol{z}_{k+1}=\operatorname{prox}_{g/\mu}\left(\boldsymbol{G}\boldsymbol{x}_{k+1}-\boldsymbol{u}_k\right) \\ \boldsymbol{u}_{k+1}=\boldsymbol{u}_k-\boldsymbol{G}\boldsymbol{x}_{k+1}+\boldsymbol{z}_{k+1} \end{cases} \tag{2.7.15}$$

which, by definition of the proximal, is clearly equivalent to (2.7.13) when $\boldsymbol{G}=\boldsymbol{I}$ (no constraints) by replacing the variable $\boldsymbol{u}$ with $-\boldsymbol{u}$ and the step size $\gamma=1/\mu$.

2.7.4 Primal-Dual methods

Primal-Dual methods aims at solving the problem

$$\min_{\boldsymbol{x}} \{F(\boldsymbol{x})+G(\boldsymbol{K}\boldsymbol{x})\} \tag{2.7.16}$$

or its equivalent alternative versions (dual and saddle-point, respectively, see Equation 2.6.12 in section 2.6.2)

$$\begin{aligned} \max_{\boldsymbol{z}} \{-F^*(-\boldsymbol{K}^*\boldsymbol{z})-G^*(\boldsymbol{z})\} \\ \max_{\boldsymbol{z}} \left\{ \min_{\boldsymbol{x}} \{\langle \boldsymbol{K}\boldsymbol{x}\,,\,\boldsymbol{z}\rangle+F(\boldsymbol{x})-G^*(\boldsymbol{z})\} \right\} \end{aligned} \tag{2.7.17}$$

In this context, the functions F, G are possibly different from the functions "f, g" previously given. The reason is that primal-dual methods allow an important flexibility when it comes to re-writing a problem "minimize $f+g$" (possibly with constraints) to an equivalent problem (2.7.16) (or (2.7.17)).

A review of primal, primal-dual and dual splitting techniques can be found in [OV14]. In this work, we focus on a recent primal-dual algorithm proposed by Chambolle and Pock [CP11]. Its basic form is given by

$$\begin{cases} \boldsymbol{y}_{k+1}=\operatorname{prox}_{\sigma\boldsymbol{G}^*}\left(\boldsymbol{y}_k+\sigma\boldsymbol{K}\tilde{\boldsymbol{x}}_k\right) \\ \boldsymbol{x}_{k+1}=\operatorname{prox}_{\tau\boldsymbol{F}}\left(\boldsymbol{x}_n-\tau\boldsymbol{K}^*\boldsymbol{y}_{k+1}\right) \\ \tilde{\boldsymbol{x}}_{k+1}=\boldsymbol{x}_{k+1}+\theta(\boldsymbol{x}_{k+1}-\boldsymbol{x}_k) \end{cases} \tag{2.7.18}$$

where $\theta\in[0,1]$ is a "momentum step". At each iteration, the Chambolle-Pock (C-P) algorithm makes one "primal step" by computing $\operatorname{prox}_F(x)$ with step τ, and one "dual step" by computing $\operatorname{prox}_{G^*}(z)$ with step σ. This algorithm can be shown to converge when $\tau\sigma<1/\left\|\boldsymbol{K}\right\|^2$ where $\left\|\boldsymbol{K}\right\|^2$ is the squared operator norm of $\boldsymbol{K}$ (in our setting, it is the maximum eigenvalue of $\boldsymbol{K}^T\boldsymbol{K}$ which is efficiently computed with the power method, see Algorithm 6.2.1).

Algorithm (2.7.18) is of special interest when $\operatorname{prox}_{\boldsymbol{F}}(\boldsymbol{x})$ and $\operatorname{prox}_{\boldsymbol{G}^*}(\boldsymbol{y})$ are easy to compute: each iteration boils down to evaluating these operators and the forward and adjoint operators $\boldsymbol{K}$, $\boldsymbol{K}^*$. As seen in section 3.1, this algorithm enables an important flexibility for rewriting a problem where the prox of f and g are not simple to a problem (2.7.16) or (2.7.16), where the prox of F and G are simple. A relaxation step can be added

to this algorithm [Con13], which can yield a faster convergence rate. A generalization of the C-P algorithm was proposed in [Con14] for solving more general problems.

The convergence rate of primal-dual methods is less well-understood than proximal point algorithms. In [CP11], the authors prove the $O(1/N)$ convergence of the C-P algorithm, but a behavior similar to $O(1/N^2)$ is often observed in practice.

Conclusion

In this chapter, we presented an unified mathematical framework describing all the known reconstruction algorithms. The advantage of the Bayesian modelling is that assumptions on the noise and volume prior knowledge are made explicit. All these methods are "statistical", and regularized methods naturally fit in this framework. There is sometimes some confusion among the "users" of reconstruction methods; in summary, all iterative methods are "statistical"[21], the only changing aspect is the noise/volume model. Regularized methods, which corresponds to the MAP rather than ML in the Bayesian framework, also bring stability: these methods do not suffer from the "noise amplification (when using too many iterations)" problem encountered with the standard least squares reconstruction. The optimization algorithms described previously can therefore be run until convergence, without choosing the number of iterations as a parameter.

The explicit separation between the modelling and the optimization process is the starting point of choosing the best algorithm for an efficient implementation, which is addressed in the next chapter.

Bibliography

[Aar+15] Wim van Aarle et al. "The {ASTRA} Toolbox: A platform for advanced algorithm development in electron tomography". In: *Ultramicroscopy* 157 (2015), pp. 35–47. ISSN: 0304-3991. DOI: http://dx.doi.org/10.1016/j.ultramic.2015.05.002. URL: http://www.sciencedirect.com/science/article/pii/S0304399115001060.

[ABF10] Manya V Afonso, José M Bioucas-Dias, and Mário AT Figueiredo. "Fast image recovery using variable splitting and constrained optimization". In: *IEEE Transactions on Image Processing* 19.9 (2010), pp. 2345–2356.

[Bac+12] Francis Bach et al. "Optimization with sparsity-inducing penalties". In: *Foundations and Trends® in Machine Learning* 4.1 (2012), pp. 1–106.

[BC11a] Heinz H Bauschke and Patrick L Combettes. *Convex analysis and monotone operator theory in Hilbert spaces*. Springer Science & Business Media, 2011.

[BC11b] Heinz H Bauschke and Patrick L Combettes. *Convex analysis and monotone operator theory in Hilbert spaces*. Springer, 2011.

[BF07] José M Bioucas-Dias and Mário AT Figueiredo. "A new TwIST: Two-step iterative shrinkage/thresholding algorithms for image restoration". In: *IEEE Transactions on Image processing* 16.12 (2007), pp. 2992–3004.

[21] although Maximum Likelihood and Maximum A Posteriori should be distinguished

[Bor+16] Vaibhav Bora et al. “Estimation of Fano factor in inorganic scintillators”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 805 (2016), pp. 72–86.

[Bou+10] Abdelkader Bousselham et al. “Photoelectron anticorrelations and sub-Poisson statistics in scintillation detectors”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 620.2 (2010), pp. 359–362.

[BT09] Amir Beck and Marc Teboulle. “A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems”. In: *SIAM J. Img. Sci.* 2.1 (Mar. 2009), pp. 183–202. ISSN: 1936-4954. DOI: 10.1137/080716542. URL: http://dx.doi.org/10.1137/080716542.

[BV09] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Ed. by Cambridge University Press. 2009.

[Can+11] Emmanuel J Candes et al. “Compressed sensing with coherent and redundant dictionaries”. In: *Applied and Computational Harmonic Analysis* 31.1 (2011), pp. 59–73.

[CD15] Antonin Chambolle and Charles Dossal. “On the convergence of the iterates of "FISTA"”. In: *Journal of Optimization Theory and Applications* Volume 166.Issue 3 (Aug. 2015), p. 25. URL: https://hal.inria.fr/hal-01060130.

[Con13] Laurent Condat. “A primal–dual splitting method for convex optimization involving Lipschitzian, proximable and linear composite terms”. In: *Journal of Optimization Theory and Applications* 158.2 (2013), pp. 460–479.

[Con14] Laurent Condat. “A generic proximal algorithm for convex optimization—application to total variation minimization”. In: *IEEE Signal Processing Letters* 21.8 (2014), pp. 985–989.

[CP09] P. L. Combettes and J.-C. Pesquet. “Proximal Splitting Methods in Signal Processing”. In: *ArXiv e-prints* (Dec. 2009). arXiv: 0912.3522 [math.OC].

[CP11] Antonin Chambolle and Thomas Pock. “A First-Order Primal-Dual Algorithm for Convex Problems with Applications to Imaging”. English. In: *Journal of Mathematical Imaging and Vision* 40.1 (2011), pp. 120–145. ISSN: 0924-9907. DOI: 10.1007/s10851-010-0251-1. URL: http://dx.doi.org/10.1007/s10851-010-0251-1.

[CR07] Emmanuel Candes and Justin Romberg. “Sparsity and incoherence in compressive sampling”. In: *Inverse problems* 23.3 (2007), p. 969.

[CRT06a] Emmanuel J Candes, Justin K Romberg, and Terence Tao. “Stable signal recovery from incomplete and inaccurate measurements”. In: *Communications on pure and applied mathematics* 59.8 (2006), pp. 1207–1223.

[CRT06b] Emmanuel J Candès, Justin Romberg, and Terence Tao. “Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information”. In: *IEEE Transactions on information theory* 52.2 (2006), pp. 489–509.

[CT06] Emmanuel J Candes and Terence Tao. “Near-optimal signal recovery from random projections: Universal encoding strategies?” In: *IEEE transactions on information theory* 52.12 (2006), pp. 5406–5425.

[DLR77] Arthur P Dempster, Nan M Laird, and Donald B Rubin. "Maximum likelihood from incomplete data via the EM algorithm". In: *Journal of the royal statistical society. Series B (methodological)* (1977), pp. 1–38.

[Don06] David L. Donoho. "For most large underdetermined systems of linear equations the minimal l1-norm solution is also the sparsest solution". In: *Communications on Pure and Applied Mathematics* 59.6 (2006), pp. 797–829. ISSN: 1097-0312. DOI: 10.1002/cpa.20132. URL: http://dx.doi.org/10.1002/cpa.20132.

[DT14] Yoel Drori and Marc Teboulle. "Performance of first-order methods for smooth convex minimization: a novel approach". In: *Mathematical Programming* 145.1 (2014), pp. 451–482. ISSN: 1436-4646. DOI: 10.1007/s10107-013-0653-0. URL: http://dx.doi.org/10.1007/s10107-013-0653-0.

[GB08] Jens Gregor and Thomas Benson. "Computational analysis and improvement of SIRT". In: *IEEE Transactions on Medical Imaging* 27.7 (2008), pp. 918–924.

[GPR67] L.G. Gubin, B.T. Polyak, and E.V. Raik. "The method of projections for finding the common point of convex sets". In: *USSR Computational Mathematics and Mathematical Physics* 7.6 (1967), pp. 1–24. ISSN: 0041-5553. DOI: http://dx.doi.org/10.1016/0041-5553(67)90113-9. URL: http://www.sciencedirect.com/science/article/pii/0041555367901139.

[Han10] Per Christian Hansen. *Discrete Inverse Problems: Insight and Algorithms*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2010. ISBN: 0898716969, 9780898716962.

[HO93] Per Christian Hansen and Dianne Prost O'Leary. "The Use of the L-Curve in the Regularization of Discrete Ill-Posed Problems". In: *SIAM Journal on Scientific Computing* 14.6 (1993), pp. 1487–1503. DOI: 10.1137/0914086. URL: http://dx.doi.org/10.1137/0914086.

[Imm96] John Immerkær. "Fast Noise Variance Estimation". In: *Computer Vision and Image Understanding* 64.2 (1996), pp. 300–302. ISSN: 1077-3142. DOI: http://dx.doi.org/10.1006/cviu.1996.0060. URL: http://www.sciencedirect.com/science/article/pii/S1077314296900600.

[KF16] Donghwan Kim and Jeffrey A Fessler. "Optimized first-order methods for smooth convex minimization". In: *Mathematical programming* 159.1-2 (2016), pp. 81–107.

[KS88] A. Kak and M. Slaney. *Principles of Computerized Tomographic Imaging*. IEEE Press, 1988.

[MBC14] Alessandro Mirone, Emmanuel Brun, and Paola Coan. "A dictionary learning approach with overlap for the low dose computed tomography reconstruction and its vectorial application to differential phase tomography". In: *PloS one* 9.12 (2014), e114325.

[MG08] R. Mazhar and P. D. Gader. "EK-SVD: Optimized dictionary design for sparse representations". In: *2008 19th International Conference on Pattern Recognition*. Dec. 2008, pp. 1–4. DOI: 10.1109/ICPR.2008.4761362.

[Mir+14] Alessandro Mirone et al. “The PyHST2 hybrid distributed code for high speed tomographic reconstruction with iterative reconstruction and a priori knowledge capabilities”. In: *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms* 324.0 (2014). 1st International Conference on Tomography of Materials and Structures, pp. 41–48. ISSN: 0168-583X. DOI: http://dx.doi.org/10.1016/j.nimb.2013.09.030. URL: http://www.sciencedirect.com/science/article/pii/S0168583X14000251.

[Moh12] Ali Mohammad-Djafari. “Bayesian approach with prior models which enforce sparsity in signal and image processing”. In: *EURASIP Journal on Advances in Signal Processing* 2012.1 (Mar. 2012), p. 52. ISSN: 1687-6180. DOI: 10.1186/1687-6180-2012-52. URL: http://dx.doi.org/10.1186/1687-6180-2012-52.

[Nuy+13] Johan Nuyts et al. “Modelling the physics in the iterative reconstruction for transmission computed tomography”. In: *Physics in medicine and biology* 58.12 (2013), R63.

[NW01] Frank Natterer and Frank Wübbeling. *Mathematical methods in image reconstruction*. SIAM, 2001.

[OV14] Daniel O’Connor and Lieven Vandenberghe. “Primal-dual decomposition by operator splitting and applications to image deblurring”. In: *SIAM Journal on Imaging Sciences* 7.3 (2014), pp. 1724–1754.

[P+14] Neal Parikh, Stephen Boyd, et al. “Proximal algorithms”. In: *Foundations and Trends® in Optimization* 1.3 (2014), pp. 127–239.

[Pal17] Pierre Paleo. *Jupyter notebooks*. https://github.com/pierrepaleo/notebooks. 2017.

[ROF92] Leonid I. Rudin, Stanley Osher, and Emad Fatemi. “Nonlinear total variation based noise removal algorithms”. In: *Physica D: Nonlinear Phenomena* 60.1 (1992), pp. 259–268. ISSN: 0167-2789. DOI: http://dx.doi.org/10.1016/0167-2789(92)90242-F. URL: http://www.sciencedirect.com/science/article/pii/016727899290242F.

[SF09] J-L Starck and Mohamed-Jalal Fadili. “An overview of inverse problem regularization using sparsity”. In: *Image Processing (ICIP), 2009 16th IEEE International Conference on*. IEEE. 2009, pp. 1453–1456.

[VRU08] Cédric Vonesch, Sathish Ramani, and Michael Unser. “Recursive risk estimation for non-linear image deconvolution with a wavelet-domain sparsity constraint”. In: *2008 15th IEEE International Conference on Image Processing*. IEEE. 2008, pp. 665–668.

[WMG17] Li Wang, Ali Mohammad-Djafari, and Nicolas Gac. “X-ray Computed Tomography using a sparsity enforcing prior model based on Haar transformation in a Bayesian framework”. In: *Fundamenta Informaticae XX* (2017). DOI: 10.3233/FI-2016-0000. URL: https://hal.archives-ouvertes.fr/hal-01490523.

[Xu+12] Qiong Xu et al. “Low-Dose X-ray CT Reconstruction via Dictionary Learning”. In: *Medical Imaging, IEEE Transactions on* 31.9 (Sept. 2012), pp. 1682–1697. ISSN: 0278-0062. DOI: 10.1109/TMI.2012.2195669.

[Yan+15] Xiaoli Yang et al. "TV-based conjugate gradient method and discrete L-curve for few-view CT reconstruction of X-ray in vivo data". In: *Opt. Express* 23.5 (Mar. 2015), pp. 5368–5387. DOI: 10.1364/OE.23.005368. URL: http://www.opticsexpress.org/abstract.cfm?URI=oe-23-5-5368.

Chapter 3

Efficient implementation of regularized reconstruction methods

In the previous chapter, regularized reconstruction method were formalized in a common Bayesian framework, as an extension of classical reconstruction algorithms. On the other hand, some modern convex optimization algorithms adapted to the reconstruction problem were reviewed. These aim at tackling a minimization problem $\underset{\boldsymbol{x}}{\operatorname{argmin}} \left\{ \frac{1}{2} \|\boldsymbol{P}\boldsymbol{x} - \boldsymbol{d}\|_2^2 + \lambda \|\boldsymbol{D}\boldsymbol{x}\|_1 \right\}$ involving the non-differentiable regularization term $\|\boldsymbol{D}\boldsymbol{x}\|_1$ while being reasonably fast.

In this chapter, we go in more detail and present the actual efficient implementation of these methods on Graphical Processing Unit (GPU). A high speed data processing is critical to cope with the always increasing amount of data outputed by modern detectors. The reconstruction methods are designed to be integrated in the PyHST2 software [Mir+14], the tomographic reconstruction program used at ESRF.

3.1 Fast Total Variation regularized tomographic reconstruction

Tomographic reconstruction with Total Variation regularization has been practically used for over a decade, and was given a theoretical ground with the CS framework [CRT06]. However, *efficient* and *exact* algorithms solving the TV reconstruction problem were only proposed quite recently with proximal algorithms mentioned in the previous chapter ([BF07], [BT09], [CP11]). Prior methods either involved to modify the objective function, thus solving a different problem, or to use a slowly convergent optimization algorithm.

In this section, we present an efficient implementation of the Chambolle-Pock algorithm for the TV reconstruction problem in the parallel geometry setting. Numerical experiments and comparison with states-of-the-art optimization algorithms show that C-P is attractive for the TV reconstruction problem. Notably, the GPU implementation is able to reconstruct 2k×2k and 4k×4k slices in a few seconds.

3.1.1 Considerations on optimization algorithms

The least-squares tomographic reconstruction problem with TV regularization is

$$\underset{\boldsymbol{x}}{\operatorname{argmin}} \left\{ \frac{1}{2} \|\boldsymbol{P}\boldsymbol{x} - \boldsymbol{d}\|_2^2 + \lambda \|\nabla \boldsymbol{x}\|_1 \right\} \tag{3.1.1}$$

where $\boldsymbol{x}$ is the latent image/volume to reconstruct, $\boldsymbol{d}$ is the acquired data (sinogram), $\lambda > 0$ the regularization parameter and ∇ is the image/volume gradient operator. Problem (3.1.1) is an instance of optimization problems discussed in previous chapter with $f(\boldsymbol{x}) = \frac{1}{2} \|\boldsymbol{P}\boldsymbol{x} - \boldsymbol{d}\|_2^2$ and $g(\boldsymbol{x}) = \lambda \|\nabla \boldsymbol{x}\|_1$. The spatial gradient operator ∇ has been defined in

Definition 15. It can be shown that with this definition, the operator norm of the spatial gradient ∇ is $\|\nabla\| = \sqrt{\lambda_{\max}(-\operatorname{div}\nabla)} = \sqrt{8}$.

Two methods are traditionally used to solve (3.1.1). On the one hand, the TV is "smoothed" by replacing the ℓ_1 norm with the *Huber function*

$$(\psi_\mu(\boldsymbol{x}))_i = \begin{cases} |x_i| & \text{if } |x_i| \geq \mu \\ \frac{x_i^2}{2\mu} + \frac{\mu}{2} & \text{otherwise} \end{cases} \tag{3.1.2}$$

so that the *smoothed TV* is [WBA09]

$$\|\nabla \boldsymbol{x}\|_1 \simeq J_\mu(\boldsymbol{x}) = \sum_i \psi_\mu(|(\nabla \boldsymbol{x})_i|) \tag{3.1.3}$$

so that the gradient of J_μ with respect to $\boldsymbol{x}$ is

$$\nabla_{\boldsymbol{x}} J_\mu(\boldsymbol{x}) = -\operatorname{div}\Psi \quad \text{where} \quad \Psi_i = \begin{cases} \frac{(\nabla \boldsymbol{x})_i}{|(\nabla \boldsymbol{x})_i|} & \text{if } |(\nabla \boldsymbol{x})_i| \geq \mu \\ \frac{(\nabla \boldsymbol{x})_i}{\mu} & \text{otherwise} \end{cases} \tag{3.1.4}$$

Smoothing the TV yields a differentiable objective function which is "almost quadratic", enabling to use a fast gradient method (conjugate gradient/BFGS).

However, this method appears to be less reliable in presence of moderate to high noise. Figures 3.1.1 and 3.1.2 illustrate the reconstruction performances on a phantom, where the sinogram was corrupted with noise. Even without noise, the subsampling artefacts remain on the reconstruction. Nevertheless, this method yields better results than non-regularized methods (SIRT).

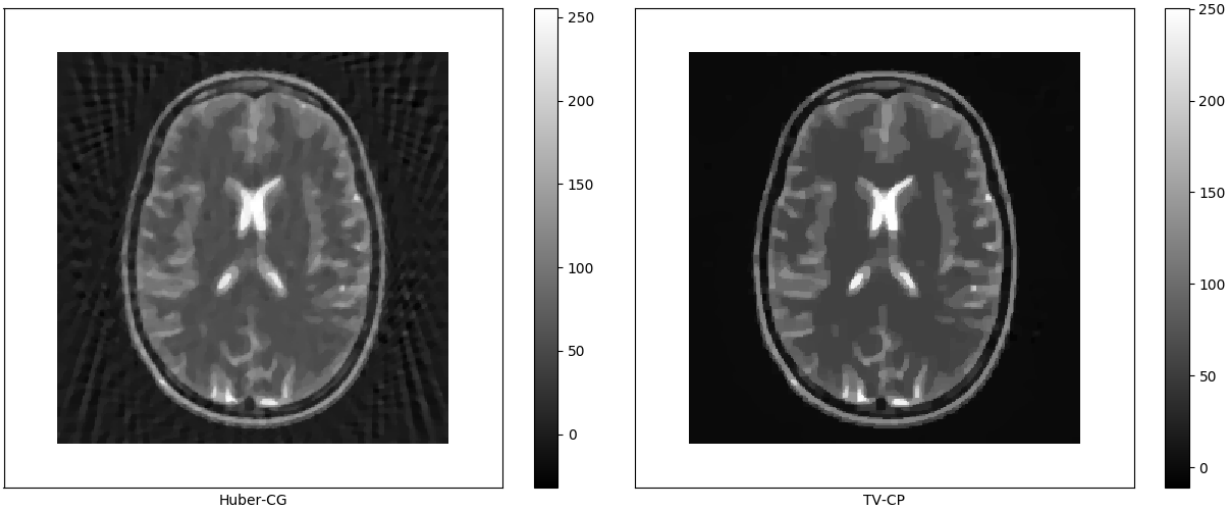


Figure 3.1.1: Reconstruction of the Brain phantom from 40 views, noiseless case. (a): Approximated TV replacing the ℓ_1 norm with the Huber function, (b): Exact TV using the C-P algorithm

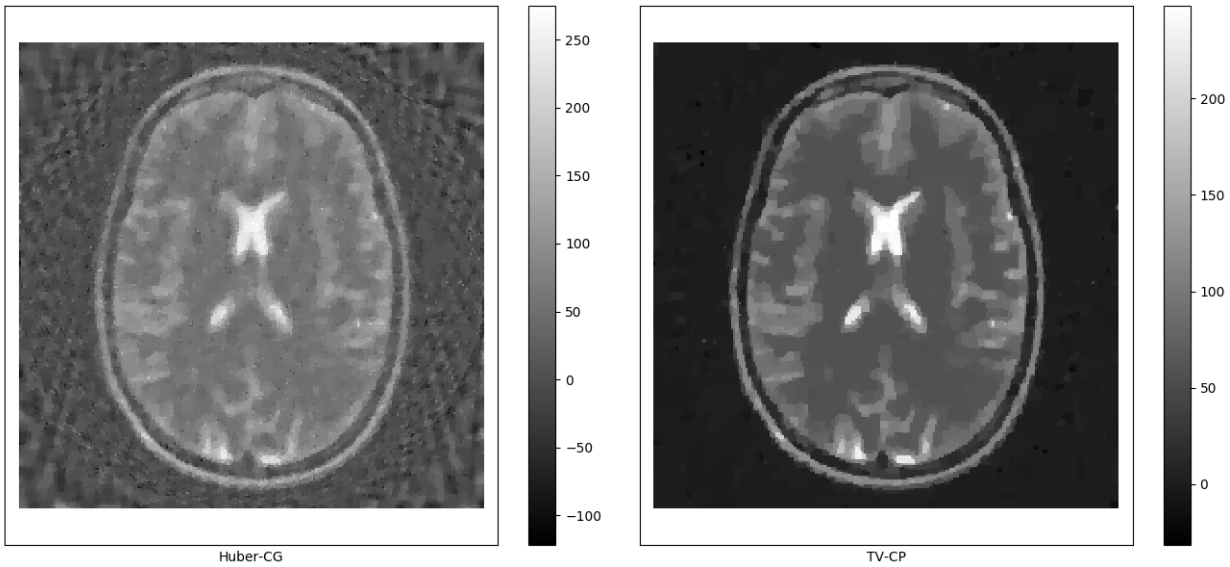


Figure 3.1.2: Reconstruction of the Brain phantom from 40 views, noisy case.
(a): Approximated TV replacing the ℓ_1 norm with the Huber function, (b): Exact TV using the C-P algorithm

On the other hand, the traditionally used method for solving (3.1.1) is the Split-Bregman algorithm (ADMM). Using this method requires to compute the prox of $\frac{1}{2}\|\boldsymbol{P}\boldsymbol{x}-\boldsymbol{d}\|_2^2$ at each iteration. Unfortunately, this computation entails an iterative process, as the operator $(\boldsymbol{I}+\gamma\boldsymbol{P}^T\boldsymbol{P})$ is not easily inverted (see Proposition 2.6.3). This means that the ADMM method is implemented as two nested loops: the outer loop for solving Problem (3.1.1), the inner loop for computing the prox of the quadratic data fidelity term. The computation of prox(g) might be simple for $g=\|\cdot\|_1$, provided that the alternative version of ADMM (2.7.15) is used with the constraint "$\nabla\boldsymbol{x}_1=\boldsymbol{x}_2$".

An alternative could be to change the optimization algorithm. As the data fidelity term is quadratic, an appealing algorithm for solving Problem (3.1.1) is the proximal gradient method (FISTA). This algorithm needs the computation of ∇f and prox(g). However, this time, the computation of prox(g) is not simple for the 2D/3D total variation. The usual way to compute the prox of TV is the Chambolle's algorithm [Cha04a], which is iterative.

In both cases of ADMM and FISTA, at least one outer loop is needed to compute a proximal. To implement an algorithm where iteration only require "matrix-vector multiplications" (i.e evaluation of linear operators), the C-P algorithm was considered.

3.1.2 TV reconstruction with the Chambolle-Pock algorithm

The first step is to cast the problem into a generic saddle-point problem:

$$\min_{\boldsymbol{x}}\max_{\boldsymbol{y}}\left\{\langle\boldsymbol{K}\boldsymbol{x}\,,\,\boldsymbol{y}\rangle+F(\boldsymbol{x})-G^*(\boldsymbol{y})\right\} \tag{3.1.5}$$

as in Problem (2.7.17), where $\boldsymbol{K}$ is a linear operator. This is done by dualizing the data fidelity term, as the squared ℓ_2 norm is its own conjugate :

$$\left(\frac{1}{2}\|\boldsymbol{P}\boldsymbol{x}-\boldsymbol{d}\|_2^2\right)^*=\max_{\boldsymbol{q}}\left\{\langle\boldsymbol{P}\boldsymbol{x}-\boldsymbol{d}\,,\,\boldsymbol{q}\rangle-\frac{1}{2}\|\boldsymbol{q}\|_2^2\right\}$$

where $\boldsymbol{q}$ is the *dual variable* of $\boldsymbol{P}\boldsymbol{x}$. The same dualization can be done for the total variation term, using another dual variable $\boldsymbol{z}$:

$$\begin{aligned}\lambda \|\nabla \boldsymbol{x}\|_1 &= \max_{\|\boldsymbol{z}\|_\infty \leq 1} \{\langle \nabla \boldsymbol{x}\,,\, \lambda \boldsymbol{z}\rangle\} \\ &= \max_{\|\boldsymbol{z}\|_\infty \leq \lambda} \{\langle \nabla \boldsymbol{x}\,,\, \boldsymbol{z}\rangle\} \\ &= \max_{\boldsymbol{z}} \left\{\langle \nabla \boldsymbol{x}\,,\, \lambda \boldsymbol{z}\rangle - i_{B_\infty^\lambda}(\boldsymbol{z})\right\} \\ &= \max_{\boldsymbol{z}} \left\{\langle \boldsymbol{x}\,,\, -\operatorname{div} \boldsymbol{z}\rangle - i_{B_\infty^\lambda}(\boldsymbol{z})\right\}\end{aligned}$$

where $i_{B_\infty^\lambda}$ is the indicator function of the ℓ_∞ ball of radius λ. The primal-dual problem is then

$$\begin{aligned}&\min_{\boldsymbol{x}} \max_{\boldsymbol{z},\boldsymbol{q}} \left\{\langle \boldsymbol{P}\boldsymbol{x} - \boldsymbol{d}\,,\, \boldsymbol{q}\rangle - \frac{1}{2}\|\boldsymbol{q}\|_2^2 + \langle \boldsymbol{x}\,,\, -\operatorname{div} \boldsymbol{z}\rangle - i_{B_\infty^\lambda}(\boldsymbol{z})\right\} \\ = &\min_{\boldsymbol{x}} \max_{\boldsymbol{z},\boldsymbol{q}} \left\{\langle \boldsymbol{x}\,,\, -\operatorname{div} \boldsymbol{z} + \boldsymbol{P}^T\boldsymbol{q}\rangle - \langle \boldsymbol{d}\,,\, \boldsymbol{q}\rangle - i_{B_\infty^\lambda}(\boldsymbol{z}) - \frac{1}{2}\|\boldsymbol{q}\|_2^2\right\}\end{aligned}$$

which is easily identifiable with (3.1.5) with

$$\begin{aligned}&G^*(\boldsymbol{z},\boldsymbol{q}) = i_{B_\infty^\lambda}(\boldsymbol{z}) + \frac{1}{2}\|\boldsymbol{q}\|_2^2 + \langle \boldsymbol{d}\,,\, \boldsymbol{q}\rangle \\ &F(\boldsymbol{x}) = 0 \\ &\boldsymbol{K}^* = \begin{pmatrix} -\operatorname{div}\,,\, \boldsymbol{P}^T\end{pmatrix} \;\Leftrightarrow\; \boldsymbol{K} = \begin{pmatrix}\nabla \\ \boldsymbol{P}\end{pmatrix}\end{aligned} \tag{3.1.6}$$

In this approach, the term F is null ; and the operator $\boldsymbol{K}$ is made of two operators ∇ and $\boldsymbol{P}$. We then have

$$\begin{aligned}&\operatorname{prox}_{\sigma G^*}(\boldsymbol{z},\boldsymbol{q}) = \left(P_{B_\infty^\lambda}(\boldsymbol{z})\,,\, \frac{\boldsymbol{q} - \sigma \boldsymbol{d}}{1+\sigma}\right) \\ &\operatorname{prox}_{\tau F}(\boldsymbol{x}) = \boldsymbol{x}\end{aligned} \tag{3.1.7}$$

since $G^*(\boldsymbol{z},\boldsymbol{q})$ is separable with respect to $\boldsymbol{z}$ and $\boldsymbol{q}$ [CP11]. The previous proximal operators in (3.1.7) and the operator $\boldsymbol{K}$ in (3.1.6) are the two only objects required to write the Chambolle-Pock algorithm – here the dual variable $\boldsymbol{y}$ in (3.1.5) is made of two dual variables $\boldsymbol{z},\boldsymbol{q}$. The final algorithm for tomographic reconstruction with TV regularization is given by Algorithm 3.1.1

Algorithm 3.1.1 Chambolle-Pock algorithm

n : number of iterations
$\boldsymbol{d}$: data (sinogram)
λ: regularization parameter
τ : step size in the primal domain (default: $1/L$)
σ : step size in the dual domain (default: $1/L$)
θ : relaxation parameter (default: 1)

procedure ChambollePock(n, τ, σ, θ)
  for $k \leftarrow 1 \dots n$ **do**
    ▷ Update dual variables $(\boldsymbol{z}, \boldsymbol{q})$: $\boldsymbol{y}_{k+1} = \text{prox}_{\sigma \boldsymbol{G}^*}\left(\boldsymbol{y}_k + \sigma \boldsymbol{K} \tilde{\boldsymbol{x}}_k\right)$
    $\boldsymbol{z}_{k+1} = P_{B^\lambda_\infty}\left(\boldsymbol{z}_k + \sigma \nabla \tilde{\boldsymbol{x}}_k\right)$
    $\boldsymbol{q}_{k+1} = \left(\boldsymbol{q}_k + \sigma \boldsymbol{P} \tilde{\boldsymbol{x}}_k - \sigma \boldsymbol{d}\right) / (1 + \sigma)$
    ▷ Update primal variable $\boldsymbol{x}$: $\boldsymbol{x}_{k+1} = \text{prox}_{\tau \boldsymbol{F}}\left(\boldsymbol{x}_k - \tau \boldsymbol{K}^* \boldsymbol{y}_{k+1}\right)$
    $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \tau \boldsymbol{P}^T \boldsymbol{q}_{k+1} + \tau \operatorname{div} \boldsymbol{z}_{k+1}$
    ▷ Relaxation step
    $\tilde{\boldsymbol{x}}_{k+1} = \boldsymbol{x}_{k+1} + \theta(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k)$
  end for
  return $\boldsymbol{x}_n$
end procedure

The Chambolle-Pock algorithm converges if $\sigma\tau \leq \frac{1}{L^2}$ where L is the norm of the operator $\boldsymbol{K}$. We have $L^2 = \|\boldsymbol{K}\|^2 = \lambda_{\max}\left(\operatorname{div} \nabla\right) + \lambda_{\max}\left(\boldsymbol{P}^T \boldsymbol{P}\right)$, where $\left\|\boldsymbol{P}^T \boldsymbol{P}\right\|$ depends on the geometry (like the number of projection angles). This is readily done with the power method (see 6.2.1).

Interestingly, this problem reformulation with $F(\boldsymbol{x}) = 0$ leaves room for constraints on the primal variable $\boldsymbol{x}$. Indeed, a constraint $\boldsymbol{x} \in \Omega$ for some convex set Ω can be encoded as $F(\boldsymbol{x}) = i_\Omega(\boldsymbol{x})$ in (3.1.5), resulting in $\text{prox}_{\tau F}(\boldsymbol{x}) = P_\Omega(\boldsymbol{x})$. For example, the positivity constraint "$x_i \geq 0$" can be added in Algorithm 3.1.1 with a single instruction "$x_i = \max(x_i, 0)$" (see Proposition 6.2.2), which is a minor and inexpensive modification. Adding a positivity constraint is known to improve the reconstruction quality [OV14] while improving the convergence rate.

3.1.3 Numerical experiments

The C-P method is now compared to the aforementioned competing methods (ADMM and FISTA) for the tomography reconstruction problem with TV regularization (3.1.1). More precisely, the algorithms used are the following: the unconstrained version of ADMM (iteration 2.7.13), FISTA with the "Optimized Gradient Method" proposed in [KF16], and a preconditioned version of C-P [PC11]. The comparison is performed according to three criteria:

- The convergence rate, i.e how many iterations are required to solve the Problem (3.1.1)
- The cost per iteration, which will also determine the total execution time
- The accuracy of the solution

The rationale of the last criterion is that algorithms involving the resolution of a subproblem in each iteration may suffer from inaccuracies when the number of inner loops is too

low. More precisely, when a proximal mapping has to be computed at each iteration with an iterative algorithm, a sufficient number of (sub-)iterations should be chosen in order to ensure the convergence to the "true" proximal at the current point. Returning an early solution might accumulate errors, which can yield an inaccurate solution at the end of the algorithm run. In theory, ADMM is rather error-tolerant in the sense that the errors on the proximal computation should be summable with an infinite number of iterations (see for example the Eckstein-Bertsekas theorem recalled in [ABF10]), but this condition is rather difficult to check in practice.

The number of iterations to reach convergence is measured in terms of value of the objective function $\frac{1}{2}\|\boldsymbol{P}\boldsymbol{x} - \boldsymbol{d}\|_2^2 + \lambda \|\nabla \boldsymbol{x}\|_1$. The faster this value decays, the higher is the convergence rate. As in [CP11], the "accuracy" of the solution is measured in terms of Mean Squared Error (MSE) between the result $\hat{\boldsymbol{x}}$ of one algorithm and a reference $\boldsymbol{x}^\sharp$ obtained by running an algorithm for a very long time. As $\boldsymbol{x}^\sharp$ is a ground truth solution of Problem (3.1.1), any algorithm should converge to $\boldsymbol{x}^\sharp$, thus, the MSE makes sense in this application.

The experiments were performed on a machine with a Intel Xeon E5-2643 v3 CPU (12 cores, 3.40GHz), and a Nvidia GeForce Titan X (Maxwell generation). The involved image is a 256×256 brain phantom proposed in [Gue+12]; the dimensions being small to make the tests faster. The fast GPU implementation of the ASTRA toolbox [Aar+16] is used for the projection and backprojection operators. Except for these operators, all the code targets CPU for a faster prototyping. This numerical experiment is available as a Jupyter notebook at [Pal17].

For each algorithm, the number of iterations was chosen so that a higher number of iterations would not bring significant change in the reconstruction MSE. Figure 3.1.3 shows a logarithmic plot of the (normalized) objective function for the different algorithms. Figure 3.1.4 shows the reconstruction result for these algorithms, along with the ground-truth solution of Problem (3.1.1). Table 3.1 shows the metrics for the three criteria discussed above.

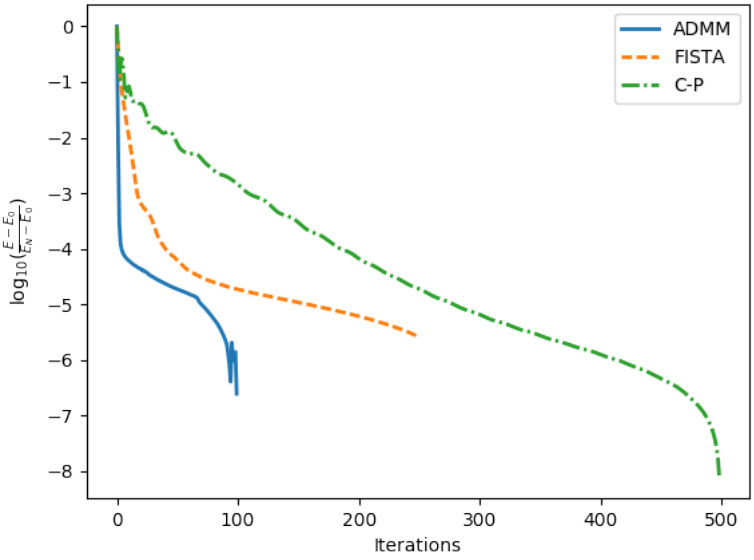


Figure 3.1.3: Comparison of the convergence rate between different optimization algorithms. For each algorithm, E denotes the objective function $\frac{1}{2}\|\boldsymbol{P}\boldsymbol{x}-\boldsymbol{d}\|_2^2+\lambda\|\nabla\boldsymbol{x}\|_1$, and E_N is its value at the last iteration. The ADMM curve is cut early, as it is the first to reach the best precision; in this case the quantity $\log_{10}(\frac{E-E_0}{E_N-E_0})$ is ill-defined.

When needed, the proximal of $f(\boldsymbol{x})$ is computed with 40 iterations of the conjugate gradient method, and the proximal of $g(\boldsymbol{x})$ is computed with 40 iterations of Chambolle's dual algorithm [Cha04b]. This number of iterations ensures a convergence to the proximal.)

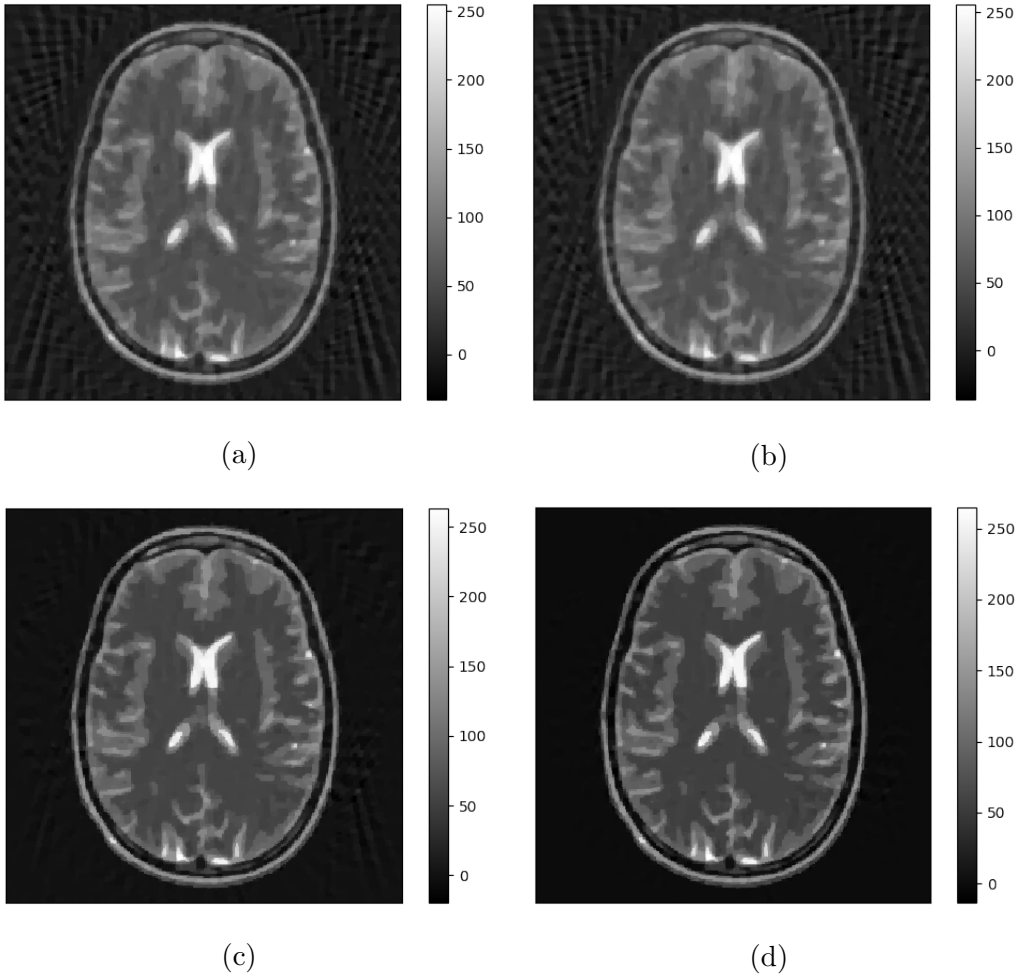


Figure 3.1.4: Results of the different optimization algorithms. (a) ADMM after 100 iterations. (b) FISTA after 250 iterations (c) C-P after 500 iterations. (d) Ground-truth solution, computed with C-P with 2000 iterations.

	Iterations	MSE	Total time
ADMM	100	8.9e1	21.4s
FISTA	250	1.1e2	23.6s
C-P	500	9.29e0	4.4s

Table 3.1: Results for the three criteria involved in the comparison of the optimization algorithms. For each algorithm, the number of iterations N was chosen such that running with more than N iterations does not significantly improves the reconstruction MSE.

From these simulations, we can draw the following conclusions:

- Regarding the convergence rate, ADMM is better than FISTA, which is better than Chambolle-Pock.
- Regarding the cost per iteration: ADMM has the higher cost, followed by FISTA, then by Chambolle-Pock.

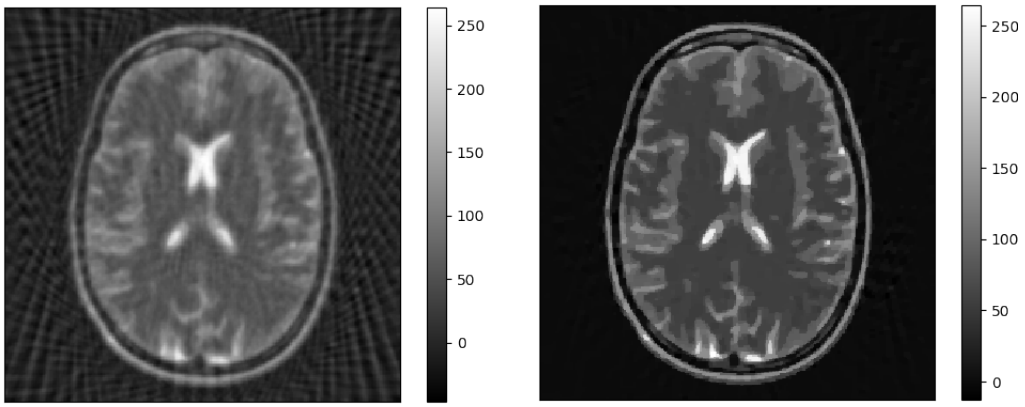
- It is important to have an accurate proximal computation for ADMM and FISTA. Not using enough iterations in the inner loops leads to a reconstruction without the "compressed sensing" benefit (i.e the reconstruction bears subsampling artefacts).

- As commonly observed, ADMM provides an extremely fast convergence in the beginning of the optimization process, then is slower to reach an accurate solution. For imaging applications, it can make sense to use a few iterations if only an approximate solution is needed. However, in the context of limited data (scarce views, low SNR), many iterations are needed to remove the noise and the subsampling artefacts.

The first two points are easy to justify. Computing the prox of f in ADMM is equivalent to inverting the (regularized) Hessian at each iteration, yielding a second-order method with a faster convergence than first-order methods [1] . FISTA uses the fact that f is differentiable for the computation of the "gradient step". Chambolle-Pock does not use any assumption, except that $\boldsymbol{P}$ is a positive operator for using its preconditioned version. Computing a prox at each iteration is costly, except for C-P where the problem is reformulated to only involve matrix-vector multiplications at each iteration.

The cost per iteration is dominated by the number of calls to $\boldsymbol{P}$ and $\boldsymbol{P}^T$ at each iteration, these correspond to the most computationally expansive operator to apply. Letting N_f and N_g denote the number of inner iterations to compute prox(f) and prox(g) respectively, the number of calls to $\boldsymbol{P}$ or $\boldsymbol{P}^T$ is $2N_f + 2N_g$ for ADMM, $2 + 2N_g$ for FISTA and 2 for C-P.

The third and fourth points can be illustrated with Figure 3.1.5, which shows the reconstruction result with ADMM after 500 iterations instead of 100.



(a) (b)

Figure 3.1.5: Reconstruction results with 500 iterations of ADMM, where different number of inner iterations N_f and N_g were used to compute prox(f) and prox(g), respectively. (a) $N_f = 10$, $N_g = 20$ (MSE = 1.78e2, total time = 44.7 s). (b) $N_f = N_g = 40$ (MSE = 1.92e0, total time = 125 s).

As a conclusion, considering the trade-off between convergence rate, accuracy and time-per-iteration, it seems that C-P is the better algorithm to use in this case. This conclusion would certainly be different in other settings. For example, in deblurring or

[1] a similar benchmark with the same conclusion can be found in [RF12]

inpainting applications, the prox of f can be computed in one step. With orthogonal Wavelet regularization, the prox of g is also very simple to compute. An additional advantage of C-P over FISTA and ADMM is that it is not required to tune the number of sub-iterations, which is a notable usability benefit.

3.1.4 Improving the convergence rate with iterative FBP

The previous section showed that C-P seems to be an interesting optimization algorithm for TV reconstruction. The primal-dual splitting scheme avoids to solve a subproblem at each iteration, resulting in a much lesser cost per iteration. However, the convergence rate is slower compared to FISTA or ADMM. The question is: can we have the "best of the two worlds", i.e modify the C-P algorithm in order to speed-up convergence ?

In section 1.4.3, it was stated that in the continuous (parallel) setting, the Radon transform $\mathcal{R}$ satisfies $\mathcal{R}^*\mathcal{R} = \Lambda^{-1}$ (Equation 1.4.7). It immediately follows that $(\mathcal{R}^*\mathcal{R})^{-1} = \Lambda$, where Λ is the Calderón's operator consisting in multiplying with the frequency magnitude in the (2D) Fourier domain.

Now recall that for minimizing a smooth function f, a simple gradient descent step (section 2.7.1) is given by $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \gamma_k \nabla f(\boldsymbol{x}_k)$, where $\gamma_k > 0$ is the descent step at iteration k. The *Newton's method*, on the other hand, has the following iteration:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \left(\nabla^2 f(\boldsymbol{x}_k)\right)^{-1} \left(\nabla f(\boldsymbol{x}_k)\right) \tag{3.1.8}$$

i.e the Newton's method is a gradient descent with a "matrix step size" equal to the inverse of the Hessian of f at $\boldsymbol{x}_k$. Computing the Hessian is cumbersome in general. However, if the objective function is quadratic, $f(\boldsymbol{x}) = \frac{1}{2}\left\|\boldsymbol{P}\boldsymbol{x} - \boldsymbol{d}\right\|_2^2$, then the Hessian of f is constant and given by $\boldsymbol{P}^T\boldsymbol{P}$. Computing the previous descent step then amounts to inverting $\boldsymbol{P}^T\boldsymbol{P}$, which is only possible with an iterative process in general.

The idea of iterative FBP is to replace $\boldsymbol{P}^T\boldsymbol{P}$ by its continuous approximation $\mathcal{R}^*\mathcal{R}$, so that $(\boldsymbol{P}^T\boldsymbol{P})^{-1} \simeq \Lambda$. This approximation only works for parallel geometry. The Newton iteration (3.1.8) becomes

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \Lambda\boldsymbol{P}^T(\boldsymbol{P}\boldsymbol{x}_k - \boldsymbol{d}) \tag{3.1.9}$$

for a quadratic f. The operator $\Lambda\boldsymbol{P}^T$ is exactly the FBP, thus, iteration (3.1.9) is an *iterative FBP*. In practice, $\Lambda\boldsymbol{P}^T = \Lambda_2\boldsymbol{P}^T$ is implemented as a *filter-then-backproject* approach with $\boldsymbol{P}^T\Lambda_1$ (see subsection 1.4.3). As commonly observed, using the filtered backprojection instead of the plain backprojection in an iterative algorithm dramatically improves the convergence rate [LT94]. The previous derivation gives an insight: the plain gradient descent becomes a fast Newton method [Cli+93]. Using much fewer iterations, with almost the same cost (the filtering process is efficiently performed with FFT) is a valuable gain for an iterative reconstruction.

From another point of view, the operator Λ in iteration (3.1.9) can also be seen as a *preconditioner*. Indeed, as seen in the first chapter, the forward projector fills more points in the low frequencies of the Fourier space. The operator Λ counter-balances this unequal density tiling of the Fourier space by weighting the frequencies.

3.1.5 On the matching of the projection and backprojection operators

In practice, implementations of the projector and backprojector are usually such that these operators are not matched[2] . Formally, if $\boldsymbol{P}$ and $\boldsymbol{B}$ denote the (implementation of)

[2] two linear operators $\mathcal{A}$ and $\mathcal{B}$ are said to be matched if for all $\boldsymbol{x}, \boldsymbol{y}$ belonging to the appropriate vector spaces, $\langle \mathcal{A}(\boldsymbol{x}) \,,\, \boldsymbol{y} \rangle = \langle \boldsymbol{x} \,,\, \mathcal{B}(\boldsymbol{y}) \rangle$. In finite dimension, it means that $\mathcal{B}$ is the transpose (or hermitian adjoint) of $\mathcal{A}$.

forward projection and backprojection operators respectively, then $\boldsymbol{B} \neq \boldsymbol{P}^T$ in general. This contrasts with the design of optimization algorithms where one naively assumes that the backprojection is the adjoint of $\boldsymbol{P}$. The mispatch between $\boldsymbol{B}$ and $\boldsymbol{P}$ is mostly due to performance reasons, as implementing the exact match of $\boldsymbol{P}$ might be computationally costly. For example, the forward projector used in PyHST uses the Joseph slice interpolation scheme [Jos82], so using the exact same interpolation weights in the backprojection as in the projection is cumbersome. In general, GPU implementations use a *ray-driven* approach for the forward projector and a *voxel-driven* approach for the backprojector [XM06], as it is preferable to have a SIMD-friendly write access pattern.

The work [ZG00] shows that for the Landweber iteration, the sign of the smallest eigenvalue[3] of $\boldsymbol{BP}$ determines the long-term convergence (a negative smallest eigenvalue causes divergence issues). The authors advise to use fast but unmatched $(\boldsymbol{P}, \boldsymbol{B})$ and regularization to stabilize the process. In [MM16], it is emphasized that using an unmatched projector-backprojector pair causes convergence issues in the long run, regardless of the objective function (regularized or not). It is observed, however, that using a ray-driven projector with a voxel-driven backprojector (which is the case in PyHST) does not make the process diverge, although the optimization algorithm might not converge to the "true" solution. The latter conclusion was also noticed from our side, although we found that geometry is far more critical for convergence: for example, an incorrect rotation center (even slightly) or mismatched 0 degrees and 180 degrees projections[4] .

3.1.6 GPU implementation

The previous numerical experiment motivated the efficient implementation of the C-P TV solver on GPU. These devices are specialized in executing *many simple* tasks in *parallel*. The last three highlighted words have a precise meaning:

- *Many* tasks usually means processing big data volumes.
- *Simple* tasks means that few *branching* (loops, conditional jumps) instructions should be performed. Central Processing Unit cores are typically designed for tasks involving complex branching, while GPU cores are designed for a regular instruction pipeline.
- *Parallelism* comes with several different degrees. Each GPU core can process its data independently of the other, or collaborate with other cores.

The architecture and programming model of GPU is out of the scope of this manuscript. A noteworthy introduction to the basic and advanced subjects of GPU programming in the context of scientific computing can be found in [Not17]. The PyHST2 software uses the CUDA framework [Nvi15].

As stated in the derivation of this algorithm, the main ingredient are the operators $\boldsymbol{P}$, ∇ and their adjoints. The PyHST2 software already provides an efficient GPU implementation of $\boldsymbol{P}$ and $\boldsymbol{P}^T$. The other main ingredients are the proximal maps of F and G^* defined in 3.1.7. Lastly, the "vector-vector operations" (addition, subtraction, multiplication with a scalar, norm computation) are handled with the CUBLAS library [NVi17].

[3] in terms of magnitude

[4] "usual" scans span an angular range of $[0, 180[$ degrees where 180 degrees is not reached ; but some scans might include it

Interestingly, the choice of the total variation (isotropic or anisotropic, see Definition 7) can be made by simply modifying the proximal of G^*. Here, the dual variable $\boldsymbol{z}$ is projected onto the ℓ_∞ ball, which corresponds to the anisotropic TV (which involves a ℓ_1 norm). Choosing the isotropic TV amounts to projecting $\boldsymbol{z}$ onto the ℓ_2 ball.

Table 3.2 shows the comparison between the GPU implementations of FISTA and C-P, where the test have been performed on the aforementioned machine. The number of iterations and the regularization parameter λ were chosen to obtain a visually appealing reconstruction without subsampling artefacts. For most configurations, a given algorithm is run with two different number of iterations in order to measure how it is close to convergence. For example, in the case of 512×512-40 projections, increasing the number of iterations does not bring significant improvements for C-P while it does for FISTA.

The MSE metric was chosen to assess the reconstruction quality. The reason is that this metric is more sensitive to the undersampling artefacts ("star" structure in the whole image) than small local noise, and thus will measure how the reconstruction is close to the ground truth with the undersampling. Therefore, we believe that MSE is a reasonable choice in this setting.

From these results, it appears that C-P is at least several times faster than FISTA for the TV reconstruction problem. It can be noted that the convergence rate of C-P is comparable with FISTA when using the iterative FBP described in 3.1.4 – in some settings, the former even has a better convergence rate. For example, on the 2048×2048 setting, 50 iterations of C-P yield a MSE of 4.91 in 0.97 second, while it requires 100 iterations if FISTA to yield a MSE of 10.3 in 6.00 seconds. Interestingly, the MSE results are better for large images with fewer iterations. An explanation could be that for this test image, there is fewer information in the high frequencies for the large versions (as the phantom is piecewise-constant). Thus, for a given image width/number of projections ratio, the accurate recovery of the large versions only involves a small fraction of the spectrum contrarily to the recovery of small versions.

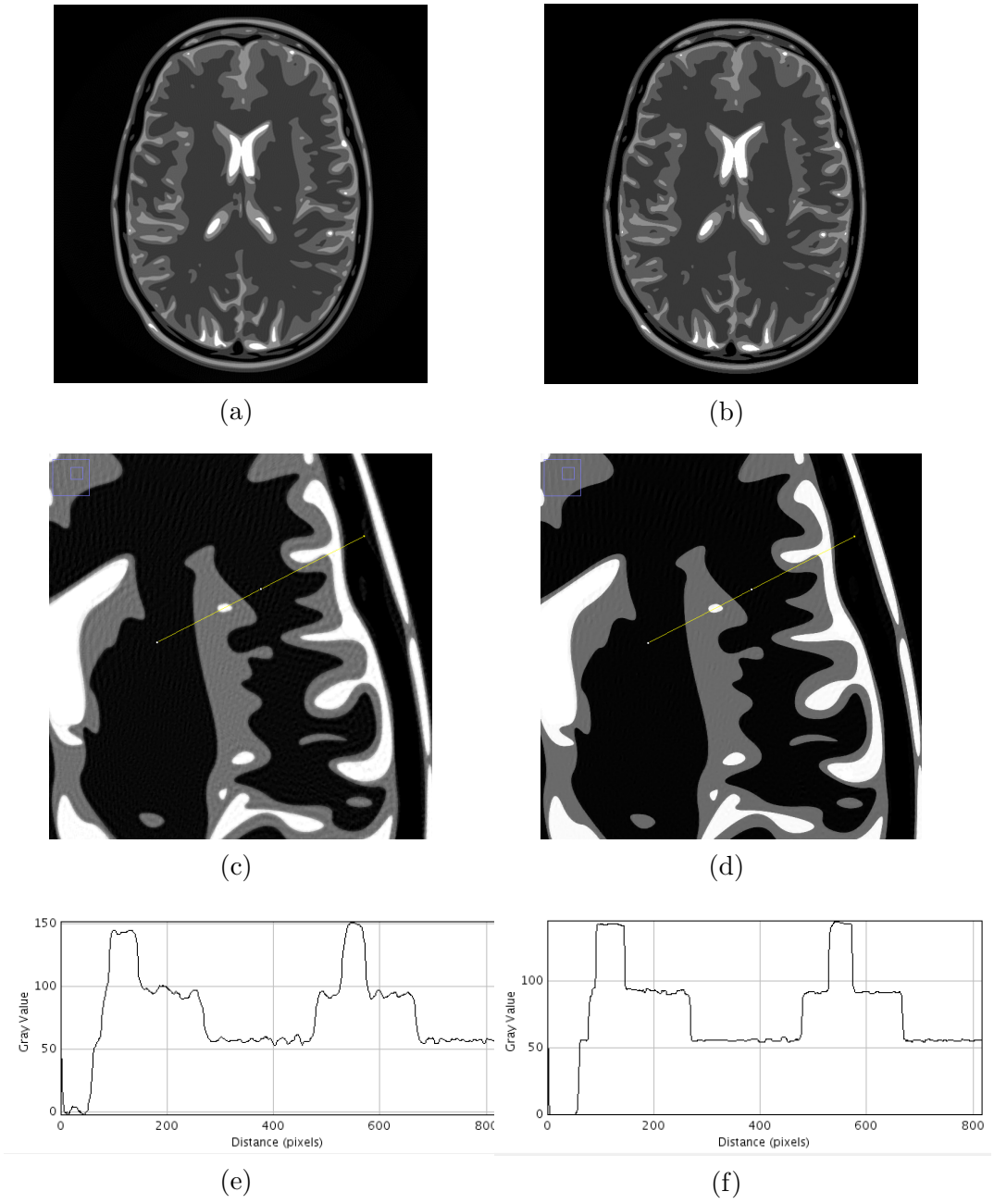


Figure 3.1.6: TV reconstruction results with 50 iterations of FISTA (a,c,d) and Chambolle-Pock (b,d,f) on a 4096×4096 brain phantom with 320 views. (a,b): reconstructed slice. (c, d): zoom in the upper right quadrant with an indication of a line profile (yellow). (e, f): line profile in the reconstruction.

3.1.7 Reconstruction results

In this subsection, the TV reconstruction capabilities is assessed on real datasets.

The first dataset (Figures 3.1.7) is a real imaging phantom scanned with the Edge Illumination method at the phase contrast imaging group of the University College London (UCL). The 2000×2000 slices are to be reconstructed from 720 projection angles. Here,

Slice width	Views	Method	Iterations	Total time (s)	MSE
512	40	CP	200	0.54	4.81e+01
		FISTA	200	2.07	6.00e+01
		CP	400	0.74	4.11e+01
		FISTA	400	3.55	2.56e+01
1024	80	CP	100	0.58	9.76e+00
		FISTA	100	2.05	2.36e+01
		CP	200	0.90	3.93e+00
		FISTA	200	3.74	2.25e+00
2048	160	CP	50	0.97	4.91e+00
		FISTA	50	3.60	2.55e+01
		CP	100	1.57	9.51e-01
		FISTA	100	6.00	1.03e+01
4096	320	CP	50	5.04	1.04e+00
		FISTA	50	13.2	1.59e+01
		CP	100	8.9	6.88e-02
		FISTA	100	24.9	8.83e+00
		FISTA	150	32.3	2.77e+00

Table 3.2: Benchmark of TV reconstruction for FISTA and C-P algorithms with different slice sizes and number of projections.

only the *absorption* data is used, so the projections SNR is low (Edge Illumination is normally a phase contrast method). Choosing a FBP filter dampening the high frequencies somewhat improves the result, but the image is still very noisy. The SIRT algorithm partly removes the noise and enhances the contrast, but even with a small number of iterations, the slice is still noisy. The TV reconstruction manages to remove the noise and to yield a better contrast than the SIRT reconstruction. The sharp edges can also be easily distinguished.

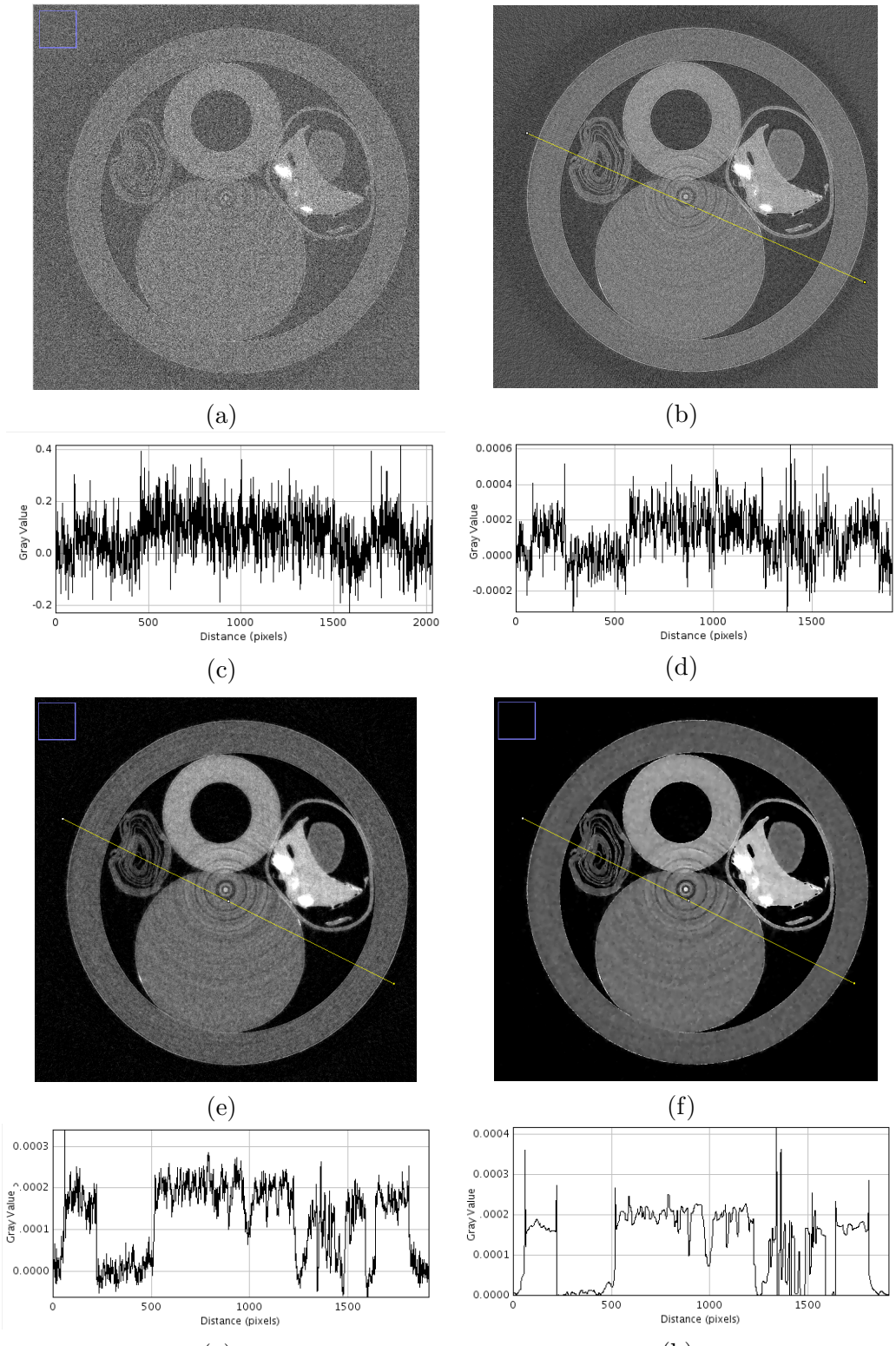


Figure 3.1.7: Reconstructed slice of a real imaging phantom (projection data courtesy: Charlotte Hagen, UCL). (a, b): reconstruction with FBP using the Ram-Lak and Hamming filters, respectively. (c, d): respective line profiles of the reconstructions in (a, b). (e, f): reconstruction with SIRT (100 iterations) and TV-C-P (150 iterations, $\lambda = 0.1$, positivity constraint), respectively. (g, h): respective line profiles of the reconstructions in (e, f).

The second dataset (Figure 3.1.8) is a scan of a snow core (ESRF ID11). The slices are 700×700 pixels and are to be reconstructed from 120 projections.

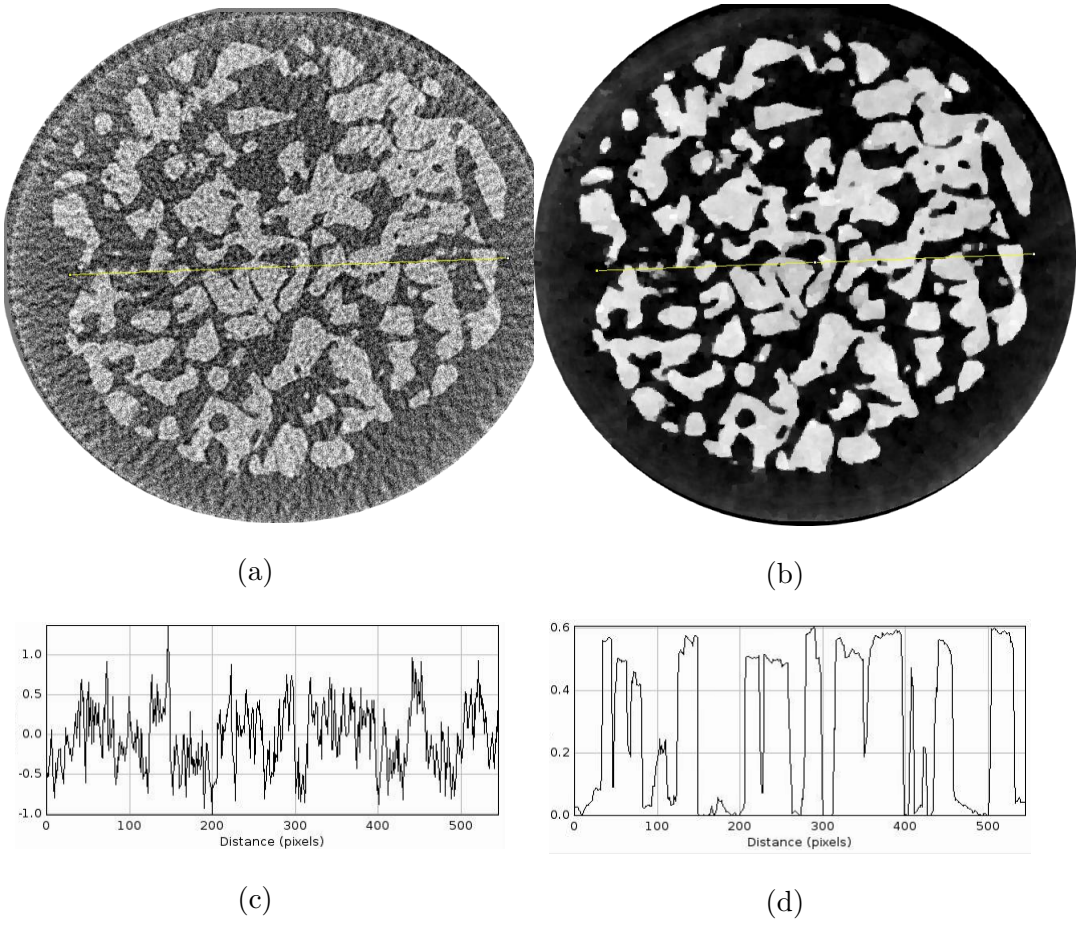


Figure 3.1.8: Reconstructed slice of a snow core scan acquired at the ESRF ID11 beamline.
(a, b): reconstructions with FBP and TV-C-P, respectively.
(c, d): profiles corresponding to the yellow lines on (a, b) for FBP and TV-C-P (300 iterations, $\lambda = 0.12$), respectively. As it can be noted, a positivity constraint was enforced.

3.2 Wavelet regularized reconstruction

The motivation of using wavelets in regularized tomographic reconstruction comes from the TV and dictionary-based reconstruction. On the one hand, TV performs well for piecewise-constant images, which corresponds to relatively simple samples. On the other hand, dictionary-base reconstruction yields prominent reconstruction capabilities [Mir+14], but a dictionary has to be learnt on a good-quality reconstruction beforehand. Besides the computational burden of this method (slow convergence due to the synthesis prior, and expensive forward/adjoint operators), a good quality reconstruction is not always available. A trade-off in terms of both reconstruction speed and quality can be achieved with wavelets, as most natural images are compressible in a wavelets domain, so there is no need to learn a dictionary.

In this second section, we present the use of wavelets as a sparsifying transform for regularized tomographic reconstruction. Although the discrete wavelet transform is reportedly used in some projects and research works (see for example [Rit+14], [Gua+16]), a work covering both the principles and implementation was lacking. As it turns out when using wavelets, implementation details – like translation invariance or thresholding functions – are crucial to get acceptable results.

The main contribution of this part is the development of an open-source, *plug and play* library for GPU DWT handling many transform types and their inverses for arbitrary signals (1D) and images (2D) shapes. We also review why wavelets are an interesting regularizer and what wavelet type is likely to be the best in the framework of CS; and address related optimization and reconstruction issues.

3.2.1 Theoretical performances of wavelets in the CS framework

As stated in subsection 2.4.5, the basic ingredients of compressive sensing are incoherence (between the acquisition and representation operators) and compressibility of the latent signal in the representation basis. This subsection is a numerical experiment aiming at *measuring* these two concepts in the tomography context.

The compressibility was defined in 2.4.5 (Definition 5). In the CS framework, a signal $\boldsymbol{x}$ is compressible in a basis $\boldsymbol{D}^T$ (for some "sparsifying transform" $\boldsymbol{D}$) if there exists $R > 0$ such that $\boldsymbol{Dx}$ belongs to the weak ℓ_p ball of radius R for some $0 < p < +\infty$. In other words, the *sorted* coefficients $(\boldsymbol{Dx})_k$ of $\boldsymbol{x}$ in the basis (or frame) $\boldsymbol{D}^T$, decay as $Rk^{-1/p}$, a power law. The compressibility in the sense of Definition 5 can be easily computed: for sorted coefficients $\boldsymbol{c}$, the radius R of the weak ℓ_p ball can be chosen as $R = |c_1| = \max_k \{|c_k|\}$, and p can be computed as $p = \max\limits_{k>1} \left\{ \dfrac{\log k}{\log |c_1| - \log |c_k|} \right\}$. Unfortunately, this may yield high values of p if the first largest coefficients have a similar value.

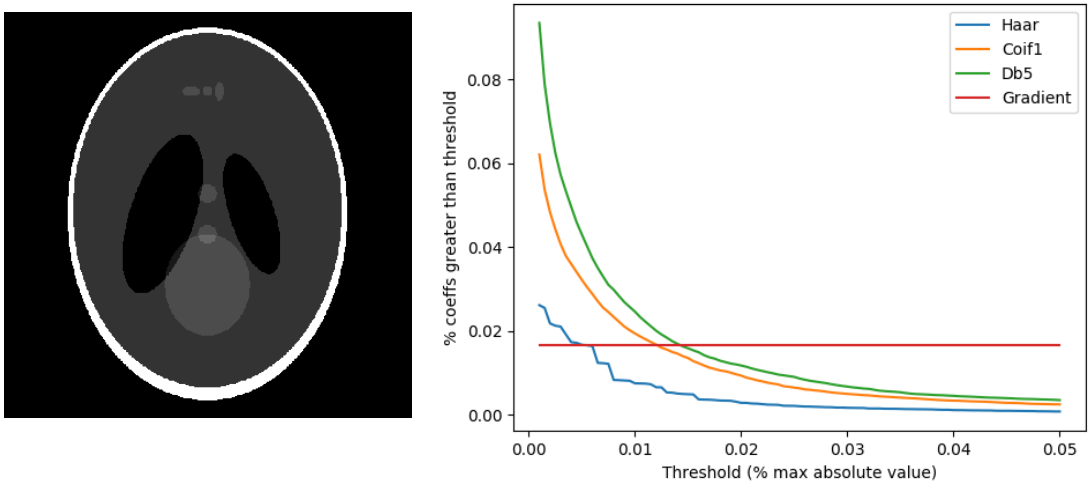
Another measure of compressibility was proposed in [Gua+16] in the context of Electron tomography. The *compressibility ratio* with threshold $\rho > 0$ of a coefficients vector $\boldsymbol{c}$ is defined as

$$\text{compressibility}_\rho(\boldsymbol{c}) = \frac{1}{n_2} \cdot \#\left\{i\,,\ |c_i| > \rho \cdot c_{\max}\right\} \tag{3.2.1}$$

where n_2 is the number of components of $\boldsymbol{c}$, and $\#$ denotes the cardinality of a set. The compressibility ratio measures the proportion of components larger than ρ times the maximum value (in terms of magnitude). If an image is compressible (in some frame), its coefficients will have a small compressibility ratio. The underlying idea is the same as for Definition 5: a compressible signal is characterized by a coefficient vector having a few large values, and many small values [5] . The work [Gua+16] reports a positive correlation between the reconstruction error and this definition of compressibility, which will be used from now.

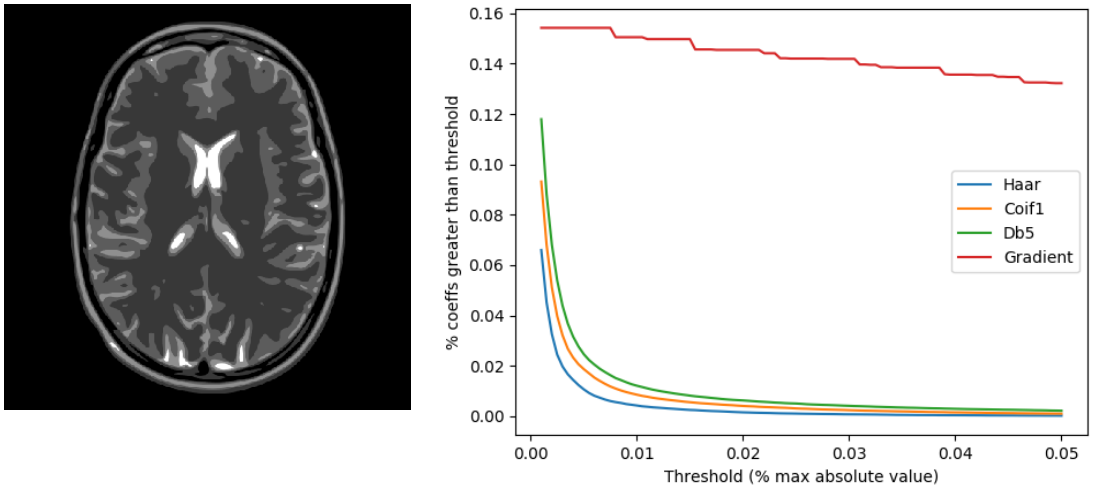
The first numerical experiment available in Jupyter notebook [Pal17] measures the compressibility ratio in various representations. In Figures 3.2.1, 3.2.2 and 3.2.3, the names "Haar", "Coif1" and "Db5" denote Wavelet filter banks names characterizing the DWT. The gradient representation simply consists in taking the spatial gradient of an image, as done in the Total Variation approach.

[5] it can be noted that the basis function should be scaled to have the same norm (energy), which is the case for the considered representations



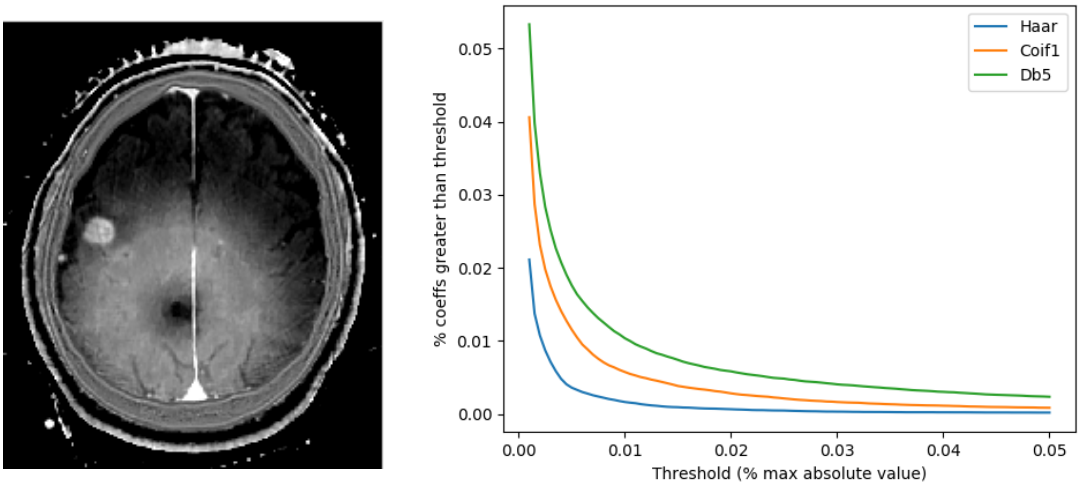
(a) (b)

Figure 3.2.1: Compressibility of the *Shepp-Logan* phantom in various representations. (a) Shepp-Logan phantom. (b) Compressibility ratio as a function of the threshold. Given a threshold, a small compressibility ratio indicates a compressible image.



(a) (b)

Figure 3.2.2: Compressibility of the brain phantom in various representations. (a) Brain phantom. (b) Compressibility ratio as a function of the threshold. Given a threshold, a small compressibility ratio indicates a compressible image.



(a) (b)

Figure 3.2.3: Compressibility of a head CT slice image in various representations. (a) Head CT slice scanned at ESRF ID17. (b) Compressibility ratio as a function of the threshold. Given a threshold, a small compressibility ratio indicates a compressible image. The gradient representation was omitted as it yields too large values to be represented together with the other representations.

The spatial gradient representation shows good compressibility results for the Shepp-Logan phantom (Figure 3.2.1), which was expected due to the piecewise-constant nature of this image. On a more elaborate model like the brain phantom (Figure 3.2.2), the gradient does not perform so well. The wavelets representations, on the other hand, show interesting compressibility properties, and also for the image of a real reconstruction (Figure 3.2.3). The Haar wavelets transform can be seen as a multi-resolution gradient, including the "approximation coefficient" (which is not a gradient but an "average image").

The Wavelets decompositions thus offer a compressibility which is more interesting than the spatial gradient representation, as compressibility holds not only for piecewise-constant images. This first experiment shows that the Haar Wavelets offer the most interesting compressibility for the tested images. The DWT is often referred as a sparsifying transform for *natural images*. The concept of natural image[6] can be loosely defined as follows: given a gray value at pixel coordinates (i, j), $v_{i,j}$, the gray values of neighbouring pixels $(i \pm \epsilon, j \pm \epsilon)$ are *correlated* with $v_{i,j}$. In words, in an image of a given scene, a bright pixel value at one location is likely to have bright pixel values in its neighbourhood. This is in contrast with *random signals*, where neighbouring samples are uncorrelated. Random images do not have any structure that can be interpreted; thus, in imaging applications, these are *non-natural* images.

The Wavelet transform exploits this correlation between adjacent samples. The more a signal is correlated ("natural"), the more its samples can be predicted from few samples [7] and the more it can be encoded efficiently by a Wavelet transform – or, in the general context of information theory, by some encoder. The DWT is therefore adapted to piecewise-smooth signals, although sharp edges can be preserved by choosing the wavelet.

[6] the concept is straightforwardly extended to other dimensions

[7] this interpretation of the DWT, based on series of "predict-update" is at the roots of the *lifting scheme*, an efficient implementation of DWT

The second numerical experiment available in [Pal17] measures the coherence between the projector and the aforementioned representations. To make the computation of the coherence easier, the operators were represented as dense matrices, which limited the size of the input/output spaces to 256×256 images. However, the coherence seems to follow a trend with respect to the representations. Results are reported in Table 3.3.

	32	64	128	256
$\boldsymbol{I}$	10.1 (31.6%)	14.8 (23.2%)	21.3 (16.6%)	30.3 (11.8%)
Haar	10.9 (34.2%)	16.1 (25.2%)	25.6 (20.0%)	39.3 (15.4%)
Coif1	13.5 (42.3%)	22.0 (34.4%)	34.4 (26.9%)	50.3 (19.7%)
Db2	12.6 (39.2%)	20.4 (31.8%)	30.2 (23.6%)	44.7 (17.5%)
Grad	6.6 (20.5%)	9.6 (15.0%)	13.9 (10.8%)	X

Table 3.3: Coherence between the projection operator $\boldsymbol{P}$ and a representation, for various sizes of $\boldsymbol{P}$. Horizontally, the coherence is computed as a function of the number of pixels in the image width (32×32, 64×64, ...). The coherence of operators acting on $N \times N$ images has a maximum value of $\sqrt{N \cdot N} = N$; thus, the coherence percentage with respect to this maximum value is also indicated. It was found that the coherence does not depend on the number of projections. Computing the coherence for sizes above 256 would entail to implement all the involved operators with a sparse representation, which was not undertaken. The "X" symbol on the last entry indicates that the computation failed due to an insufficient amount of memory.

From Table 3.3, it appears that the gradient has the best incoherence with the projection operator. This observation is compatible with the early success of TV in tomographic reconstruction. The identity $\boldsymbol{I}$ ("spikes basis") shows the second best incoherence; however, most images are not sparse in the natural representation (identity basis). The Haar wavelet transform comes third in terms of incoherence. The fact that the Haar wavelets has the best incoherence among all wavelets is not surprising, as it is relatively close to the Gradient representation[8].

The conclusion of these two numerical experiments is that the Haar Wavelet decomposition in the theoretical framework of CS, as it offers the best trade-off between compressibility and incoherence with the tomography projector. This motivates the high-performance implementation of the DWT for being integrated in tomography reconstruction projects.

3.2.2 The Discrete Wavelet Transform

In this subsection, we recall how the computation of the discrete wavelet transform is performed – the reader familiar with DWT might skip it. Let $\boldsymbol{S}$ be a signal, one dimensional (time-dependent) for clarity. In practice, the signal is represented by discrete samples $(S_1, \ldots, S_N)$, hereafter denoted $\boldsymbol{S}(n)$. One stage of the Discrete Wavelet Transform maps the signal $\boldsymbol{S}(n)$ to two signals $\boldsymbol{S}_a(k)$ and $\boldsymbol{S}_d(k)$: the approximation coefficients and the detail coefficients, respectively.

To obtain the approximation coefficients, the input signal $\boldsymbol{S}(n)$ is convolved with a low-pass filter $\boldsymbol{g}(n)$. Conversely, the detail coefficients are computed with a convolution with a high-pass filter $\boldsymbol{h}(n)$. These filters are related to the scaling and wavelet function

[8] the Haar representation can be seen as a *multi-scale* gradient representation, with an extra gradient direction (diagonal)

through the two-scale relations [She96]. A multi-resolution analysis can be carried out by repeating the process on the approximation coefficients $S_a(k)$, as depicted on Figure 3.2.4.

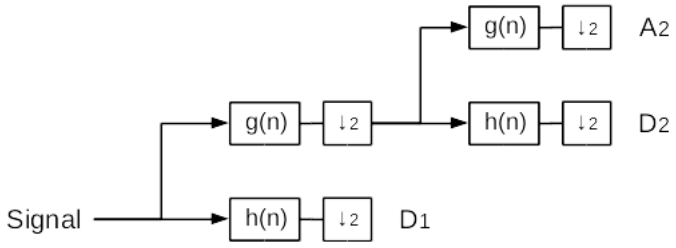


Figure 3.2.4: Discrete Wavelet transform of a signal based on a filter bank. $\boldsymbol{h}$ is the high-pass filter and $\boldsymbol{g}$ is the low-pass filter. The operation $\downarrow 2$ denotes the subsampling by a factor of two. $\boldsymbol{A}_k$ and $\boldsymbol{D}_k$ are the vectors of approximation and detail coefficients at scale k, respectively.

This approach for computing the discrete wavelet transform uses a filter bank, and is sometimes called two-channel subband coder. The reconstruction process, i.e computing the signal $\boldsymbol{x}(n)$ from its approximation and detail coefficients at various scales, follows the same principle by replacing the convolution-subsampling steps with upsampling-convolution steps, where the filters $\boldsymbol{g}(n)$ and $\boldsymbol{h}(n)$ are replaced with reconstruction filters $\tilde{\boldsymbol{g}}(n)$ and $\tilde{\boldsymbol{h}}(n)$.

Comparing with the definition of CWT, it can be shown [Dau+92] that DWT boils down to computing CWT choosing dilation steps $a = 2^j a_0$ and time shifts $\tau = k2^j\tau_0$ for $j, k \in \mathbb{N}$. The DWT then follows a dyadic sampling of the frequency axis, as the scales are a power of two. Since the analysis frequency is halved at each scale, it is unnecessary to keep all the signal samples, hence the decimation steps ($\downarrow 2$) in the DWT process. The standard DWT is said to be dyadic (or critically sampled): at each level, the number of coefficients is halved by subsampling the convolution results. For example given a signal of length N decomposed with three scales, the detail coefficients have length $N/2$, $N/4$ and $N/8$ at scales 1, 2, 3, respectively.

Another crucial property of the DWT is perfect reconstruction, i.e the ability to reconstruct the signal up to numerical error from the coefficients. These two properties, combined with the fact that only linear operations – convolution and subsampling – are involved, make the DWT a linear and invertible transform.

Applying the filter-based process (Figure 3.2.4) to two dimensional images outputs yields *four* coefficients vectors: the approximation coefficients, and the detail coefficients corresponding to three orientations: horizontal, vertical, and diagonal. These coefficients are hereby denoted $(\boldsymbol{A}, \boldsymbol{H}, \boldsymbol{V}, \boldsymbol{D})$. The number of coefficients *per dimension* is halved, meaning that each of the coefficient vectors $(\boldsymbol{A}, \boldsymbol{H}, \boldsymbol{V}, \boldsymbol{D})$ at level one has four times less values than the input image. In the second level, the filter bank is applied on A, and each vector of the output $(\boldsymbol{A}_2, \boldsymbol{H}_2, \boldsymbol{V}_2, \boldsymbol{D}_2)$ has four times less values than $\boldsymbol{A}$, and so on. Figure 3.2.5 illustrates the computation of one step of the 2D DWT.

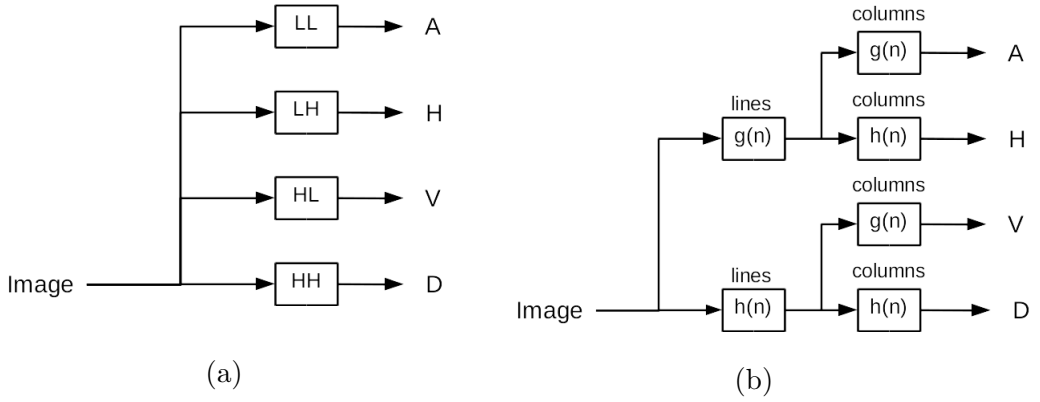


Figure 3.2.5: 2D separable Discrete Wavelet Transform. For clarity, the subsampling steps $\downarrow 2$ have been omitted. (**a**) LL, LH, HL and HH are the filters computing the approximation coefficients (A) and the horizontal (H), vertical (V), diagonal (D) detail coefficients respectively. (**b**) When the filters are separable, convolutions can be advantageously computed by first performing one dimensional convolutions on each line, then one dimensional vertical convolutions on each column. Formally, for separable convolution, we have LL $= \boldsymbol{g}\boldsymbol{g}^T$, LH $= \boldsymbol{g}\boldsymbol{h}^T$, HL $= \boldsymbol{h}\boldsymbol{g}^T$ and HH $= \boldsymbol{h}\boldsymbol{h}^T$.

From this filter bank implementation of DWT, it is clear that convolution is the key operation of DWT. The convolution of a two dimensional image $\boldsymbol{I}(x, y)$ with a kernel $\boldsymbol{K}(x, y)$ is defined by equation (3.2.2), where the summations actually take place on the support of $\boldsymbol{I}(x, y)$ with boundary conditions.

$$(I * K)(x, y) = \sum_{u=-\infty}^{\infty} \sum_{v=-\infty}^{\infty} I(u, v) K(x - u, y - v) \tag{3.2.2}$$

The DWT, despite its attractive properties, lacks of translation invariance in the sense that the wavelet coefficients of a translated signal $\boldsymbol{x}(t - \tau)$ will not be the translated coefficients of $\boldsymbol{x}(t)$. This lack of translation invariance can lead to artifacts when the wavelet coefficients are modified. Translation invariance can be achieved by averaging the results of DWT with all possible shifts (much less in practice), or by computing a *stationary wavelet transform* (SWT) [CD95]. The SWT is computed by increasing the "distance between the samples" at each level when convolving: this is done by both removing the downsampling after the convolution, and inserting zeros between filter coefficients. For this reason, this method is called the *a-trous* algorithm [She92]. This transform is redundant: the total number of output coefficients is higher than the number of input samples. It involves more computations and a larger memory consumption, but achieves exact shift invariance.

3.2.3 Implementation of the Discrete Wavelet Transform

The interesting theoretical properties of the Wavelet Transform motivated a high performance GPU implementation. In order to re-use this implementation for other regularized inverse problems (not only tomography), this project was implemented as a library called PDWT (Parallel Discrete Wavelet Transform). At the time (2016), to the best of our knowledge, no GPU implementation of the DWT provided the following wanted features:

- Source code freely and publicly available, with a permissive license

- Easy to integrate in an existing project
- DWT decomposition and reconstruction, with arbitrary image shapes (not only powers of two)
- Availability of different filter banks, not only the Haar transform
- Translation invariance, either by cycle spinning (see 3.2.5) or stationary wavelet transform

The implementation of PDWT targets modern Nvidia GPUs and uses the CUDA programming language. The library is primarily designed to handle two dimensional (2D) transforms, although it can handle one dimensional transforms as well.

In the following, the parallel stencil access pattern necessary for convolution is described. Then, the GPU implementation of different transforms is detailed with the CUDA terminology. The description of CUDA execution model is out of the scope of this paper; the reader can refer to [Nvi15] for example.

Parallel convolution and stencil pattern

The key operation of DWT, as described in 3.2.2, is convolution. The standard filter kernels involved in most Wavelet Transforms have a relatively small support (less than per 40 samples per dimension), therefore, convolution is more efficiently implemented in the direct domain (i.e without Fast Fourier Transform). The parallel implementation of the convolution follows a *stencil* memory access pattern: each output sample at location (x, y) is a weighted sum on a square neighborhood centered on (x, y), the weights being the coefficients of the kernel $\boldsymbol{K}$.

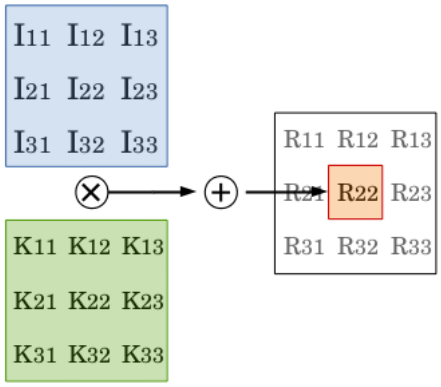


Figure 3.2.6: Convolution stencil pattern. The blue (top-left) square is a subset of the input image, the green (bottom-left) square is the (mirrored) 3×3 convolution kernel, and the white (right) square is the corresponding subset of the output image. Each output sample is the sum of the element-wise multiplications of the mirrored kernel and the corresponding subset of the image. On this example, we have $R_{2,2} = \sum_{j=1}^{3} \sum_{i=1}^{3} K_{i,j} I_{i,j}$.

This memory access pattern is especially interesting, especially in one dimension, for it enables *coalesced memory access*. A memory access is said to be coalesced if adjacent work items (cuda threads in this implementation) access to contiguous memory locations.

On GPUs, memory transactions are done by *group of work items* (*warps* in CUDA terminology): each memory read/write entails to actually process T bytes, where $T = 128$ for modern Nvidia GPUs. Thus, aligning the memory access is essential for performances. For example, accessing 32 values of a `float32` image *line* can be done in one memory transaction, while accessing 32 values of an image *column* requires 32 memory transactions. The reason is that GPU languages (CUDA, OpenCL) are a row-major: the image values are stored row-wise. Thus, successive elements in a row are contiguous in memory, while successive elements in a column are not.

The convolution can be computed as a separable convolution if the convolution kernel is separable. In two dimensions, the kernel K is separable if it is the outer product of two one dimensional kernels K_x and K_y. The convolution is then computed by performing a one dimensional convolution on all the lines, then a one dimensional convolution on all the columns. Separable convolution has a notable computational advantage over nonseparable convolution. If the image size is n^2 and the kernel size is k^2, nonseparable convolution requires $n^2 \times k^2$ operations while separable convolution requires $2 \times n^2 \times k$ operations.

In separable convolution, the stencil pattern is one dimensional. Memory access are coalesced for horizontal convolution, and one might think that the access are not coalesced for vertical convolution. However, for Nvidia cards from the Fermi generation, benchmarks show that horizontal convolution and vertical convolution take the same amount of time. We believe that the reason is that "smarter" caching mechanism, friendly with SIMD lines in whatever direction, occur from this generation. This hypothesis is substantiated by the fact that kernels involving both horizontal and vertical access patterns (at the same time) have a much slower execution time. Thus, parallel separable convolution has a major computational advantage over parallel nonseparable convolution.

Overall execution model

The functions compiled and executed on the GPU device are called kernels (CUDA kernel in this implementation), which are not to be confused with the convolution kernels like $g(n)$, $h(n)$. Ensuring good performances entails to carefully analyze two main possible bottlenecks: input/output (IO) and computations. A program is said IO-limited or compute-limited when the bottleneck is the IO throughput or the computational power, respectively.

The IO limitations are alleviated by using coalesced memory accesses when possible, and by storing the convolution kernels coefficients in a cached memory called constant memory. Besides, all devices buffers, where the image and coefficients are stored, are pre-allocated in a structure. All the device operations like forward transform, inverse transform and thresholding are performed on the buffers of this structure, so that memory allocations and copies do not interfere with the actual computing.

The computations are optimized by a careful design of the kernels, which is the object of the following sections.

Discrete Wavelet Transform

The separable forward discrete wavelet transform (DWT) is implemented following the Mallat multiresolution algorithm [Mal89] with the following principle. The input image has a size (N_c, N_r) where N_c and N_r are the number of columns and rows of the image, respectively. After the first level of the transform, each coefficients image $(\boldsymbol{A}, \boldsymbol{H}, \boldsymbol{V}, \boldsymbol{D})$ has a size $(N_c/2, N_r/2)$.

In the first stage, horizontal convolution with the wavelet kernel is performed on all the lines of the input image. This convolution is merged with the subsampling step, which means that it does not consists in computing the convolution and then the subsampling the result. Instead, the sample k of the output current line is computed from a neighborhood centered on sample $2 \times k$ in the input. Thus, an intermediary image has to be created for the result of horizontal convolution and horizontal subsampling. Therefore, a grid of $(N_c/2, N_r)$ threads is launched; each thread computes the value of a pixel (for the four images $\boldsymbol{A}, \boldsymbol{H}, \boldsymbol{V}, \boldsymbol{D}$).

In the second stage, the intermediary image of size $(N_c/2, N_r)$ is taken as an input for the vertical convolution and vertical subsampling. A grid of $(N_c/2, N_r/2)$ threads is launched, each thread computing the value of one sample of each four image coefficients $\boldsymbol{A}, \boldsymbol{H}, \boldsymbol{V}, \boldsymbol{D}$. This principle is illustrated on Figure 3.2.7.

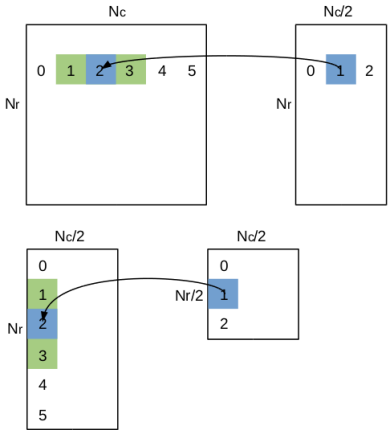


Figure 3.2.7: Illustration of the two-stage separable convolution-subsampling with a 6×6 image and a 3×3 filter kernel. In the first stage (top), the output current line at index 1 (blue) is computed from the subset of 3 input samples (green), centered on index $1 \times 2 = 2$. The result (right) has half the number of columns of the input. The same principles goes for the second stage, replacing lines with columns.

The output of the first level of DWT consists in four coefficient images $\boldsymbol{A}, \boldsymbol{H}, \boldsymbol{V}, \boldsymbol{D}$; each having size $(N_c/2, N_r/2)$. To compute a second level of the transform, the process previously described is applied on approximation coefficients $\boldsymbol{A}$.

Two temporary arrays need to be allocated for this separable transform. Indeed, the DWT consists in convolving the image with four directional 2D filters (LL, LH, HL, HH). In the separable case, these filters are generated from two 1D filters $\boldsymbol{g}(n)$ and $\boldsymbol{h}(n)$, as described on Figure 3.2.5. The first stage is an horizontal convolution and subsampling with filter $g(n)$, which creates a temporary image of size $(N_c/2, N_r)$; and an horizontal convolution and subsampling with filter $h(n)$, which creates another such temporary image. These two images are then vertically convolved-subsampled with filters $\boldsymbol{g}(n)$ and $\boldsymbol{h}(n)$ to create the four possible combinations of operations, yielding the coefficients A, H, V, D. This is detailed in Algorithm 3.2.1 where tmp1 and tmp2 are the aforementioned temporary arrays, and pdwt_pass1, pdwt_pass2 are kernels computing the horizontal convolution-subsampling and vertical convolution-subsampling, respectively.

Algorithm 3.2.1 Forward separable DWT

I: input image
q: number of decomposition levels

procedure DWT(I, q) ▷ First level
  pdwt_pass1(I, tmp1, tmp2) ▷ Horizontal convolution and subsampling
  pdwt_pass2(tmp1, tmp2, A, H_1, V_1, D_1) ▷ Vertical convolution and subsampling
  ▷ Other levels
  for $k \leftarrow 2, q$ **do**
    pdwt_pass1(A, tmp1, tmp2) ▷ Horizontal convolution and subsampling
    pdwt_pass2(tmp1, tmp2, A, H_k, V_k, D_k) ▷ Vertical convolution and subsampling
  end for
end procedure
return (A, H_1, V_1, D_1, ..., H_q, V_q, D_q)

The two kernels implementing the two steps of the transform can be used for all the wavelets, as the wavelets are fully determined by the filter coefficients $g(n)$ and $h(n)$. The DWT with the Haar wavelet has a dedicated kernel which does not need to handle boundary conditions, and is thus faster.

The extension of the principle depicted on Figure 3.2.7 for non-separable convolution is straightforward. Instead of performing a two stages convolution-subsampling reducing the number of lines and columns, all the steps are merged in a single pass. As highlighted in section 3.2.3, this transform will not benefit from the line/columns cache mechanism, making it slower. However, non-separable convolution has been implemented for extensibility, as it enables to add non-separable user-defined wavelets.

Inverse Discrete Wavelet Transform

The (two dimensional) inverse wavelet transform maps a set of coefficient images $(\boldsymbol{A}, \boldsymbol{H}_1, \boldsymbol{V}_1, \boldsymbol{D}_1, \ldots, \boldsymbol{H}_q, \boldsymbol{V}_q, \boldsymbol{D}_q)$ back to an image. Under certain properties, the wavelet transform satisfies the *perfect reconstruction property*: the reconstructed image is equal to the image which gave the coefficients, up to numerical errors.

The inversion follows the same principle depicted on Figures 3.2.4 and 3.2.5, except that the convolution-subsampling operation is replaced with an upsampling-convolution operation. Let $S(n)$ be a sampled signal (1D for simplicity), and $A(n)$, $D(n)$ be its approximation and detail coefficients resulting of one level of DWT. These coefficients were formally obtained with

$$\begin{aligned} \boldsymbol{A} &= (\boldsymbol{S} * \boldsymbol{g}) \downarrow 2 \\ \boldsymbol{D} &= (\boldsymbol{S} * \boldsymbol{h}) \downarrow 2 \end{aligned} \tag{3.2.3}$$

where $\boldsymbol{g}, \boldsymbol{h}$ are the low-pass and high-pass filter kernels, respectively. Then, the signal $S(n)$ can be reconstructed (up to numerical error) with

$$\boldsymbol{S} = (\boldsymbol{A} \uparrow 2) * \tilde{\boldsymbol{g}} \; + \; (\boldsymbol{D} \uparrow 2) * \tilde{\boldsymbol{h}} \tag{3.2.4}$$

where $\tilde{\boldsymbol{g}}, \tilde{\boldsymbol{h}}$ are the low-pass and high-pass filter kernels used for the reconstruction, respectively. The inversion (3.2.4) bears similarities with the algebraic transpose of (3.2.3). If $\boldsymbol{C_g}$ and $\boldsymbol{C_h}$ denote the Toeplitz matrix associated to filter kernels $\boldsymbol{g}(n)$ and $\boldsymbol{h}(n)$, the

forward transform can be algebraically expressed as

$$\begin{pmatrix} \boldsymbol{A} \\ \boldsymbol{D} \end{pmatrix} = \begin{bmatrix} \mathcal{D}_2 \boldsymbol{C_g} \\ \mathcal{D}_2 \boldsymbol{C_h} \end{bmatrix} \boldsymbol{S} \tag{3.2.5}$$

where the input and output signals are stacked as 1D vectors, and $\mathcal{D}_2$ denotes the downsampling $\downarrow 2$. The inverse DWT operation can then be written as

$$S = \begin{bmatrix} C_{\tilde{g}} \mathcal{D}_2^T & ; & C_{\tilde{h}} \mathcal{D}_2^T \end{bmatrix} \begin{pmatrix} A \\ D \end{pmatrix} = \left(C_{\tilde{g}} \mathcal{D}_2^T \mathcal{D}_2 C_g \; + \; C_{\tilde{h}} \mathcal{D}_2^T \mathcal{D}_2 C_h \right) S \tag{3.2.6}$$

where the filters $\tilde{g}$ and $\tilde{h}$ are designed so that the forward-inverse process is the identity :

$$C_{\tilde{g}} \mathcal{D}_2^T \mathcal{D}_2 C_g \; + \; C_{\tilde{h}} \mathcal{D}_2^T \mathcal{D}_2 C_h = \mathrm{Id} \tag{3.2.7}$$

From equations (3.2.5) and (3.2.6), it is clear that in the case where $C_{\tilde{g}} = C_g^T$ and $C_{\tilde{h}} = C_h^T$ (i.e $\tilde{g}, \tilde{h}$ are the matched filters of g, h), then the inverse is exactly the algebraic transpose. The transform is said to be orthogonal: if $\mathcal{W}$ denotes the whole wavelet transform, then $\mathcal{W}^T \mathcal{W} = \mathrm{Id}$.

Understanding that the inverse transform is almost the algebraic transpose of the forward transform (exactly in case of orthogonal wavelet transform), the implementation of the inverse transform is straightforward. Figure 3.2.8 depicts the inverse 2D separable DWT. The forward DWT operator $\begin{bmatrix} \mathcal{D}_2 C_g \\ \mathcal{D}_2 C_h \end{bmatrix}$ depicted on Figure 3.2.5 splits, convolves and subsamples the input signal. The transpose $\begin{bmatrix} C_{\tilde{g}} \mathcal{D}_2^T & ; & C_{\tilde{h}} \mathcal{D}_2^T \end{bmatrix}$ upsamples, convolves and sums the input coefficients.

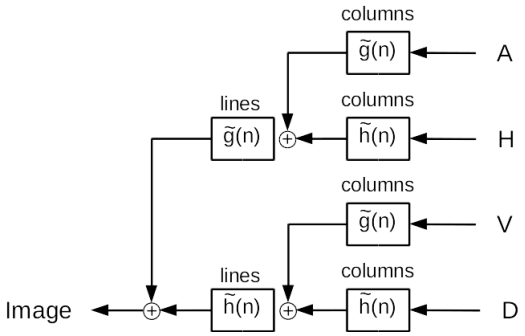


Figure 3.2.8: 2D separable Inverse Discrete Wavelet Transform. For clarity, the upsampling steps $\uparrow 2$ have been omitted. The diagram reads from right to left: the coefficients $(\boldsymbol{A}, \boldsymbol{H}, \boldsymbol{V}, \boldsymbol{D})$ are column-convolved with filters $\tilde{\boldsymbol{g}}(n)$ and $\tilde{\boldsymbol{h}}(n)$. The results are summed according to the splitting made in forward DWT: $\boldsymbol{H} * \tilde{\boldsymbol{h}}$ is added to $\boldsymbol{A} * \tilde{\boldsymbol{g}}$ and $\boldsymbol{D} * \tilde{\boldsymbol{h}}$ is added to $\boldsymbol{V} * \tilde{\boldsymbol{g}}$. The result are line-convolved again and summed to get the reconstructed image.

The parallel separable inversion follows the same two-stages process as the separable forward transform. Given the four coefficients sets $(\boldsymbol{A}, \boldsymbol{H}, \boldsymbol{V}, \boldsymbol{D})$ each of size $(N_c/2, N_r/2)$, *upsampled convolutions* are performed along the columns. Upsampled convolution here consists in convolving with an upsampled input, which is the transpose operation of downsampling the result of a convolution. In the forward transform, the downsampling of the convolution result was achieved by launching a thread grid with twice less threads than input samples (per dimension), performing the convolution with a "step" (*stride*) of 2 for

a 2-downsampling. Here, for the first stage of the inversion, a grid of $(N_c/2, N_r)$ threads is launched. Each thread will write a sample in two temporary buffers of size $(N_c/2, N_r)$: one for $\boldsymbol{A} * \tilde{\boldsymbol{g}} + \boldsymbol{H} * \tilde{\boldsymbol{h}}$ and one for $\boldsymbol{V} * \tilde{\boldsymbol{g}} + \boldsymbol{D} * \tilde{\boldsymbol{h}}$ as described on Figure 3.2.8.

The upsampled convolution is done with two separate convolutions with the even and odd samples indexes of the filters. If $A = \begin{pmatrix} a_1 & , & a_2 & , & \dots & , & a_M \end{pmatrix}$ is the approximation coefficient and $\overline{A} = \begin{pmatrix} a_0 & , & 0 & , & a_1 & , & 0 & , & \dots & , & a_{M-1} & , & 0 \end{pmatrix}$ denotes the upsampled version of $\boldsymbol{A}$, then the convolution with $\tilde{\boldsymbol{g}} = \begin{pmatrix} \tilde{\boldsymbol{g}}_0 & , & \dots & , & \tilde{\boldsymbol{g}}_{P-1} \end{pmatrix}$ is

$$\begin{aligned} \left(\overline{\boldsymbol{A}} * \tilde{\boldsymbol{g}}\right)(k) &= \sum_{i=0}^{P-1} \tilde{g}_{P-1-i} \overline{a}_{k-P/2+i} \qquad \text{where} \quad \overline{a}_i = \begin{cases} a_{i/2} & \text{if } i \text{ is even} \\ 0 & \text{otherwise} \end{cases} \\ &= \begin{cases} \displaystyle\sum_{j=0}^{P/2-1} (\tilde{g}_e)_{P/2-1-j} a_{k/2-P/4+j} & \text{if } i \text{ is even} \\ \displaystyle\sum_{j=0}^{P/2-1} (\tilde{g}_o)_{P/2-1-j} a_{k/2-P/4+j} & \text{otherwise} \end{cases} \end{aligned} \tag{3.2.8}$$

where $(\tilde{g}_e)$ and $(\tilde{g}_o)$ denote the filters extracted from $\tilde{g}$ by retaining only even and odd coefficients, respectively. This convolution thus consists in alternating between two convolutions with the even-indexed and odd-indexed samples of $\tilde{g}$. If $P = 4$, then $\tilde{g} = \begin{pmatrix} \tilde{g}_1 & , & \tilde{g}_2 & , & \tilde{g}_3 & , & \tilde{g}_4 \end{pmatrix}$ and the first output samples are :

$$\begin{aligned} &1: \quad \tilde{g}_2 a_0 + \tilde{g}_0 a_1 \qquad &&2: \quad \tilde{g}_3 a_0 + \tilde{g}_1 a_1 \\ &3: \quad \tilde{g}_2 a_1 + \tilde{g}_0 a_2 \qquad &&4: \quad \tilde{g}_3 a_1 + \tilde{g}_1 a_2 \end{aligned} \tag{3.2.9}$$

As there are twice more threads launched than samples in the input coefficient, threads with even (resp. odd) indexes will take part in the computation of the first (resp. second) convolution.

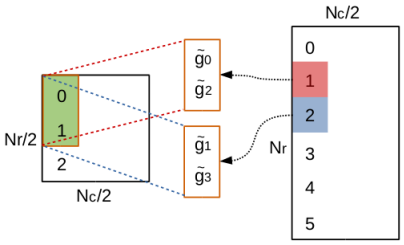


Figure 3.2.9: Illustration of the first stage of the separable inversion. The input coefficient (left) has a size $(N_c/2, N_r/2)$ and the temporary buffer (right) has a size $(N_c/2, N_r)$ due to the vertical oversampling. Even-indexed threads write at the even-indexed samples locations of the output buffer (red) by performing a convolution with the even-indexed filter samples (top-center orange box). The same goes for odd-indexed threads, performing a convolution with the odd-indexed filter samples (bottom-center orange box) to write at odd-indexed output locations (blue).

Stationary Wavelet Transform

Although providing a fast and memory-efficient representation, the standard discrete wavelet transform suffers from *shift variance* in the sense that the wavelet coefficient of a shifted version $S(t-\tau)$ of a signal $S(t)$ are not directly related to the shifted wavelet

coefficients of $S(t)$. Even the energy by scale is not the same for the coefficients of $S(t-\tau)$ and $S(t)$ [SBK05]. This sensitivity to translation is detrimental to applications like signal analysis or denoising.

To overcome this limitation of the DWT, a redundant version of the transform has been proposed [CD95]. The principle is to remove the subsampling steps in the forward transform (Figure 3.2.4), and conversely the upsamplers in the "inverse" transform. This library implements the Undecimated Wavelet Transform, also known as Stationary Wavelet Transform (SWT) following the *à trous* algorithm [She92] [SF09b].

The SWT is an over-complete transform, which means that the transform output has more samples than the input. More precisely, given a discrete one-dimensional signal of length $N > 0$, applying $l \geq 1$ decomposition levels yields $(l+1)N$ samples as there is no subsampling. In two dimensions, applying $l \geq 1$ decomposition levels on a $N_c \times N_r$ image yields $(3l+1)N_r \times N_c$ samples. In q dimensions, the *redundancy factor* of level $l \geq 1$ SWT is $(2^q - 1)l + 1$. This high redundancy comes at the expense of memory. More recent transforms like the complex wavelet transform were designed to achieve both translation invariance and a lower redundancy [SBK05].

The implementation is similar to the standard DWT described in 3.2.3, except for two points. First, there is no subsampling during the transform. As convolution and subsampling were performed at the same time by one CUDA kernel for DWT, new CUDA kernels had to be written for SWT. Second, the wavelet low-pass and high-pass filters are *upsampled* by a factor of two at each scale in the *à trous* algorithm [She92]. For example, for a filter $\boldsymbol{g} = (g_0, g_1, \ldots, g_n)$, the filter used at scale two is $(\boldsymbol{g} \uparrow 2) = (g_0, 0, g_1, 0, \ldots, g_n, 0)$, the filter used at scale three is $(\boldsymbol{g} \uparrow 4) = (g_0, 0, 0, 0, \ldots, g_n, 0, 0, 0)$ and so on. Upsampling the filter is equivalent to read the data with a strided access, so there is no need to allocate new filters. Here the stride is 2^{l-1} where l is the current decomposition level, starting from one.

The memory read footprint is higher for SWT, as for a filter of length n, $2^{l-1} \cdot n$ data samples have to be read at decomposition level $l \geq 1$ for each output sample. The memory write, however, is still coalesced by design: thread number k write the convolution result sample at location k (see subsection 3.2.3).

As the SWT is an over-complete transform, a given set of coefficients is not necessarily the result of the SWT of some input data – the output space is bigger than the input space. Hence, the inverse of the SWT is not unique. However, a canonical inversion is given by the tight frame property: for well chosen filter pairs, the SWT is a *semi-orthogonal* transform in the sense of Equation 3.2.10.

$$\mathcal{S}^T \mathcal{S} = (2^q)^l \, \mathrm{Id} \tag{3.2.10}$$

where $\mathcal{S}$ denotes the forward SWT, q is the transform number of dimensions and $l \geq 1$ is the number of decomposition levels. The inverse SWT implemented in this library, denoted by $\mathcal{I}$, is chosen to be $\mathcal{I} = 2^{-q \cdot l} \mathcal{S}$. From this choice of $\mathcal{I}$, for all signals A, B of respective SWT coefficients C_A, C_B, Equation 3.2.11 holds.

$$\langle \mathcal{S} \cdot A \, , \, C_B \rangle = (2^q)^l \, \langle A \, , \, \mathcal{I} \cdot C_B \rangle \tag{3.2.11}$$

where $\langle \cdot \, , \, \cdot \rangle$ denotes the Euclidean inner product. Notably, the energy is larger in the coefficients domain. The inverse SWT hence follows the same principle as the inverse DWT, except that no upsampling is done and the filters are modified following the *à trous* algorithm.

Nonseparable Wavelet Transform

Although the standard 2D DWT is computed with a separable approach (see subsection 3.2.3), the PDWT library also supports nonseparable transforms, which can be interesting for defining a set of custom filters. Instead of performing two transforms steps (horizontal convolution/subsampling, then vertical convolution/subsampling), all the process is performed by one kernel. As highlighted in subsection 3.2.3, the implementation will not benefit from an efficient cache mechanism and the transform will be slower than a separable transform. The nonseparable transform is also implemented for SWT.

3.2.4 Speed performances assessment of PDWT

Comparison with CPU implementations

In the speed comparisons, the reference we chose is PyWavelets [Fil06], a Python package for wavelet transform following the same syntax and conventions as Matlab. PyWavelets has a Python interface, but the core calculations are implemented in C. Another reference candidate could be Matlab, but PyWavelets makes the comparison easier as both our library and PyWavelets are interfaced in Python. Tests carried on the R2016a version of Matlab show that the computation times of Matlab and PyWavelets are comparable.

The first benchmark series was run on a "workstation" machine with a Intel Xeon E5-1607 v2 - 3.00GHz CPU (4 physical cores) and a Nvidia GeForce GTX 750 Ti GPU. This machine is hereafter denoted by "machine 1". The second series was run on a high performance machine with a Intel Xeon E5-2643 v3 - 3.40GHz CPU (12 physical cores) and a Nvidia GeForce GTX Titan X GPU. This machine is hereafter denoted by "machine 2".

Two wavelets were tested: the Haar wavelet and the Daubechies wavelet with 20 vanishing moments. The reason is that the former has the minimal convolution kernel size (2 pixels per dimension), while the latter has the maximum convolution kernel size supported by PDWT (40 pixels per dimension), theoretically giving extremal computation times. As the convolutions are computed in the image domain, convolution with larger kernel sizes entail more computations, hence larger execution times are expected. Image sizes vary from $(128, 128)$ to $(4096, 4096)$ pixels, and the maximum decomposition levels are computed for each size. It can however be noted that PDWT does not require the image to have dimensions which are power of two; and that the image size is only limited by the GPU memory.

Three transforms types were tested: the standard DWT (Figure 3.2.10), its inverse (Figure 3.2.11) and SWT (Figure 3.2.12). The inverse SWT was not tested as PyWavelets currently does not have a native implementation of it.

As the PDWT aims at being used on GPU only, these benchmark do not take CPU-GPU memory transfers times into account. For completeness, Figures 3.2.13 and 3.2.14 show the execution times with the CPU-GPU memory transfers times. The computation performances are hidden by the memory transfer times, which depend only on the image size; thus there is little difference between the 'haar" and "db20" wavelets. It shows that even with explicit CPU-GPU transfers – which make little sense for iterative solving of regularized inverse problem as shown in the Applications part – speed-ups of the order of 10 can be expected with respect to a CPU implementation.

The "db20" decomposition might be faster than the "haar" decomposition on small image sizes. The reason is that the maximum decomposition level on a $(128, 128)$ image is $\lfloor \log_2 \left(\frac{128}{2-1} \right) \rfloor = 7$ for "haar" and $\lfloor \log_2 \left(\frac{128}{40-1} \right) \rfloor = 1$ for "db20". The convolution with

the "haar" kernel, although faster, will be performed 7 times when the convolution with the "db20" kernel will only be performed once.

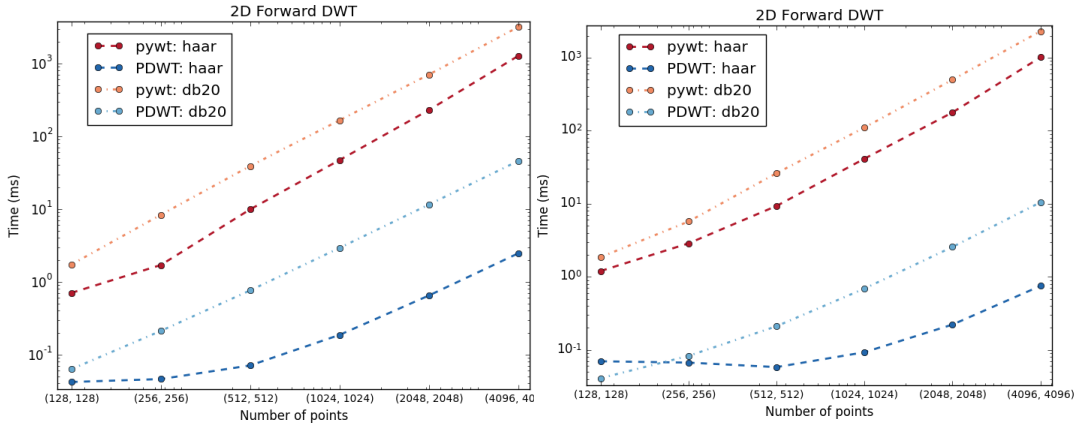


Figure 3.2.10: Execution time results for forward 2D DWT. (a) Machine 1. (b) Machine 2.

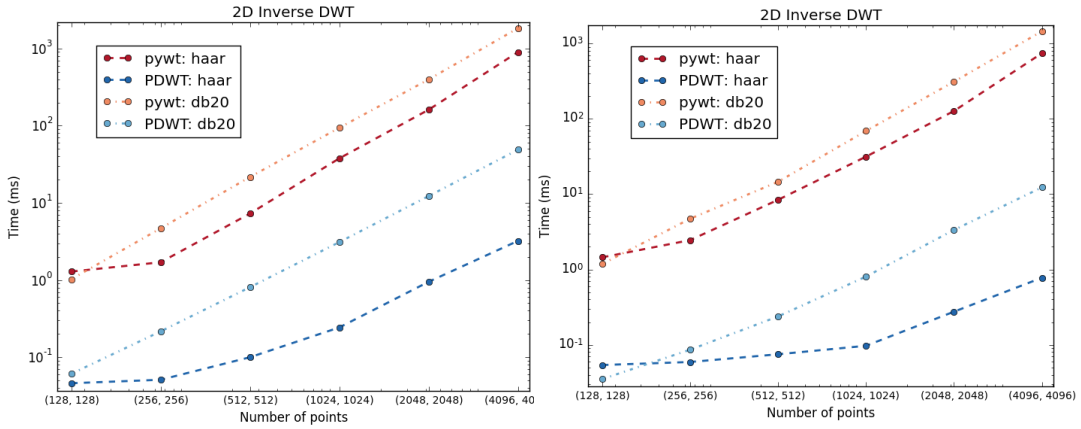
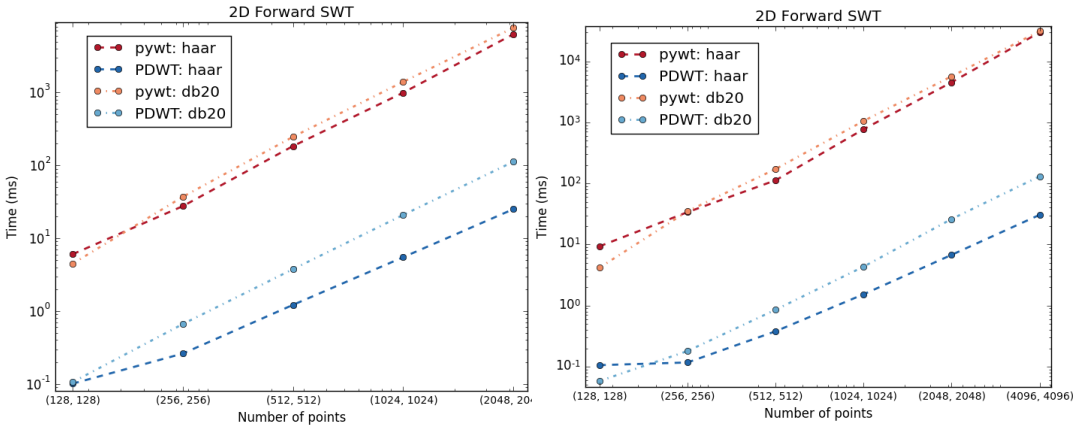
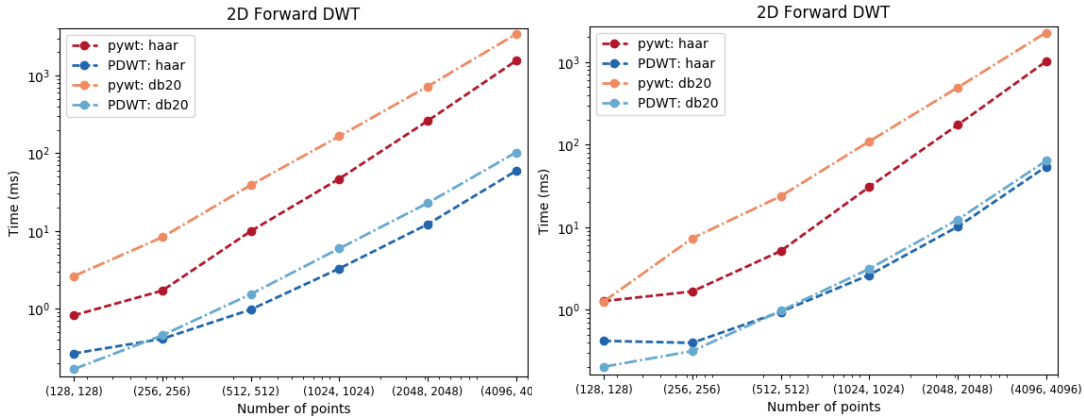


Figure 3.2.11: Execution time results for forward Inverse 2D DWT. (a) Machine 1. (b) Machine 2.



(a) (b)

Figure 3.2.12: Execution time results for forward 2D SWT. (a) Machine 1. The maximal size is $(2048, 2048)$ as there is not enough memory for storing the SWT of a $(4096, 4096)$ image on a GTX 750 Ti (2 GB memory). (b) Machine 2.



(a) (b)

Figure 3.2.13: Execution time results for forward 2D DWT, taking the CPU-GPU memory transfer time into account. (a) Machine 1. (b) Machine 2.

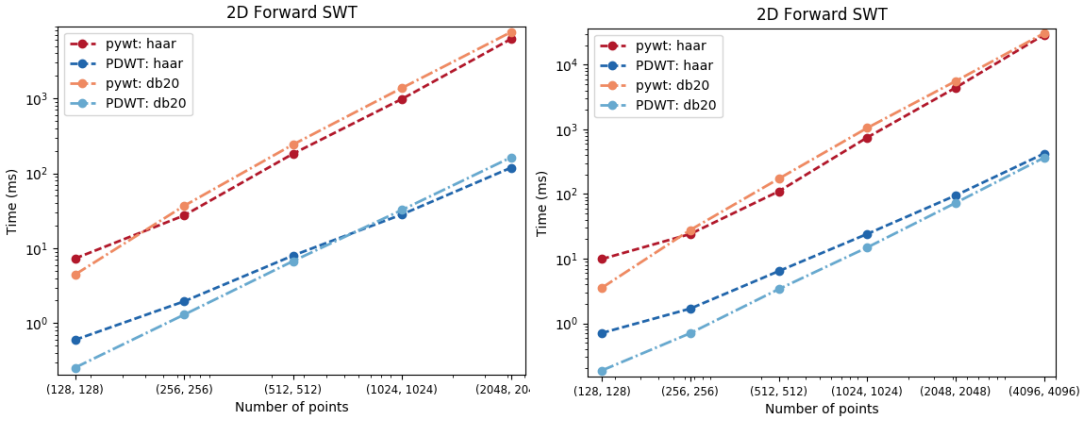


Figure 3.2.14: Execution time results for forward 2D SWT, taking the CPU-GPU memory transfer time into account. (a) Machine 1. (b) Machine 2.

Comparison with a GPU implementation of the 5/3 and 9/7 transforms

Many GPU implementation works of the discrete wavelet transform are published in the literature (see for example [Fra+10]); however, these works seldom provide the source codes. Moreover, implementations usually focus on specific wavelets, for example Haar or Daubechies-Feauveau. Thus, it was difficult to compare our general-purpose wavelet transform with other GPU implementations.

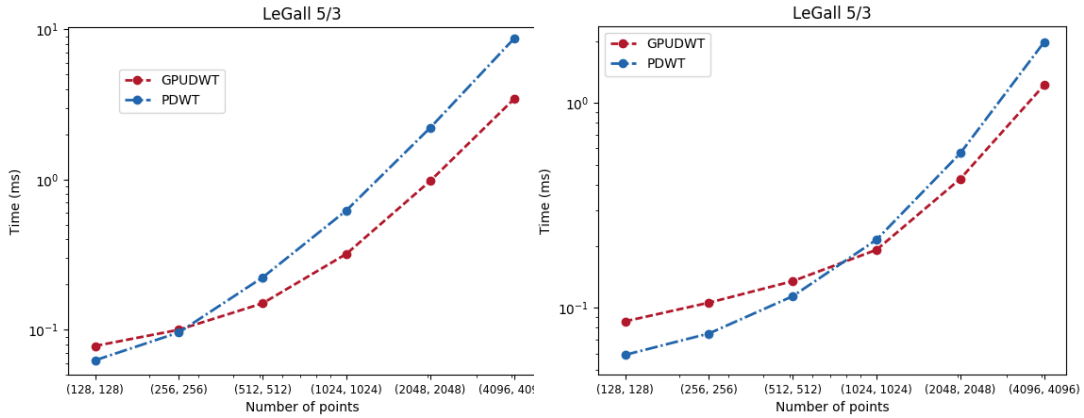
The only publicly available GPU code we could find is "GPUDWT" [Mat09]. It implements the LeGall (LeGall 5/3) and Cohen Daubechies-Feauveau (CDF 9/7) transforms, used in the JPEG2000 standard [TM12]. The transforms are implemented following a lifting scheme [Swe98]. These transforms can be implemented with the classical filter bank approach using the filters defined in Equations 3.2.12 and 3.2.13 for LeGall 5/3 and CDF 9/7, respectively [Bar06].

$$
\begin{aligned}
g_{53} &= \frac{1}{8}\left(-1 \;,\; 2 \;,\; 6 \;,\; 2 \;,\; -1\right) & h_{53} &= \frac{1}{2}\left(-1 \;,\; 2 \;,\; -1\right) \\
\tilde{g_{53}} &= \frac{1}{2}\left(1 \;,\; 2 \;,\; 1\right) & \tilde{h_{53}} &= \frac{1}{8}\left(-1 \;,\; -2 \;,\; 6 \;,\; -2 \;,\; -1\right)
\end{aligned}
\tag{3.2.12}
$$

$$g_{97} = \begin{pmatrix} 0.026748757411 \\ -0.016864118443 \\ -0.078223266529 \\ 0.266864118443 \\ 0.602949018236 \\ 0.266864118443 \\ -0.078223266529 \\ -0.016864118443 \\ 0.026748757411 \end{pmatrix} \quad h_{97} = \begin{pmatrix} 0.091271763114 \\ -0.057543526229 \\ -0.591271763114 \\ 1.11508705 \\ -0.591271763114 \\ -0.057543526229 \\ 0.091271763114 \end{pmatrix}$$

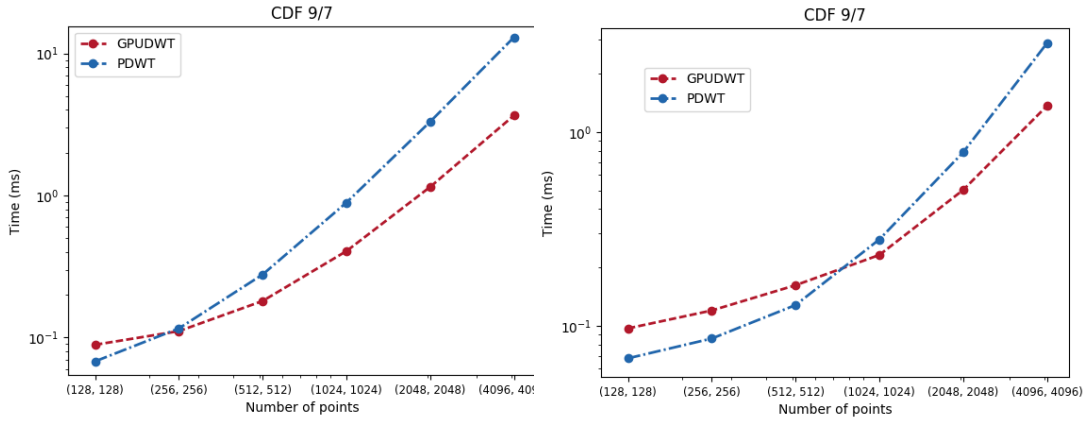
$$g_{\tilde{97}} = \begin{pmatrix} -0.091271763114 \\ -0.057543526229 \\ 0.591271763114 \\ 1.11508705 \\ 0.591271763114 \\ -0.057543526229 \\ -0.091271763114 \end{pmatrix} \quad h_{\tilde{97}} = \begin{pmatrix} 0.026748757411 \\ 0.016864118443 \\ -0.078223266529 \\ -0.266864118443 \\ 0.602949018236 \\ -0.266864118443 \\ -0.078223266529 \\ 0.016864118443 \\ 0.026748757411 \end{pmatrix} \tag{3.2.13}$$

Built-in 5-3 and 9-7 biorthogonal filters pairs could have been used (`bior2.2` and `bior4.4` respectively), but the PDWT library enables to define and use custom filters.



(a) (b)

Figure 3.2.15: Execution time results for LeGall 5/3 transform (a) Machine 1. (b) Machine 2.



(a) (b)

Figure 3.2.16: Execution time results for Cohen-Daubechies-Feauveau 9/7 transform (a) Machine 1. (b) Machine 2.

From Figures 3.2.15 and 3.2.16, the GPUDWT implementation beats the PDWT implementation. We can highlight two reasons. First, GPUDWT has dedicated kernels for each transform, while PDWT uses generic convolution-downsampling kernels. Second, GPUDWT uses a lifting scheme, which is theoretically more computationally efficient than the convolution-based approach. It is interesting to note, however, that the filter bank approach outperforms GPUDWT for small image sizes, and then becomes less efficient for larger sizes. The turning point depends on the amount of cache of the GPU; here the GTX Titan X has more cache (384.0 KiB) than the GTX 750 Ti (80.0 KiB).

3.2.5 Iterative reconstruction with wavelets regularization

This section discusses practical issues when using wavelets as a regularization for tomographic reconstruction. The DWT/SWT were implemented on GPU to provice an efficient evaluation of the operator $\boldsymbol{W}$ for the wavelets-regularized tomographic reconstruction problem

$$\underset{\boldsymbol{x}}{\operatorname{argmin}} \left\{ \frac{1}{2} \left\| \boldsymbol{P}\boldsymbol{x} - \boldsymbol{d} \right\|_2^2 + \lambda \left\| \boldsymbol{W}\boldsymbol{x} \right\|_1 \right\} \tag{3.2.14}$$

which is a special instance of the usual reconstruction problem (2.4.4) with $\boldsymbol{D} = \boldsymbol{W}$.

Shift invariance and thresholding artefacts

It is known that when regularizing inverse problems – even denoising – with wavelets, the reconstruction can bear *thresholding artefacts*. These artefacts appear as "salt and pepper noise" in the result, and are due to aliasing and lack of shift invariance of the regular DWT (see 3.2.2). These two effects have no consequences when inverting the wavelet transform on noiseless coefficients, as the DWT is carefully designed to yield a perfect reconstruction. However, wavelets processing entails to modify the coefficients (usually by thresholding); and both aliasing and shift variance artefacts can occur [SBK05]. These issues can be addressed in two ways:

- using a technique called *cycle spinning*, which aims at reducing these effects by averaging reconstructions of shifted versions of the image

- using another transform with better shift invariance and aliasing cancellation properties, like the undecimated wavelet transform [CD95] or the dual tree complex wavelet transform [SBK05]

Examples are illustrated on Figure 3.2.17.

Let $\boldsymbol{x}$ be a $N \times N$ pixels image to reconstruct. The principle of the first technique is to reconstruct circular shifts of $\boldsymbol{x}$ (there are N^2 such images), and to average them [CD95]. This approach is of course intractable, as reconstructing $\boldsymbol{x}$ is in itself a costly process in the context of iterative tomographic reconstruction. Instead, the circular shifts are performed between the iterations: given a reconstruction estimate at iteration k, $\boldsymbol{x}_k$, the next estimate is computed on a circular shift of $\boldsymbol{x}_k$. Formally, if φ defines the abstract process of computing $\boldsymbol{x}_{k+1}$ from $\boldsymbol{x}_k$, then $\boldsymbol{x}_{k+1} = \sigma^{-1}(\varphi(\sigma(\boldsymbol{x}_k)))$, where σ is a function performing a circular shift of the image. A random shift is used at each iteration, so that many different shifts are made after hundred(s) of iterations. A drawback is that the objective function is not monotonically decreasing: because of the shifts, some low-amplitude jumps can be observed. Fortunately, the algorithm is still globally converging experimentally, although we did not investigate further the convergence guarantees.

The second approach consists in replacing the DWT with an undecimated transform (SWT, see 3.2.3). Although it completely solves the aliasing and shift invariance issues, this approach has two downsides. First, the memory consumption is much higher: while DWT is an invertible transform, SWT is redundant by a factor of $3l$ where $l \in \mathbb{N}$ is the number of decomposition levels. In other words, the decomposition of $\boldsymbol{x}$ yields $3 \times l + 1$ coefficients images of $N \times N$ pixels. In our parallel geometry setting, this approach is tractable. For other 3D geometries where a 3D transform has to be performed, the problem is worse: there are 8 coefficient images per level. This also implies a longer processing time, although in practice the bottleneck of iterative tomographic reconstruction are usually the projection and backprojection operators. The second drawback is on the optimization side. The optimization problem for (analysis) wavelets regularized tomographic reconstruction is

$$\underset{\boldsymbol{x}}{\operatorname{argmin}} \left\{ \frac{1}{2} \|\boldsymbol{P}\boldsymbol{x} - \boldsymbol{d}\|_2^2 + \lambda \|\boldsymbol{W}\boldsymbol{x}\|_1 \right\} \tag{3.2.15}$$

with the usual notations (see for example 3.1.1). Recall from 2.6.2 that the proximal mapping of $\boldsymbol{x} \mapsto \lambda \|\boldsymbol{W}\boldsymbol{x}\|_1$ is straightforward to compute if $\boldsymbol{W}$ is an orthogonal transform, as it is the case for DWT with orthogonal filters. In this case, proximal algorithms solving (3.2.15) can directly use the proximal mapping of the regularization term in this case. For example, FISTA is the algorithm of choice as Problem (3.2.15) has a simple (quadratic) smooth term and a regularization term with simple proximal (see the discussion on proximal algorithms in 3.1.1). Generally, when φ is a scalar function, the prox of $\boldsymbol{x} \mapsto \varphi(\boldsymbol{L}\boldsymbol{x})$ can be expressed as a function of $\operatorname{prox}_{\varphi}(\boldsymbol{x})$ if the linear transform $\boldsymbol{L}$ is semi-orthogonal, i.e $\boldsymbol{L}\boldsymbol{L}^T = \nu\boldsymbol{I}$ for some $\nu > 0$ [CP09]. If (n_2, n) denote the number of lines and columns of $\boldsymbol{L}$ respectively, the previous equality implies $n_2 \leq n$, i.e $\boldsymbol{L}$ *cannot* be an over-complete transform [9]. In other words, if $\boldsymbol{W}$ is the (overcomplete) SWT, the equality $\boldsymbol{W}\boldsymbol{W}^T = \nu\boldsymbol{I}$ never holds for any $\nu > 0$. This precludes the fast computation of the prox of $\boldsymbol{x} \mapsto \lambda \|\boldsymbol{W}\boldsymbol{x}\|_1$.

A first solution could be to use the synthesis formulation, as in the dictionary-based reconstruction

$$\underset{\boldsymbol{w}}{\operatorname{argmin}} \left\{ \frac{1}{2} \left\|\boldsymbol{P}\boldsymbol{W}^T\boldsymbol{w} - \boldsymbol{d}\right\|_2^2 + \lambda \|\boldsymbol{w}\|_1 \right\} \tag{3.2.16}$$

[9] it can easily be seen with the matrix rank. Assume $n_2 > n$. Since $\boldsymbol{L}\boldsymbol{L}^T = \nu\boldsymbol{I}$, we have $\operatorname{rank}(\boldsymbol{L}\boldsymbol{L}^T) = \operatorname{rank}(\boldsymbol{L}) = n_2$. On the other hand, $\operatorname{rank}(\boldsymbol{L}) \leq \min(n, n_2)$, so $n_2 \leq \min(n, n_2) = n$, which contradicts the initial assumption. Hence, $\boldsymbol{L}$ cannot be an overcomplete transform

However, it is not clear whether the synthesis formulation yields results *as least as good as* the analysis formulation in terms of quality[10]. Numerical experiments from our side suggest that the analysis formulation is actually better, which is supported by [SF09a]. This question nevertheless does not have a clear answer in the general inverse problem setting [EMR07]. Optimization considerations on this issue can be found in [Cha+09]. On the implementation side, the synthesis formulation also entail to implement operations (addition, scalar multiplication) in the coefficient domain, which involves more computations for an overcomplete transform.

In order to keep the analysis formulation, we can re-write Problem (3.2.15) as for TV in 3.1.2 to use the Chambolle-Pock algorithm. Another possibility is to solve Problem (3.2.16) with the extra constraint $\boldsymbol{w} \in \text{range}(\boldsymbol{W})$, which makes it equivalent to (3.2.15). This constraint is encoded as a convex indicator function, and the proximal operator is the projection onto the set $\{\boldsymbol{w},\ \exists \boldsymbol{x},\ \boldsymbol{w} = \boldsymbol{W}\boldsymbol{x}\}$. We chose the first approach. Let us note that a trade-off between redundancy and shift invariance can be achieved with the dual tree complex wavelet transform [SBK05]. This transform has recently be implemented and used for clinical CT imaging [Not17] and shows promising results for 3D reconstruction.

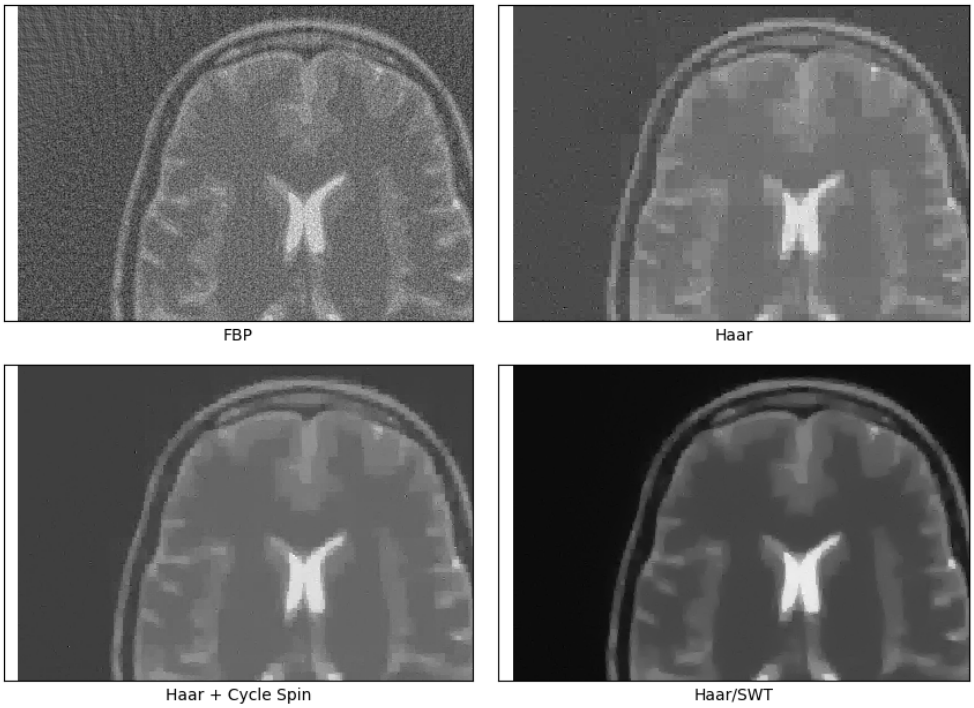
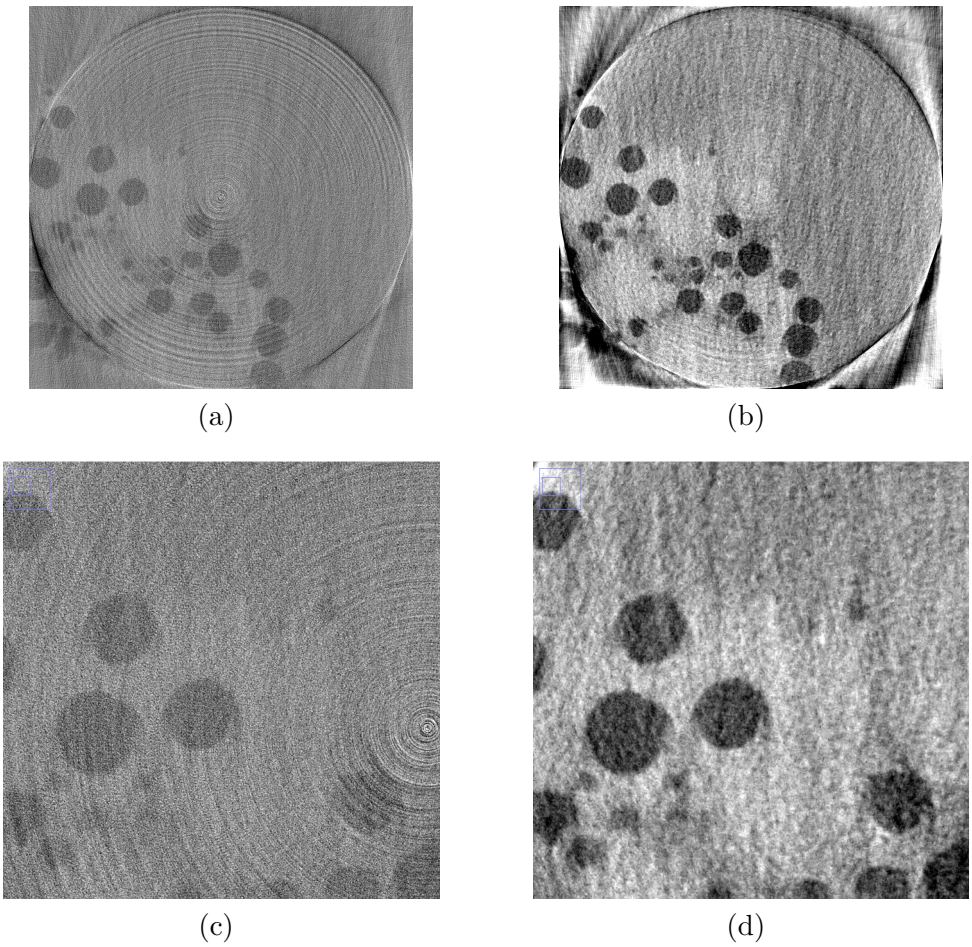


Figure 3.2.17: Reconstructions of a noisy sinogram of the 512×512 "Brain" phantom from 40 views.
Top-left: FBP. Top-Right: Haar wavelets DWT regularization. The image bears downsampling ("blocky regions") and shift variance ("salt and pepper noise") artefacts. Bottom-left: Haar wavelets DWT regularization with cycle spinning. The shift variance artefacts are mostly gone, but the aliasing remains. Bottom-right: Haar wavelets SWT regularization. Both shift variance and aliasing artefacts are gone.

[10] they are not equivalent since $\boldsymbol{W}$ is no more an orthonormal transform

Reconstruction of real datasets

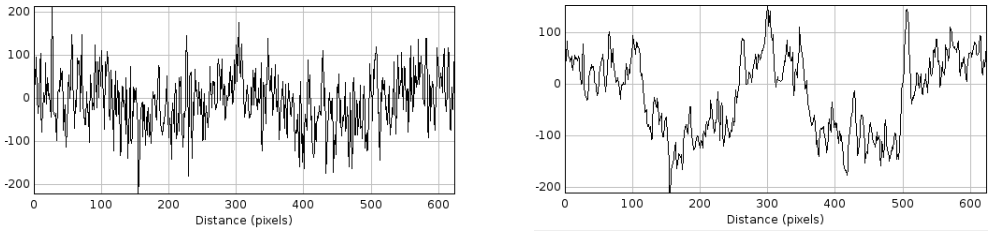
The reconstruction performances are assessed on real datasets acquired at ESRF tomography beamlines. Figures 3.2.18 and 3.2.19 shows how regularized reconstruction can enhance the contrast and thus facilitate the segmentation. In this context, the dataset has few projections (250 views for 2048×2048 pixels), with a low SNR.



(a) (b)

(c) (d)

Figure 3.2.18: Reconstruction results for a PCT micro-scan of a drosophila (data courtesy: Alexandra Pacureanu, ID16-a), 2048×2048 pixels, from 250 views.
(a) FBP, (b) Wavelets (DWT) reconstruction after a rings correction method (see 3.2.6). The Daubechies 2 ("db2") wavelet was chosen to enforce some smoothness.
(c, d): close-up of (a, b), respectively.



(a) (b)

Figure 3.2.19: Line profiles corresponding to Figure 3.2.18. (a) FBP, (b) Wavelets reconstruction after a rings correction method (see 3.2.6). The Daubechies 2 ("db2") wavelet was chosen to enforce some smoothness.

Figure 3.2.20 shows reconstructions of an in-vivo alveoli scan in a rabbit lungs in the context of the study of the Acute Respiratory Distress Syndrome. In this setting, the scan is cumbersome because the heart and respiratory movements can induce motion artefacts, thus, a fast scan has to be performed. With wavelets regularization, reducing the number of projections is possible without sacrificing the reconstruction quality allowing for faster scans. The Wavelets regularization somehow smooths the reconstruction, but a less smooth reconstruction can be obtained with a "sharp" wavelet family like Haar.

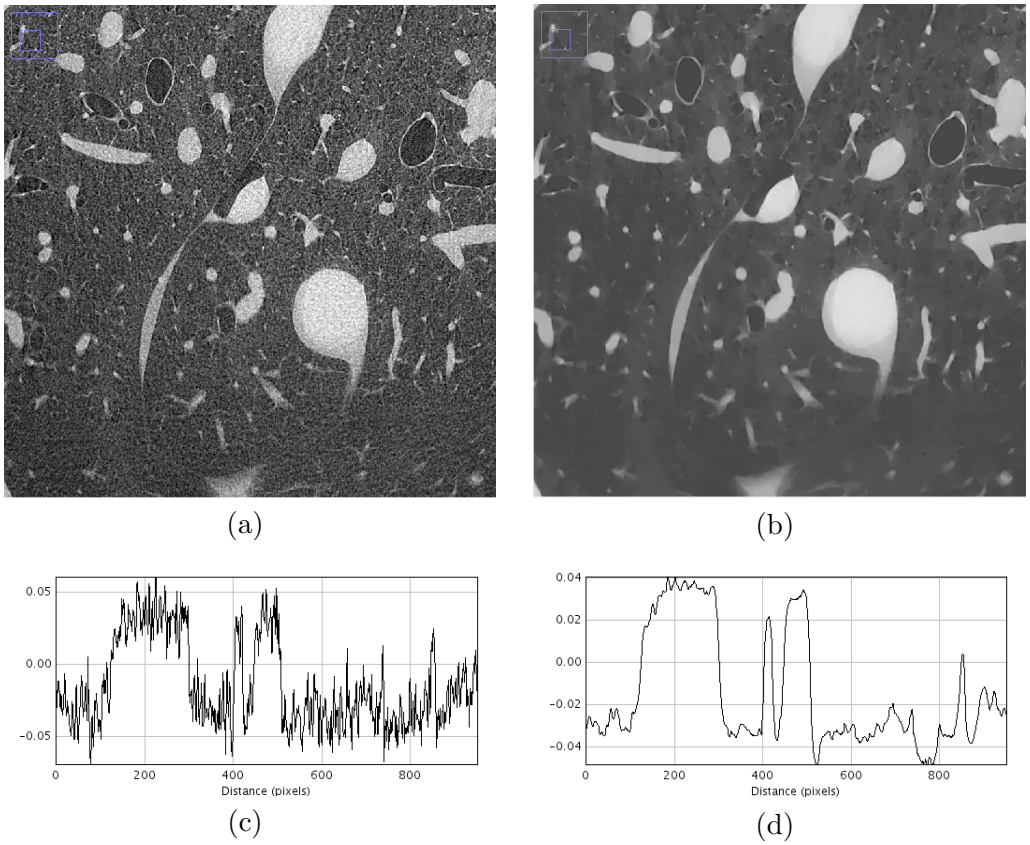


Figure 3.2.20: Reconstruction results for a PCT scan of rabbit lungs (data courtesy: Ludovic Broche, ID17), 2048×2048, from 150 views.
(a): FBP, (b): wavelets (SWT/Coifman 1) reconstruction.
(c, d): line profiles corresponding to (a, b), respectively.

Figure 3.2.21 shows the reconstruction of the phase counterpart of Figures 3.1.7.As the Edge Illumination method yields a differential sinogram (the refraction angle is proportional to the spatial gradient of the X-ray transform of the refractive index δ), the usual reconstruction algorithms for absorption data cannot be used. Instead, the inverse problem is reformulated to include the differential nature of the data[11]

$$\underset{\boldsymbol{x}}{\operatorname{argmin}} \left\{ \frac{1}{2} \left\| \nabla \boldsymbol{P}\boldsymbol{x} - \boldsymbol{d} \right\|_2^2 + \lambda \left\| \boldsymbol{W}\boldsymbol{x} \right\|_1 \right\} \tag{3.2.17}$$

[11] this idea was proposed by Daniël Pelt at a workshop, and is formalized for example in [Hah+15]

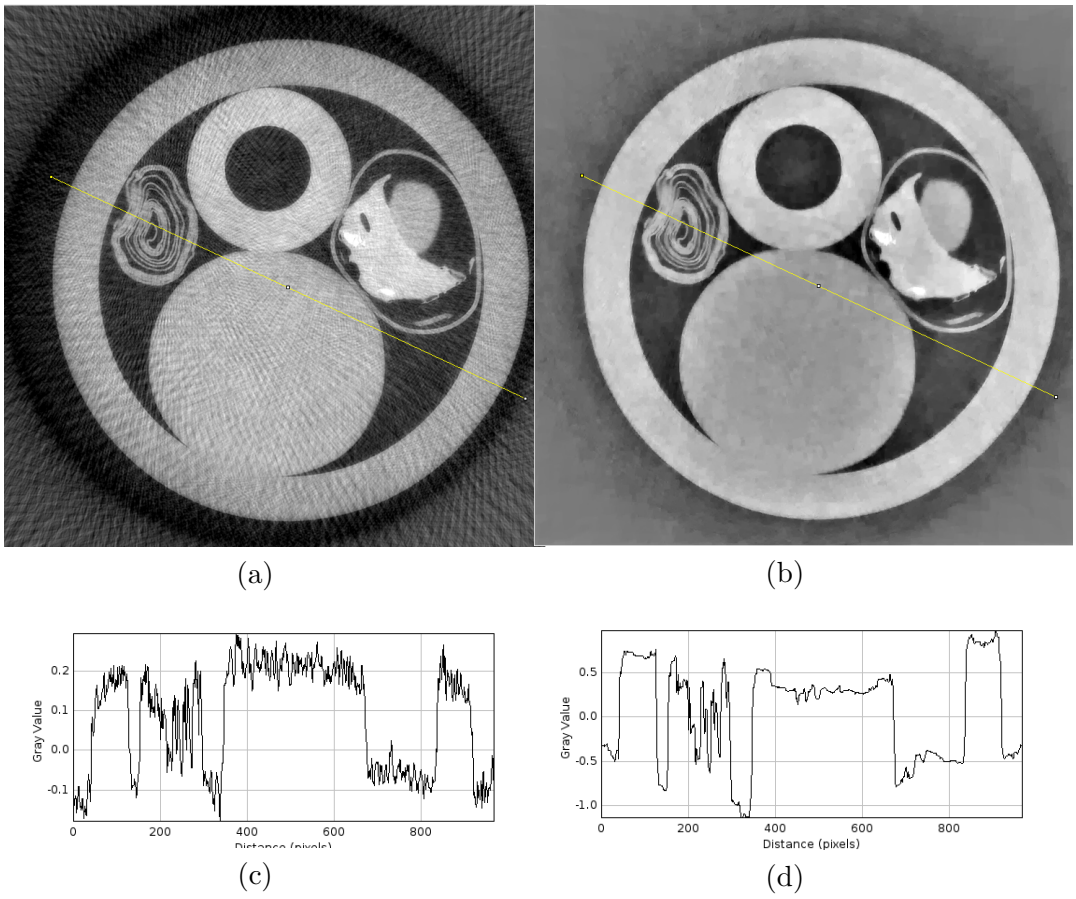


Figure 3.2.21: Reconstructed slice of a real imaging phantom (projection data courtesy: Charlotte Hagen, UCL), with subsampled data. (a): Hilbert transform and backprojection, (b): Iterative reconstruction with the Haar wavelets regularization, using the gradient in the forward model. (c, d): line profiles corresponding to (a, b), respectively.

Reconstruction timings

This part gives reconstruction times for various data sizes. Table 3.4 gives some reconstruction times for various slice size in the case of undersampled data. The convergence is relatively fast, and can benefit from the same trick as in 3.1.4. The synthetic dataset and the timing methods are the same as for Table 3.2, so the execution times can be compared. It appears that the wavelets reconstruction is indeed competitive with TV-C-P in terms of speed. The MSE is also similar, although TV appears to converge faster to a better solution, which is not surprising as the phantom is a piecewise-constant image.

Width	Views	Iterations	MSE (DWT)	MSE (SWT)	Time (DWT)	Time (SWT)
512	40	250	5.14e+01	4.72e+01	0.623 s	0.778 s
1024	80	100	8.92e+00	5.22e+01	0.620 s	0.951 s
2048	160	50	1.97e+00	1.12e+01	0.922 s	2.01 s
4096	320	50	5.59e-01	2.55e-01	5.60 s	10.4 s

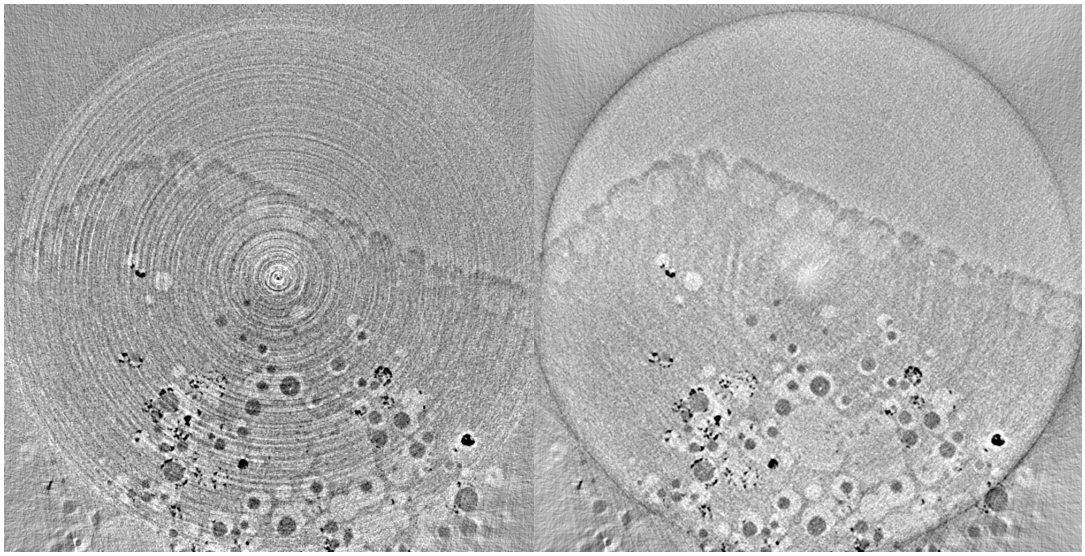
Table 3.4: Iterative reconstruction times for DWT (with cycle spinning) and SWT as a function of the data size.

3.2.6 Application to sinogram filtering

Rings artefacts come from a measurement problem giving rise to spurious vertical lines in the sinogram. This issue is discussed more in depth in chapter 4.1. In this part, we briefly show how the implemented wavelets library can be used for efficient rings artefacts removal. More specifically, the work [Mün+09]. describes a rings artefacts removal method based on singoram filtering, before reconstructing the slice. It consists in detecting and removing vertical lines in the sinogram by the means of the discrete wavelet transform. To do so, the vertical coefficients are filtered at each scale by a high-pass (complementary) Gaussian filter in the vertical direction. Indeed, as rings artefacts are modelled as vertical lines, the sinogram at the corresponding locations bears essentially low frequencies. A Python implementation of this method can be found in the Tomopy project [Gür+14].

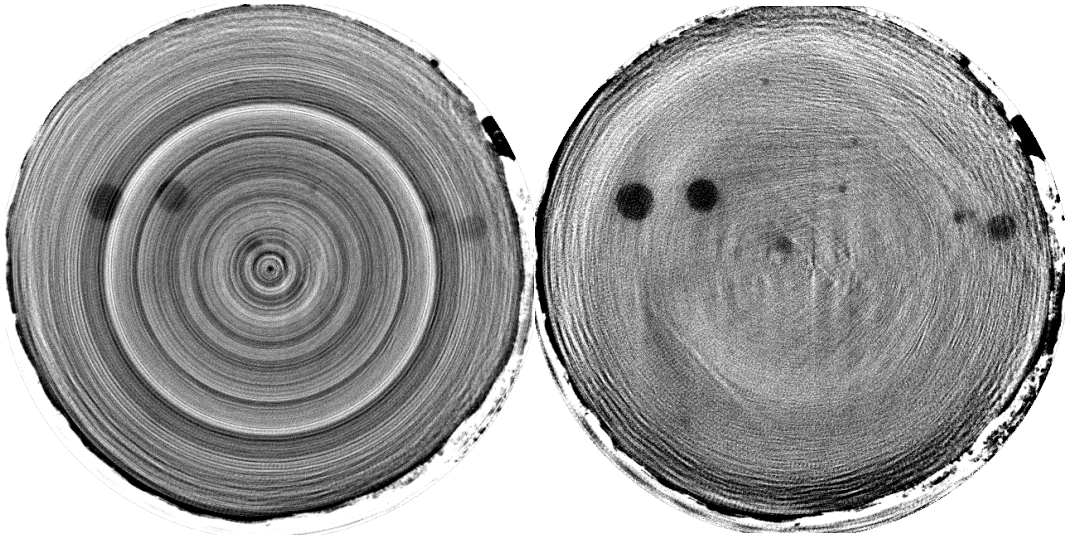
Having a general-purpose GPU DWT is very helpful, as all the other operations (fourier transform, filtering) already have available GPU implementations. Moreover, practical use of the method [Mün+09] suggests that that some wavelets are more efficient to isolate the artefacts depending on the context; thus, it is valuable to have a GPU transform where the wavelet can be tuned.

Figures 3.2.22 and 3.2.23 show the rings artefacts removal capability of the Fourier-Wavelets method. In some cases, post-processing of the data like segmentation is not possible without a rings artefacts removal procedure. Using this sinogram filtering method can give back the possibility of post processing.



(a) (b)

Figure 3.2.22: Rings artefacts correction on a PCT micro-scan of a drosophila (data courtesy: Alexandra Pacureanu, ID16-a).
(a): FBP, (b): Münch Et Al rings correction followed by a SIRT reconstruction



(a) (b)

Figure 3.2.23: (data courtesy: Julie Villanova, ID16-b) (a): FBP, (b): Münch Et Al rings correction followed by FBP

3.3 A conjugate subgradient for ℓ_2-ℓ_1 optimization

In section 3.1.1, we reviewed state-of-the-art convex optimization algorithms in the context of TV tomographic reconstruction, which is an *analysis* formulation. Dictionary-based reconstruction, on the other hand, corresponds to a *synthesis* formulation. From the previous considerations, it appears that FISTA is the algorithm of choice for the synthesis problem, as the gradient of the fidelity term only involves evaluations of the operators (and their adjoints), and the prox of $\boldsymbol{w} \mapsto \lambda \|\boldsymbol{w}\|_1$ is straightforward. However, as observed in practice, the convergence is relatively slow, especially with over-complete dictionaries. This section describes a new efficient optimization algorithm tailored for the "ℓ_2-ℓ_1 problem", or LASSO problem, aiming at solving

$$\underset{\boldsymbol{w}}{\operatorname{argmin}} \left\{ \frac{1}{2} \|\boldsymbol{A}\boldsymbol{w} - \boldsymbol{b}\|_2^2 + \lambda \|\boldsymbol{w}\|_1 \right\} \tag{3.3.1}$$

In our setting, problem (3.3.1) corresponds to the synthesis formulation of a regularized tomographic reconstruction problem, but the applications of this algorithm are broader.

This section presents a new algorithm dedicated to problem (3.3.1). This work was published in [MP17].

3.3.1 Introduction

The dictionary-based tomographic reconstruction problem is

$$\underset{\boldsymbol{w}}{\operatorname{argmin}} \left\{ \frac{1}{2} \left\|\boldsymbol{P}\boldsymbol{D}^T\boldsymbol{w} - \boldsymbol{d}\right\|_2^2 + \lambda \|\boldsymbol{w}\|_1 \right\} \tag{3.3.2}$$

where $\boldsymbol{d}$ is the acquired data (sinogram), $\boldsymbol{P}$ the projection operator, $\boldsymbol{D}^T$ is a dictionary (a redundant frame, see 2.5.2), and $\boldsymbol{w}$ is the vector of coefficients of the underlying slice/volume in the dictionary domain. The non-differentiable ℓ_1 term $\|\boldsymbol{w}\|_1$ precludes from using fast methods like the conjugate gradient.

Many state-of-the-art non-smooth convex optimization algorithms can be unified in the framework of proximal algorithms (see 2.6.2) which roughly replace a *gradient step* with a *proximal step*. However, the ℓ_1 term $\|\boldsymbol{w}\|_1$ is very simple, as the variables in $\boldsymbol{w}$ are not coupled by an operator inside the norm. Its subgradient is straightforward:

$$\left(\partial \|\cdot\|_1 (\boldsymbol{x})\right)_i = \begin{cases} -1 & x_i < 0 \\ [-1,1] & x_i = 0 \\ 1 & x_i > 0 \end{cases} \tag{3.3.3}$$

As the regularization term is simple, Problem (3.3.2) deviates only "slightly" from a quadratic minimization, for which efficient algorithms are available. This motivated the extension of the CG algorithm to ℓ_2-ℓ_1 problems like (3.3.2), where a simple[12] ℓ_1 term is added to the ℓ_2 term. The resulting algorithm is called Conjugate Subgradient (CSG) algorithm, as it replaces the gradient a (differentiable) objective function with a subgradient if a (non-differentiable) ℓ_2-ℓ_1 objective function.

In the next section, after a brief recall of the conjugate gradient algorithm, we derive the CSG. We then show how this new algorithm is adapted to ill-conditioned ℓ_2-ℓ_1 problems.

3.3.2 The nonlinear conjugate gradient algorithm

In this section, we settle the notations by recalling the standard conjugate gradient algorithm. The derivation of the conjugate gradient algorithm from Krylov subspaces is out of the scope of this section; instead, we give the principle and pseudo-code of CG assuming that the underlying concepts are known.

Let $\boldsymbol{x}$ denote the (vector) variable of the function F. For the remainder of this section, the function to minimize is $F(\boldsymbol{x}) = f(\boldsymbol{x}) + g(\boldsymbol{x})$ with $f(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2$ and $g(\boldsymbol{x}) = \beta\|\boldsymbol{x}\|_1$, so the optimization problem is[13]

$$\underset{\boldsymbol{x}}{\operatorname{argmin}} \left\{ F(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 + \beta\|\boldsymbol{x}\|_1 \right\} \tag{3.3.4}$$

The CG algorithm builds a set of conjugate directions $(\boldsymbol{p}_k)_{k=1\ldots n}$ where n is the number of iterations. Once the conjugate direction $\boldsymbol{p}_k$ is calculated at iteration k, the variable is updated with $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k$. The scalar α_k is the step size at iteration k, computed with a line search. The gradient of F is then evaluated in $\boldsymbol{x}_{k+1}$ to compute the next conjugate direction $\boldsymbol{p}_{k+1}$. The computation of $\boldsymbol{p}_{k+1}$ actually only depends on the previous direction, which makes the conjugate gradient algorithm practically usable. For a differentiable function F, the standard conjugate gradient is given in its numerical form by Algorithm 3.3.1.

[12] no variables coupling by an operator

[13] in the context of tomographic reconstruction, the variable was denoted $\boldsymbol{w}$ as it corresponds to coefficients in a frame. In this general context, no assumption are made, hence the different variables names $\boldsymbol{A}$, $\boldsymbol{x}$ and $\boldsymbol{b}$

Algorithm 3.3.1 Conjugate gradient

F: differentiable function
n: number of iterations

procedure CONJGRAD(F, n)
  Compute an initial guess x_0
  $\boldsymbol{g}_0 = -\nabla F(\boldsymbol{x}_0)$ ▷ Steepest direction at iteration 0
  $\boldsymbol{p}_0 = \boldsymbol{g}_0$
  for $k \leftarrow 0, n$ **do**
    $\alpha_k = \underset{\alpha}{\operatorname{argmin}} \{F(x_k + \alpha \boldsymbol{p}_k)\}$ ▷ Line search
    $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k$ ▷ Update variable
    $\boldsymbol{g}_{k+1} = -\nabla F(\boldsymbol{x}_{k+1})$ ▷ Update Steepest direction
    $\beta_k = \dfrac{\boldsymbol{g}_{k+1}^T(\boldsymbol{g}_{k+1} - \boldsymbol{g}_k)}{\boldsymbol{g}_k^T \boldsymbol{g}_k}$ ▷ Update β, for ex. with the Polak-Ribiere rule
    $\boldsymbol{p}_{k+1} = \boldsymbol{g}_{k+1} + \beta_k \boldsymbol{p}_k$ ▷ New conjugate direction
  end for
  return $\boldsymbol{x}_n$
end procedure

3.3.3 From conjugate gradient to conjugate subgradient

In the basic subgradient method

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \gamma_k \boldsymbol{p}_k \qquad \boldsymbol{p}_k \in \partial F(\boldsymbol{x}_k) \tag{3.3.5}$$

the direction $\boldsymbol{p}_k$ is any subgradient $\partial F(\boldsymbol{x}_k)$, which is a drawback of this method since there is no indication of which subgradient should be chosen. As a result, the conjugate subgradient is not a descent method: the objective function can increase during the optimization process [SM03].

To build an algorithm based on the conjugate gradient, one has to define an unique descent direction at each iteration, which means choosing between all the possible subgradients ∂F when F is not differentiable. The basic idea is to rely on the quadratic part ∇f of the gradient. Once the gradient of the smooth part $\nabla f(x)$ is calculated, the subgradient of the L1 part g is evaluated with :

$$\partial g(\boldsymbol{x}) = \begin{cases} \operatorname{sign}(\boldsymbol{x}) & \text{if } \boldsymbol{x} \neq 0 \\ \operatorname{sign}(\nabla f(\boldsymbol{x})) & \text{if } \boldsymbol{x} = 0 \end{cases} \tag{3.3.6}$$

where the sign function is applied componentwise. With this particular choice of subgradient (3.3.6), the subderivative of $F = f + g$ is always single-valued. The motivation of such a choice is that when the variable $\boldsymbol{x}$ comes near the singularity of $g = \|\cdot\|_1$, every direction (subgradient) is possible; the ambiguity is removed by relying on the derivative of the quadratic term f.

When using the CG method, using a preconditioner can dramatically improve the convergence rate. In our method, the preconditioner relies on the magnitude of the quadratic part of the gradient ∇f.

From the variables $\boldsymbol{x}_{k+1}$, $\boldsymbol{p}_k$, $\boldsymbol{q}_k$ (see Algorithm 3.3.1), three new preconditioned variables $\overline{\boldsymbol{x}}_{k+1}$, $\overline{\boldsymbol{p}}_{k+1}$, $\overline{\boldsymbol{q}}_{k+1}$ are built with the following preconditioner :

$$\begin{cases} \boldsymbol{D} = \begin{cases} 1 & \text{if } |\nabla f(\boldsymbol{M}_k \odot \boldsymbol{x}_{k+1})| < \beta \text{ and } \boldsymbol{x}_k \cdot \boldsymbol{x}_{k+1} < 0 \\ 0 & \text{otherwise} \end{cases} \\ \boldsymbol{M}_{k+1} = \min\left(\boldsymbol{M}_k \cdot (1 - \gamma \boldsymbol{D} + \delta(1 - \boldsymbol{D})),\, 1\right) \\ \boldsymbol{S}_{k+1} = \begin{cases} 0 & \text{if } |\nabla f(\boldsymbol{M}_k \odot \boldsymbol{x}_{k+1})| < \beta \text{ and } |\boldsymbol{x}| < \varepsilon \\ 1 & \text{otherwise} \end{cases} \\ \boldsymbol{V}_{k+1} = \dfrac{\boldsymbol{M}_{k+1}}{\boldsymbol{M}_k} \end{cases} \tag{3.3.7}$$

$$\begin{cases} \overline{\boldsymbol{x}}_{k+1} = \dfrac{\boldsymbol{x}_{k+1}}{\boldsymbol{V}_{k+1}} \cdot \boldsymbol{S}_{k+1} \\ \overline{\boldsymbol{p}}_{k+1} = \boldsymbol{p}_k \cdot \boldsymbol{V}_{k+1}^a \cdot \boldsymbol{S}_{k+1} \\ \overline{\boldsymbol{q}}_{k+1} = \boldsymbol{q}_k \cdot \boldsymbol{V}_{k+1} \cdot \boldsymbol{S}_{k+1} \end{cases} \tag{3.3.8}$$

where all the vector-vector operations are componentwise, and the symbol $\odot$ denotes a componentwise (diagonal) matrix-vector multiplication. For example, the matrix $\boldsymbol{M}_k$ is a matrix, and $\boldsymbol{M}_k \odot \boldsymbol{x}_{k+1}$ denotes the multiplication of all the components of $\boldsymbol{x}$ with the diagonal of $\boldsymbol{M}_k$.

The rationale of this preconditioner can be summarized as follows:

- When the gradient magnitude of the quadratic part ∇f is important, the components of the variables are updated as in the conjugate gradient method – without variable substitution – since the quadratic part is predominant over the non-smooth part.

- When $|\nabla f|$ is small, the standard conjugate gradient method would be disturbed by frequent crossings of regions where the gradient of g is discontinuous. The used rule is that the preconditioning factors are increasingly shrunk by a factor $\gamma < 1$ as long as they should be updated. The criterion is to check if the previous preconditioned variable ($\overline{\boldsymbol{x}}_k$) and the variable updated after the line search ($\boldsymbol{x}_{k+1}$) have an opposite sign. This variable substitution is implemented by the coefficient vector (or diagonal matrix) $\boldsymbol{M}_k$. We note that, although our problem is not constrained, there is some similarity between our idea of choosing descent directions that try to avoid gradient discontinuities, and the Conditional Gradient Descent method [Bub14] which chooses the descent direction which maximizes the decrease of the linearized objective function within the domain borders.

- The exponent a, used in the determination of the vector $\overline{\boldsymbol{p}}_{k+1}$ is a tunable number. The vector $\overline{\boldsymbol{p}}_{k+1}$ is used in the composition of the $\boldsymbol{p}_{k+1}$ descent direction (see Algorithm 3.3.1). By using a number $a > -1$ we tend to avoid constructing descent directions which bring us too fast to non-smooth regions. Keeping $a = -1$ corresponds to using the previous descend direction as in standard conjugate gradient method.

- Another rule is that during this phase (small quadratic gradient), the components which are "small enough" (below a threshold ϵ) are set – and will remain as long as the force on them is weak– to zero. This rule is especially important for the convergence toward solutions with high sparsity. This rule is implemented by the matrix $\boldsymbol{S}_k$.

The conjugate subgradient algorithm for LASSO optimization is given by Algorithm 3.3.2

Algorithm 3.3.2 Conjugate subgradient

F: function to optimize, $F(\boldsymbol{x}) = f(\boldsymbol{x}) + g(\boldsymbol{x})$ with f the quadratic part and g the L1 part
γ, δ, ϵ: parameters for update the preconditioner (see (3.3.7))
n: number of iterations

procedure CONJSUBGRAD(F, (γ, δ, ϵ), n)
  Compute an initial guess $\overline{x}_0$
  $\boldsymbol{g}_0 = -\nabla F(\boldsymbol{x}_0)$ ▷ Steepest direction at iteration 0
  $\boldsymbol{p}_0 = \boldsymbol{g}_0$
  $\boldsymbol{M}_0 = 1$ ▷ Element-wise
  for $k \leftarrow 0, n$ **do**
    $\boldsymbol{q}_k = \boldsymbol{M}_k \odot \boldsymbol{A}^T \boldsymbol{A}(\boldsymbol{M}_k \odot \boldsymbol{p}_k)$
    Compute $\alpha_k = \underset{\alpha}{\operatorname{argmin}} \{F(\boldsymbol{M}_k \odot (\overline{\boldsymbol{x}}_k + \alpha \boldsymbol{p}_k))\}$
    $\boldsymbol{x}_{k+1} = \overline{\boldsymbol{x}}_k + \alpha_k \boldsymbol{p}_k$
    Update preconditioners ($\boldsymbol{M}_{k+1}$, $\boldsymbol{S}_{k+1}$, $\boldsymbol{V}_{k+1}$) using (3.3.7)
    Update ($\overline{\boldsymbol{x}}_{k+1}$, $\overline{\boldsymbol{p}}_{k+1}$, $\overline{\boldsymbol{q}}_{k+1}$) using (3.3.8)
    $\boldsymbol{g}_{k+1} = -\nabla F(\overline{\boldsymbol{x}}_{k+1} \odot \boldsymbol{M}_{k+1}) \odot \boldsymbol{S}_{k+1} \odot \boldsymbol{M}_{k+1}$
    $\beta = -\dfrac{\overline{\boldsymbol{q}}_{k+1}^T \boldsymbol{g}_{k+1}}{\overline{\boldsymbol{q}}_{k+1}^T \overline{\boldsymbol{p}}_{k+1}}$
    $\boldsymbol{p}_{k+1} = \boldsymbol{g}_{k+1} + \beta \overline{\boldsymbol{p}}_{k+1}$
  end for
  return x_n
end procedure

where the multiplications with $\boldsymbol{A}$ and $\boldsymbol{A}^T$ are "regular" matrix-vector multiplications.

3.3.4 Line search

The line search is a crucial step of gradient methods. The variables are updated with the previously computed conjugate direction $\boldsymbol{p}_k$. The step α_k in this direction should be such as

$$\alpha_k = \underset{\alpha}{\operatorname{argmin}} \{F(\boldsymbol{M}_{k+1} \odot \boldsymbol{x}_{k+1})\} \qquad \text{where } \boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha \boldsymbol{p}_k \tag{3.3.9}$$

The computation of (3.3.9) can be done "blindly" with a generic line search, but here one can benefit from both the quadratic nature of f and the convex property of g. We discuss how to do it in this session, discarding for conciseness, and without loss of generality, the preconditioner $\boldsymbol{M}$.

Regarding the quadratic part f, it is easily shown that

$$f(\boldsymbol{x}_k + \alpha \boldsymbol{p}_k) = \frac{1}{2} \|\boldsymbol{A}(\boldsymbol{x}_k + \alpha \boldsymbol{p}_k) - \boldsymbol{b}\|_2^2 = a_2 \alpha^2 + a_1 \alpha + a_0 \tag{3.3.10}$$

$$\text{where } a_2 = \frac{1}{2} \boldsymbol{p}_k^T \boldsymbol{A}^T \boldsymbol{A} \boldsymbol{p}_k, \;\; \boldsymbol{a}_1 = \boldsymbol{p}_k^T \boldsymbol{A}^T (\boldsymbol{A}\boldsymbol{x}_k - \boldsymbol{b}), \;\; \boldsymbol{a}_0 = \frac{1}{2} \|\boldsymbol{A}\boldsymbol{x}_k - \boldsymbol{b}\|_2^2$$

The coefficients a_2 and a_1 can be computed once for all before the line search; actually, they are also used elsewhere in the algorithm so they have to be computed anyway. The

evaluation of $\frac{\mathrm{d}f}{\mathrm{d}\alpha}$, the derivative of f with respect to the scalar α, only requires these two coefficients, and thus has virtually no cost.

An interesting property of smooth quadratic function $f(\boldsymbol{x}) = \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2$ is

$$\nabla f(\boldsymbol{x}_{k+1}) = \nabla f(\boldsymbol{x}_k) + \alpha_k \boldsymbol{A}^T \boldsymbol{A}\boldsymbol{p}_k \tag{3.3.11}$$

The vector $\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{p}_k$ is already known from the computation of $\boldsymbol{p}_k^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{p}_k$. Hence the update of the gradient $\nabla f(\boldsymbol{x}_{k+1})$ from the previous gradient $\nabla f(\boldsymbol{x}_k)$ is cheap.

Therefore, for a smooth quadratic function, the line search is straightforward:

$$\begin{aligned} 0 = \frac{\mathrm{d}f}{\mathrm{d}\alpha}(\boldsymbol{x}_{k+1}) &= \nabla f(\boldsymbol{x}_{k+1})^T \cdot \frac{\mathrm{d}}{\mathrm{d}\alpha}\boldsymbol{x}_{k+1} \\ &= \boldsymbol{p}_k^T \left(\nabla f(\boldsymbol{x}_k) + \alpha \boldsymbol{A}^T\boldsymbol{A}\boldsymbol{p}_k\right) \qquad \text{using (3.3.11)} \end{aligned}$$

which gives

$$\alpha_k = \frac{-\boldsymbol{p}_k^T \nabla f(\boldsymbol{x}_k)}{\boldsymbol{p}_k^T \boldsymbol{A}^T \boldsymbol{A}\boldsymbol{p}_k} \tag{3.3.12}$$

Now, getting back to the whole function $F = f + g$, a one-step line search like (3.3.12) is not possible since one cannot extract α from $\partial g(\boldsymbol{x}_{k+1})$. However, due to the convexity of g, an upper bound of α_k can be computed using the following property :

Proposition 3.3.1

For all k, we have $\boldsymbol{p}_k^T \partial g(\boldsymbol{x}_{k+1}) \geq \boldsymbol{p}_k^T \partial g(\boldsymbol{x}_k)$.

Proof. In this proof, The superscript denotes a component of a vector. Since g is convex, every component ∂g^i of its subgradient is increasing. Thus, we have $\partial g(x_{k+1})^i \geq \partial g(x_k)^i$ if and only if $x_{k+1}^i \geq x_k^i$, i.e $p_k^i \geq 0$ (since $\alpha_k \geq 0$). Thus :

- If $p_k^i \geq 0$, then $x_{k+1}^i = x_k^i + \alpha_k p_k^i \geq x_k^i$, so $\partial g(x_{k+1})^i \geq \partial g(x_k)^i$, so $p_k^i \cdot \partial g(x_{k+1}) \geq p_k^i \cdot \partial g(x_k)$.
- Similarly, if $p_k^i \leq 0$, then $\partial g(x_{k+1})^i \leq \partial g(x_k)^i$ so $p_k^i \cdot \partial g(x_{k+1})^i \geq p_k^i \cdot \partial g(x_k)^i$.

Doing the scalar product, we have in any case $\boldsymbol{p}_k^T \partial g(\boldsymbol{x}_{k+1}) \geq \boldsymbol{p}_k^T \partial g(\boldsymbol{x}_k)$ □

Using this property, we can derive the same calculation as for (3.3.12) :

$$\begin{aligned} 0 = \frac{\mathrm{d}F}{\mathrm{d}\alpha}(\boldsymbol{x}_{k+1}) &= \frac{\mathrm{d}f}{\mathrm{d}\alpha}(\boldsymbol{x}_{k+1}) + \frac{\mathrm{d}g}{\mathrm{d}\alpha}(\boldsymbol{x}_{k+1}) \\ &= \boldsymbol{p}_k^T \nabla f(\boldsymbol{x}_k) + \alpha \boldsymbol{p}_k^T \boldsymbol{A}^T \boldsymbol{A}\boldsymbol{p}_k + \boldsymbol{p}_k^T \partial g(\boldsymbol{x}_{k+1}) \\ &\geq \boldsymbol{p}_k^T \left(\nabla f(\boldsymbol{x}_k) + \partial g(\boldsymbol{x}_k)\right) + \alpha \boldsymbol{p}_k^T \boldsymbol{A}^T \boldsymbol{A}\boldsymbol{p}_k \end{aligned} \tag{3.3.13}$$

Thus

$$\alpha_k \leq \alpha_k^u = \frac{-\boldsymbol{p}_k^T \partial F(\boldsymbol{x}_k)}{\boldsymbol{p}_k^T \boldsymbol{A}^T \boldsymbol{A}\boldsymbol{p}_k} \tag{3.3.14}$$

For the last inequality in (3.3.13), property 3.3.1 has been applied. The upper bound α_k^u is convenient for a line search using the bisection method. For example, the line search can be done using the *regula falsi* method at the beginning when the differentiable L2 part is predominant, and then the bisection method when the L1 part becomes more important.

3.3.5 Applications

In this section, numerical examples are provided to compare the convergence of this new method with FISTA and ADMM which are state-of-art convex non-smooth optimization methods.

Example on ill-conditioned matrix

This example illustrates the convergence rate of the conjugate subgradient algorithm for problem (3.3.4), where the matrix $\boldsymbol{A}$ is built to be ill-conditioned. The code to compute this example can be found at [MP]. In this example $\boldsymbol{A}$ is a $500 \times 1k$ matrix and $\boldsymbol{A}^T\boldsymbol{A}$ is a $1k \times 1k$ symmetric matrix, with a large condition number. The eigenvalues of $\boldsymbol{A}^T\boldsymbol{A}$ are plotted on Figure 3.3.1.

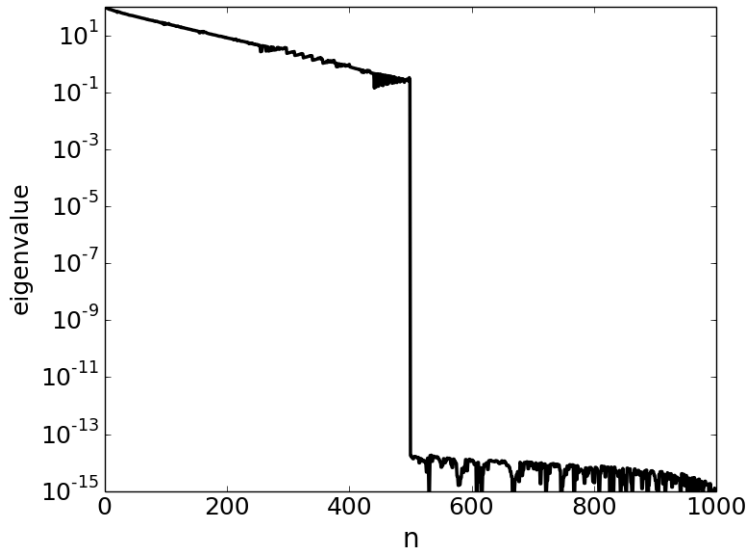


Figure 3.3.1: Logarithmic plot of the eigenvalues of the matrix $\boldsymbol{A}^T\boldsymbol{A}$

The regularization parameter, in the LASSO objective function, was $\beta = 0.1$. The CSG algorithm was run with the parameters $\gamma = 0.85$ and $\delta = 0.04$, and the exponent for direction $\overline{p}$ was $a = 1$. The FISTA algorithm was applied in its restarted variant: we reset the acceleration parameter each time the step would increase the objective function. The best scaling parameter of ADMM was found to be $\rho = L/350$ where L is the maximum eigenvalue of $\boldsymbol{A}^T\boldsymbol{A}$.

Figure 3.3.2 shows the objective function values $F(\boldsymbol{x}) - F(\boldsymbol{x}_\infty)$ for 2000 iterations for the three methods: CSG, FISTA and ADMM.

It can be seen that CSG achieves the solution in about 800 iterations, while FISTA needs much more iterations to converge. Also, the objective function values are always smaller for CSG.

We see that our CSG method is competitive also compared to ADMM. Although ADMM converges in half the number of steps required by CSG, one must consider that, for large systems, the matrix $(\boldsymbol{A}^T\boldsymbol{A} + \rho\boldsymbol{I})^{-1}$ cannot always be computed. This is particularly true in tomography where, $\boldsymbol{A}$ and $\boldsymbol{A}^T$ (projection and back-projection operators) are sparse, but $(\boldsymbol{A}^T\boldsymbol{A})^{-1}$ is dense. One should, in such case, calculate the action of $(\boldsymbol{A}^T\boldsymbol{A} + \rho\boldsymbol{I})^{-1}$ by an iterative method like the conjugate gradient, which has a high cost per iteration.

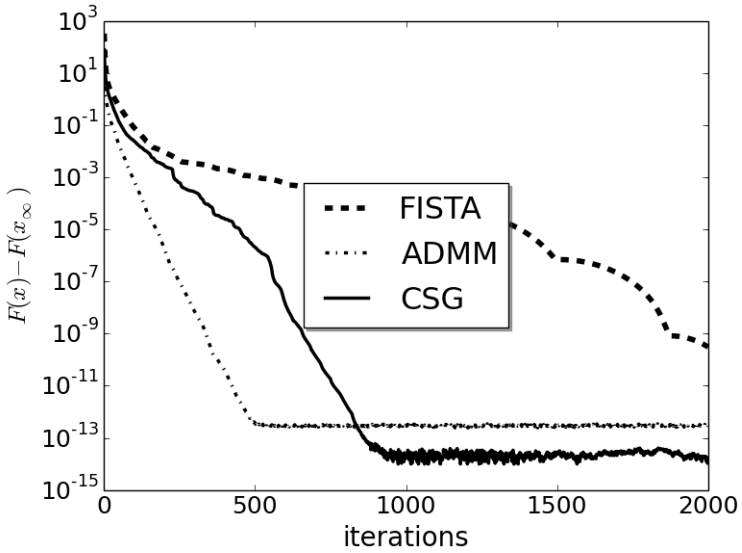


Figure 3.3.2: Logarithmic plot of objective function values for CSG, FISTA, ADMM algorithms

Tomographic reconstruction with the dictionary regularization and ring-artifacts correction

We now examine how the CSG algorithm performs for iterative tomographic reconstruction with dictionary regularization and rings artefacts correction. The rings artefacts correction is incorporated in the objective function (3.3.15), more details are given in 4.1.

$$\underset{\boldsymbol{w},\boldsymbol{r}}{\operatorname{argmin}} \left\{ F(\boldsymbol{w}) = \left\| \boldsymbol{P}\boldsymbol{D}^T\boldsymbol{w} + \boldsymbol{U}\boldsymbol{r} - \boldsymbol{d} \right\|_2^2 + \beta \left\| \boldsymbol{w} \right\|_1 + \beta_r \left\| \boldsymbol{r} \right\|_1 \right\} \tag{3.3.15}$$

where a ring vector $\boldsymbol{r}$ is added to each projection line of the sinogram – the rings artifacts are modeled, in the sinogram, as constant values along the projection angle axis. The sinogram has dimensionality (N_p, N) where N_p is the number of projections and N is the number of pixels in one dimension of the slice. The operator $\boldsymbol{U}$ repeats the ring variables on each line of the sinogram. It was observed that the additional variable $\boldsymbol{r}$ slows down the convergence rate for standard algorithms like FISTA, which notably motivated the design of CSG.

In this example, the standard 512×512 test image *Lena* was used, and 80 projections were used to generate the sinogram. Additionally, rings artifacts were simulated by adding lines in the sinogram.

The functional (3.3.15) was minimized with two techniques implemented in the PyHST2 code [Mir+14]: FISTA algorithm and the conjugate subgradient algorithm (CSG). In this test, an over-complete dictionary has been used, resulting in an ill-conditioned problem which is a difficult setting for optimization algorithms. Moreover we observed that, for this kind of problem, the "energy transfer" from the reconstructed image to the auxiliary variables capturing the spurious artifacts ($\boldsymbol{r}$) occurs in the final part of the convergence and is slow with FISTA. With CSG, the best convergence properties were obtained with $a = 0$.

Figure 3.3.3 shows the plot of the normalized objective function $F(\boldsymbol{w}) - F(\boldsymbol{w}_\infty)$ for 8000 iterations. Both methods converge to the same final value since the same functional $F(\boldsymbol{w})$ is minimized, but the last stage of the optimization process is much faster for

the conjugate subgradient algorithm. Figure 3.3.4 shows the reconstructed images with Filtered Back Projection and the Dictionary Learning technique, for parameters $\beta = 0.7$ and $\beta_r = 10$. It can be noted that the rings artifacts are almost entirely removed.

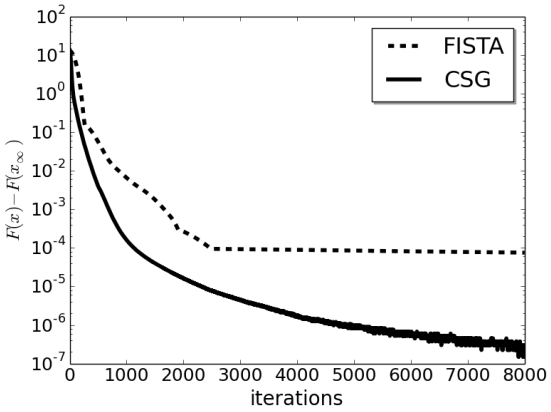
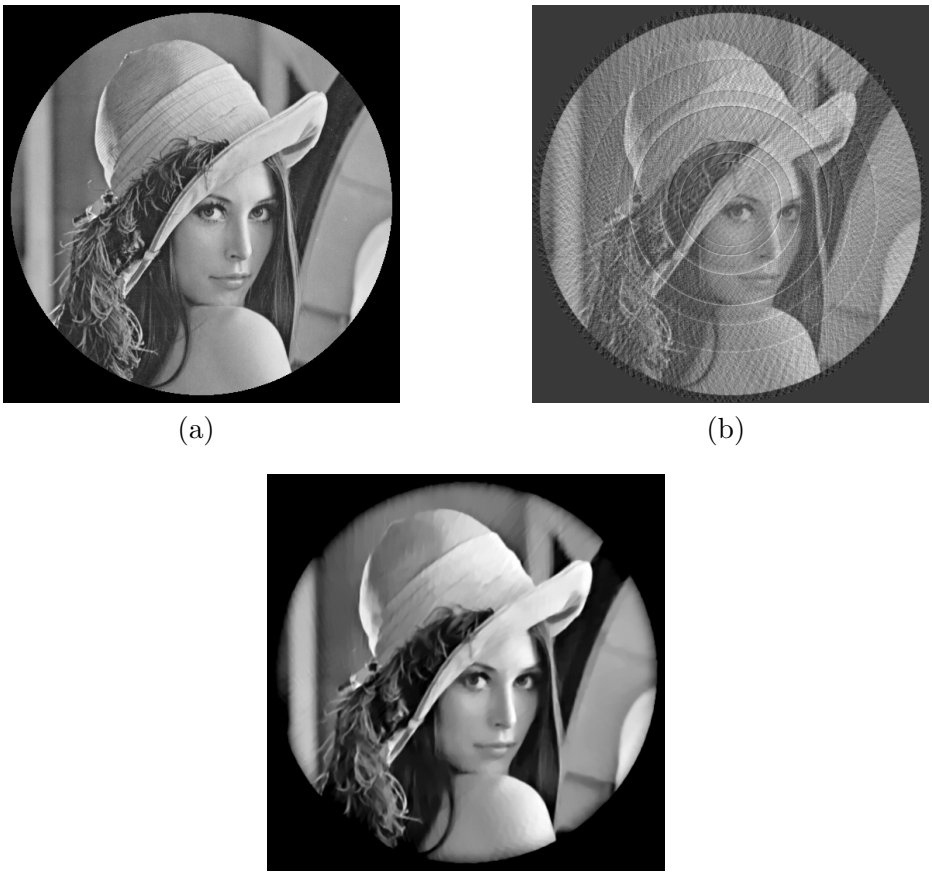


Figure 3.3.3: Logarithmic plot of the values of the objective function for both methods.



(a) (b)

(c)

Figure 3.3.4: Phantom of Lena reconstructed with 80 projection angles. Lines were added to the sinogram to simulate ring artifacts. (a) Phantom of Lena (b) FBP (c) DL

The CSG algorithm is therefore successfully adapted for the iterative rings artefacts correction which will be developed in section 4.1.

3.4 Filter-based proximal computation for ADMM

In parallel geometry, the operator $\mathcal{R}^*\mathcal{R}$ is shift invariant (see section 1.4.3). As a consequence, the least squares inversion can be computed efficiently, leading to a "filtering" process instead of a whole iterative process. A comprehensive work exploiting this property can be found in [Pel16]. In this section, we use the approach of this work to efficiently compute the proximity operator of $\boldsymbol{x} \mapsto \frac{\gamma}{2}\|\boldsymbol{P}\boldsymbol{x} - \boldsymbol{d}\|_2^2$, which is given by $\left(\boldsymbol{I} + \gamma \boldsymbol{P}^T\boldsymbol{P}\right)^{-1}\left(\boldsymbol{x} + \gamma \boldsymbol{P}^T\boldsymbol{d}\right)$. This prox is required in algorithms like ADMM in the context of TV regularization, and computing it usually requires another iterative solving. In this case, we show that similarly to deconvolution, the previous prox can be computed efficiently, leading to an interesting speed-up of the ADMM algorithm.

3.4.1 Least-squares inversion of shift-invariant operators

Let $\boldsymbol{A}$ be a shift invariant linear operator – more details on these operators can be found in Appendix 6.3.2. There exists a vector $\boldsymbol{h}_{\boldsymbol{A}}$, hereby simply $\boldsymbol{h}$, such that evaluating $\boldsymbol{A}$ on a vector $\boldsymbol{x}$ can be computed as $\boldsymbol{h} \odot \boldsymbol{x}$, or even more efficiently as $\boldsymbol{F}^*\left((\boldsymbol{F}\boldsymbol{x}) \odot (\boldsymbol{F}\boldsymbol{h})\right)$ where $\boldsymbol{F}$ is the unitary DFT[14] and $\odot$ is the vector-vector elementwise multiplication operation. For example, $\boldsymbol{A}$ can be the model of a spatial blur; in this case, $\boldsymbol{h}$ is called the point spread function. Given an observed blurry image $\boldsymbol{b}$, the least-squares deblurring is a LIP instance

$$\underset{\boldsymbol{x}}{\operatorname{argmin}} \left\{ \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 \right\} \tag{3.4.1}$$

whose (maximum likelihood) solution is given applying the pseudo-inverse $\hat{\boldsymbol{x}} = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{b}$. In general, the pseudo-inverse $(\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T$ cannot be computed or even stored, as for a N pixels image $\boldsymbol{x}$, this operator has N^2 components. In this case, however, $\boldsymbol{A}$ can be represented by a simple vector $\boldsymbol{h}$. It can then be shown (see Appendix 6.3.3) that the pseudo-inverse can be computed as

$$(\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T = \boldsymbol{F}^*\boldsymbol{D}_{\boldsymbol{A}}^{-1}\boldsymbol{F} \tag{3.4.2}$$

where $\boldsymbol{D}_{\boldsymbol{A}}$ is the diagonal matrix containing the FT of $\boldsymbol{h}$ on the diagonal. Applying the pseudoinverse (3.4.2) on a vector $\boldsymbol{x}$ boils down to "dividing by the PSF in the Fourier domain".

This approach has of course numerical issues, which is why regularized pseudoinverse are used instead. An example of regularized pseudoinverse is $(\boldsymbol{I} + \gamma\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T$, which is the solution mapping of

$$\frac{\gamma}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 + \frac{1}{2}\|\boldsymbol{x}\|_2^2 \tag{3.4.3}$$

It can be shown (see Appendix 6.3.3) that the inversion can be computed as

$$(\boldsymbol{I} + \gamma\boldsymbol{A}^T\boldsymbol{A})^{-1} = \boldsymbol{F}^*\boldsymbol{H}^{-1}\boldsymbol{F} \tag{3.4.4}$$

where $\boldsymbol{H} = \boldsymbol{I} + \gamma|\boldsymbol{D}_{\boldsymbol{A}}|^2$ is the diagonal matrix where the diagonal component k equals $1 + \gamma|(\boldsymbol{F}\boldsymbol{h})(k)|^2$. There again, the whole inversion can be *efficiently computed* and *stored* as an operator (meaning that it is computed once and can be re-applied on any vector), since it is characterized by a vector $\boldsymbol{h}$ (the impulse response).

[14] in practice, computing the convolution through FT is not beneficial when the kernel $\boldsymbol{h}$ has a few non-zero components

3.4.2 Least-squares inversion in tomography

In the continuous case, it was shown in 1.4.3 that the Radon transform $\mathcal{R}$ satisfies

$$\begin{aligned} \mathcal{R}^*\mathcal{R} &= \Lambda_2^{-1} \\ \mathcal{R}\mathcal{R}^* &= \Lambda_1^{-1} \end{aligned} \tag{3.4.5}$$

where Λ_2 and Λ_1 are the two-dimensional and one-dimensional Calderón's operator (i.e elemenwise multiplication with $\nu \mapsto |\nu|$ in the Fourier domain), respectively[15]. In this context, the FBP can be seen as a least-squares inversion described in section 3.4.1, where $\boldsymbol{A} = \boldsymbol{P}$ is the discretization of the Radon transform. More importantly, Equation (3.4.5) shows that the Radon transform is a shift-invariant operator, so its least-squares inverse can be efficiently computed – at least in the continuous case.

In the computed tomographic reconstruction setting, the operator $\boldsymbol{P}$ is a discretization of $\mathcal{R}$. The operator $\boldsymbol{P}^T\boldsymbol{P}$ is only approximatively translation invariant in pararallel geometry. Therefore, the "transfer function" is not the inverse Calderón's operator Λ_2 anymore, but the multiplicative inverse of the FT of $\boldsymbol{P}^T\boldsymbol{P}\boldsymbol{\delta}_2$ (where $\boldsymbol{\delta}_2$ is a discrete 2D Dirac/Kronecker function) which notably depends on the number of projections[16]. Still, the matrix $\boldsymbol{P}^T\boldsymbol{P}$ has an almost-circulant structure, which seems promising for an efficient least-squares inversion. In the following, we investigate two approaches.

Direct least-squares inversion

When the data has few projection angles, the FBP does not provide a satisfactory reconstruction: there is a large mismatch between the continuous case $(\mathcal{R}^*\mathcal{R})^{-1} = \Lambda_2$ (infinitely many angles) and the discrete case $(\boldsymbol{P}^T\boldsymbol{P})^{-1}$ (scarse angles). In this part, we test the "divide by the transfer function" method described in Appendix 6.3.3.

More precisely, we consider $(\boldsymbol{P}\boldsymbol{P}^T)^{-1}$ instead of $(\boldsymbol{P}^T\boldsymbol{P})^{-1}$ in order to follow a "filter-then-backproject" $(\boldsymbol{P}^T(\boldsymbol{P}\boldsymbol{P}^T)^{-1})$ approach rather than a "backproject-then-filter" $((\boldsymbol{P}^T\boldsymbol{P})^{-1}\boldsymbol{P}^T)$ approach (the equivalence in the continuous case is given by Equation (1.4.13)). Indeed, the backprojection operation should theoretically be carried on an infinite image support; practical truncation to the reconstructed image support entails strong convolution artefacts at the edges. Using the "filter-then-backproject" approach avoids this numerical issue.

The goal is to compute the *operator* $(\boldsymbol{P}\boldsymbol{P}^T)^{-1}$ in the discrete setting. The transfer function of $\boldsymbol{P}\boldsymbol{P}^T$ is angle-dependent in contrast with the continuous case. To compute it, we generate a sinogram where each line (angle) is a 1D Kronecker function $\boldsymbol{\delta}_1$. This sinogram is backprojected and forward projected to get the impulse response[17] denoted $\boldsymbol{h}$. The DFT of $\boldsymbol{h}$ is then computed: $\boldsymbol{h}_{\boldsymbol{F}} = \boldsymbol{F}_1\boldsymbol{h}$, giving the transfer function[18]. The "least-squares filter" is the elementwise inverse of this transfer function. Figure 3.4.1 shows examples of inverted transfer functions.

[15] one-dimensional Calderón's operator means that the filtering is applied on only one variable of the sinogram, as in the FBP

[16] if the projections are uniformly distributed in $[0, \pi]$

[17] the convolution in the sinogram domain consists in 1D convolutions with each line

[18] The Fourier transform $\boldsymbol{F}_1$ in the sinogram domain consists in computing the 1D FT of each line

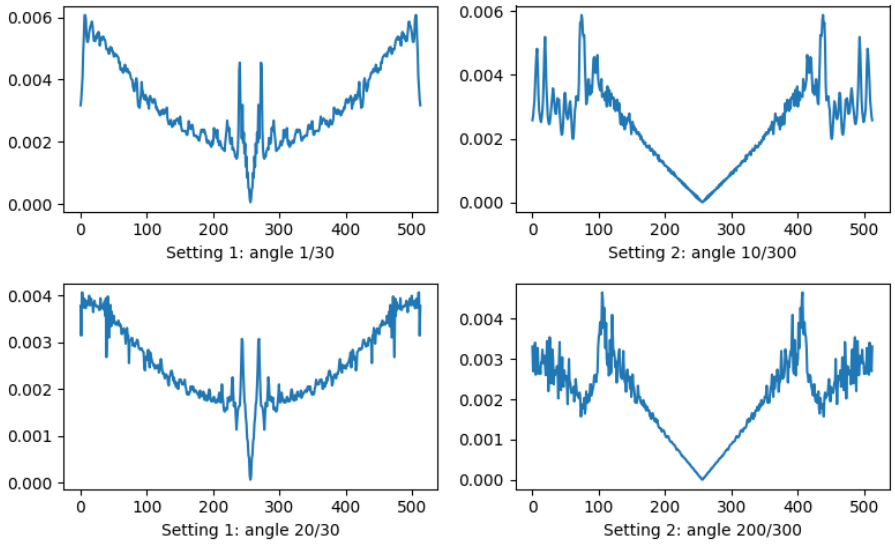


Figure 3.4.1: Angle-dependent filter $1/|\boldsymbol{h_F}|$ computed with two settings: **Setting 1** is a projector on 512×512 grid with 300 angles, **Setting 2** is a projector on 512×512 grid with 30 angles. The four quadrants show a line of $1/|\boldsymbol{h_F}|$ at two different angles for the two settings. The horizontal axes are the numerical frequencies, where the zero frequency is in the middle. With more projections, the filter gets closer to the Calderón's filter $\nu \mapsto |\nu|$.

Given a sinogram $\boldsymbol{d}$, the least-squares reconstruction is then simply computed as

$$\boldsymbol{x}_{\mathrm{ML}} = \boldsymbol{P}^T \boldsymbol{F}_1^*(\boldsymbol{F}_1\boldsymbol{d} \odot \boldsymbol{h_F}^{-1}) \tag{3.4.6}$$

where $\boldsymbol{h_F}^{-1}$ is the elementwise inverse of the sinogram-like vector $\boldsymbol{h_F}$. Figure 3.4.2 shows reconstructions of noisy sinograms using this method. Although it yields results similar to an iterative least squares reconstruction (SIRT/CG), it cannot prevent "noise amplification". This noise amplification can be avoided when using iterative reconstructions by setting a small number of iterations, giving a trade-off between a blurry solution (few iterations) and a noisy solution (many iterations). This approach does not take into account a number of iterations, so the noise amplification cannot be avoided.

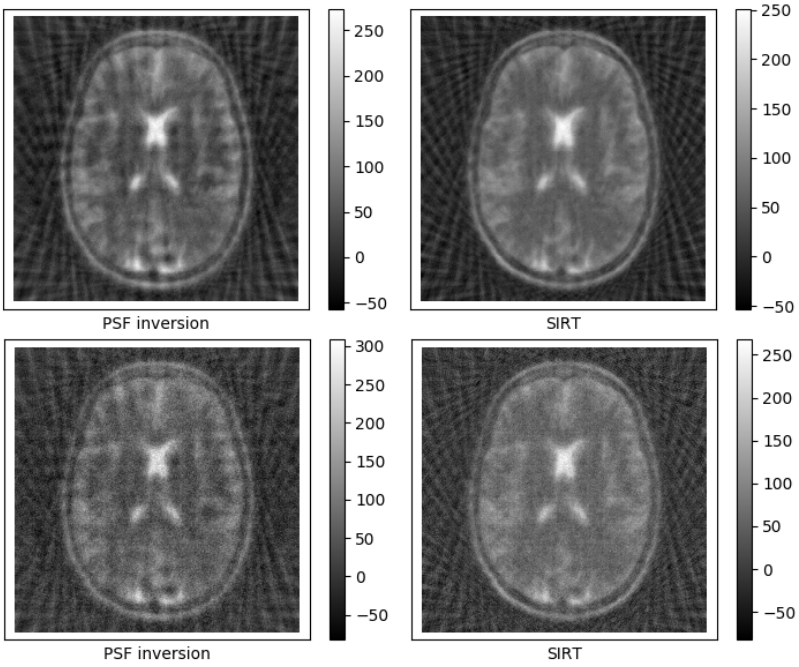


Figure 3.4.2: Reconstruction of the noisy 512×512 MRI phantom from 30 projection angles. The noise std is 1% and 3% of the noiseless sinogram maximum value for the top row and bottom row, respectively.

Iterative least-squares inversion

Instead of dividing by the transfer function in order to perform the least squares inversion, one might also use an approximated version of $(\boldsymbol{P}^T\boldsymbol{P})^{-1}$. Indeed, iterative algorithms minimizing $f(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{P}\boldsymbol{x} - \boldsymbol{d}\|_2^2$ converge to $(\boldsymbol{P}^T\boldsymbol{P})^{-1}\boldsymbol{P}^T\boldsymbol{d}$ (see Appendix 6.3.4). As the latter quantity is ill-defined, iterations are stopped early to numerical issues in the solution. Therefore, the impulse response $\boldsymbol{h}$ of the previous section becomes $\boldsymbol{h}_n$, an impulse response depending on the number of iterations n.

An iterative least-squares inversion in the context of iterative tomographic reconstruction is proposed in the work [PB15]. It starts from the standard Landweber iteration for minimizing f:

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{x}_k - \alpha\nabla f(\boldsymbol{x}_k) \\ &= \boldsymbol{x}_k - \alpha\boldsymbol{P}^T(\boldsymbol{P}\boldsymbol{x}_k - \boldsymbol{d}) \\ &= (\boldsymbol{I} - \alpha\boldsymbol{P}^T\boldsymbol{P})\boldsymbol{x}_k + \alpha\boldsymbol{P}^T\boldsymbol{d} \end{aligned} \tag{3.4.7}$$

where $\alpha > 0$ is the gradient step. Equation (3.4.7) defines an arithmetico–geometric sequence. If $\boldsymbol{x}_0 = \boldsymbol{0}$, the iterate $\boldsymbol{N}$ is straightforwardly given by

$$\begin{aligned} \boldsymbol{x}_n &= \left[\sum_{n=0}^{n-1}(\boldsymbol{I} - \alpha\boldsymbol{P}^T\boldsymbol{P})^k\right](\alpha\boldsymbol{P}^T\boldsymbol{d}) \\ &= \left[\alpha\sum_{n=0}^{n-1}(\boldsymbol{I} - \alpha\boldsymbol{P}^T\boldsymbol{P})^k\right]\boldsymbol{P}^T\boldsymbol{d} \end{aligned} \tag{3.4.8}$$

which is a backproject-then-filter approach, the "filter" depending both on the projector $\boldsymbol{P}$ (number of projections) and the number of iterations n. As the matrix $\boldsymbol{P}^T\boldsymbol{P}$ is almost shift-invariant, it has an "almost-circulant" structure, which is also the case for $\boldsymbol{I} - \alpha\boldsymbol{P}^T\boldsymbol{P}$,

and all the iterates $\sum_k (\boldsymbol{I}-\alpha \boldsymbol{P}^T\boldsymbol{P})^k$. Thus, for a given number of iterations n, the operator $\boldsymbol{G} = \left[\alpha \sum_{n=0}^{n-1}(\boldsymbol{I}-\alpha \boldsymbol{P}^T\boldsymbol{P})^k\right]$ is a "filter" and can be characterized by a single vector, its impulse response. In other words, n iterations of the SIRT/least squares reconstruction can be pre-computed as a filter [PB15].

To implement this "least-squares filter", n iterations (3.4.7) (without the term $\alpha \boldsymbol{P}^T\boldsymbol{d}$) are applied on a 2D Kronecker function $\boldsymbol{\delta}_2$, yielding an impulse response $\boldsymbol{h}_n$ which can be used as a filter in the image domain. However, for the same reason as described previously[19], a "filter-then-backproject" approach is followed based on Equation (1.4.2): the impulse response in the image domain $\boldsymbol{h}_n$ is forward projected to obtain an impulse response in the sinogram domain $\boldsymbol{P}\boldsymbol{h}_n$. A one dimensional convolution is performed at each angle (sinogram lines), and the result is backprojected.

This "sirt-filter" method is more numerically stable than the previous one, as it accounts for a number of iterations, thus preventing the instability of the inversion process. Successful applications are reported in [Pel16], which is a promising basis for the fast computation of the regularized pseudoinverse $(\boldsymbol{I}+\gamma \boldsymbol{P}^T\boldsymbol{P})^{-1}$.

3.4.3 Fast filter-based proximal computation

The aim of this section is to build a filter approximating the operator

$$\left(\boldsymbol{I}+\gamma \boldsymbol{P}^T\boldsymbol{P}\right)^{-1}\left(\boldsymbol{x}+\gamma \boldsymbol{P}^T\boldsymbol{d}\right) \tag{3.4.9}$$

In the non-iterative case ("dividing with the psf in the Fourier domain"), the inverse is given by Equation 3.4.4 where $\boldsymbol{A}=\boldsymbol{P}$. In the iterative case, the filter after n iterations is computed with the same fashion as for Equation 3.4.8. It can be shown (Appendix 6.3.4) that the appropriate iterate is

$$\boldsymbol{x}_n = \alpha \sum_{k=0}^{n-1}((1-\alpha)\boldsymbol{I}-\alpha\gamma \boldsymbol{P}^T\boldsymbol{P})^k \tag{3.4.10}$$

where α is the gradient descent step which should satisfy $\alpha < 1/(1+\gamma \left\|\boldsymbol{P}^T\boldsymbol{P}\right\|)$.

For the reason given previously, the operator $(\boldsymbol{I}+\gamma \boldsymbol{P}\boldsymbol{P}^T)^{-1}$ is computed instead. However, the resulting filter should be applied on an image in contrast to the "SIRT-filter" which is applied on a sinogram. In the latter case, the filter approximating $(\boldsymbol{P}^T\boldsymbol{P})^{-1}$ could be projected to be convolved (line-wise) with the sinogram to reconstruct (consequence of Equation (1.4.2)). In this case, the proximal given by Equation (3.4.9) has to be applied on an image $\boldsymbol{x}$, so we cannot apply the "filter-then backproject rather than backproject-then-filter" principle. Fortunately, we can use the Woodburry matrix identity which gives in our case

$$\begin{aligned}\left(\boldsymbol{I}+\gamma \boldsymbol{P}^T\boldsymbol{P}\right)^{-1} &= \boldsymbol{I}-\boldsymbol{P}^T\left(\gamma^{-1}+\boldsymbol{P}\boldsymbol{P}^T\right)^{-1}\boldsymbol{P} \\ &= \boldsymbol{I}-\gamma \boldsymbol{P}^T\left(\boldsymbol{I}+\gamma \boldsymbol{P}\boldsymbol{P}^T\right)^{-1}\boldsymbol{P}\end{aligned} \tag{3.4.11}$$

where the identity matrices domains differ depending on the context. Thus, we can still compute $\left(\boldsymbol{I}+\gamma \boldsymbol{P}\boldsymbol{P}^T\right)^{-1}$ instead of $\left(\boldsymbol{I}+\gamma \boldsymbol{P}^T\boldsymbol{P}\right)^{-1}$ even if it has to be applied on an image.

The reference proximal computation is done with the CG algorithm (see Section 3.1.1). The "prox-filter" $\boldsymbol{h}_n$ is computed with n (several hundreds) iterations of the gradient

[19] the operator $\mathcal{R}^*$ theoretically backprojects the images on an infinite grid, which is not the case in the discrete setting, so convolution border effect occur

descent

$$\begin{cases} \tilde{\boldsymbol{h}}_{k+1} = (1-\alpha)\tilde{\boldsymbol{h}}_k - \alpha\gamma \boldsymbol{P}\boldsymbol{P}^T\tilde{\boldsymbol{h}}_k \\ \boldsymbol{h}_{k+1} = \boldsymbol{h}_k + \tilde{\boldsymbol{h}}_{k+1} \end{cases} \tag{3.4.12}$$

where $\tilde{\boldsymbol{h}}_0 = \boldsymbol{\delta}_1$ is a Dirac in the sinogram domain. The proximal is then applied on images with Equation (3.4.11) where $\left(\boldsymbol{I} + \gamma \boldsymbol{P}\boldsymbol{P}^T\right)^{-1}$ is replaced with a (series of 1D) convolution with $\boldsymbol{h}_n$.

Figure 3.4.3 shows an example of result of proximal applied on an image with the two methods (CG and prox-filter). When applying the prox $\left(\boldsymbol{I} + \gamma \boldsymbol{P}^T\boldsymbol{P}\right)^{-1}(\boldsymbol{x} + \gamma \boldsymbol{P}^T\boldsymbol{d})$, the sinogram $\boldsymbol{d}$ is the sinogram of the 256×256 MRI phantom with 40 projection angles, and the image $\boldsymbol{x}$ is the "ascent" image from the scipy Python module – this image was chosen so that the "mixing" effect between sinogram and image can be seen. These figures can be computed with the Jupyter notebook [Pal17]. From these figure, the "prox-filter" provides very good approximation of the true proximal.

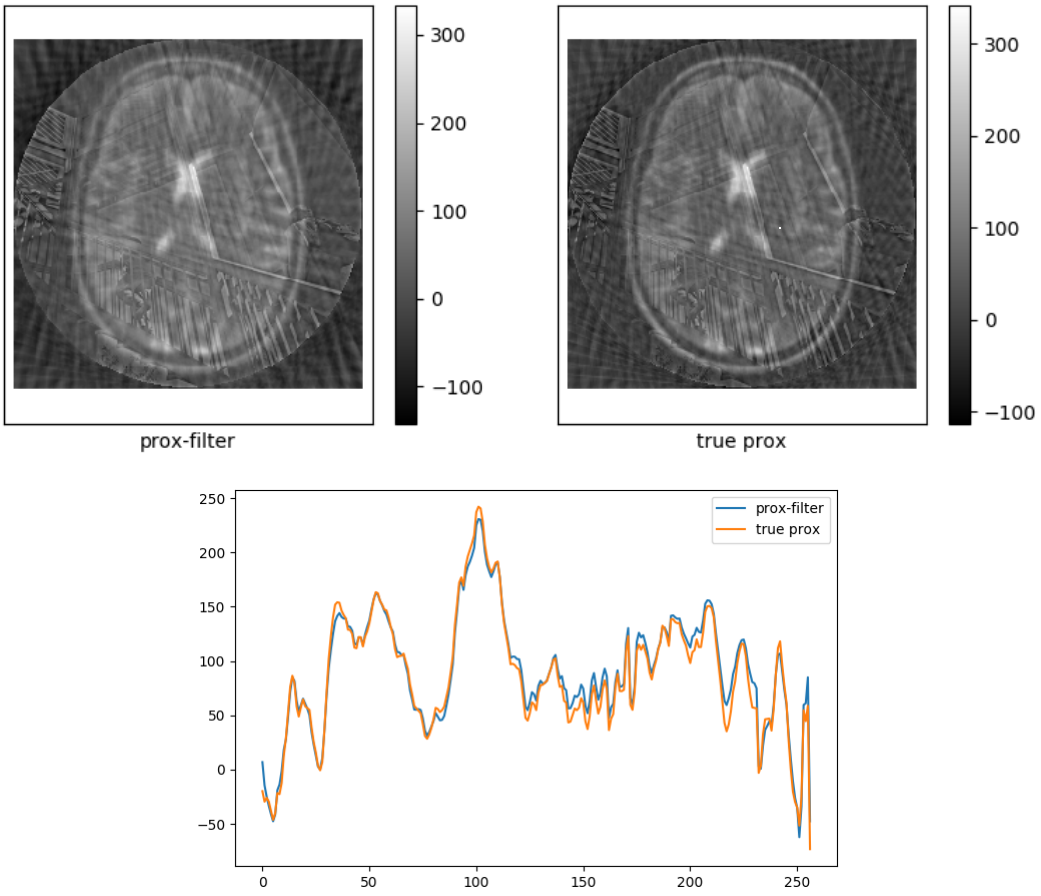


Figure 3.4.3: Comparison of the results with two methods for computing the proximal: conjugate gradient and "prox-filter". Top row: comparison of image results with both methods. Bottom row: middle line profile of the images: prox-filter (blue) and true prox (orange).

The prox-filter is then "plugged" in the ADMM algorithm used in section 3.1.1 (instead of CG), making the overall process run more than twice faster. Figure 3.4.4 compares the reconstruction results of the 256×256 MRI phantom with 40 projection angles, with two

version of ADMM. One is using CG to compute the prox (hereby "ADMM-CG"), and the other is using the proposed prox-filter to compute the prox (hereby "ADMM-prox-filter"). Although the "ADMM-prox-filter" reconstruction is not as good as the ADMM-CG, the result is still promising, and benefits from the "compressed sensing property" (there is no subsampling artefacts).

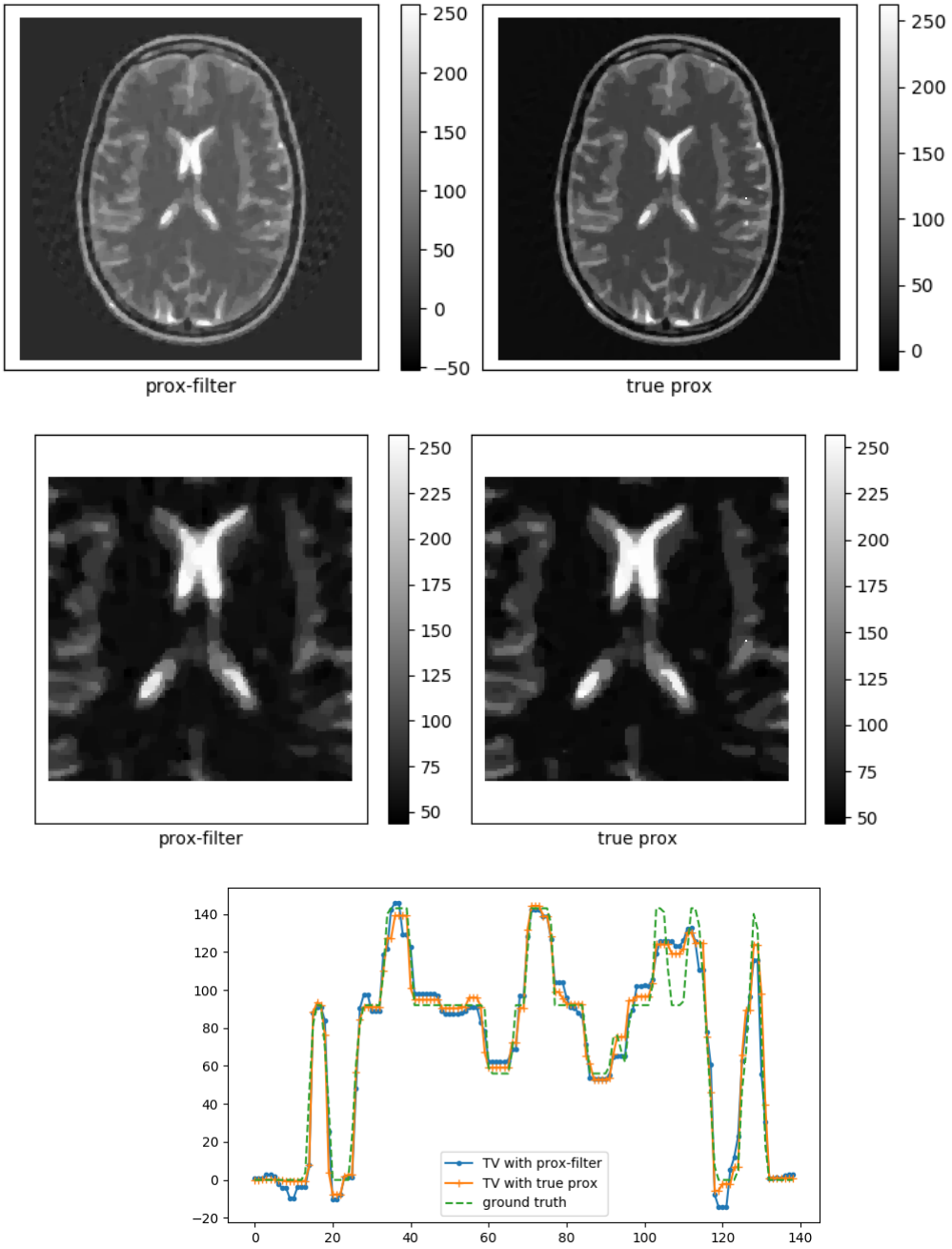


Figure 3.4.4: Reconstruction results with ADMM using "prox-filter" (left column) and the CG algorithm (right column) respectively.

In the numerical experiments, we nevertheless found a limitation to this approach: it appears that when computing the prox-filter through Equations (3.4.11) and (3.4.12), a "zero mask" had to be applied to the resulting image. This mask sets to zero all the

pixels outside the circle ζ inscribed in the image support. The ADMM algorithm fails to converge if this mask is not applied. We believe that the reason why the "masking" is necessary is that in the discrete setting, the "valid" support of the (regularized) inversion of $\boldsymbol{P}^T\boldsymbol{P}$ is precisely the circle ζ inscribed in the image support. Indeed, the set of image pixels which have a projection hitting all the angles is ζ; therefore, $\boldsymbol{P}^T\boldsymbol{P}$ is only invertible in this support.

The fast computation of the proximal of $\boldsymbol{x} \mapsto \frac{\gamma}{2} \|\boldsymbol{P}\boldsymbol{x} - \boldsymbol{d}\|_2^2$ with this proposed method enables to use ADMM with a lesser cost per iteration, as tens of iterations of CG are replaced with a series of 1D convolutions. If the constrained version of ADMM is used (Equation (2.7.15) of section 2.7.3), the algorithm gets rid of all its sub-iterations. Another application is the regularization with orthogonal wavelets transform: as the proximal of the regularization term is straightforward, *both* proximal are computed quickly, leading to an algorithm with fast convergence and low cost per iteration.

Conclusion

In this chapter, we presented efficient implementations of iterative reconstruction methods. On the algorithmic side, we analyzed competing algorithms for TV reconstruction (3.1.1), and we fully exploited the property of parallel geometry to speed-up both the convergence rate (3.1.4) and execution time (3.4). We also proposed a new algorithm to speed-up the convergence rate of dictionary reconstruction (3.3). On the computational side, we implemented the relevant operators on GPU for a fast processing (3.1.6 and 3.2)

From the work in this chapter, we can propose the following combination of model/algorithms depending on the setting:

- Sample with few well-defined phases: TV regularization with C-P algorithm
- More complex/biological samples where a good quality reconstruction of a similar dataset is available: dictionary-based reconstruction with CSG algorithm
- Complex samples where no similar datasets are available: wavelets regularization with FISTA or ADMM algorithms.

The combination of state-of-the-art optimization algorithms and high-throughput GPU implementation of the operators provide reconstruction methods that are fast enough to be used in production on beamlines.

Bibliography

[Aar+16] Wim van Aarle et al. "Fast and flexible X-ray tomography using the ASTRA toolbox". In: *Opt. Express* 24.22 (Oct. 2016), pp. 25129–25147. DOI: 10.1364/OE.24.025129. URL: http://www.opticsexpress.org/abstract.cfm?URI=oe-24-22-25129.

[ABF10] Manya V Afonso, José M Bioucas-Dias, and Mário AT Figueiredo. "Fast image recovery using variable splitting and constrained optimization". In: *IEEE Transactions on Image Processing* 19.9 (2010), pp. 2345–2356.

[Bar06] Mauro Barni. *Document and Image Compression (Signal Processing and Communications)*. Ed. by Mauro Barni. CRC Press, Inc., 2006. ISBN: 978-0-8493-3556-3.

[BF07] José M Bioucas-Dias and Mário AT Figueiredo. "A new TwIST: Two-step iterative shrinkage/thresholding algorithms for image restoration". In: *IEEE Transactions on Image processing* 16.12 (2007), pp. 2992–3004.

[BT09] Amir Beck and Marc Teboulle. "A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems". In: *SIAM J. Img. Sci.* 2.1 (Mar. 2009), pp. 183–202. ISSN: 1936-4954. DOI: 10.1137/080716542. URL: http://dx.doi.org/10.1137/080716542.

[Bub14] Sébastien Bubeck. "Theory of convex optimization for machine learning". In: *arXiv preprint arXiv:1405.4980* (2014).

[CD95] Ronald R Coifman and David L Donoho. *Translation-invariant de-noising*. Springer, 1995.

[Cha+09] Lotfi Chaari et al. "Solving inverse problems with overcomplete transforms and convex optimization techniques". In: *SPIE*. San Diego, California, United States, Aug. 2009. URL: https://hal.archives-ouvertes.fr/hal-00826119.

[Cha04a] Antonin Chambolle. "An algorithm for total variation minimization and applications". In: *Journal of Mathematical imaging and vision* 20.1-2 (2004), pp. 89–97.

[Cha04b] Antonin Chambolle. "An algorithm for total variation minimization and applications". In: *Journal of Mathematical imaging and vision* 20.1 (2004), pp. 89–97.

[Cli+93] N. H. Clinthorne et al. "Preconditioning methods for improved convergence rates in iterative reconstructions". In: *IEEE Transactions on Medical Imaging* 12.1 (Mar. 1993), pp. 78–83. ISSN: 0278-0062. DOI: 10.1109/42.222670.

[CP09] P. L. Combettes and J.-C. Pesquet. "Proximal Splitting Methods in Signal Processing". In: *ArXiv e-prints* (Dec. 2009). arXiv: 0912.3522 [math.OC].

[CP11] Antonin Chambolle and Thomas Pock. "A First-Order Primal-Dual Algorithm for Convex Problems with Applications to Imaging". English. In: *Journal of Mathematical Imaging and Vision* 40.1 (2011), pp. 120–145. ISSN: 0924-9907. DOI: 10.1007/s10851-010-0251-1. URL: http://dx.doi.org/10.1007/s10851-010-0251-1.

[CRT06] Emmanuel J Candès, Justin Romberg, and Terence Tao. "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information". In: *IEEE Transactions on information theory* 52.2 (2006), pp. 489–509.

[Dau+92] Ingrid Daubechies et al. *Ten lectures on wavelets*. Vol. 61. SIAM, 1992.

[EMR07] Michael Elad, Peyman Milanfar, and Ron Rubinstein. "Analysis versus synthesis in signal priors". In: *Inverse problems* 23.3 (2007), p. 947.

[Fil06] Gregory R. Lee Filip Wasilewski Ralf Gommers. *PyWavelets*. https://github.com/PyWavelets/py 2006.

[Fra+10] Joaquín Franco et al. "Parallel 3D fast wavelet transform on manycore GPUs and multicore CPUs". In: *Procedia Computer Science* 1.1 (2010), pp. 1101–1110.

[Gua+16] Matthew D Guay et al. "Compressed Sensing Electron Tomography for Determining Biological Structure". In: *Scientific reports* 6 (2016).

[Gue+12] Matthieu Guerquin-Kern et al. "Realistic analytical phantoms for parallel magnetic resonance imaging". In: *IEEE Transactions on Medical Imaging* 31.3 (2012), pp. 626–636.

[Gür+14] Doga Gürsoy et al. "TomoPy: a framework for the analysis of synchrotron tomographic data". In: *Journal of synchrotron radiation* 21.5 (2014), pp. 1188–1193.

[Hah+15] Dieter Hahn et al. "Statistical iterative reconstruction algorithm for X-ray phase-contrast CT". In: *Scientific reports* 5 (2015).

[Jos82] Peter M Joseph. "An improved algorithm for reprojecting rays through pixel images". In: *IEEE transactions on medical imaging* 1.3 (1982), pp. 192–196.

[KF16] Donghwan Kim and Jeffrey A Fessler. "Optimized first-order methods for smooth convex minimization". In: *Mathematical programming* 159.1-2 (2016), pp. 81–107.

[LT94] David S Lalush and Benjamin MW Tsui. "Improving the convergence of iterative filtered backprojection algorithms". In: *Medical physics* 21.8 (1994), pp. 1283–1286.

[Mal89] Stephane G Mallat. "A theory for multiresolution signal decomposition: the wavelet representation". In: *IEEE transactions on pattern analysis and machine intelligence* 11.7 (1989), pp. 674–693.

[Mat09] Jiří Matela. "GPU-Based DWT Acceleration for JPEG2000". In: *MEMICS 2009 Proceedings*. Brno, 2009, pp. 136–143. ISBN: 978-80-87342-04-6.

[Mir+14] Alessandro Mirone et al. "The PyHST2 hybrid distributed code for high speed tomographic reconstruction with iterative reconstruction and a priori knowledge capabilities". In: *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms* 324.0 (2014). 1st International Conference on Tomography of Materials and Structures, pp. 41–48. ISSN: 0168-583X. DOI: http://dx.doi.org/10.1016/j.nimb.2013.09.030. URL: http://www.sciencedirect.com/science/article/pii/S0168583X14000251.

[MM16] Filippo Arcadu and Marco Stampanoni and Federica Marone. "On the crucial impact of the coupling projector-backprojector in iterative tomographic reconstruction". In: *CoRR* abs/1612.05515 (2016). URL: http://arxiv.org/abs/1612.05515.

[MP] Alessandro Mirone and Pierre Paleo. *python script: CSG.py*. https://github.com/pierrepaleo/csg.

[MP17] A. Mirone and P. Paleo. "A conjugate subgradient algorithm with adaptive preconditioning for the least absolute shrinkage and selection operator minimization". In: *Computational Mathematics and Mathematical Physics* 57.4 (Apr. 2017), pp. 739–748. ISSN: 1555-6662. DOI: 10.1134/S0965542517040066. URL: https://doi.org/10.1134/S0965542517040066.

[Mün+09] Beat Münch et al. "Stripe and ring artifact removal with combined wavelet—Fourier filtering". In: *Opt. Express* 17.10 (May 2009), pp. 8567–8591. DOI: 10.1364/OE.17.008567. URL: http://www.opticsexpress.org/abstract.cfm?URI=oe-17-10-8567.

[Not17] Thibault Notargiacomo. "Computationally Efficient Sparse Prior in Regularized Iterative Tomographic Reconstruction". PhD thesis. Grenoble Images Parole Signal Automatique, 2017.

[Nvi15] Nvidia. *The CUDA C programming guide*. http://docs.nvidia.com/cuda/cuda-c-programming-guide. 2015.

[NVi17] NVidia. *The cuBLAS library*. https://developer.nvidia.com/cublas. 2017.

[OV14] Daniel O'Connor and Lieven Vandenberghe. "Primal-dual decomposition by operator splitting and applications to image deblurring". In: *SIAM Journal on Imaging Sciences* 7.3 (2014), pp. 1724–1754.

[Pal17] Pierre Paleo. *Jupyter notebooks*. https://github.com/pierrepaleo/notebooks. 2017.

[PB15] Daniel Pelt and Joost Batenburg. "Accurately approximating algebraic tomographic reconstruction by filtered backprojection". In: *Scientific Computing* Conference paper (2015). Ed. by M. King; S. Glick; K. Mueller. URL: https://ir.cwi.nl/pub/23742.

[PC11] Thomas Pock and Antonin Chambolle. "Diagonal preconditioning for first order primal-dual algorithms in convex optimization". In: *2011 International Conference on Computer Vision*. IEEE, 2011, pp. 1762–1769.

[Pel16] D.M. Pelt. "Filter-based reconstruction methods for tomography". PhD thesis. University of Leiden, 2016.

[RF12] Sathish Ramani and Jeffrey A Fessler. "A splitting-based iterative algorithm for accelerated statistical X-ray CT reconstruction". In: *IEEE transactions on medical imaging* 31.3 (2012), pp. 677–688.

[Rit+14] S Rit et al. "The Reconstruction Toolkit (RTK), an open-source cone-beam CT reconstruction toolkit based on the Insight Toolkit (ITK)". In: *Journal of Physics: Conference Series* 489.1 (2014), p. 012079. URL: http://stacks.iop.org/1742-6596/489/i=1/a=012079.

[SBK05] Ivan W Selesnick, Richard G Baraniuk, and Nick C Kingsbury. "The dual-tree complex wavelet transform". In: *IEEE signal processing magazine* 22.6 (2005), pp. 123–151.

[SF09a] Ivan W. Selesnick and Mário A. T. Figueiredo. *Signal restoration with overcomplete wavelet transforms: comparison of analysis and synthesis priors*. 2009. DOI: 10.1117/12.826663. URL: http://dx.doi.org/10.1117/12.826663.

[SF09b] Jean-Luc Starck and Jalal Fadili. "Numerical issues when using wavelets". In: *Encyclopedia of Complexity and Systems Science*. Springer, 2009, pp. 6352–6368.

[She92] M. J. Shensa. "The discrete wavelet transform: wedding the a trous and Mallat algorithms". In: *IEEE Transactions on Signal Processing* 40.10 (Oct. 1992), pp. 2464–2482. ISSN: 1053-587X. DOI: 10.1109/78.157290.

[She96] Y. Sheng. *Wavelet Transform*. Ed. by A. D. Poularikas. The Electrical Engineering Handbook Series. CRC press, 1996, pp. 747–827.

[SM03] Lin Xiao Stephen Boyd and Almir Mutapcic. "Subgradient methods". In: *Notes for EE392o* (2003).

[Swe98] Wim Sweldens. "The lifting scheme: A construction of second generation wavelets". In: *SIAM Journal on Mathematical Analysis* 29.2 (1998), pp. 511–546.

[TM12] David Taubman and Michael Marcellin. *JPEG2000 image compression fundamentals, standards and practice: image compression fundamentals, standards and practice*. Vol. 642. Springer Science & Business Media, 2012. ISBN: 978-0-7923-7519-7. DOI: 10.1007/978-1-4615-0799-4.

[WBA09] Pierre Weiss, Laure Blanc-Féraud, and Gilles Aubert. "Efficient schemes for total variation minimization under constraints in image processing". In: *SIAM journal on Scientific Computing* 31.3 (2009), pp. 2047–2080.

[XM06] Fang Xu and Klaus Mueller. "A comparative study of popular interpolation and integration methods for use in computed tomography". In: *Biomedical Imaging: Nano to Macro, 2006. 3rd IEEE International Symposium on*. IEEE. 2006, pp. 1252–1255.

[ZG00] Gengsheng L Zeng and Grant T Gullberg. "Unmatched projector/backprojector pairs in an iterative reconstruction algorithm". In: *IEEE transactions on medical imaging* 19.5 (2000), pp. 548–555.

Chapter 4

Artefacts removal and local tomography

In this chapter, we explore how the classical optimization-based reconstruction algorithms can be extended to account for artefacts. In general, real-world acquired data is often degraded with "noise" sources linked to the acquisition process. These effects lead to what is called *artefacts* in the reconstructed data, in the sense of those are features we do not want or expect to see. Thus, if the artefacts source can be accurately described in the forward model, the reconstruction process can hopefully eliminate them.

In the first section, the forward model is modified to take rings artefacts into account. A structured noise $\boldsymbol{r}$ is added to the sinogram $\boldsymbol{Px}$, leading to the optimization problem[1]

$$\underset{\boldsymbol{x},\boldsymbol{r}}{\operatorname{argmin}} \left\{ \frac{1}{2} \left\| \boldsymbol{Px} - \boldsymbol{d} \right\|_2^2 \right\} + \beta \left\| \boldsymbol{Dx} \right\|_1 + \beta_r \left\| \boldsymbol{r} \right\|_1$$

In the second section, we describe a correction algorithm for local tomography artefacts removal.

$$\underset{\boldsymbol{w}}{\operatorname{argmin}} \left\{ \left\| \boldsymbol{C}\tilde{\boldsymbol{P}}\boldsymbol{G}\boldsymbol{w} - \boldsymbol{d} \right\|_2^2 + \phi(\boldsymbol{w}) \right\}$$

where the slice $\tilde{\boldsymbol{x}} = \boldsymbol{Gw}$ is the synthesis of variables in a coarse image basis, $\tilde{\boldsymbol{P}}$ is a projection operator on a larger grid, and $\boldsymbol{C}$ is a truncation operator simulating the local tomography acquisition. The function ϕ encodes a known zone constraint.

In both cases, the problem differ from the standard reconstruction problem by variables and/or operators, but are still in the unified mathematical framework developed in the second chapter.

4.1 Rings artefacts

In this part, we show that the forward model can account for rings artefacts by handling them as a structured noise. This work was published in [PM15].

4.1.1 Rings artefacts in tomographic reconstruction

During a tomographic acquisition process, some flaws in the experimental setup can lead to unwanted artefacts appearing on the reconstructed slice. Ring-shaped features are a well-known example of such artefacts. Even after preprocessing steps like flat-field correction and median filtering, these artefacts can remain and are detrimental to the reconstruction quality. Therefore, multiple techniques have been developed to tackle this problem.

Generally speaking, ring artefacts have various possible causes. The presence of defective pixels in the detector leads to sharp artefacts, while dust on scintillator crystal can

[1] in the case of an analysis formulation where $\boldsymbol{D}$ can be the gradient operator ∇ or a forward wavelets transform

form large artefacts. Experimental flaws can also include vibration of the monochromator or tilt of the rotation axis. In almost all cases, the defects appear as lines in the sinogram since there are independent of the projection angle. These spurious lines give raise to ring-shaped artefacts in the reconstructed object. Due to the nature of the sinogram, these rings are always centered in the image.

Various techniques have been proposed in the literature to reduce or suppress the rings artefacts. As reported in [RLH12], these techniques can be classified into two groups : sinogram preprocessing and reconstructed images post-processing. The preprocessing methods aim at detecting and correcting the spurious lines in the sinogram before applying the reconstruction process, thus, rings do not form if the method succeeds. A recent work [EB14] reports a regularized approach for rings artefacts reduction using a total variation denoising of the sinogram before calling the reconstruction routine. It is a generalization of the algorithm proposed in [Tit+10] which consists in a regularization of the sinogram. This can also be classified in the sinogram pre-processing techniques.

On the other hand, post-processing techniques work directly on the reconstructed image, trying to extract the concentric circles and filter them. These methods often perform a transformation into polar coordinates to transform the concentric circles into straight lines [PKK09].

A comprehensive comparison of ring artefact removal methods can be found in [RLH12]. Although these methods certainly provide satisfactory results in their limited framework, the authors report that no existing method is really suitable for reliably correcting all rings, since they always introduce other distortions. Thus, the ring removal problem cannot be considered as solved and is subject to continual efforts. In this work, a new approach to correct the ring artefacts in a compressed-sensing framework is presented. In this technique, the correction is intrinsically part of the reconstruction process, hence can be neither be viewed as sinogram pre-processing nor slice post-processing. The basic idea is to split the sinogram into two components, one containing the genuine sinogram and the other containing the artefacts component. This approach bears some similarities with a recent work [Moh+14] where the artefacts model is also included in the objective function which is optimized in a non homogeneous iterative coordinate descent. However, the aforementioned method uses a ℓ_2 minimization, while regularized methods typically use a prior like TV, which is adapted for undersampled data.

In this work, we use two rings artefact removal methods for comparison purpose: one sinogram filtering and one slice processing approach. The reference sinogram preprocessing technique is the wavelet-Fourier filtering [Mün+09], hereby denoted Fourier-Wavelets (FW). This methods first compute the wavelet decomposition at a level L of the sinogram. The vertical detail coefficients V_i at level $i \in [[1, L]]$ emphasize the spurious lines that give raise to rings artefacts. In these coefficients, a spurious line is nearly constant along the projection angle, thus, it has only low frequencies in the Fourier domain. Filtering these few low frequencies in the Fourier domain enables to suppress the line after taking the inverse Fourier transform. The filter used is a high-pass Gaussian filter whose standard deviation σ tunes the bandwidth. Then, the sinogram is reconstructed from these filtered wavelet coefficients. In the tests, σ denotes the standard deviation of the Gaussian filter and L is the number of levels of the wavelet decomposition. The reference image correction technique used here is Rings Correction in Polar Coordinates [PKK09], hereby denoted Rings Correction in Polar Coordinates (RCP). It transforms the image into polar coordinates and performs a low-pass filtering in the radial direction. The filtered image is then subtracted from the original image, and a threshold is applied to ignore non-artefact structures. The result is filtered in the azimuthal direction. After a transformation into

Cartesian coordinates, the image should only contain rings artefacts ; these are subtracted from the original image. A C++ implementation can be found at [Bla14].

4.1.2 Simultaneous iterative reconstruction and correction of rings artefacts

In this section, we present how rings correction can be handled directly in the reconstruction process by integrating additional variables in the functional to minimize. This approach is independent of the regularization used and can therefore be applied in various frameworks like total variation and dictionary-based reconstruction. In the general framework of regularized reconstruction, let $f(\boldsymbol{x}, \boldsymbol{d})$ denote the data fidelity term (where $\boldsymbol{d}$ is the acquired sinogram and $\boldsymbol{x}$ is the latent image/volume to be reconstructed), and $g(\boldsymbol{x})$ denote the regularization term (see section 2.1.2).

In this approach, the rings correction consists in splitting the sinogram into two components: the "genuine" sinogram, and the spurious straight lines giving rings after back-projection. Indeed, although ring artefacts have various causes, they often appear in the sinogram as lines which are almost constant along the projection angle (see Figure 4.1.1). These artefacts form a *structured noise* which can be taken into account in the forward model. Thus, a natural approach is to model the rings by constant lines in the sinogram.

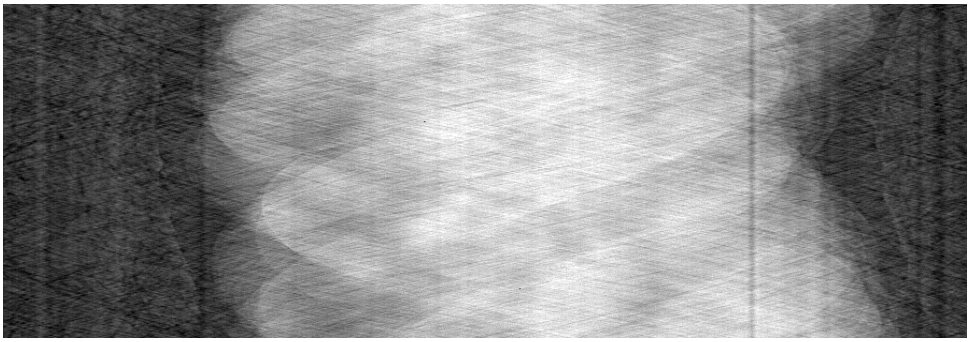


Figure 4.1.1: Sinogram of a CT scan of dendritic crystals (data courtesy: Daniil Kazantsev, University of Manchester / Diamond I13). The rings artefacts appear as vertical stripes.

In iterative techniques, the rings correction can be handled by an additional variable vector $\boldsymbol{r}$ in the fidelity term $f(\boldsymbol{y}, \boldsymbol{x})$: rings variables are added to the sinogram for each projection. For a Gaussian noise prior, the fidelity term for one projection reads

$$f(\boldsymbol{x}, \boldsymbol{d}, \boldsymbol{r}) = \frac{1}{2} \|\boldsymbol{P}\boldsymbol{x} + \boldsymbol{U}\boldsymbol{r} - \boldsymbol{d}\|_2^2 \tag{4.1.1}$$

If N is the number of pixels of one dimension of the image $\boldsymbol{x}$, then $\boldsymbol{x}$ is a $(N^2, 1)$ vector (see section 1.3.2). The sinogram $\boldsymbol{d}$ is a $(N_p \times N, 1)$ vector, where N_p is the number of projection angles. The rings vector $\boldsymbol{r}$ is a $(N, 1)$ variable, where each component is repeated along all the N_p angles according to Figure 4.1.3. The operator $\boldsymbol{U}$, which is a $(N_p \times N, N)$ matrix, implements the operation illustrated on Figure 4.1.3.

Interestingly, the adjoint of the "tiling operator" $\boldsymbol{U}$ is a "summation operator". The evaluation of $\boldsymbol{U}^T\boldsymbol{u}$, where $\boldsymbol{u}$ has the dimensionality of a sinogram, consists in performing the sum of $\boldsymbol{u}$ along the lines. It means that the component k of $\boldsymbol{v} = \boldsymbol{U}^T\boldsymbol{u}$ ($\boldsymbol{v}$ is has the dimensionality of a sinogram line) is the sum of the column k of $\boldsymbol{u}$. Summing a sinogram

along its lines is sometimes used as a sinogram pre-processing technique, as the rings artefacts are more prominent after the summation (see Figure 4.1.2). Normalizing the sinogram with this sum sometimes corrects the artefacts, although the quantitativeness is lost. In the proposed optimization-based framework, the rings variables are often updated through the gradient of the fidelity term, which involves this summation step $\boldsymbol{U}^T$.

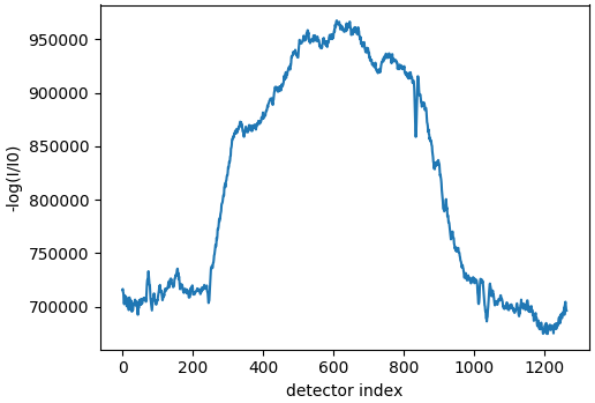


Figure 4.1.2: Profile of the summation along the lines of the singoram in Figure 4.1.1

We emphasize the fact that the sinogram decomposition into a genuine sinogram $\boldsymbol{Px}$ and spurious rings $\boldsymbol{r}$ is not a pre-processing technique ; the rings removal is intrinsically part of the reconstruction process. At each iteration, the image $\boldsymbol{x}$ and the rings variables $\boldsymbol{r}$ are adapted to minimize the energy $F(\boldsymbol{x}, \boldsymbol{r})$.

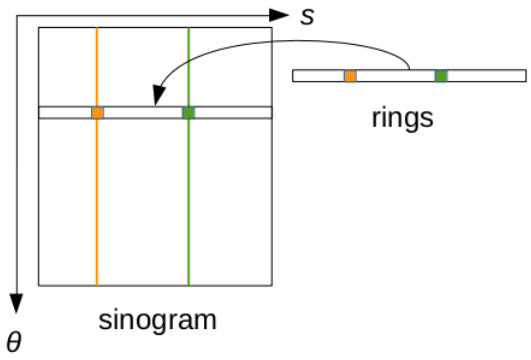


Figure 4.1.3: Principle of the rings separation. θ is the projection angle and s is the detector bin index. The vertical orange and green lines represent spurious lines giving raise to ring artefacts. The decomposition $\boldsymbol{Px} + \boldsymbol{r}$ forces the ring values to be captured in the vector $\boldsymbol{r}$ (independent of the projection angle). In the end, only the part without the rings $\boldsymbol{r}$ is returned.

This splitting is done in the reconstruction process, so the two components are updated after each iteration. In the end, only the valid sinogram component is kept while the rings variables are discarded.

By incorporating the rings correction in the reconstruction process, a consistence between the reconstructed slice and the estimated rings artefacts is maintained. Sinogram pre-processing techniques modify the acquired data to filter the unwanted lines, which often introduces new artefacts. On the other hand, image correction techniques can also

add new artefacts when circular features are detected as artefacts; and the forward and backward Cartesian-polar coordinate transforms lead to a loss of precision even with interpolation. When the rings correction is done in the reconstruction process, the data is not modified, and the rings artefacts are estimated in the forward model as a structured noise.

The inclusion of rings correction into the iterative reconstruction is reviewed for three types of regularization: Total Variation, Dictionary and Wavelets. We then illustrate the rings removal capability of the proposed methods on synthetic and real data.

4.1.3 Iterative rings correction in the TV framework

When the sparsity-inducing prior is the total variation, the functional $\tilde{f}(\boldsymbol{x}, \boldsymbol{r})$ is

$$\tilde{f}(\boldsymbol{x}, \boldsymbol{r}) = f(\boldsymbol{x}, \boldsymbol{r}) + \beta \,\mathrm{TV}\,(\boldsymbol{x}) + \beta_r \,\|\boldsymbol{r}\|_1 \tag{4.1.2}$$

where β is a parameter weighting the relative importance of spatial regularization, and β_r is a penalization parameter for the rings.

While sinogram preprocessing techniques filter the lines parallel to the projection angle, this approach forces the sinogram to be decomposed as a sinogram $\boldsymbol{P}\boldsymbol{x}$ and rings variables $\boldsymbol{r}$. The sparsity constraint $\beta_r \,\|\boldsymbol{r}\|_1$ forces the rings variables to have only a few not null components, since the ℓ_1 norm is a convex relaxation of the sparsity-inducing ℓ_0 norm.

The minimum of $\tilde{f}(\boldsymbol{x}, \boldsymbol{r})$ can be found with various convex optimization algorithms. In the context of TV regularization, we use the Chambolle-Pock algorithm (see section 3.1.2).

Most of the time, the fidelity term is a ℓ_2 squared distance (see section 2.4.4). The explicit function to minimize is then given by Equation (4.1.3)

$$\underset{x}{\operatorname{argmin}} \left\{ \frac{1}{2} \,\|\boldsymbol{P}\boldsymbol{x} + \boldsymbol{U}\boldsymbol{r} - \boldsymbol{d}\|_2^2 + \lambda \,\|\nabla \boldsymbol{x}\|_1 + \lambda_r \,\|\boldsymbol{r}\|_1 \right\} \tag{4.1.3}$$

There are at least two ways of building a C-P algorithm instance from (4.1.3), but we will choose the most "straightforward". We have:

$$\begin{aligned}
&\frac{1}{2} \,\|\boldsymbol{P}\boldsymbol{x} + \boldsymbol{U}\boldsymbol{r} - \boldsymbol{d}\|_2^2 + \lambda \,\|\nabla \boldsymbol{x}\|_1 \\
&= \max_{\boldsymbol{q}} \left\{ \langle \boldsymbol{P}\boldsymbol{x} + \boldsymbol{U}\boldsymbol{r} - \boldsymbol{d} \,,\, \boldsymbol{q} \rangle - \frac{1}{2} \,\|\boldsymbol{q}\|_2^2 \right\} + \max_{\boldsymbol{z}} \left\{ \langle \nabla \boldsymbol{x} \,,\, \boldsymbol{z} \rangle - i_{\overset{\lambda}{\infty}}(\boldsymbol{z}) \right\} \\
&= \max_{\boldsymbol{q},\boldsymbol{z}} \left\{ \left\langle \begin{bmatrix} \boldsymbol{P} & \boldsymbol{U} \\ \nabla & \boldsymbol{0}_r \end{bmatrix} \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{r} \end{pmatrix} \,,\, \begin{pmatrix} \boldsymbol{q} \\ \boldsymbol{z} \end{pmatrix} \right\rangle - \frac{1}{2} \,\|\boldsymbol{q}\|_2^2 - \langle \boldsymbol{d} \,,\, \boldsymbol{q} \rangle - i_{\overset{\lambda}{\infty}}(\boldsymbol{z}) \right\}
\end{aligned} \tag{4.1.4}$$

Including the term $\lambda_r \,\|\boldsymbol{r}\|_1$, problem (4.1.3) corresponds to the saddle point problem (3.1.5) with

$$\begin{aligned}
\boldsymbol{K} &= \begin{bmatrix} \boldsymbol{P} & \boldsymbol{U} \\ \nabla & 0_r \end{bmatrix} \\
F(\boldsymbol{x}, \boldsymbol{r}) &= \lambda_r \,\|\boldsymbol{r}\|_1 \\
G^*(\boldsymbol{q}, \boldsymbol{z}) &= \frac{1}{2} \,\|\boldsymbol{q}\|_2^2 + \langle \boldsymbol{d} \,,\, \boldsymbol{q} \rangle + i_{\overset{\lambda}{\infty}}(\boldsymbol{z})
\end{aligned} \tag{4.1.5}$$

As the above functions are separable with respect to their variables, the corresponding proximal operators are given by

$$\begin{aligned}
\mathrm{prox}_{\tau F}\,(\boldsymbol{x}, \boldsymbol{r}) &= (\boldsymbol{x} \,,\, \mathcal{S}_{\tau\lambda_r}(\boldsymbol{r})) \\
\mathrm{prox}_{\sigma G^*}\,(\boldsymbol{q}, \boldsymbol{z}) &= \left(\frac{\boldsymbol{q} - \sigma \boldsymbol{b}}{1 + \sigma} \,,\, P_{B^{\lambda}_{\infty}}(\boldsymbol{z}) \right)
\end{aligned} \tag{4.1.6}$$

where $\mathcal{S}$ is the soft thresholding operator (see Proposition 2.6.3). This approach has the advantage of not adding a third dual variable. Operator $\boldsymbol{K}$ in Equation 4.1.5 and proximal maps in Equation 4.1.6 are all what is needed to implement the C-P algorithm.

4.1.4 Iterative rings correction in Dictionary-based reconstruction

Similarly to total variation (4.1.3), the proximal operator is separable with respect to patch variables $\boldsymbol{w}$ and rings variables $\boldsymbol{r}$. The optimization problem becomes

$$\underset{\boldsymbol{w},\boldsymbol{r}}{\operatorname{argmin}} \ \|\boldsymbol{P}\boldsymbol{D}^*\boldsymbol{w} + \boldsymbol{U}\boldsymbol{r} - \boldsymbol{d}\|_2^2 + \beta_{\mathrm{DL}} \|\boldsymbol{w}\|_1 + \beta_r \|\boldsymbol{r}\|_1 + \rho \cdot h(\boldsymbol{w}) \tag{4.1.7}$$

where $\boldsymbol{w}$ denotes the dictionary coefficients in the frame $\boldsymbol{D}^*$, and $\rho \cdot h(\boldsymbol{w})$ is a (differentiable) function promoting overlap between patches [Mir+14]. The scalar parameters β_{DL}, β_r and ρ weight the influence of the dictionary regularization, rings regularization and patches overlapping, respectively. In this case, the non-smooth term is a simple ℓ_1 norm which has a closed-form proximal operator (the soft thresholding operator). Thus, the FISTA algorithm can be advantageously used. As the regularization is separable in β_{DL} and β_r, the proximal simply consists in an additional soft threshold, which does not add a significant complexity to the algorithm. The function to optimize has two (vector) variables. The gradient of the fidelity term is

$$\nabla f(\boldsymbol{w},\boldsymbol{r}) = \begin{pmatrix} \nabla_{\boldsymbol{w}} f(\boldsymbol{w},\boldsymbol{r}) \\ \nabla_{\boldsymbol{r}} f(\boldsymbol{w},\boldsymbol{r}) \end{pmatrix} = \begin{pmatrix} \boldsymbol{D}\boldsymbol{P}^T\boldsymbol{e} \\ \boldsymbol{U}^T\boldsymbol{e} \end{pmatrix} + \rho \begin{pmatrix} \nabla_{\boldsymbol{w}} h(\boldsymbol{w}) \\ \boldsymbol{0} \end{pmatrix} \tag{4.1.8}$$

where $\boldsymbol{e} = \boldsymbol{P}\boldsymbol{D}^*\boldsymbol{w} + \boldsymbol{U}\boldsymbol{r} - \boldsymbol{d}$ is the "error term" (operand of the norm) at each iteration. FISTA, which is based on a (projected) gradient descent scheme (see), has a step size $1/L$ where $L > 0$ is the largest eigenvalue of the Hessian of the data fidelity term. This Hessian is given by Equation (4.1.9)

$$\begin{bmatrix} \nabla_{\boldsymbol{w}}^2 & \nabla_{\boldsymbol{w}}\nabla_{\boldsymbol{r}} \\ \nabla_{\boldsymbol{r}}\nabla_{\boldsymbol{w}} & \nabla_{\boldsymbol{r}}^2 \end{bmatrix} (f + h) = \begin{bmatrix} \boldsymbol{D}\boldsymbol{P}^T\boldsymbol{P}\boldsymbol{D}^* & \boldsymbol{U}^T\boldsymbol{P}\boldsymbol{D}^* \\ \boldsymbol{D}\boldsymbol{P}^T\boldsymbol{U} & \boldsymbol{U}^T\boldsymbol{U} \end{bmatrix} + \begin{bmatrix} \nabla_{\boldsymbol{w}}^2 h(\boldsymbol{w}) & 0 \\ 0 & 0 \end{bmatrix} \tag{4.1.9}$$

and is, as usual, computed with several tens of iterations of the power method.

4.1.5 Iterative rings correction in the Wavelets framework

In the wavelets framework, the analysis formulation of the reconstruction problem with rings correction is

$$\underset{\boldsymbol{x},\boldsymbol{r}}{\operatorname{argmin}} \ \frac{1}{2} \|\boldsymbol{P}\boldsymbol{x} + \boldsymbol{U}\boldsymbol{r} - \boldsymbol{d}\|_2^2 + \beta \|\boldsymbol{W}\boldsymbol{x}\|_1 + \beta_r \|\boldsymbol{r}\|_1 \tag{4.1.10}$$

In the case of the standard DWT[2], the proximal of $\beta \|\boldsymbol{W}\boldsymbol{x}\|_1$ is straightforward; thus, the FISTA method can be used (the differentiable term gradient is easy to compute and the prox is simple). The proximal, separable in $\boldsymbol{x}$ and $\boldsymbol{r}$, is given by

$$\operatorname{prox}_{\beta\|\boldsymbol{W}\boldsymbol{x}\|_1 + \beta_r\|\boldsymbol{r}\|_1}(\boldsymbol{x},\boldsymbol{r}) = \left(\boldsymbol{W}^T\mathcal{S}_{\beta}(\boldsymbol{W}\boldsymbol{x}),\ \mathcal{S}_{\beta_r}(\boldsymbol{r})\right) \tag{4.1.11}$$

which, in words, consists in thresholding the wavelets coefficients of $\boldsymbol{x}$ (with threshold β) before transforming them back, and thresholding the rings variable $\boldsymbol{r}$ (with threshold β_r).

In the case of SWT, the equality $\boldsymbol{W}^T\boldsymbol{W} = \mu\boldsymbol{I}$ does not hold anymore, so the prox cannot be simply computed. Instead, a strategy similar to the splitting done for the Chambolle-Pock TV solver can be adopted. The Equations 4.1.5 and 4.1.6 can be used by replacing the operator ∇ with the SWT.

[2] with orthogonal filters

4.1.6 Results - simulated data

We present here results on simulated data, and compare our method with two mainstream techniques of rings correction: FW, a sinogram pre-processing based on Fourier-Wavelet de-striping [Mün+09]; and RCP, an image correction using polar coordinates transformation [PKK09][3] We also present the results on simulated undersampled data, on which our method is well adapted.

In simulated data, we use the standard test image "Lena" containing both smooth components and texture details, making it more challenging to reconstruct than usual phantoms like Shepp-Logan phantom. For all these tests, the image size is 512×512 pixels and 800 projections were used for the reconstructions ; except for the fourth test (Figure 4.1.13) where only 80 projections were used. The tests are divided in increasing levels of difficulty for rings removal methods.

In some figures, we drew red arrows to indicate the location of faint rings which can still be seen after correction.

First test case - small constant lines

In the first test, constant lines are added in the sinogram. These lines independent from the projection angle give raise to rings artefacts in the reconstructed slice. Since the spurious lines have a constant value, they should be well handled by pre-processing techniques.

Figure 4.1.4 shows the results for the first test case. The rings are reduced by the sinogram pre-processing technique (4.1.4.c), but they do not totally disappear. Besides, additional artefacts appear after the correction. The RCP performs slightly better (4.1.4.e) ; a strong artefact is added to the right but the result is qualitatively better. The Total Variation regularization entirely removes the rings (4.1.4.g). It can be seen on 4.1.4.h that other rings were actually added to the slice, but their amplitude is very small according to the scale (color bar), so they are not detrimental to the reconstruction quality. The Dictionary Learning reconstruction (4.1.4.i) removes the rings, but the difference image 4.1.4.j shows that the slice is slightly blurred: the very fine details are smoothed.

[3] In the case of RCP, the thresholding parameters are set so that all the image pixels can be considered as possibly part of an artefact. The important remaining parameters are the maximum ring width W, and the maximum angular arc θ_0 we expect the rings to have.

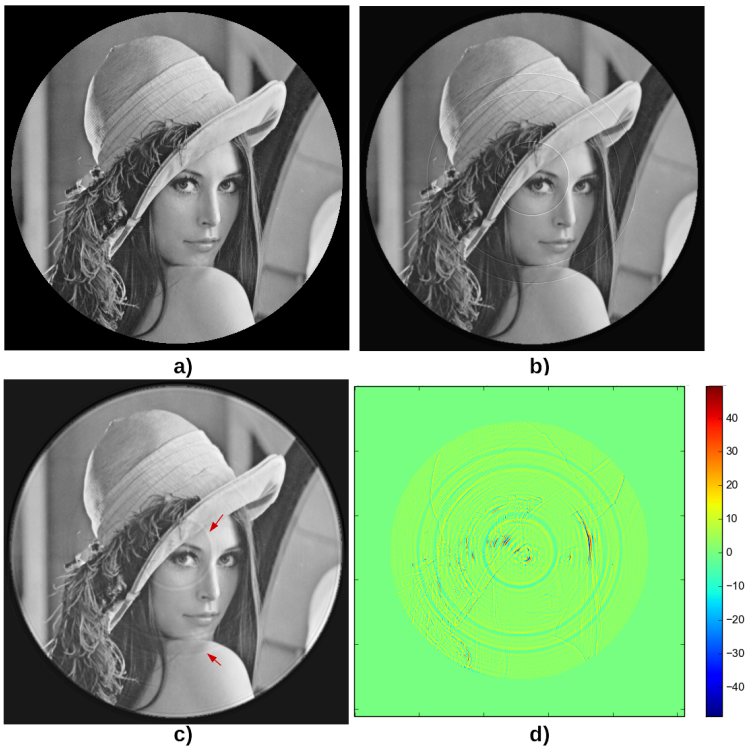


Figure 4.1.4: First test case.
(a) Original phantom. (b) Result of filtered backprojection after adding constant lines in the sinogram. (c) Image back-projected after applying the Munch *et al.* de-striper algorithm with $\sigma = 3.5$, $L = 2$ and the "Daubechies 15" wavelet. (d) Difference between the phantom and the corrected image. The PSNR is 26.6.

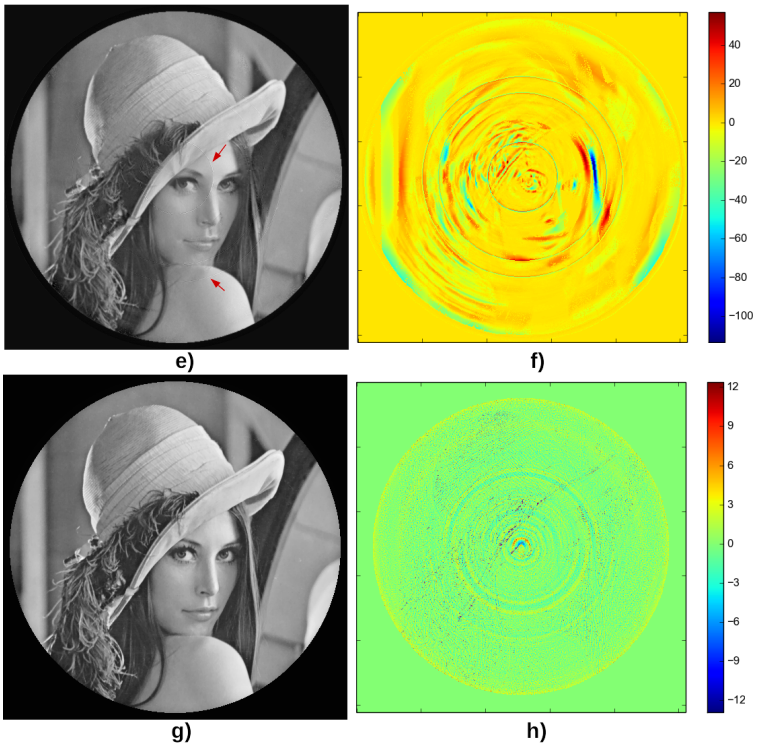


Figure 4.1.5: First test case (continued).
(e) Result of the correction with the RCP technique with $W = 10$ and $\theta_0 = 10$. (f) Difference between the phantom and the corrected image. The PSNR is 29.6. (g) Result of the reconstruction using the Total Variation regularization, with parameters $\beta = 0.5$, $\beta_r = 0.05$. (h) Difference between the phantom and the corrected image. The PSNR is 39.0.

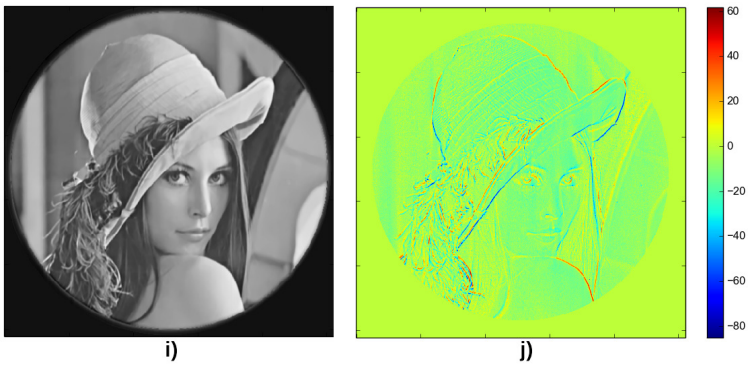


Figure 4.1.6: First test case (continued).
(i) Result of the reconstruction using the Dictionary Learning technique with $\beta = 0.3$, $\beta_r = 0.5$, $\rho = 1$ (j) Difference between the phantom and the reconstructed image. The PSNR is 29.2.

Second test case - varying lines

In the second test, lines with variable intensity are added to the sinogram. This makes the correction more difficult for pre-processing techniques, especially if the lines have sharp

variations (i.e high frequencies components).

Figure 4.1.7 shows the results for the second test case. The sinogram filtering adds many spurious rings (4.1.7.c). The RCP technique removes most of the rings, but small rings details remain. The difference image 4.1.7.f shows some artefifacts which may be the result of the different transformations between Cartesian and polar coordinates. The Total Variation regularization (4.1.7.g) entirely removes the rings artefacts ; the result is qualitatively very close to the original phantom. On the other hand, the Dictionary Learning reconstruction does not manage to perfectly correct the slice: a remaining ring can be seen on Lena's cheek (Figures 4.1.7.i and 4.1.7.j). A solution can be to increase the value of the parameter β (i.e β_{DL} in eq. (4.1.7)), but it would lead to a more blurry image. This suggests that the Total Variation is more suited that Dictionary Learning to correct the rings on a image containing many texture components.

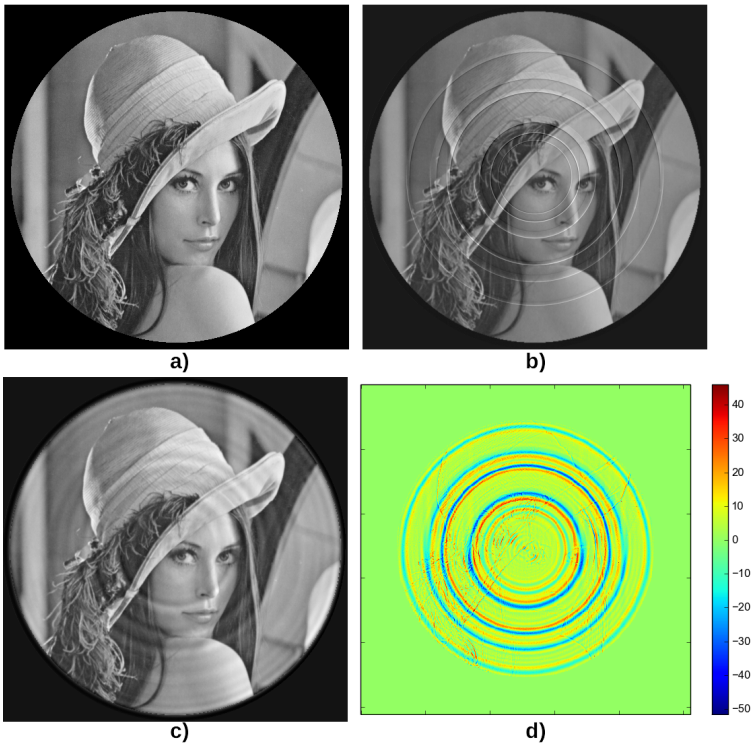


Figure 4.1.7: Second test case.
(a) Original phantom. (b) Result of filtered backprojection after adding lines of variable width and intensity in the sinogram. (c) Image back-projected after applying the Munch *et al.* de-striper algorithm, with $\sigma = 1.5$, $L = 2$ and the "Daubechies 15" wavelet. (d) Difference between the phantom and the corrected image. The PSNR is 29.2.

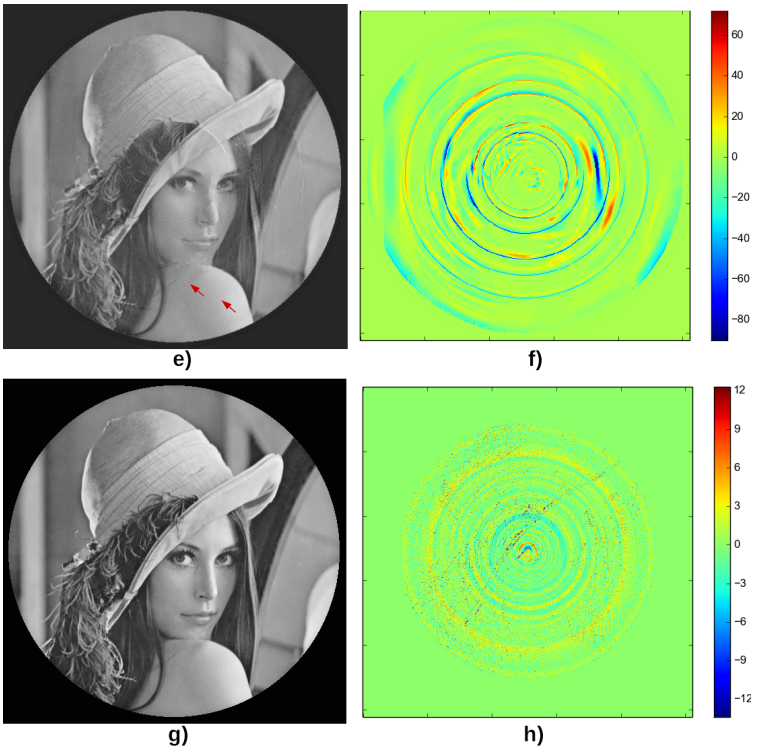


Figure 4.1.8: Second test case (continued).
(e) Result of the correction using the RCP technique with $W = 10$ and $\theta_0 = 10$. (f) Difference between the phantom and the corrected image. The PSNR is 25.1. (g) Result of the reconstruction using the Total Variation regularization, with parameters $\beta = 0.5$, $\beta_r = 0.05$. (h) Difference between the phantom and the corrected image. The PSNR is 39.4.

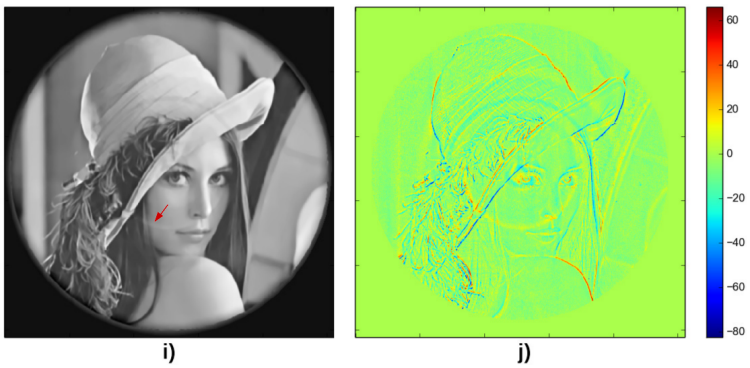


Figure 4.1.9: Second test case (continued).
(i) Result of the reconstruction using the Dictionary Learning technique with $\beta = 0.7$, $\beta_r = 0.5$, $\rho = 1$ (j) Difference between the phantom and the reconstructed image. The PSNR is 30.6.

Third test case - circular features

In the third test, circular features are added in the phantom before adding spurious lines in the sinogram. This case is more challenging because correction methods should not remove

any feature coming from the phantom (they belong to the "true" image), while actually removing rings coming from the sinogram (they come from a flaw in the experimental setup).

Figure 4.1.10 shows the results for the third test case. Here two features are added to the original phantom (4.1.10.a): a black disk and a circular "ring". These features are part of the phantom, they should not be filtered by rings correction techniques. Lines added to the sinogram are not constant along the projection angle, and their width can be several pixels. This leads to a back-projected image (4.1.10.b) with large rings. The RCP technique (4.1.10.e) is more efficient than the sinogram preprocessing (4.1.10.c).

The Total Variation (4.1.10.g) removes the rings, but also nearly remove the circular feature of the phantom 4.1.10.a, which gives the blue circle in the difference image 4.1.10.h. The black disk is well preserved.

The Dictionary Learning technique (4.1.10.i) does no gives as good results as the Total Variation regularization. The black disk is blurred, and the rings are not entirely corrected.

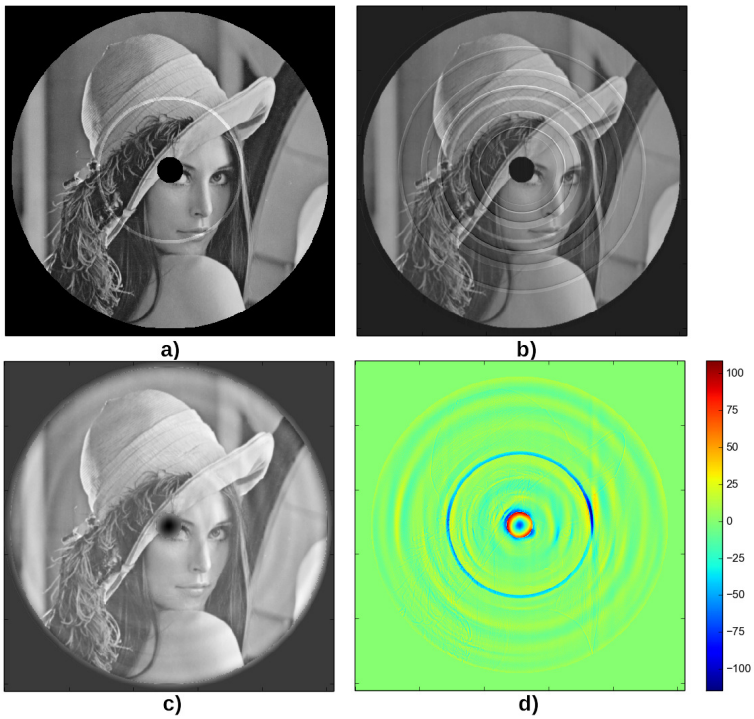


Figure 4.1.10: Third test case.
(a) Original phantom. (b) Result of filtered backprojection after adding lines of variable width and intensity in the sinogram. (c) Image back-projected after applying the Munch *et al.* de-striper algorithm, with $\sigma = 2.5$, $L = 5$ and the "Daubechies 20" wavelet. (d) Difference between the phantom and the corrected image. The PSNR is 23.2.

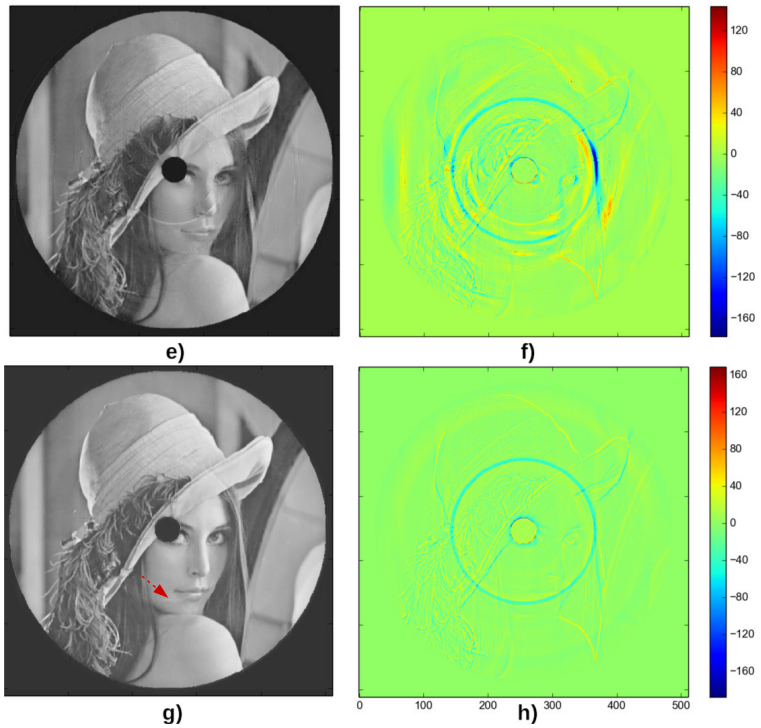


Figure 4.1.11: Third test case (continued).
(e) Result of the correction using the RCP technique, with $W = 10$ and $\theta_0 = 10$. (f) Difference between the phantom and the corrected image. The PSNR is 21.2 (g) Result of the reconstruction using the Total Variation regularization, with parameters $\beta = 0.5$, $\beta_r = 0.05$. (h) Difference between the phantom and the corrected image. The PSNR is 29.9.

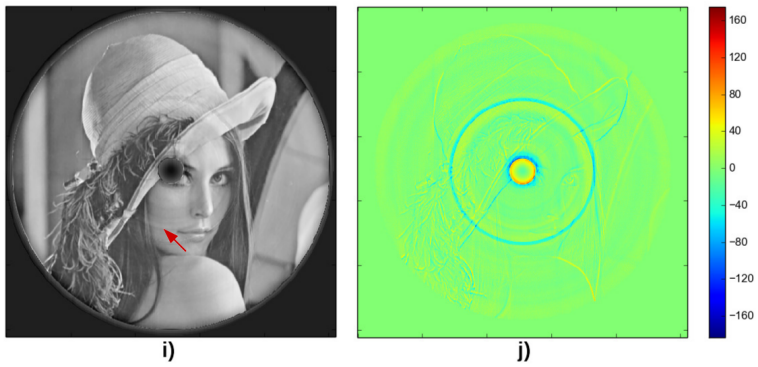


Figure 4.1.12: Third test case (continued).
(i) Result of the reconstruction using the Dictionary Learning technique with parameters $\beta = 0.05$, $\beta_r = 3 \cdot 10^{-3}$, $\rho = 10$ (j) Difference between the reconstruction and the phantom. The PSNR is 28.3.

Fourth test case - undersampled data

Iterative reconstruction with regularization is especially interesting when it comes to reconstruct a volume from few projections. In the previous test cases, the 512×512 image needed $\frac{\pi}{2}512 \simeq 800$ projections to be accurately reconstructed with the filtered back-

projection. Figure 4.1.13 shows the result of the third test case with 80 projections instead of 800. Filtered backprojection (Figure 4.1.13.a) leads to star artefacts due to the data undersampling. The reconstruction is much more satisfactory with Dictionary Learning or TV regularisation. In these iterative methods, the ring artefacts correction can be turned off (Figure 4.1.13.b) or on by simply adjusting one parameter. In all cases, the black disk is preserved and the ring artefacts correction did not suffer from the few number of projections. One can notice that in this case, DL produces smoother results than TV, but does not entirely remove the rings (Figure 4.1.13.c) while TV is able to entirely remove the rings (Figure 4.1.13.d).

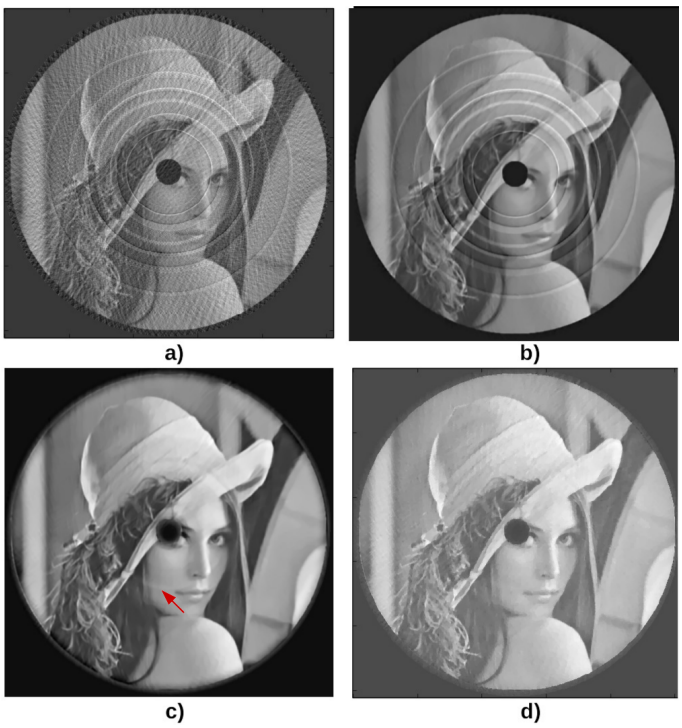


Figure 4.1.13: Reconstruction of the third case phantom (4.1.10.a) with 80 projections instead of 800. (a) Result of the reconstruction using the Filtered Back-Projection. (b) Result of the reconstruction using the Dictionary Learning technique, without rings correction, with parameters $\beta = 0.05$, $\rho = 10$. (c) Result of the reconstruction using the Dictionary Learning with rings correction, with parameters $\beta = 0.05$, $\beta_r = 3 \cdot 10^{-3}$, $\rho = 10$. (d) Result of the reconstruction using the Total Variation regularization with parameters $\beta = 1$, $\beta_r = 0.05$.

A plot of the rings variables during the total variation reconstruction shows that the rings are actually captured by the ring vector $\boldsymbol{r}$. The six peaks representing the detected lines in the sinogram are clearly visible in Figure 4.1.14. Their location correspond to the actual locations where the rings were added in the sinogram.

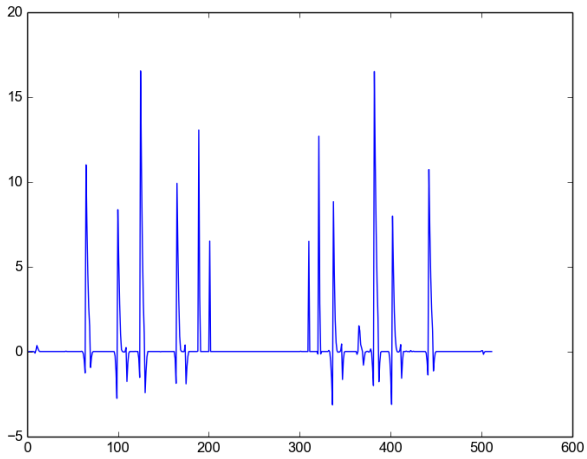


Figure 4.1.14: Vector of rings variables for second test case (Figure 4.1.7.b). The horizontal axis goes from zero to the number of bins of the detector (512). The peaks location correspond to the actual position of the rings added in the simulated data, which were 65-70, 100-110, 125-130, 165-175, 201-202, 321-323 (in pixels). This setting was a 180 degrees scan, so for each location $b \in [0, 511]$, a line was added at the "complementary" bin index $511 - b$ in order to have full rings (otherwise, only half rings appear in the reconstruction). This complementarity is correctly captured by the optimization procedure, as each peak has the same height as its "complementary peak".

Beside the visual aspect of the corrected image, we use the Peak Signal to Noise Ratio (PSNR) as a measure of the correction quality. Although PSNR gives a score to the overall similarity between the corrected image and the original phantom, it is inconsistent with the eye perception of quality. For example, RCP performed better than sinogram filtering in these tests, but got a lowest PSNR for cases 2 and 3. The structural similarity index gives the same kind of results. The reasons of this inconsistence can be the following. The Munch filter has a blurring effect – since it modifies the wavelet detail coefficients – which is detrimental to the image overall quality. However, the blurring effect is averaged in the MSE/SSIM calculation. On the other hand, the Polar filter does less blurring, but there are strong local errors, leading to a high MSE. Quantitative quality assessment is a difficult issue in tomographic reconstruction, and to our knowledge, no satisfactory metric adapted to tomographic reconstruction has been proposed yet.

For these tests, sinogram pre-processing technique did not yield good results. This can be due to the fact that the sinogram lines were captured by the wavelet approximation coefficients rather than by the detail coefficients, making the filtering ineffective. By trying with lines of smallest amplitude, the Fourier-Wavelet method actually worked without adding big artefacts in the reconstructed image.

4.1.7 Results - real data

We give here some results for reconstructions performed on real data.

Figure 4.1.15 shows the reconstruction results on a syntactic foam sample. The slice is 2048×2048 pixels, and 2449 projections were used. In this case, the rings are "large" in the extent that the radius difference between the exterior and the interior of the ring is several pixels. This means that the spurious lines in the sinogram have several pixel of width along the detector bins axis, forming "bands". However, the intensities of the lines forming a band has too many variations to be efficiently filtered by sinogram pre-processing

techniques. This case is also difficult for slice-correction algorithms which detect circular features, since the sample itself have circular features which should not be removed. The RCP technique (4.1.15.b), however, only detects circular features whose center is the image center. The Total Variation technique (4.1.15.c) removes most of the rings, but a relatively high β had to be chosen, which led to a somewhat blurred result. The Dictionary-based reconstruction (4.1.15.d) performs a better correction. Note that the dictionary has been learned offline on Lena image, and yet provided a satisfactory reconstruction.

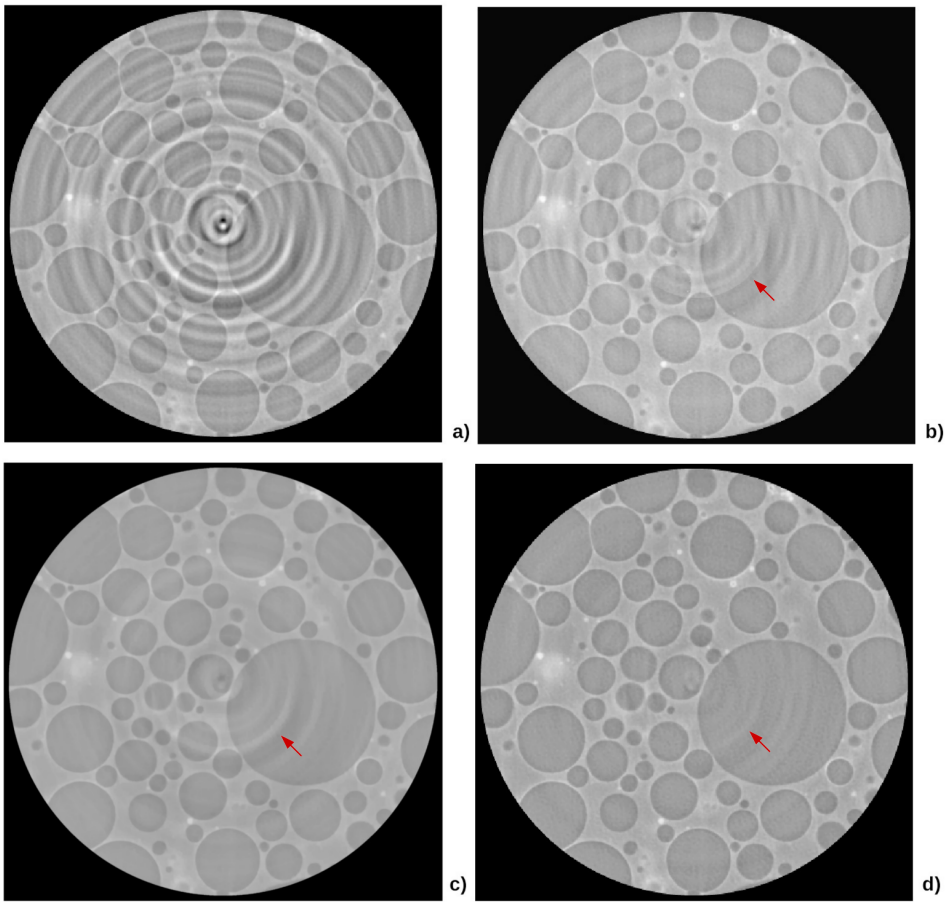


Figure 4.1.15: Syntactic foam sample (credits: Elodie Boller, ESRF ID19). (a) Filtered back-projection. (b) Correction with RCP technique, using the parameters $W = 60$ and $\theta_0 = 60$. (c) Reconstruction with Total Variation technique, using the parameters $\beta = 0.35$ and $\beta_r = 1 \cdot 10^{-6}$ (d) Reconstruction with Dictionary Learning technique, using the parameters $\beta_r = 0.1$, $\beta_{\mathrm{DL}} = 0.035$ and $\rho = 20$.

Figure 4.1.16 shows the reconstruction results on rhynie chert fossil sample. The slice is 2048×2048 pixels, and 2000 projections were used. This situation is almost the opposite of the previous case: the rings artefacts have a small intensity in the reconstructed slice, and the sample borders form a nearly circular polygonal shape. These border have a huge amplitude with respect to the rest of the sample, and the transition between the border and the interior/exterior is very sharp. Thus, slice correction techniques would try to remove the borders before any other feature in the slice depending on the thresholding parameters.

The rings correction was difficult for the total variation reconstruction: the procedure

added rings tangent to one of the slice borders. It turned out that the problem was due to the rotation center for (back)projection improperly set, leading to accumulating errors in the iterative reconstruction. Indeed, total variation and dictionary learning reconstruction require to compute the projection for the functional, and the back-projection for the functional gradient. If the rotation center for these operations is not the same that the one used for actually rotate the sample, slight errors appear in the (back)projection ; these error accumulate with the number of iterations and take the form of circular features (Figure 4.1.16.c).

After setting the correct rotation center, we were able to remove the ring artefacts (Figure 4.1.16.d), especially the one near the center of Figure 4.1.16.a. In this case, the RCP technique did quite well (Figure 4.1.16.b).

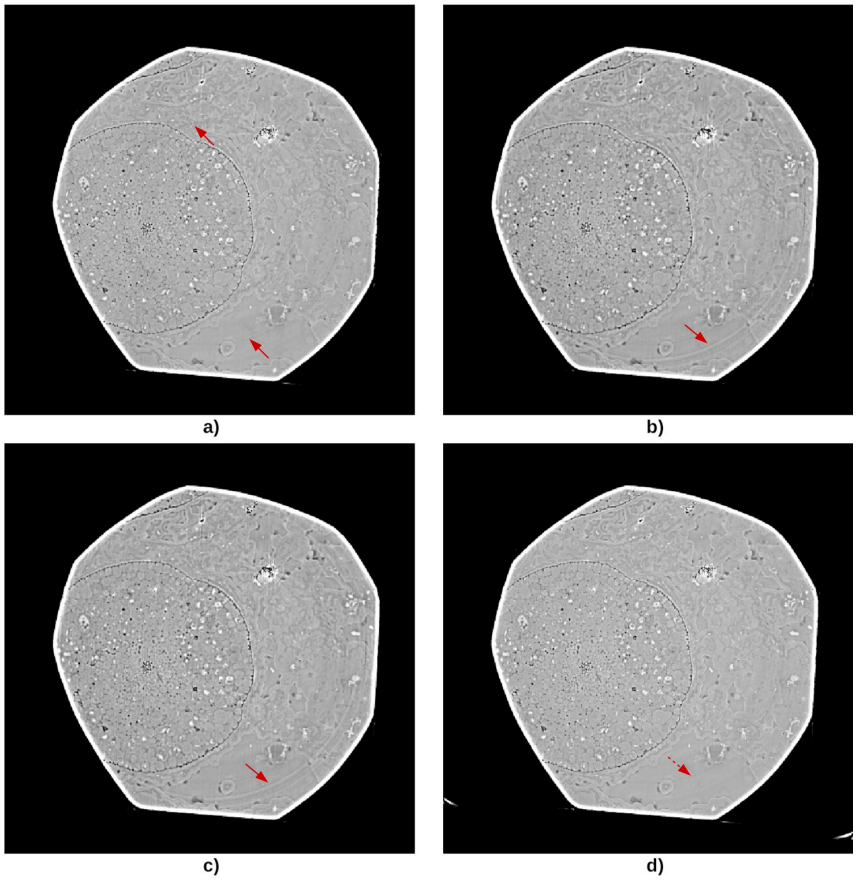
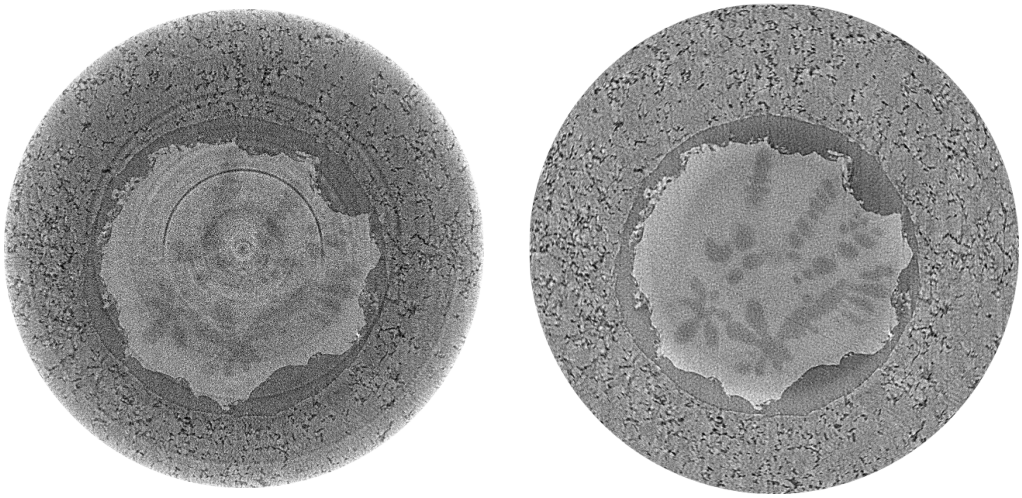


Figure 4.1.16: Rhynie chert fossil sample (credits: Paul Tafforeau, ESRF ID19). (a) Filtered back-projection with the correct rotation axis. (b) Correction with the RCP technique, using the parameters $W = 10$ and $\theta_0 = 10$. (c) Reconstruction with the Total Variation regularization using the incorrect rotation axis. (d) Reconstruction with the Total Variation regularization using the correct rotation axis. The parameters were $\beta = 3 \cdot 10^{-3}$ and $\beta_r = 3 \cdot 10^{-4}$.

Figure 4.1.17 shows the reconstruction results on dendritic crystals. The slice is 1265×1265, scanned with 360 projections.



(a) (b)

Figure 4.1.17: Reconstruction and rings artefacts correction on a dendritic crystals scan (data courtesy: Daniil Kazantsev, University of Manchester/Diamond I13). (a) FBP. (b) Reconstruction with TV regularization and rings artefacts correction. The TV regularization enhanced the contrast, and the rings were corrected.

4.1.8 Execution time and convergence rate

In this section we measure the execution time required to obtain an acceptable reconstruction. All the tests are performed on a machine with an `Intel Xeon CPU E5-1607 v2 @ 3.00GHz` processor and a `GeForce GTX 750 Ti` graphic card.

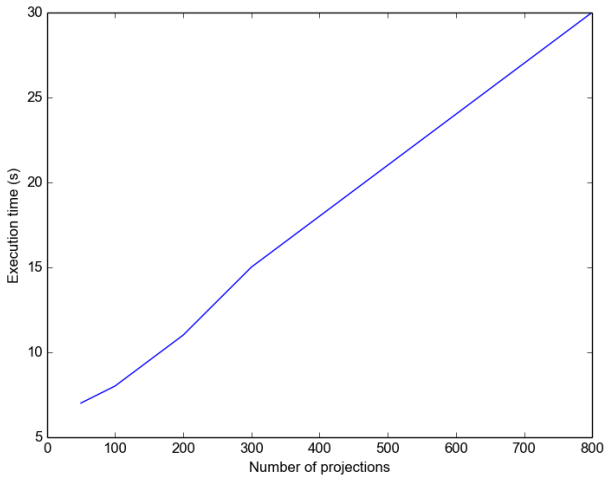
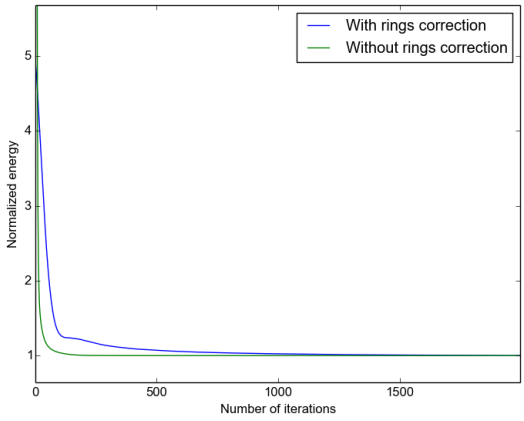


Figure 4.1.18: Execution time as a function of the number of projections for 1000 iterations. The image used is the 512×512 test image "Lena" corrupted with the rings presented in the second test case.

We measured that the execution time is the same with rings correction and without rings correction: including the ring artefacts correction in the functional has no additional cost in the reconstruction. The execution time is proportional to the number of projections,

as it can be guessed with Figure 4.1.18, since more data has to be processed by the operators.

The values of the objective function as a function of the number of iterations is an illustration of the convergence rate. For Total Variation reconstruction, the objective function is given by (4.1.2), it includes both the fidelity term (Euclidean distance) and the regularization term (L1 norm of the image gradient). For Dictionary Learning, it is given by (4.1.7).



(a)

Figure 4.1.19: Energy as a function of the number of iterations for the Total Variation tomographic reconstruction. The energy is normalized by the energy of the last iteration in order to have the same scale in the two cases. The image used is the 512×512 test image "Lena" corrupted with the rings presented in the second test case. Evolution of energy with 800 projections.

Figure 4.1.19 shows the evolution of the objective function (4.1.2) as a function of the number of iterations. With rings correction, the reconstruction process needs more iterations to converge. For simulated and real data, it turned out that a satisfactory reconstruction can be achieved with less than 1000 iterations without rings correction. When the rings correction is activated, it takes about 2000 iteration to correctly remove the ring artefacts. Thus, while the rings correction has no additional cost per iteration, it takes nevertheless more iterations to converge to an image with removed ring artefacts. The "energy transfer" between the fidelity term $\|\boldsymbol{y} - (P\boldsymbol{x} + \boldsymbol{r})\|_2^2$ and the L1 norm of the rings $\|\boldsymbol{r}\|_1$ is actually quite slow.

The reconstruction parameters like the total variation penalization and rings correction weight depend on the data. For most data in parallel geometry, the same parameters can be used for all the slices. Thus, one single slice can be used for the parameters optimization. The parameters are chosen manually to have a good reconstruction quality. An approach towards automatic parameters optimization might be the L-curve method [HO93], which is not used here.

4.1.9 Conclusion

A new way to correct the rings artefacts has been proposed. This techniques does fit well in the scope of iterative regularized reconstruction, since it is especially adapted when the number of projection is limited. Including the rings artefacts correction in the

iterative reconstruction process has shown to be efficient while requiring no extra pre or post-processing steps. Besides, additional artefacts are less likely to appear thanks to the regularization. This method can be adapted to any regularized approach, since the only things to do are modifying the functional and the iterative correction step accordingly.

Although a simultaneous reconstruction and rings correction is appealing, there are some limitations: first, the method is less powerful for very large rings artefacts (in this case, the Fourier-Wavelet method yields better results), and the convergence rate is slower when the rings correction is enabled. This slower convergence speed motivated the design of the conjugate subgradient algorithm (see section 3.3).

4.2 Local tomography

In this section, we propose an interior tomography reconstruction method based on a known subregion, compatible with a high efficient implementation. This method iteratively refines a reconstruction, aiming at reducing the local tomography artefacts To cope with the ever increasing data volumes, this method is highly optimized on two aspects: firstly, the problem is reformulated to reduce the number of variables, and secondly, the operators involved in the optimization algorithms are efficiently implemented. Results show that 4096^2 slices can be processed in tens of seconds, while being beyond the reach of equivalent exact local tomography method. The method and implementation used in this work was published in [PDM17] and [PM17].

4.2.1 Introduction

Region of Interest (ROI) tomography, also called local tomography, naturally arises when imaging objects that are too large for the detector field of view (FOV), as depicted on Figure 4.2.1. It notably occurs in medical imaging, where only a small part of a body is imaged for radiation dose concerns.

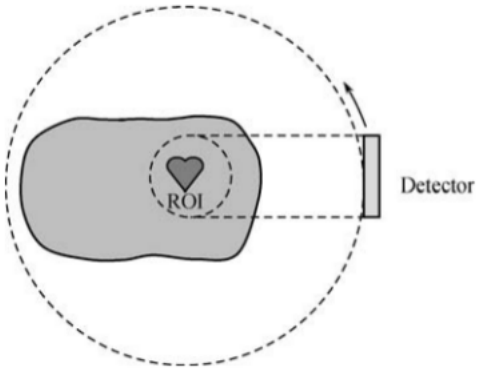


Figure 4.2.1: Local tomography setup when the detector covers only a ROI of the object. Image: [Zen10]

Since the projection data does not cover the entire object, it is said to be *truncated* with respect to a scan that would cover the entire object. The aim is then to reconstruct the ROI from this "truncated" data.

However, due to the nature of the tomography acquisition, the acquired data is not sufficient to reconstruct the ROI in general: for each angle, rays go through the entire object, not only the ROI. Thus, the data does not only contain information on the ROI, but also contribution from the parts of the object external to the ROI. For example, on Figure 4.2.1, the detector gets data from parts of the object located at the left of the ROI. These contributions from the external parts actually preclude from reconstructing exactly the ROI from the acquired data in general.

The problem of reconstructing the interior of an object from truncated data is referred as the *interior problem*. It is well known that the interior problem does not have a unique solution in general. If $\boldsymbol{P}$ denotes the projection operator, $\boldsymbol{d}$ the acquired data and $\boldsymbol{x}$ a solution of the problem $\boldsymbol{P}(\boldsymbol{x}) = \boldsymbol{d}$, then $\boldsymbol{x}$ is defined up to a set of *ambiguity functions* $\boldsymbol{u}$ such that $\boldsymbol{P}(\boldsymbol{x} + \boldsymbol{u}) = \boldsymbol{d}$. An example is given in [CD10] where $\boldsymbol{u}$ is non-zero in the ROI, but $\boldsymbol{P}(\boldsymbol{u}) = \boldsymbol{0}$ in the detector zone corresponding to the ROI: two solutions differing by $\boldsymbol{u}$

would produce the identical interior data. In [WY13], it is emphasized that the ambiguity is an *infinitely differentiable* function whose variation increases when going outside the ROI. The non-uniqueness of the solution of the interior problem prevents quantitative analysis of the reconstructed slices.

Methods tackling the ROI tomography problem can mainly be classified in two categories. The first category methods aim at completing the data by extrapolating the sinogram. These are often oriented toward easy and practical use, although having no theoretical guarantees. The second category of methods rely on prior knowledge on the object. Many theoretical efforts were made on these methods, providing for example uniqueness and stability results.

Other works use wavelets to localize the Radon transform [Ras+97] [SD05] or focus on the detection of discontinuities, the best known being probably Lambda-tomography [YW06]. An extensive work in local methods for reconstructing discontinuities of an object was conducted in [Bil07].

4.2.2 Sinogram extrapolation methods

In a classical tomography acquisition, the whole object is imaged. If nothing is surrounding the object, the rays are not attenuated by the exterior of the object; thus the sinogram values for each angle go to zero on the left and right parts (after taking the negative logarithm of the normalized intensity, and ignoring the various imperfections). In a local tomography acquisition, however, the data is "truncated" with respect to what would have been a whole scan. The incompleteness of the data induces artefacts on the reconstructed image. The first obvious artefact is visible as a bright rim on the exterior of the image. This bright rim is the result of the abrupt transition in the truncated sinogram: the filtration process suffers from a Gibbs phenomenon. Another artefact is referred as the *cupping effect*: an unwanted background appears in the reconstructed image, which makes further analysis like segmentation challenging. These two artefacts occur simultaneously, but they have different causes. The bright rim comes from the truncation, while the cupping comes from the contribution of the external part.

Figure 4.2.2 and 4.2.3 illustrates these artefacts. A synthetic slice is projected, and the resulting sinogram is truncated to simulate a ROI tomography setup. The filtering step enhances the transition between the ROI and the truncated part which is set to zero. The difference between the filtered whole sinogram and the filtered truncated sinogram also shows the cupping effect, which appears as a low-frequency bias.

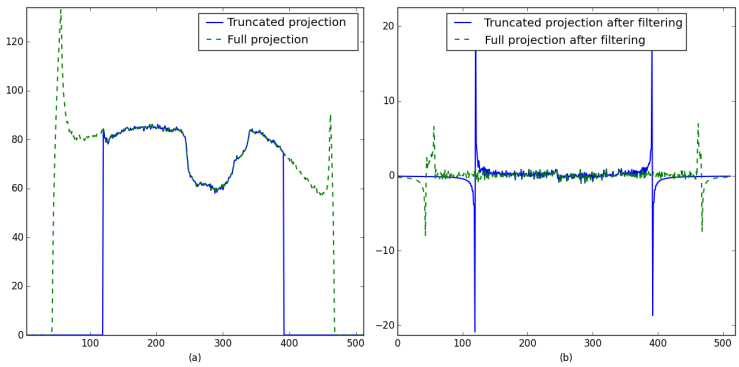
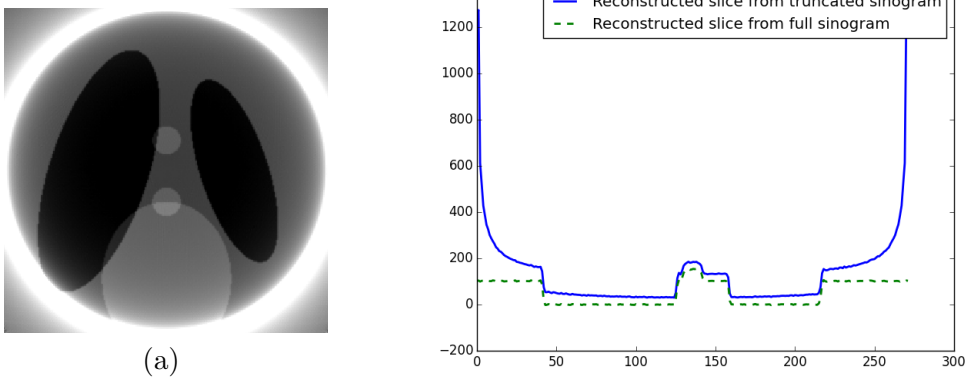


Figure 4.2.2: Illustration of the truncation artefacts on a line of the sinogram of the *Shepp-Logan* phantom. (a): Whole sinogram corresponding to a scan where all the object is imaged (green), and truncated sinogram (blue). (b): After the ramp-filtering.



(a) (b)

Figure 4.2.3: (a): Reconstruction of the truncated sinogram with filtered back-projection. The contrast has been modified to visualize the interior of the slice. (b): Line profile of the reconstruction

Sinogram extrapolation methods primarily aim at eliminating the bright rim resulting from the truncation by ensuring a smooth transition between the ROI and the external part. Besides, efforts have been put into the estimation of the missing data in order to reduce the cupping effect. These techniques are referred as sinogram extrapolation methods: the external part is estimated from the truncated data with some extrapolating function.

Extrapolating function can be for example constant (the outermost left/right values are replicated), polynomial, $\cos^2$. In [SY11], a mixture of exponential and quadratic functions are used to estimate the external part, possibly iteratively. Projection of a circle, for which a closed-form formula is known, can also be used [Van09]. A common approach is using the values of the left/right part of the sinogram to estimate the external part, that is, replicating the borders values.

In general, sinogram extrapolation methods do not take into account the sinogram theoretical properties. For example, given an object being nonzero only inside a circle of a given radius, the sinogram decreases to zero at the left and right boundaries. Generally speaking, a sinogram of complete measurements satisfies the Helgason-Ludwig consistency conditions (4.2.1) [Van09]:

$$H_n(\theta) = \int_{-\infty}^{\infty} s^n p(\theta, s)\, \mathrm{d}s \tag{4.2.1}$$

is a homogeneous polynomial of degree n in $\sin\theta$ and $\cos\theta$, for all $n \geq 0$. An alternative formulation is given by equation (4.2.2) :

$$H_{n,k}(\theta) = \int_0^{\pi} \int_{-\infty}^{\infty} s^n e^{jk\theta} p(\theta, s)\, \mathrm{d}s\, \mathrm{d}\theta \;=\; 0 \tag{4.2.2}$$

for $k > n \geq 0$ and $k - n$ even. In [VDV06], (4.2.2) is used as a quantitative measure of the sinogram consistency, and is optimized as an objective function. Figure 4.2.4 illustrates the Helgason-Ludwig order zero condition, i.e the sum of a sinogram along the detector bins (columns) should be independent of the projection angle.

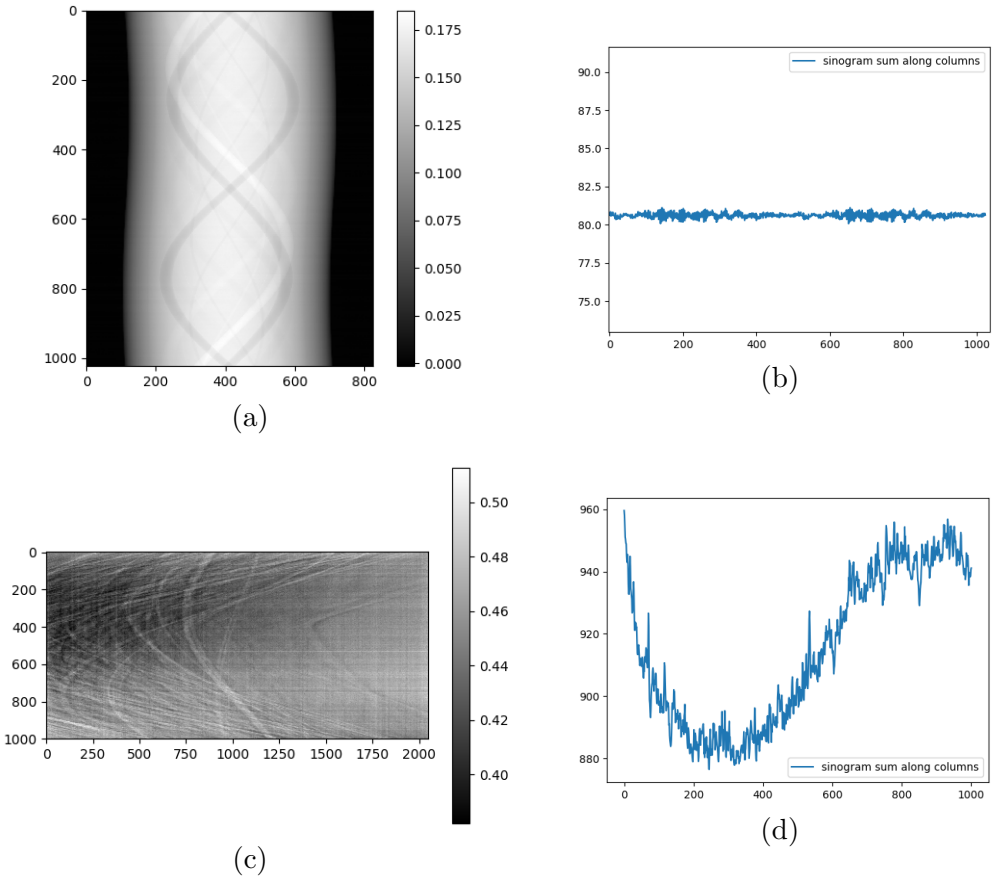


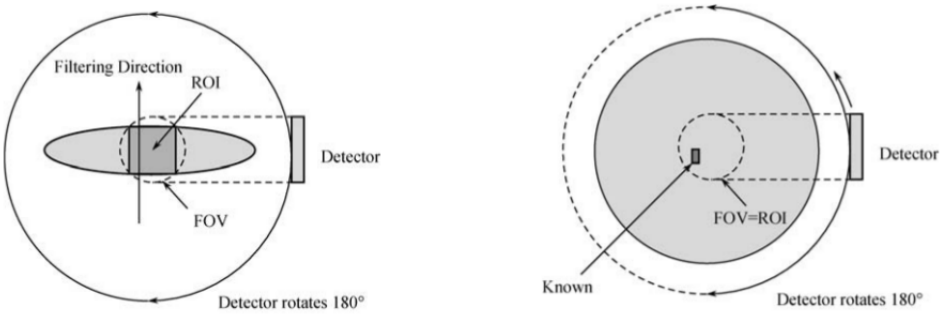
Figure 4.2.4: Comparison of sinograms sums along columns, in nonlocal and local settings. (a) Sinogram resulting of the scan of a Catphan 600 medical phantom, non-local setting (courtesy: ID17) (b) Sum of along the columns (c) Sinogram corresponding to the slice on Figure 3.2.22, local setting (d) Sum along the columns

For many applications, constant extrapolation provides acceptable results [Kyr+11], although cupping artefact makes the segmentation challenging.

4.2.3 Prior knowledge based interior tomography

It was long believed that ROI tomography cannot be solved exactly, because of the nature of Radon inversion through FBP: the reconstruction of each voxel requires the knowledge of all the (complete) lines passing through this voxel. However, in the last decade, it has been shown that multiple nonequivalent reconstruction formulas allow partial reconstruction from partial data in the 2D case [CD10]. Alternatively to Filtered Back Projection reconstruction, which requires complete data, Virtual Fan Beam (VFB) and Differentiated Back-Projection (DBP) were developed based on the Hilbert projection equality [Cla+09].

Moreover, uniqueness theorems based on analytical continuation of the Hilbert Transform were stated and progressively refined in [Noo+02], [Cla+04], [NCP04], [ZPS05], [Def+06], [Ye+07], [Kud+08], [TYT12]. They ensure an exact and stable reconstruction of the ROI given some assumptions. These assumptions can be of geometric nature, or in the form of a prior knowledge.



(a) (b)

Figure 4.2.5: (a): Setup where the DBP can reconstruct the ROI. As the scanner field of view extends the ROI on both sides, the finite inverse Hilbert Transform can be computed. (b): Setup of interior tomography when the FOV does not extend the object. Only the knowledge of a sub-region can provide an exact reconstruction. Images: [Zen10]

Geometry-based prior knowledge is related to the acquisition geometry. For example, in DBP based reconstruction, a point can be reconstructed if it lies on a line segment extending outside the object on both sides, and all lines crossing the segment are measured [CD10], as shown in Figure 4.2.5 (a). Similar results were obtained under less restrictive assumptions, for example the field of view extending the ROI on only one side [Ye+07].

These geometry-based methods do not work, however, when the FOV does not extend the object (Figure 4.2.5 (b)). In this case, it has been shown [Cou+08] that a prior knowledge on the function inside the ROI enables exact and stable reconstruction of the ROI. This knowledge can be in the form of the function values inside a sub-region of the ROI [Kud+08] or can be about the properties of the function to reconstruct, for example sparsity in some domain.

This latest kind of knowledge has led to compressive sensing based ROI tomography. In [Yan+10], [YW09], [Lee+15], Total Variation method is used to reconstruct the ROI. In [NSK07], the function is assumed to be sparse in the wavelet domain, and a multi-resolution scheme reduces the number of unknown by keeping only fine-scale wavelet coefficients inside the ROI. In [KQR15], it is shown that piecewise constant functions are determined *everywhere* by their ROI data, the underlying hypothesis being formulated as sparsity in the Haar domain.

4.2.4 Local tomography and artefacts

The most common local tomography reconstruction method is extrapolating the sinogram before computing the filtered backprojection (FBP), hereafter denoted *padded FBP*. The extrapolation is usually done by replicating the sinogram boundary values. This prevents truncation artefact (Gibbs phenomenon) from occurring, and often provides acceptable results [Kyr+11].

However, this technique can fail when the ROI is surrounded by anisotropic and/or strongly absorbing material, or when the reconstruction has intrinsically low contrast (for example different parts with the same linear absorption coefficient).

The notable local tomography artefact is the cupping effect. On a reconstructed image, local tomography artefacts appear as a varying contrast. The gray values are typically higher far from the center than close to the center, forming a “cup”. The cupping is also visible when plotting an image line passing through the center, as a function of the pixel

location. Such lines are hereby called *profiles*; for example, the vertical line profile is the vertical line of the image passing through the center.

Figure 4.2.6 shows the Shepp-Logan phantom with a region of interest. Figure 4.2.7 shows the reconstruction with padded FBP and Figure 4.2.8 shows line profile of the reconstruction. The cupping effect is clearly visible in both the reconstruction image and profile. This cupping effect can be detrimental for the post-reconstruction analysis, for example segmentation.

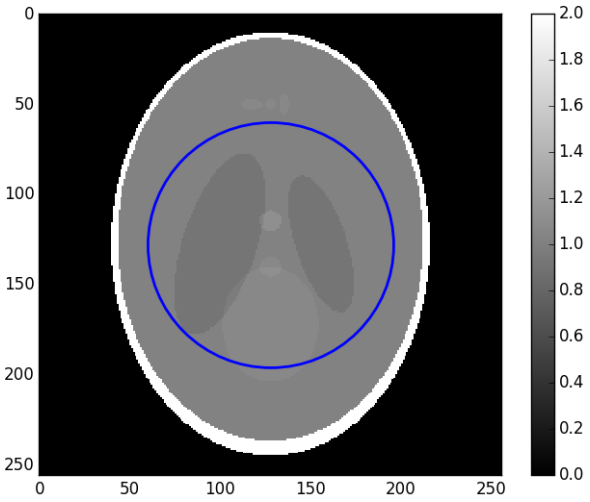


Figure 4.2.6: Shepp-Logan phantom, 256^2 pixels. The right bar indicates the gray values, which for real data can be the linear attenuation coefficient values. The blue circle is the region of interest covered by the detector field of view.

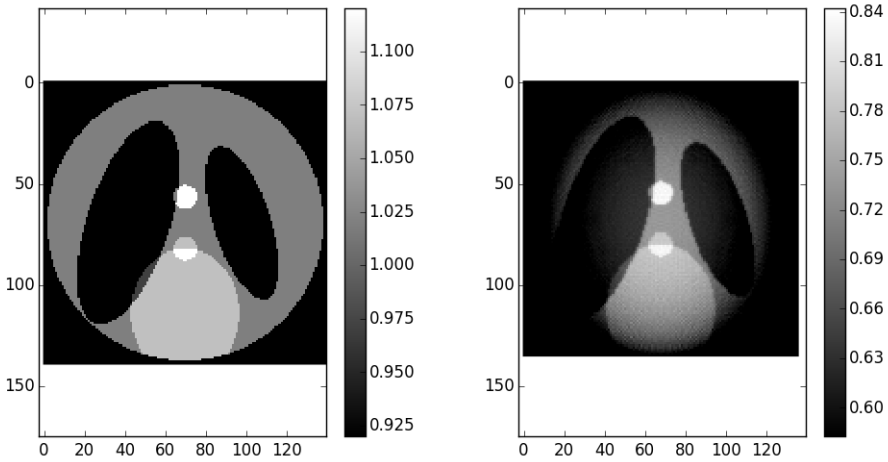


Figure 4.2.7: Zoom on the region of interest defined by the blue circle on Figure 4.2.6. The support is 136^2 pixels. Left: ground-truth zoom. Right: reconstruction with padded FBP. The image is brighter close to the center than far from the center, which is characteristic of the cupping effect. Contrast was adapted with respect to the center of the images.

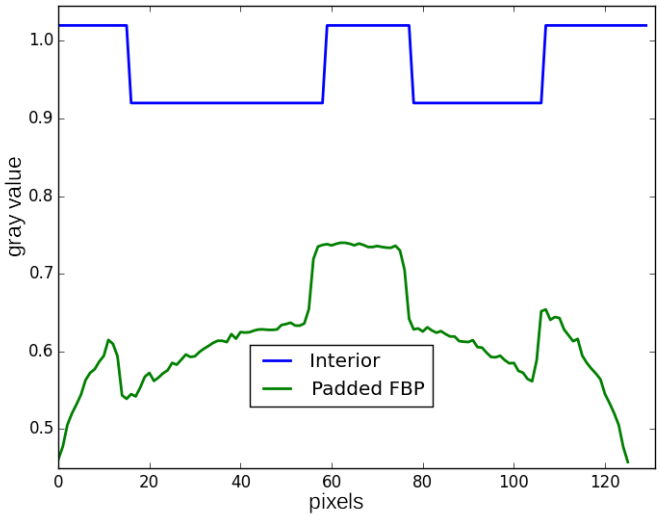


Figure 4.2.8: Horizontal line profile in the region of interest. Blue: ground truth. Green: padded FBP reconstruction. The error on the padded FBP reconstruction appears as a low frequency mean bias.

In this work, we examine a family of exact reconstruction methods based on a known subregion. We implement a method handling a reduced number of unknowns by expressing the image in a coarse basis in order to correct the cupping effect.

4.2.5 Standard iterative reconstruction

Iterative methods in tomography are based on optimization algorithms solving problem (4.2.3)

$$\boldsymbol{P}\boldsymbol{x} = \boldsymbol{d} \tag{4.2.3}$$

with the usual notations. In this context, the reconstructed slice $\boldsymbol{x}$ is an image of support $N \times N = N^2$, where N is the number of pixels of the detector horizontally. The sinogram $\boldsymbol{d}$ support is $N \times N_p$, where N_p is the number of projections. Thus, the projector is theoretically an operator of dimensions $(N \times N_p,\, N^2)$, assuming that slices are stacked as one dimensional N^2 vectors, and sinograms are stacked as one dimensional $N \times N_p$ vectors.

As (4.2.3) is ill-posed in general, a surrogate problem is solved instead, for example problem (4.2.4).

$$\underset{\boldsymbol{x}}{\operatorname{argmin}} \left\{ \|\boldsymbol{P}\boldsymbol{x} - \boldsymbol{d}\|_2^2 + \phi(\boldsymbol{x}) \right\} \tag{4.2.4}$$

where $\|\cdot\|_2^2$ is the squared Frobenius norm and $\phi(\boldsymbol{x})$ is a function bringing stability to the solution. In local tomography, problem (4.2.3) is even more ill-posed due to the incompleteness of the data $\boldsymbol{d}$, as explained in the introductory part. In order for the solution to be acceptable, the exterior of the ROI has to be estimated. This can be done by extending the support of $\boldsymbol{x}$ to iteratively estimate the exterior by solving (4.2.5)

$$\underset{\tilde{\boldsymbol{x}}}{\operatorname{argmin}} \left\{ \left\| \boldsymbol{C}\tilde{\boldsymbol{P}}\tilde{\boldsymbol{x}} - \boldsymbol{d} \right\|_2^2 + \phi(\tilde{\boldsymbol{x}}) \right\} \tag{4.2.5}$$

where $\tilde{\boldsymbol{x}}$ is an image with extended support $N_2 \times N_2 = N_2^2$, where $N_2 > N$, and $\tilde{\boldsymbol{P}}$ is a wider projector adapted to this new geometry. To compute the data fidelity term (here the Euclidean distance between $\tilde{\boldsymbol{P}}\tilde{\boldsymbol{x}}$ and $\boldsymbol{d}$), the size of the projected solution has to be consistent with the acquired data. Thus, the projection is cropped by the means of an operator $\boldsymbol{C}$ to recover the original local geometry. The cropping operator $\boldsymbol{C}$ maps an extended sinogram of support $N_2 \times N_p$ to a sinogram of support $N \times N_p$, by keeping only the N central columns. This models the truncation in the local tomography setup, where the detector is not large enough to image the entire object support N_2. In practice, the cropping operation is implemented inside the projector $\tilde{\boldsymbol{P}}$ by simply restricting the projection to the detector limited field of view N. In the formulas, the cropping operator $\boldsymbol{C}$ is explicitly separated from the projector $\tilde{\boldsymbol{P}}$ to highlight the local setup in the forward model.

Efficient implementations of the projection and backprojection operators enable to solve problem (4.2.5). The ASTRA toolbox [Aar+16], for example, has versatile geometry capabilities and built-in algorithms for solving (4.2.5) for $\phi(\boldsymbol{x}) = \boldsymbol{0}$.

In this work, we consider the case where a subregion is known. This prior knowledge on the volume can be used to constrain the sets of solutions. A uniqueness theorem was stated in [Kud+08] along with a reconstruction algorithm based on differentiated backprojection and projection onto convex sets to invert the finite Hilbert transform. This algorithm, however, is difficult to implement; and no implementation is readily available for experiments.

We focus on a simpler approach based on formalism (4.2.5). In this formulation, the prior knowledge can be encoded in several ways. The first is to enforce the values of $\tilde{\boldsymbol{x}}$ in the known region, for example using an indicator function. The second is to add a term penalizing the distance between the values of $\tilde{\boldsymbol{x}}$ in the known region and the actual values. We adopt the latter approach, which was proposed for example in [RK10].

Let Ω denote the domain where the values of the volume are known. It is a subset (possibly a union of subsets) of the image support N^2, and we denote N_Ω its cardinality, that is, the total number of known pixels. Let $\boldsymbol{x}_{|\Omega}$ denote the values of $\boldsymbol{x}$ inside the known region. The prior knowledge is encoded by $\phi(\boldsymbol{x}) = \lambda \left\| \boldsymbol{x}_{|\Omega} - \boldsymbol{u}_0 \right\|_2^2$ where $\boldsymbol{u}_0$ denotes the known values inside Ω and $\lambda \geq 0$ is a parameter weighting the fidelity to the known zone. Both $\boldsymbol{x}_{|\Omega}$ and $\boldsymbol{u}_0$ have N_Ω components.

Figures 4.2.9 and 4.2.10 shows the reconstruction result on the Shepp-Logan phantom with such choice of $\phi(x)$. The cupping effect is almost removed, but the reconstructed slice is also noisy, which is a known effect of least squares minimization on an ill-posed problem when running too many iterations [VRU08]. On the other hand, many iterations are required to reduce the cupping effect.

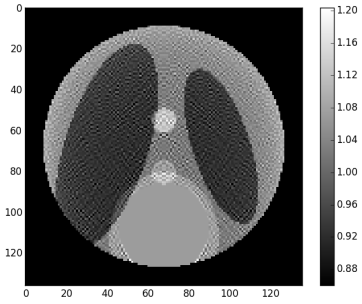


Figure 4.2.9: Result of iterative reconstruction with standard least squares minimization. Contrast was adapted with respect to the center of the image.

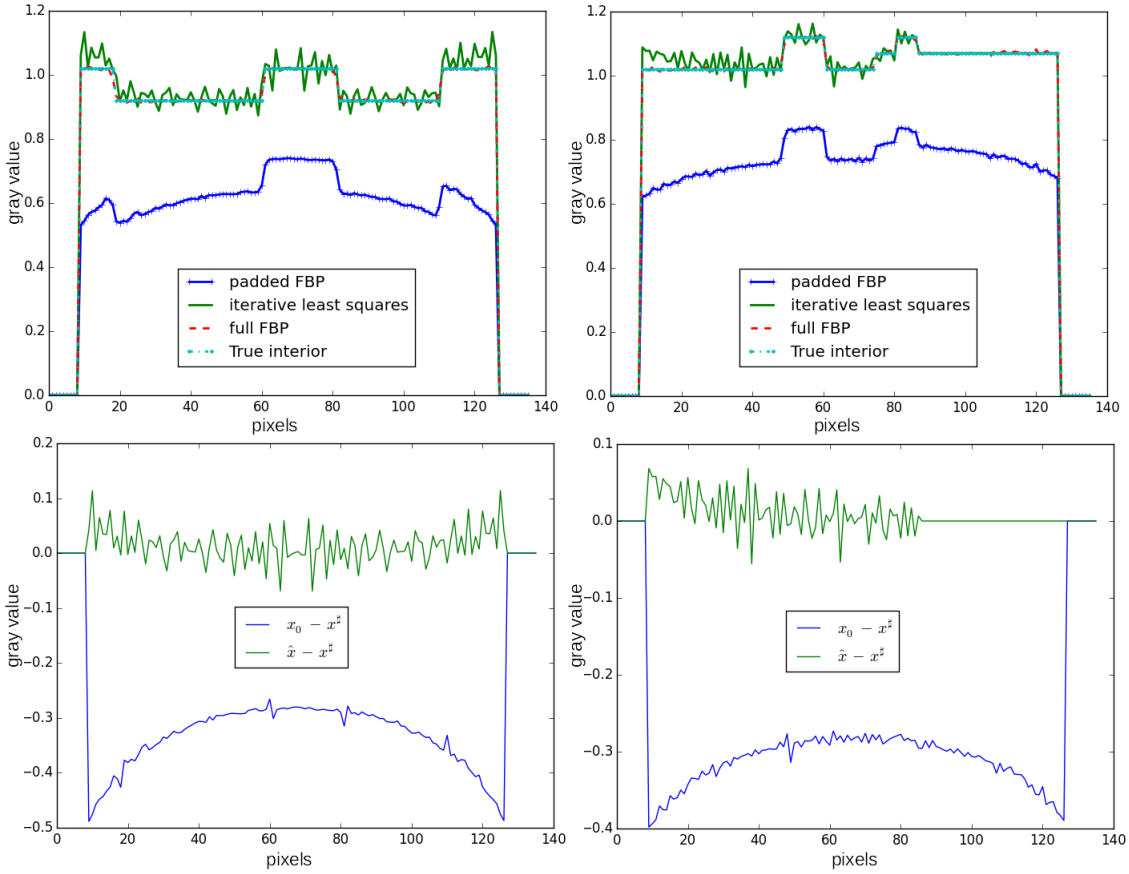


Figure 4.2.10: Line profiles of reconstructions of the Shepp-Logan phantom in a local tomography setup. Top line: reconstruction profiles for padded FBP (blue) and iterative least squares (green). Left: middle line of the reconstructed image. Right: middle column of the reconstructed image. Bottom line: difference profiles between the ground truth $x^\sharp$ and the reconstructions with padded FBP x_0 (blue) and iterative least squares $\hat{x}$ (green). Left: middle line of the difference image, Right: middle column of the difference image. The iterative least squares reconstruction almost removes the cupping effect, but a high frequency noise can be seen in the profiles.

A workaround on this problem is adding a regularization term to stabilize the solution.

In the case of TV regularization, the function $\phi(\boldsymbol{x})$ in (4.2.5) can then be written $\phi(\boldsymbol{x}) = \lambda \left\| \boldsymbol{x}_{|\Omega} - \boldsymbol{u}_0 \right\|_2^2 + \beta \left\| \nabla \boldsymbol{x} \right\|_1$ where $\beta \geq 0$ weights the regularization. Figures 4.2.11 and 4.2.12 show the result of reconstruction with this method. The reconstruction is much more accurate and bears almost no difference with respect to the ground truth, which is an illustration of the uniqueness theorem stated in [Kud+08].

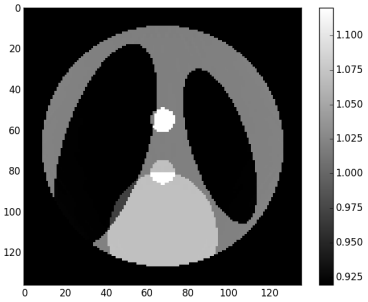


Figure 4.2.11: Result of iterative reconstruction with total variation regularization. Contrast was adapted with respect to the center of the image.

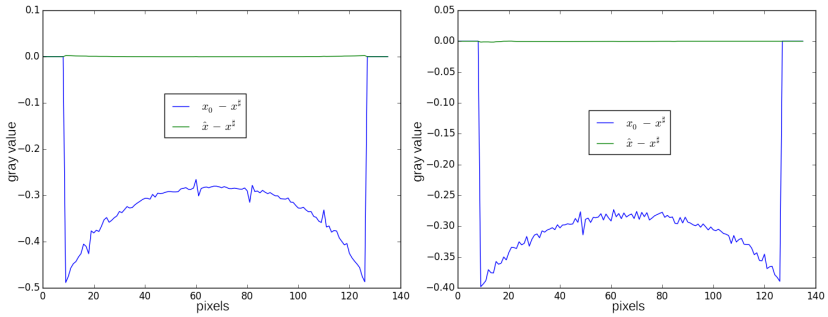


Figure 4.2.12: Line profiles of differences between reconstructions and ground truth. Blue: difference between padded FBP $\boldsymbol{x}_0$ and ground truth $\boldsymbol{x}^\sharp$. Green: difference between iterative total variation minimization $\hat{x}$ and ground truth $\boldsymbol{x}^\sharp$. Left: line profile, Right: column profile. Total Variation minimization removes both high and low frequency errors, exactly recovering the region of interest.

This approach, hereby denoted Standard Iterative Reconstruction (SIR), has two drawbacks. The first is using a prior which might not be accurate: in this example, Total Variation promotes piecewise-constant images and is thus not adapted for complex samples. The second drawback is on the computational side. Adding a non-differentiable prior involves to change the optimization algorithm for another probably less efficient in the sense that more iterations are required to reach convergence. In the examples, the preconditioned Chambolle-Pock algorithm described in [PC11] was used for the TV minimization. Approximatively 3 000 iterations are required to approximately get rid of the cupping effect (when approximatively 500 are required in the case of a complete scan), and more than 10 000 iterations are required to get the line profiles shown on Figure 4.2.12. This approach is impracticable for modern datasets with increasing amount of data: on the one hand, projection and backprojection become costly operations, on the other hand even more iterations are required due to the higher number of variables.

4.2.6 An iterative correction algorithm for local tomography

In the proposed method, we also use an iterative algorithm, but only for *correction*, not for *reconstruction*. Based on the observation that FBP with extrapolation provides satisfactory reconstruction of medium and high frequencies of the slice, the proposed method aims at improving the reconstructed slice by removing the local tomography artefacts visible as low frequencies (cupping effect). This correction is performed by representing the reconstruction error in a coarse basis, reducing the number of degrees of freedom of the problem.

Estimating the reconstruction error

Let $\boldsymbol{x}_0$ be a reconstruction of the region of interest with the padded FBP technique, and $\boldsymbol{x}^\sharp$ be the true values of the region of interest. Both are slices of support N^2 pixels. The reconstruction error, unknown in practice, is denoted $\boldsymbol{e} = \boldsymbol{x}^\sharp - \boldsymbol{x}_0$. This error mainly consists in low frequency artefacts (the cupping effect).

The proposed method aims at correcting the low frequencies artefacts by representing them in a coarse basis, in order to decrease the number of degrees of freedom. The forward model is

$$\underset{\boldsymbol{x}_e}{\operatorname{argmin}} \left\{ \left\| \boldsymbol{C}\tilde{\boldsymbol{P}}(\tilde{\boldsymbol{x}_0} + \boldsymbol{x}_e) - \boldsymbol{d} \right\|_2^2 + \phi(\boldsymbol{x}_e) \right\} \tag{4.2.6}$$

where $\boldsymbol{x}_e$ is a correction term added to the initial reconstruction. Here again, $\tilde{\boldsymbol{x}_0}$ denotes an extension of the support of $\boldsymbol{x}_0$, $\tilde{\boldsymbol{P}}$ is a projection operator adapted to this extended geometry, and $\boldsymbol{C}$ is a truncation operator. As the initial reconstruction is constant, problem (4.2.6) can be rewritten

$$\underset{\boldsymbol{x}_e}{\operatorname{argmin}} \left\{ \left\| \boldsymbol{C}\tilde{\boldsymbol{P}}\boldsymbol{x}_e - \boldsymbol{f} \right\|_2^2 + \phi(\boldsymbol{x}_e) \right\} \tag{4.2.7}$$

where $\boldsymbol{f} = \boldsymbol{d} - \boldsymbol{C}\tilde{\boldsymbol{P}}\tilde{\boldsymbol{x}_0}$. Problem (4.2.7) can be understood as fitting the (approximate) reconstruction error $\boldsymbol{f}$. As the reconstruction error in the ROI is $\boldsymbol{e} = \boldsymbol{x}^\sharp - \boldsymbol{x}_0$, we can write

$$\begin{aligned} \tilde{\boldsymbol{e}} &= \tilde{\boldsymbol{x}^\sharp} - \tilde{\boldsymbol{x}_0} \\ \tilde{\boldsymbol{P}}\tilde{e} &= \tilde{\boldsymbol{P}}\tilde{\boldsymbol{x}^\sharp} - \tilde{\boldsymbol{P}}\tilde{\boldsymbol{x}_0} \\ \boldsymbol{C}\tilde{\boldsymbol{P}}\tilde{e} &= \boldsymbol{d} - \boldsymbol{P}\boldsymbol{x}_0 \end{aligned} \tag{4.2.8}$$

where $\tilde{\boldsymbol{x}^\sharp}$ denotes the whole volume, so that $\boldsymbol{d} = \boldsymbol{C}\tilde{\boldsymbol{P}}\tilde{\boldsymbol{x}^\sharp}$ models the local tomography acquisition. If x_0 is extended to $\tilde{x_0}$ by inserting zeros, then $\boldsymbol{C}\tilde{\boldsymbol{P}}\tilde{\boldsymbol{x}_0} = \boldsymbol{P}\boldsymbol{x}_0$ as there is no contribution from the external part. However, the quantity of interest is the reconstruction error ($\boldsymbol{e}$) in the ROI, not in the whole volume ($\tilde{\boldsymbol{e}}$). Since the projection of $\boldsymbol{e}$ is different from the cropped projection of $\tilde{\boldsymbol{e}}$, the term $\boldsymbol{d}-\boldsymbol{P}\boldsymbol{x}_0$ only approximates the projection of the reconstruction error in the ROI. This quantity is nevertheless used as an approximation of the projection of the reconstruction error in the ROI. Once the optimal correction term $\hat{\boldsymbol{x}_e}$ is found, the resulting reconstruction is simply computed as $\boldsymbol{x} = \tilde{\boldsymbol{C}}(\tilde{\boldsymbol{x}_0} + \hat{\boldsymbol{x}_e})$ where $\tilde{\boldsymbol{C}}$ is a cropping operator in the image domain, mapping images of support N_2^2 to images of support N^2.

Reducing the degrees of freedom

The principle of the implemented method is to refine an initial solution of the local tomography problem, knowing that middle and high frequency features are usually well

recovered. By focusing on the low frequencies, the complexity of problem (4.2.5) can be reduced by solving a simpler problem. Complexity reduction is achieved by expressing the reconstruction error in a coarse basis.

Gaussian function were chosen as a representation basis, which finds a computational justification detailed in 4.2.7. The reconstruction error e is estimated by $\hat{e}$ as a convolution between a finite discrete Dirac comb and a two dimensional Gaussian function g_σ defined by Equation (4.2.9)

$$g_\sigma(u,v) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{u^2+v^2}{2\sigma^2}\right) \tag{4.2.9}$$

where u, v denote discrete indexes in the image, and $\sigma > 0$ is the standard deviation characterizing the Gaussian function. The estimate of the reconstruction error at location (u_0, v_0), $\hat{e}(u_0, v_0)$, is then given by Equation (4.2.10)

$$\hat{e}(u_0,v_0) = \sum_{u,v} c_{u,v} g_\sigma(u_0 - u\cdot s,\, v_0 - v\cdot s) \tag{4.2.10}$$

where $c_{u,v}$ are coefficients multiplying the Gaussian functions g_σ, and s is the spacing (in pixels) between points of the Dirac comb. The summation in (4.2.10) actually occurs on a finite support. In our implementation, the Gaussian function is truncated at 3σ at each side, so the sum takes place on a $\lfloor 6\sigma+1 \rfloor \times \lfloor 6\sigma+1 \rfloor$ pixels square.

This representation of correction features as Gaussian blobs is actually not a basis in the mathematical sense: some images cannot be represented by a linear combination of Gaussians. However, this representation is very close to a basis for $\sigma \simeq s$ [Bal+02]. In our case, we choose $s = 0.65 \times \sigma$, meaning that there is a significant overlap between the Gaussians. The discrete Gaussian kernel is truncated at 3σ, so its length is $\lceil 6\sigma+1 \rceil$ samples.

Estimation (4.2.10) is done such that projection of $\hat{e}$ has minimal Euclidean distance with $d - Px_0$. Let G denote the operator mapping the coefficients $c_{i,j}$ to the image $\hat{e}$ through convolution formula (4.2.10). The coefficients vector c is estimated by solving Problem (4.2.11)

$$\underset{\boldsymbol{c}}{\operatorname{argmin}} \left\{ \left\| \boldsymbol{C}\tilde{\boldsymbol{P}}\boldsymbol{G}\boldsymbol{c} - \boldsymbol{f} \right\|_2^2 + \phi(\boldsymbol{c}) \right\} \tag{4.2.11}$$

where $\boldsymbol{f} = \boldsymbol{d} - \boldsymbol{P}\boldsymbol{x}_0$ and $\phi(\boldsymbol{c})$ is a constraint function on the coefficients which is detailed later.

Thus, Problem (4.2.11) is solved instead of Problem (4.2.5). In Problem (4.2.11), the unknowns are the coefficients c of the coarse basis. As there are much less coefficients c in the coarse representation than pixels in the extended image support N_2^2, the degrees of freedom is accordingly reduced – in a first approximation, by a factor s^2.

Solving (4.2.11) requires the computation of the operators $\boldsymbol{C}$, $\tilde{\boldsymbol{P}}$, $\boldsymbol{G}$, and possibly their adjoints. The implementation of the crop operator $\boldsymbol{C}$ is straightforward, as it consists in truncating the sinogram to the size of the acquired data. In practice, it consists in modifying the projector $\tilde{\boldsymbol{P}}$ so that the projections are limited to the reduced detector field of view N_2. The operator $\boldsymbol{G}$ can be described as follows. Coefficients $c_{u,v}$ are placed every $s > 0$ pixel on an image of the size of the extended reconstruction $\tilde{\boldsymbol{x}_0}$. This image (a two dimensional Dirac comb in the continuum case) is then convolved by the kernel g_σ. Lastly, an efficient implementation of the projection and backprojection operators is needed to solve (4.2.11). This is discussed in the implementation section.

Adding the known zone constraint

The known zone constraint is simply encoded as a distance between the known zone $\boldsymbol{u}_0$ and the restriction of the estimate to the known zone support, $\boldsymbol{G}\boldsymbol{c}_{|\Omega}$. The final optimization problem is

$$\underset{\boldsymbol{c}}{\operatorname{argmin}} \left\{ \left\| \boldsymbol{C}\tilde{\boldsymbol{P}}\boldsymbol{G}\boldsymbol{c} - \boldsymbol{f} \right\|_2^2 + \beta \left\| \boldsymbol{G}\boldsymbol{c}_{|\Omega} - \boldsymbol{u}_0 \right\|_2^2 \right\} \tag{4.2.12}$$

4.2.7 High-performance implementation

After having reduced the number of degrees of freedom for problem (4.2.6), we describe an efficient implementation of the involved operators based on look-up tables (LUT).

Projection a of Gaussian tiling

The choice of a Gaussian basis for a coarse representation of the correction term is based on a characteristic of the Gaussian kernel: it is both rotationally invariant and separable [KS92]. These two properties provide a computational advantage: the order of projection and convolution can somehow be exchanged.

More precisely, given an image $\boldsymbol{y}$ consisting of the Gaussian coefficients evenly placed with a spacing s, the standard way to compute $\tilde{\boldsymbol{P}}\boldsymbol{G}\boldsymbol{y}$ is first performing the convolution $\boldsymbol{G}\boldsymbol{y}$ defined by (4.2.10) and then projecting with $\tilde{\boldsymbol{P}}$. An equivalent computation, however, can be done by first projecting the image of isolated points $\boldsymbol{y}$, and then convolving each line of the resulting sinogram by a one dimensional Gaussian function. This is illustrated in Figure 4.2.13.

This latter approach has two advantages. First, the two dimensional convolution (or two series of one dimensional convolution in this separable case) is replaced by a series of one dimensional convolutions. Secondly, the projection here only consists in projecting isolated points. This operation can be optimized by designing a *point projector* based on look-up tables.

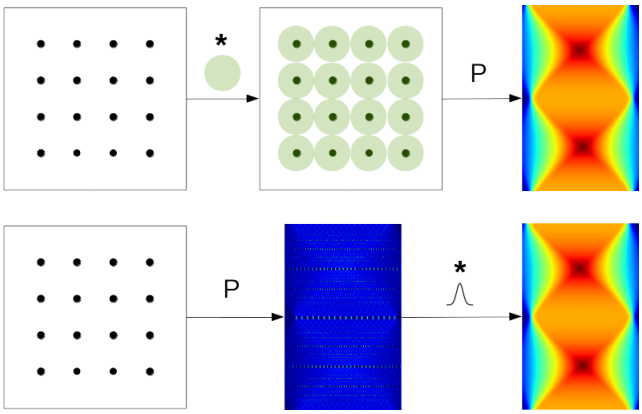


Figure 4.2.13: Illustration of the alternative way of computing the projection of a tiling of Gaussian functions. In the first approach (top line), coefficients are evenly placed on the image support (left). This image is then convolved by the 2D Gaussian kernel (green circles), which gives an intermediate image (center). This image is projected to obtain a sinogram (right). In the second approach (bottom line), isolated coefficients (left) are projected. Each line of the resulting sinogram (middle) is convolved by a 1D Gaussian kernel, to obtain the sinogram (right).

LUT-based point projector

As previously discussed, the operators involved in forward model (4.2.12) are a cropping operator, a one dimensional convolution and a projector. The convolution can be efficiently implemented, either in the Fourier space or in the direct space when one of the functions has a small support. Therefore, a fast projector is essential for solving (4.2.12) in an iterative fashion. In our case, the object to project has a very special structure, as it consists in points spaced by several pixels. Thus, standard projectors of tomography softwares can be replaced by a more efficient implementation, hereby called *point projector*, based on look-up tables.

In the remainder, the notations used are the following. The support of the original image is N^2. The number of projections is N_p, so the acquired sinogram has size $N \times N_p$. The size of the extended image is N_2^2 where $N_2 \geq N$. The number of Gaussian functions used to tile the support is N_g. The spacing between Gaussian blobs on the image is s; thus we have $N_g \simeq \left(\frac{N_2}{s}\right)^2$ in a first approximation. We also use the following indexes convention: Gaussian coefficients are numbered with $i \in [0, N_g[$, and sinogram indexes are numbered with $k \in [0, N_s[$ where $N_s = N_2 \times N_p$ is the size of the (extended) sinogram.

Each Gaussian coefficient number $i \in [0, N_g[$ is projected on (at most) N_p positions in the sinogram. Therefore, a look-up table $\boldsymbol{J}$ is built so that for each i, $J[i]$ is the "list" of locations in sinogram hit by this point after projection. The LUT $\boldsymbol{J}$ is an array of size $N_g \times N_p$. Each entry $J_{i,j}$ corresponds to a position, in the sinogram, that is hit by a projected point $i \in [0, N_g[$. For example, entry $J_{0,2}$ is an index in the sinogram that is hit by point 0; and entry $J_{5,j}$ are an indexes hit by point 5 for all j. This is illustrated by Figures 4.2.14, 4.2.15.

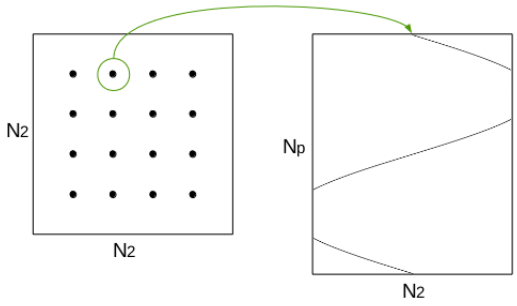


Figure 4.2.14: Principle of the point projector. Gaussian basis coefficients are placed on the image of N_2^2 support (left), with even spacing. Each isolated point is projected on at most N_p positions in the sinogram (right).

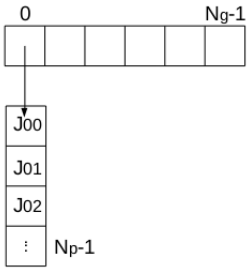


Figure 4.2.15: Illustration of the Look-Up Table $\boldsymbol{J}$. The Gaussian coefficients placed on the image are stored in a vector of size N_g (top). Each coefficient point (indexed in $[0, N_g[$) is projected on at most N_p positions in the sinogram. For each $i \in [0, N_g[$, the structure $J[i] = \{J_{i,0}, J_{i,1}, \ldots\}$ (bottom) contains the list of the sinogram positions hit by projection of i. For example, the Gaussian coefficient number 0 is projected on sinogram positions $J_{0,0}$, $J_{0,1}$ ....

When computing the sinogram, however, the look-up table J is best accessed "backward": for a given position $k \in [0, N_s[$ in the sinogram, we have to determine which points are hitting it through projection. To this end, two look-up tables $\boldsymbol{J}$ and **Pos** are built. For $k \in [0, N_s[$, **Pos**$[k]$ indicates a position in LUT J, and $J[p_k]$ is a coefficient number $i \in [0, N_g[$ being projected at position k. Therefore, the LUT J does not contain sinogram indexes anymore, but rather coefficient indexes. This is illustrated in Figure 4.2.16. The LUT $\boldsymbol{J}$ is re-ordered such that the interval $[p_k, p_{k+1} - 1]$ gives access to an indexes range in J; this index range is the set of all coefficients indexes being projected on sinogram index k.

The point projector is described by Algorithm 4.2.1. The matrix $\boldsymbol{W}$, indexed in the same way as $\boldsymbol{J}$, contains the weights of the projections: depending on the position of a point in the image and the projection angle, its projection does not exactly fall into a sinogram pixel. The matrix $\boldsymbol{W}$ thus encodes the geometric contribution of the projection of the points.

This projection scheme basically consists in storing the explicit projection matrix $\tilde{\boldsymbol{P}}$ with a *Compressed Sparse Row* (CSR) format [Bar+94], where LUT $\boldsymbol{J}$ corresponds to "`col_ind`", LUT **Pos** corresponds to "`row_ptr`" and matrix W contains the values. Storing the entire "linear-algebra" projection matrix without compression would entail to store $(N_2^2) \times (N_2 \cdot N_p)$ elements, which is impracticable (for example more than one terabyte is required for a 1024^2 slice). However, as each slice point is projected on at most N_p sinogram positions, this matrix actually has at most $N_2^2 \times N_p$ nonzero elements. Additionally, as the slice is reduced in a coarse basis, there are $\left(\frac{N_2}{s}\right)^2 \times N_p$ nonzero values to store in this case. The format described above is used to store these elements. Algorithm 4.2.1 is thus no more than a matrix-vector multiplication with a sparse matrix in CSR format.

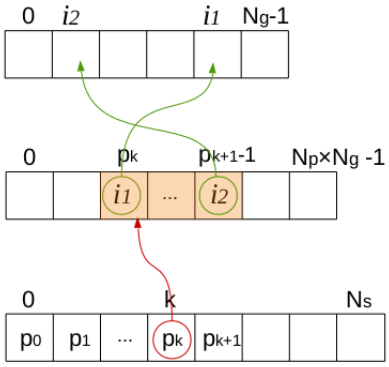


Figure 4.2.16: Illustration of the LUT-based point projector. To determine which points are projected on position $k \in [0, N_s[$ of the sinogram, the matrix Pos (bottom) is accessed at index k, and contains the value $\text{Pos}[k] = p_k$. This value p_k is a position in LUT J (middle), so that $J[p_k] = i_1$ is the index of one coefficient being projected at index k in sinogram. The process is repeated for $p_k + 1$, until $p_{k+1} - 1$. The corresponding range in J (shaded orange) indicates coefficient indexes that all are projected on sinogram index k.

Algorithm 4.2.1 Point projector

sino: sinogram, of size $N_s = N_2 \times N_p$
J: LUT, of size $N_g \times N_p$
coeffs: coefficients vector of the Gaussian basis, of size N_g
W: projection weights, of size $N_g \times N_p$

```
1: procedure POINTPROJECTOR(sino, J, coeffs, W)
2:     for k ∈ [0, N_s − 1] do
3:         pos1 = Pos[k]
4:         pos2 = Pos[k+1]
5:         for j ∈ [pos1, pos2[ do
6:             sino[k] += W[j] * coeffs[J[j]]
7:         end for
8:     end for
9: end procedure
```

This approach for computing the point projector is friendly in a memory-write point of view: after accumulating the contributions of all coefficients projected on position k, the sinogram at index k, sino[k], is updated accordingly. This is especially important for GPU implementation, as consecutive threads access contiguous memory locations, which is a coalesced access pattern. On GPUs, each memory transaction actually entails accessing L bytes, so coalesced access to 32 bits scalars results in a read or write of $L/4$ addresses in a single transaction (for example $L = 128$ for modern NVidia GPUs).

Implementation of the adjoint operators

As a gradient-based optimization algorithm is used for solving (4.2.12), the adjoint of operator $\boldsymbol{C\tilde{P}G}$ has to be computed. This operator $\boldsymbol{G}^T\tilde{\boldsymbol{P}}^T\boldsymbol{C}^T$ consists in extending

the sinogram with zeros, point-backprojecting and retrieving the Gaussian components from the backprojected image. As mentioned above, the operator $\boldsymbol{G}$ can be described as $\boldsymbol{G} = \boldsymbol{H_\sigma U}$ where $\boldsymbol{U}$ is an upsampling operator (here with a factor s), and H_σ is the convolution with 2D Gaussian kernel (4.2.9). Thus, $\boldsymbol{G}^T = \boldsymbol{H_\sigma}^T \boldsymbol{U}^T$ which is a downsampling followed by a convolution with kernel (4.2.9). The actual computation is then $\boldsymbol{G}^T \tilde{\boldsymbol{P}}^T \boldsymbol{C}^T = \boldsymbol{H_\sigma}^T \boldsymbol{U}^T \tilde{\boldsymbol{P}}^T \boldsymbol{C}^T = \boldsymbol{U}^T \tilde{\boldsymbol{P}}^T \boldsymbol{H_\sigma^1} \boldsymbol{C}^T$ where $\boldsymbol{H_\sigma^1}$ is a one dimensional convolution on the sinogram rows.

As previously, these operations can be merged. As $\boldsymbol{G}^T \tilde{\boldsymbol{P}}^T \boldsymbol{C}^T$ returns a Gaussian coefficients vector from a sinogram, only the coefficients are of interest here. Therefore, the point-backprojector $\tilde{\boldsymbol{P}}^T$ is merged with the downsampling $\boldsymbol{U}^T$ as previously. For a given coefficient, we have to find which sinogram entries backproject on the coefficient position. This approach avoids to compute useless $N_g \times (s-1)^2$ backprojections points on the image, as it is downsampled afterwise.

The point backprojector is implemented, as previously, with a LUT J_2 of size $N_g \times N_p$ and a LUT **Pos2** of size $N_g + 1$. The matrix $\mathbf{J_2}$ is re-ordered so that for all $i \in [0, N_g[$, the interval $[\text{Pos2}[i], \text{Pos2}[i+1] - 1]$ corresponds to an index range in LUT $\boldsymbol{J_2}$. This is illustrated in Figure 4.2.17.

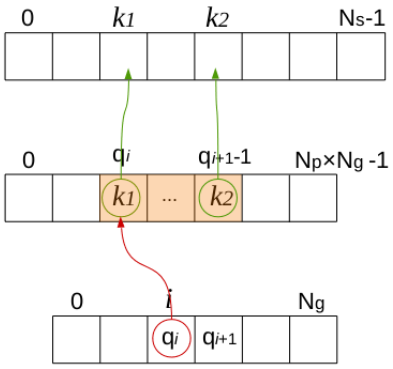


Figure 4.2.17: Illustration of the LUT-based point backprojector. To determine which sinogram points are backprojected on coefficient $i \in [0, N_g[$, the matrix **Pos2** (bottom), is accessed at index i, and contains the value $\text{Pos2}[i] = q_i$. This value q_i is a position in LUT $\boldsymbol{J}_2$ (middle), so that $J_2[q_i] = k_1$ is the index of one sinogram entry being backprojected at index i of the coefficients vector. The process is repeated for $q_i + 1$ until $q_{i+1} - 1$. The corresponding range in $\boldsymbol{J_2}$ (shaded orange) indicates sinogram indexes that are all backprojected on coefficient index i.

The point-backprojector is given by Algorithm 4.2.2. Again, the backprojection from a sinogram to a Gaussian coefficients vector corresponds to a matrix-vector multiplication with a matrix in CSR format. The matrix $\boldsymbol{W}_2$, containing the geometric weights of the backprojector, can be viewed as the Column Sparse Storage (CSC) version of the matrix $\boldsymbol{W}$.

Algorithm 4.2.2 Point backprojector

coeffs: coefficients vector of the Gaussian basis, of size N_g
sino: sinogram, of size $N_s = N_2 \times N_p$
J2: LUT, of size $N_g \times N_p$
W2: backprojection weights, of size $N_g \times N_p$

```
1: procedure POINTBACKPROJECTOR(coeffs, sino, J2, W2)
2:     for i ∈ [0, N_g − 1] do
3:         pos1 = Pos2[i]
4:         pos2 = Pos2[i+1]
5:         for j ∈ [pos1, pos2[ do
6:             coeffs[i] += W2[j] * sino[J2[j]]
7:         end for
8:     end for
9: end procedure
```

Parallel implementation

In modern experiments carried on X-ray light sources, the data volumes, produced by new generations of detectors, always overwhelm the computing power. Simply waiting for more powerful machines is of little hope, as advances in detectors overruns the Moore's law. Instead, an algorithmic work has to be accomplished to exploit parallelism of modern architectures. In the last decade, the advent of general-purpose GPU (GPGPU) computing was advantageously used, especially in tomography.

The proposed method has been implemented in the PyHST2 software [Mir+14] used at ESRF for tomographic reconstruction, with the CUDA language targeting Nvidia GPUs. The point-projector and point-backprojector, which are the most time-consuming operators, are implemented as efficient CUDA kernels. As for Algorithms 4.2.1 and 4.2.2, the CUDA point-projector and point-backprojector are implemented as matrix-vector multiplication with a matrix in CSR format.

We describe here the implementation of the point-projector, i.e the computation of the sinogram values sino$[k]$ for $k \in [0, N_s[$. The point-backprojector follows the same principle. To compute the sinogram value sino$[k]$, the LUT $\boldsymbol{J}$ has to be accessed from p_k to $p_{k+1} - 1$ as illustrated on Figure 4.2.17. This memory range is accessed in parallel by threads of the many-cores GPU with the following principle. Each thread reads $m \geq 1$ values in the LUT. With these values $J[j]$, where $j = p_k, p_k + 1, \ldots$, the coefficients vector is accessed at coeffs$[J[j]]$. The threads are grouped in blocks, and each thread updates a temporary array in shared memory with the contributions read in coeffs$[J[j]]$. Then, in each block, the shared array is accumulated by one thread. The result is added to sino$[k]$. This is illustrated in Figure 4.2.18.

The parallelization is done on the read of matrix J, as it is the biggest data structure of the method. As it has been re-arranged so that the inverval $[\text{pos}[k], \text{pos}[k+1] - 1]$ is a contiguous memory range in J, the described implementation has an efficient memory access pattern.

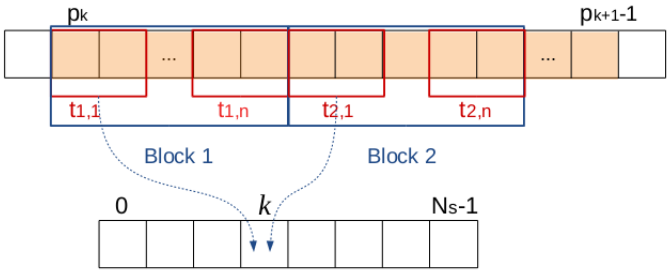


Figure 4.2.18: Illustration of the GPU LUT-based point projector. The memory range $[p_k, p_{k+1}-1]$ in LUT $\boldsymbol{J}$ (top, shaded orange) contains all the indexes needed to be accessed in the coefficients vector to compute sino$[k]$. In this illustration, each thread reads $m = 2$ values in the LUT (red rectangles). The threads are grouped in blocks of n threads (blue rectangles). In the block 1, threads $t_{1,1}, \ldots, t_{1,n}$ update a temporary shared array with their contribution. The same is done in block 2, where another temporary shared array is used. Then, one thread per block accumulates the results of the shared array, and adds the results to sino$[k]$. The addition has to be atomic, as threads from several groups might access sino$[k]$ at the same time.

Multi-resolution Gaussian basis

The correction term $\boldsymbol{x}_e$ in model (4.2.6) is a tiling of Gaussian functions: $\boldsymbol{x}_e = \boldsymbol{G}\boldsymbol{c}$ where $\boldsymbol{c}$ is the vector of coefficients in the Gaussian basis, and $\boldsymbol{G}$ is the operator previously described. In a first approach, all the Gaussian functions (4.2.9) have the same variance σ^2, so that operator $\boldsymbol{G}$ is linear and problem (4.2.12) is convex. The coefficients are placed on a support of size N_2^2 before being (theoretically) convolved with a 2D Gaussian kernel. The spacing between points is s, so that the number of required coefficients is approximately $N_g \simeq \left(\frac{N_2}{s}\right)^2$.

Another approach is using different variances depending on the position in the image. As only the support $N \leq N_2$ of the original reconstruction $\boldsymbol{x}_0$ has to be corrected, Gaussians with a larger support (larger σ) can be used on the exterior of the ROI, further reducing the number of unknowns. By using small Gaussians (small σ) inside the ROI, local features can be estimated in the correction term $\boldsymbol{x}_e$, while large Gaussians are used to roughly estimate the contribution of the exterior of the ROI. The new operator $\boldsymbol{G}$ can be written

$$\boldsymbol{G} = \sum_j H_{\sigma_j} \boldsymbol{U}_j \tag{4.2.13}$$

where $\sigma_1, \sigma_2, \ldots$ is a series of standard deviations for the Gaussians, and $\boldsymbol{U}_j$ are upsampling operators with different factors. This representation is similar to a multi-resolution scheme also used in [NSK07]. This multi-resolution basis allows to further reduce the number of variables in vector $\boldsymbol{c}$: in this case, $N_g < \left(\frac{N_2}{s}\right)^2$. In our implementation, the standard deviations are progressively doubled until reaching the diameter of the ROI, and then remain constant outside the ROI.

Implementation of (4.2.13) is straightforward. The coefficients in vector $\boldsymbol{c}$ are classified according to the distance to the center, forming subsets of coefficients $c^1, c^2, \ldots$. Each subset is point-projected, and line-convolved with the corresponding $\sigma_1, \sigma_2, \ldots$. The resulting sinograms are summed to obtain the projection of $\boldsymbol{G}\boldsymbol{c}$.

Optimization algorithm

Efficient optimization algorithms can be used to solve the quadratic problem (4.2.12). We use the conjugate gradient (CG) algorithm, requiring the computation of the adjoint of the involved operators previously described. CG also entail matrix-vector multiplications, which are efficiently implemented with the CSR representation of point-projector and back-projector.

In the GPU implementation, all the involved arrays are single precision (float 32 bits) as most GPUs are relatively not efficient with 64 bits operations. However, the conjugate gradient algorithm involves scalar products. These operations are implemented by dedicated kernels returning double precision values, as error accumulation is noticeable when accumulating on large arrays in single precision.

4.2.8 Results and discussion

In this section, we discuss the results of the proposed method for local tomography reconstruction. Three test cases are used: the Shell-Logan phantom with low contrast, the Lena image with high contrast but strong absorbing material outside the ROI, and a truncated sinogram of a real dataset scan. Synthetic sinograms are generated by projecting an object and truncating the sinogram to the radius of a given region of interest in the image. A benchmark is also performed for the GPU implementation on the first test case.

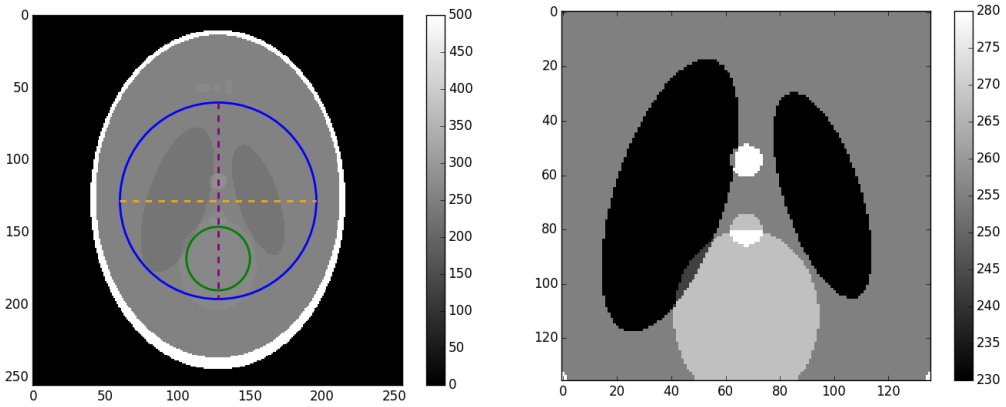
The following notations are used: σ is the standard deviation of the Gaussians of the basis, s is the grid spacing, N_2 is the size (width or height in pixels) of the extended image and R is the radius (in pixels) of the known region. In practice, the size of the "original image" (which corresponds to the size of an image that would contain the whole object in practice) is unknown, hence N_2 is always chosen different from the width of the original test image.

In all cases, the input image is projected with a projector covering the entire object. The resulting synthetic sinogram is then truncated to the radius of the region of interest. The truncated sinogram is the input of the methods. The proposed method is compared to the padded FBP.

First test case

The first test involves the standard Shepp-Logan phantom (Figure 4.2.19), 256×256 pixels. The region of interest is embedded inside the "absorbing outer material" (ellipse with the highest gray values) to simulate a local tomography acquisition. For an easier interpretation of the line profiles in the final reconstructed images, the values of the standard phantom are multiplied by 250 so that all the values are between 0 and 500. The width of the extended for reconstruction image is $N_2 = 260$.

This phantom is the "original" Shepp-Logan phantom, not the "modified Shepp-Logan phantom" where the contrast is improved. In our case, low contrast is important for the tests, as the cupping effect is stronger and directly visible in the reconstructed slice. For high contrast images, the cupping effect only affects few low frequency components, and is thus less detrimental to the reconstruction quality.



(a) (b)

Figure 4.2.19: (a): Shepp-Logan phantom of size 256×256. The outer circle is the region of interest, the inner circle is the known subregion. The dashed lines indicate the profiles which are to be plotted in the reconstructed slice. (b): View of the region of interest (with adapted contrast).

Figure 4.2.20 shows the result of the reconstruction with padded FBP and with the proposed method. The Gaussian coefficients were computed with $\sigma = 4$ on a grid of spacing $s = 6$. The known region radius is $R = 20$ pixels, and the extended image width is $N_2 = 260$ pixels.

By visual inspection, this method do not induce new artifacts in the reconstruction. Figure 4.2.21 shows a line profile of this reconstruction. The cupping effect is visible for the padded FBP, and it has been removed with the proposed method. More importantly, the average reconstructed values are distributed around the true interior values. This provides an illustration of the uniqueness theorem: knowing the values of a subregion of the ROI enables to exactly reconstruct (up to numerical errors) the ROI. The reconstruction with the proposed method bears the same high frequencies as the FBP with full data, which is a good indication that this method do not induce new artifacts. The fact that the reconstruction has the same mean values than the true interior could enable quantitative analysis of the reconstructed volume, which is not easily achievable in local tomography.

Figure 4.2.20 shows the reconstruction results for the setup of Figure 4.2.19. As it can be seen, the cupping effect is mostly removed with respect to the padded FBP technique. Importantly, the proposed method does not create additional artefacts when correcting the cupping effect. Figure 4.2.21 shows line profiles of these reconstructions. The known zone constraint provides a reconstruction with an almost zero mean bias.

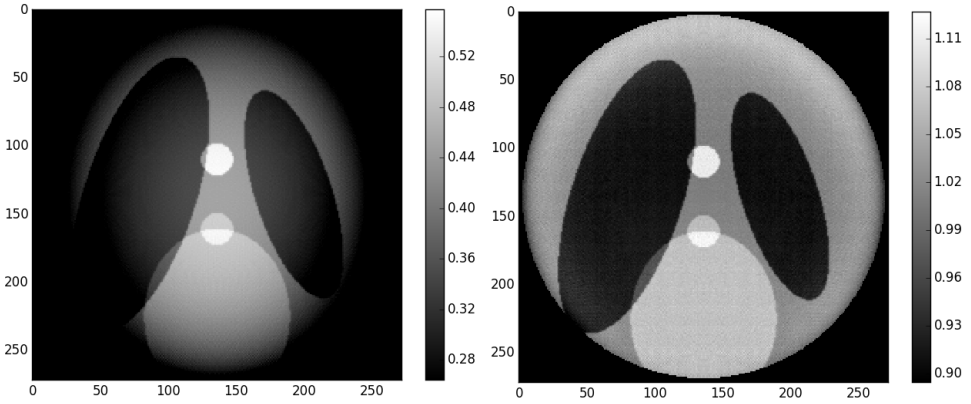


Figure 4.2.20: ROI reconstruction results. Left: padded FBP, right: proposed. For both images, the contrast was adapted with respect to the center.

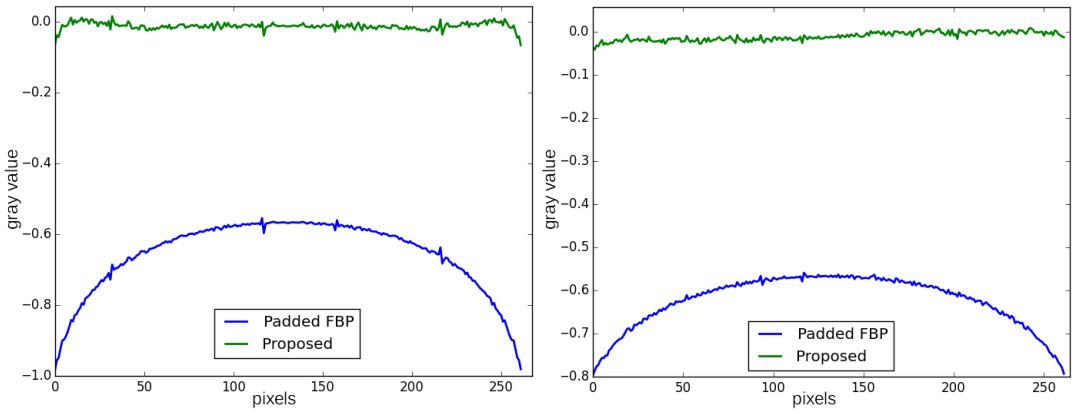


Figure 4.2.21: Line profiles of reconstructions with the proposed method and the padded FBP. The proposed method was executed with $N_g = 1345$ (left) and $N_g = 729$ (right), corresponding to a relatively coarser basis.

The known zone constraint is important to remove the mean bias. As it can be seen on Figures 4.2.22, the cupping effect remains if no constraint is applied. In this case, the uniqueness theorem does not apply when there is no constraint, hence there are no guarantee that the method converge to an acceptable solution.

First test case - benchmark

In this section, we compare the proposed method with the SIR method described in section 4.2.5, in term of speed.

The image size varies in the benchmarks, and the radii of ROI and known zone also vary accordingly. The known zone has be chosen as a uniform zone.

In the following benchmark, the following notations are used. N is the horizontal size of the initial reconstruction, i.e the diameter of the acquired ROI, which means that the acquired sinogram has a size $N \times N_p$. N_0 is the horizontal size of the whole object support, unknown in practice (for example $N_0 = 512$ in the case of the 512^2 Shepp-Logan phantom). N_2 is the horizontal size of the extended reconstruction ($N_2 > N$), which

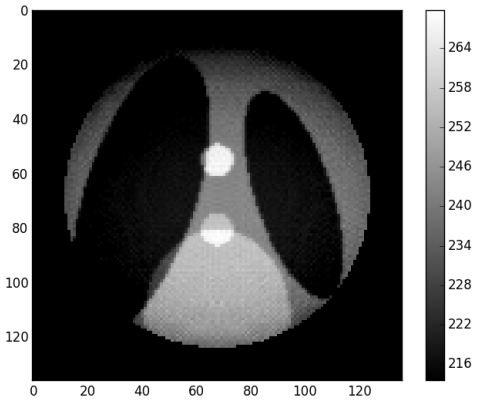


Figure 4.2.22: Reconstruction of the Shepp-Logan phantom without known zone constraint

should approximate N_0. Lastly, N_g is the number of Gaussian functions used for the proposed method.

We describe the problem setup and the settings of the proposed algorithm with a series of "Setting" keywords in Table 4.1:

- Setting 1: $N_0 = 512$, $N = 272$, $N_2 = 572$, $N_g = 1345$, iterations $= 200$
- Setting 1b: Same as Setting 1, except $N_g = 729$
- Setting 2: $N_0 = 1024$, $N = 544$, $N_2 = 1144$, $N_g = 1345$, iterations $= 300$
- Setting 2b: same as Setting 2, except $N_g = 2081$
- Setting 3: $N_0 = 2048$, $N = 1088$, $N_2 = 2288$, $N_g = 1345$, iterations $= 500$
- Setting 3b: same as Setting 3, except $N_g = 805$
- Setting 4: $N_0 = 4096$, $N = 2176$, $N_2 = 4576$, $N_g = 2081$, iterations $= 500$
- Setting 4b: same Setting 3, except $N_g = 1037$

All the test were performed on a machine with a Intel Xeon CPU E5-2643 12 cores 3.40GHz, and a Nvidia Geforce GTX Titan X GPU. As the LUT can be used for all the slices of a volume, the computation of the LUT is not taken into account. The optimizations algorithm used are the preconditioned Chambolle-Pock method [PC11] for pixel domain exact method, and Conjugate Gradient for the proposed method.

We report both the number of iterations needed to converge to the objective function minimum, and the total execution time. This gives information on both the efficiency on the optimization algorithm to converge for the given problem, and on the complexity of each iteration, as the time for one iteration is roughly the total execution time divided by the number of iterations.

Table 4.1 summarizes the results of the two methods for various setups. For each original phantom size, the two methods are tested with two sets of different parameters. For 512^2, 1024^2, 2048^2, 4096^2 original phantom shapes, the number of projections are respectively 800, 1500, 2500 and 4000.

The Python prototype implementation of this method was run with the parameters of Table 4.1. It yields the following execution times: 11.3 seconds for a 512^2 image, 83.1

	Proposed method			SIR method		
Settings	CPU Time	GPU Time	PSNR	Its	Time (s)	PSNR
Setting 1	10.2	2.31	35.5	4000	123	36.79
Setting 1b	5.86	2.01	34.93	3000	106	35.94
Setting 2	36.71	5.14	28.03	4000	523	31.56
Setting 2b	60.7	11.4	30.25	8000	1094	37.85
Setting 3	235	33.5	27.73	4000	3570	15.13
Setting 3b	129	19.2	24.75	7000	6237	20.71
Setting 4	1028	109	24.11	4000	N.A.	N.A.
Setting 4b	870	97.6	22.74	7000	N.A.	N.A.

Table 4.1: Execution time for various local tomography setups. The execution times are in seconds. The "N.A." entries mean that the method was not executed until convergence as it took too much time.

seconds for a 1024^2 image, 842 seconds for a 2048^2 image, and 3630 seconds for a 4096^2 image. Although it is still better than the "pixel domain approach", it suffers from very long execution times for large images. The LUT computation time, which is in the order of several minutes for a 4096×4096 slice, does not appear in Table 4.1 as the same LUT can be reused for the whole volume.

In the example of 512^2 phantom size, the proposed method is executed with an acquired sinogram of width 272 pixels. The slice is extended to 572 pixels, and the Gaussian basis is configured to have 1345 functions in total. 200 iterations yield the reconstruction of Figure 4.2.20 in 10.2 seconds (without taking the LUT computation time). On the other hand, the standard pixel-domain method is executed with 4000 iterations and yields a reconstruction similar to Figure 4.2.11, although of slightly lesser quality, in 123 seconds. The test is then run for a smaller number of Gaussians: the execution time is reduced, but the quality is slightly degraded. This is due to the fact that the number of Gaussians is determined by the spacing s, which itself is linked to the standard deviation σ. Decreasing the number of unknowns (N_g) speeds up the computations, but also increases the width of the Gaussians, so the reconstruction error might not be appropriately fitted.

The proposed method essentially has one parameter: the initial value for σ[4] A good estimate of the extended slice size N_2 can be obtained by first computing a FBP on a very large reconstruction grid: in this case, the object support can be inferred from visual inspection once for all the volume. Given a size N_2, small initial σ lead to larger computation times as there are more functions in the basis, so the LUT are bigger. Larger initial σ decreases the computation time but might yield coarser results. Figure 4.2.23 shows an example of the influence of the number of Gaussians N_g on the result in the case of a 1024^2 original phantom. As seeing the profile, the cupping removal is slightly better when N_g is bigger (smaller Gaussians) and the error profile is overall closer to zero.

[4] as explained in the multi-resolution subsection, the standard deviations are then progressively doubled until reaching the ROI radius; and then kept to a maximal value outside the ROI.

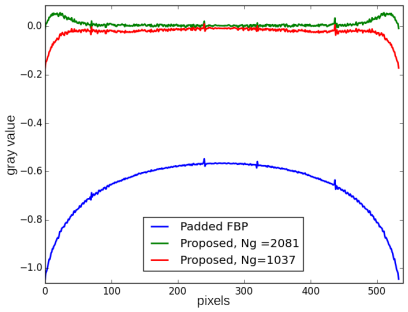


Figure 4.2.23: Line profile of reconstruction of a 1024^2 phantom with 544^2 pixels ROI, with different number of Gaussian functions.

The SIR method (with pixel domain variables) starts to be impracticable from 2048^2 pixels slices, as thousands of iterations are required to yield an acceptable image quality, leading to hours of processing per slice. The execution times for 4096^2 slices could not be measured as they take too much time; therefore the PSNR are not available in these cases. This method is actually implemented in Python with the ASTRA Toolbox, meaning that only the projection and backprojection are performed on GPU, so the implementation suffers from memory transfers between CPU and GPU. If fully implemented on GPU, one could expect a 5-10 speed-up for this method; nevertheless the proposed method would still be ahead.

For both methods the PSNR is progressively decreasing as the size of the slice increase, yet the reconstructions are satisfying. We believe this is a consequence of the cupping being not entirely corrected on the slice borders, which brings more and more contribution as the number of pixels increase.

Second test case

The second test involves the test image "Lena", 512×512 pixels, bearing both smooth regions and high frequencies textures. Ellipses with strong intensity values has been superimposed in the exterior of the ROI to simulate absorbing material.

Figure 4.2.24 shows the test setup. The known region has be chosen as slowly varying as possible, as in real acquisitions the known region is likely to be air or coarse features. The width of the extended image is $N_2 = 520$.

Figure 4.2.25 shows the difference between the true interior and the reconstruction with the proposed method, with and without the known zone constraint. In this case, the differences between the proposed reconstruction and the true interior are not significant enough to display the curves of absolute line profiles, so only the difference profiles are shown. The parameters $\sigma = s = 3$ has been used for these reconstructions. Figure 4.2.26 shows the reconstructed images. As the contrast is high, the cupping effect is not prominent on the reconstructions ; however, an excessive brightness can be seen on the bottom side.

It is also interesting to visualize the reconstruction of the whole extended image. As it can be seen on Figure 4.2.27, the Gaussian basis even yields an approximation of the exterior. This approximation is actually important for modeling the contribution of the external part in the acquired sinogram. The bias correction is thus closely related to the modeling of the external part.

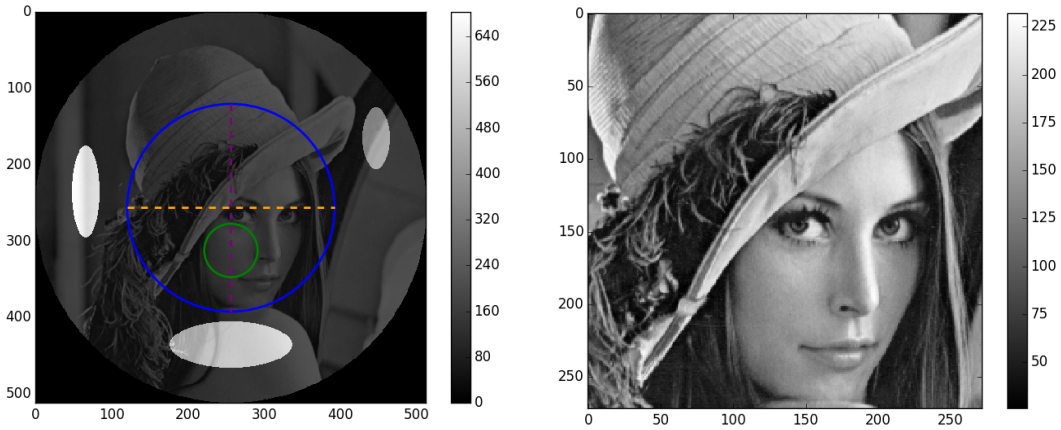


Figure 4.2.24: (a) Phantom "Lena". Ellipses with high gray values has been added to accentuate the local tomography setup. The outer circle is the ROI, and the inner circle is the known region. The dashed lines indicate the profiles which are to be plotted in the reconstructed slice. (b) View of the region of interest with adapted contrast.

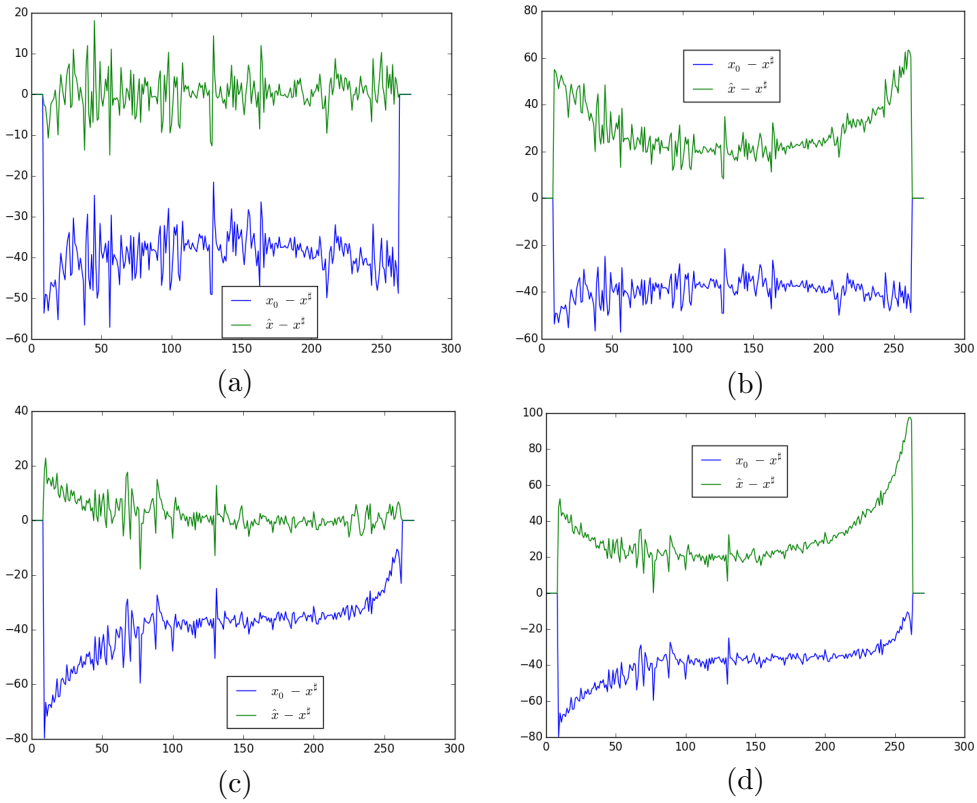


Figure 4.2.25: Profiles of difference between the reconstruction and the true interior for the *Lena* image. x_0, $\hat{x}$ and $x^\sharp$ are the padded FBP, the proposed reconstruction and the true interior, respectively. In blue: difference between the padded FBP and the true interior. In green: difference between the reconstruction with the proposed method with $\sigma = s = 3$ and the true interior. First row: profiles of the middle line of the image for (a) $R = 35$, (b) no constraint ($R = 0$) Second row: profiles of the middle column for (c) $R = 35$, (d) no constraint ($R = 0$)

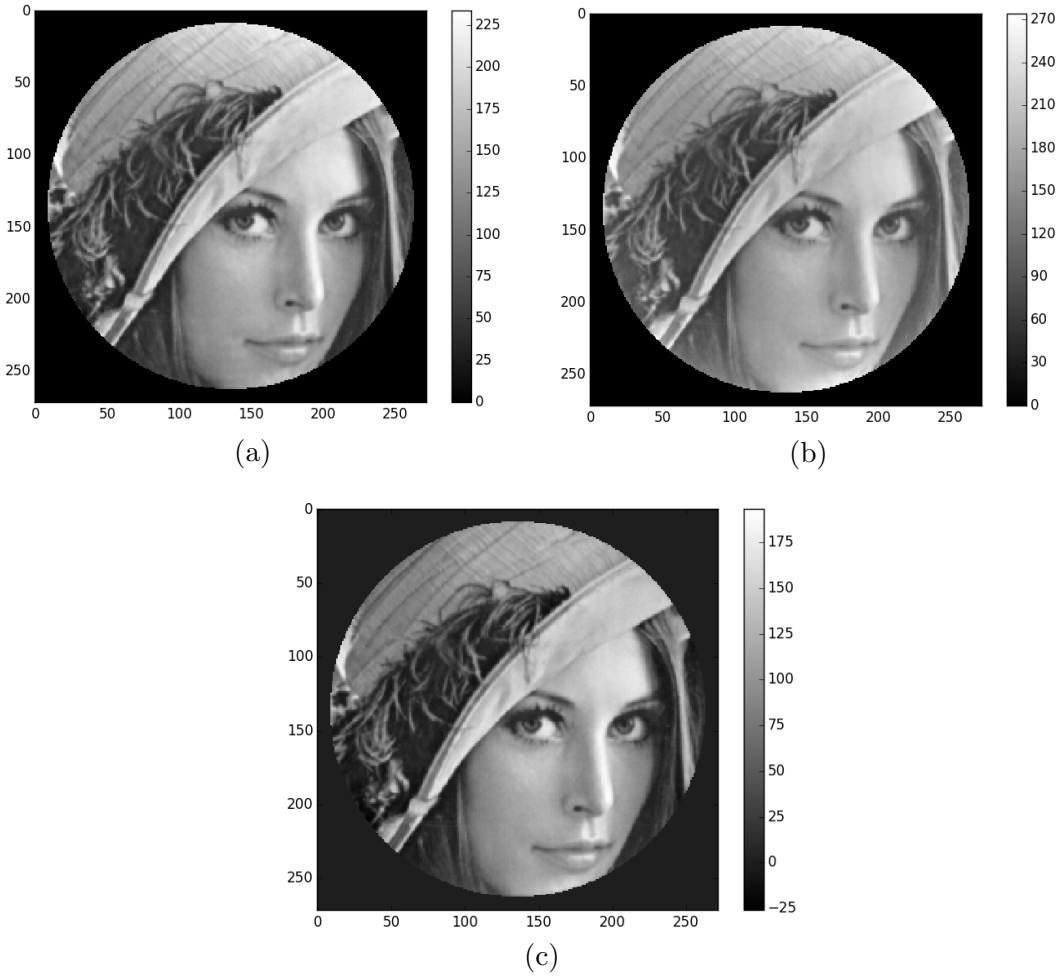


Figure 4.2.26: Reconstructions with (a) proposed method, known zone of radius $R = 35$ (b) proposed method, no known zone ($R = 0$), (c) padded FBP.

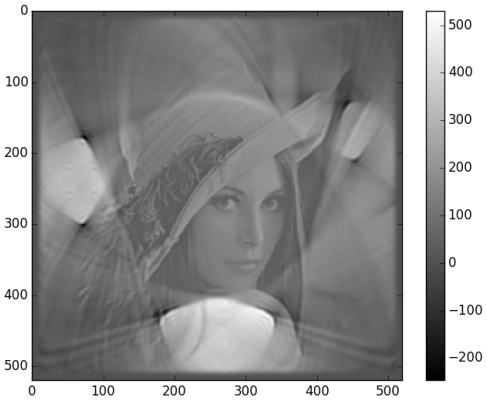
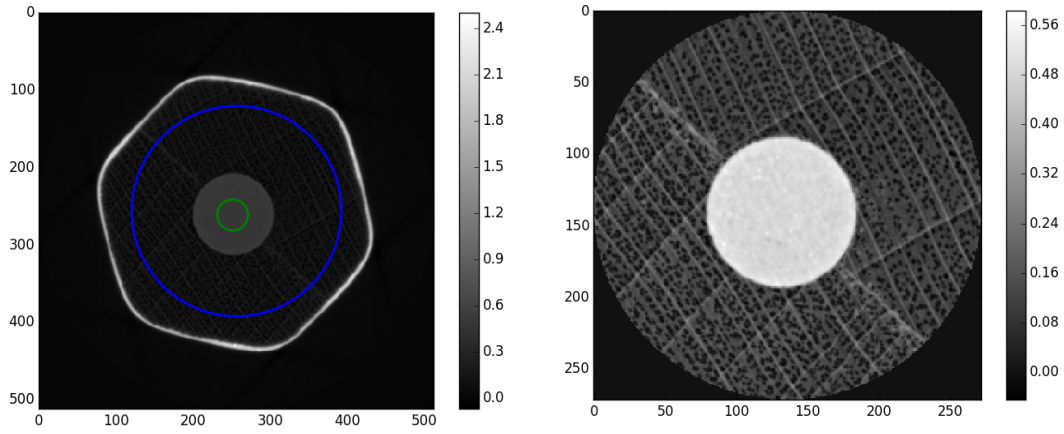
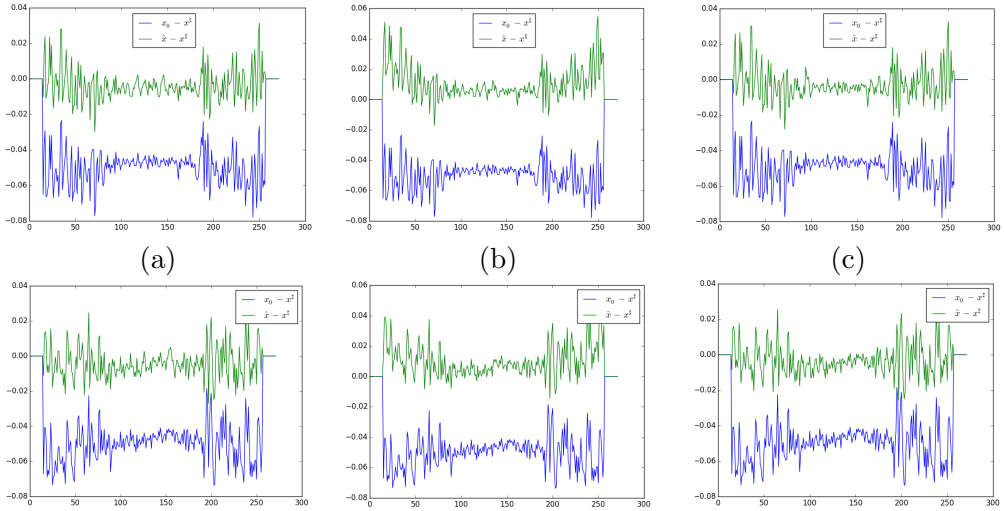


Figure 4.2.27: Extended image after reconstruction, without cropping to the region of interest. The parameters used were $\sigma = s = 3$ and $R = 35$.



(a) (b)

Figure 4.2.28: (a) *Pencil* test image. In red: region of interest. In green: known subregion. (b) View of the region of interest.



(a) (b) (c)

(d) (e) (f)

Figure 4.2.29: Profiles of difference between the reconstruction and the true interior for the *pencil* image. x_0, $\hat{x}$ and $x^\sharp$ are the padded FBP, the proposed reconstruction and the true interior, respectively. In blue: difference between the padded FBP and the true interior. In green: difference between the reconstruction with the proposed method with $\sigma = s = 3$ and the true interior. First row: profiles of the middle line of the image for (a) $R = 20$, (b) $R = 10$, (c) $R = 40$. Second row: profiles of the middle column for (d) $R = 20$, (e) $R = 10$, (f) $R = 40$

Third test case

The third test involves the image of a pencil resulting from a scan at the ESRF ID19 beamline, 512×512 pixels, shown on Figure 4.2.28. The width of the extended image is $N_2 = 520$.

Figure 4.2.29 shows profiles of the difference between the reconstruction and the true interior. On this image, a greater radius also improves the cupping removal. The profile of a line through the reconstructed image is depicted on Figure 4.2.30.

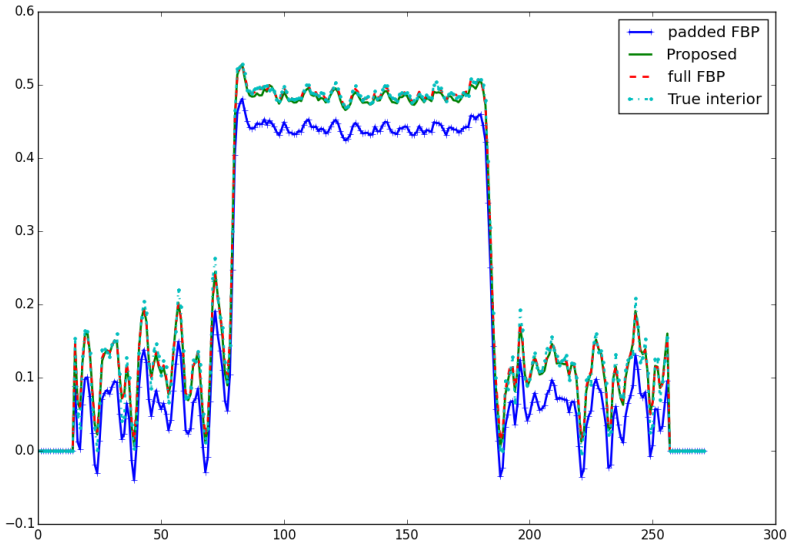
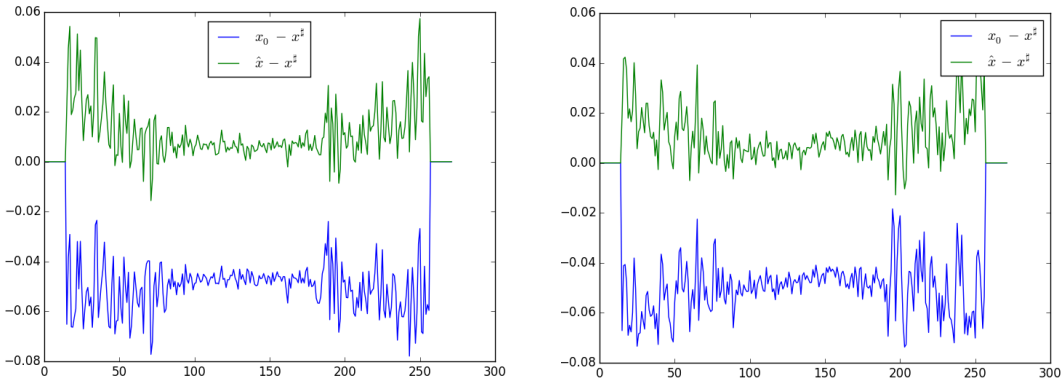


Figure 4.2.30: Line profiles for the *pencil* image. The proposed method were applied with parameters $\sigma = s = 3$ and $R = 40$.



(a) (b)

Figure 4.2.31: Profiles of difference between the reconstruction without known zone constraint and the true interior for the *pencil* image. (a) Line profile. (b) Column profile.

As a last remark, Figure 4.2.31 shows the result of this method without using the known zone constraint, that is, without applying the constraint $g_{|\Omega_g} = g_0$.As expected, there is a not-null mean bias, even if it has been reduced with respect to padded FBP.

Beside visual inspection, reconstructions can be quantitatively compared to the true interior of the test image. Table 4.2 shows the comparison with two image metrics: peak signal to noise ratio (PSNR) and the structural similarity index (SSIM). As these metrics are indicators of an *average* distance between two images, we believe it is well suited for this purpose of evaluating how the low frequencies are corrected by the proposed method.

These results suggest that the proposed method yield better overall reconstruction quality than padded FBP. In particular, it does not induce spurious distortion in the reconstruction. For the "Lena" test case, a similar reconstruction quality was obtained with $(\sigma, s) = (4, 6)$ with respect to $(\sigma, s) = (3, 3)$; which indicate that a coarser basis do not always yield worse reconstruction results.

For all the reconstructions with the proposed methods, the conjugate gradient op-

Image	Reconstruction method	Parameters	PSNR	SSIM
Shepp-Logan	Padded FBP		20.09	0.5751
Shepp-Logan	Proposed	$\sigma = 4$, $s = 6$, $R = 20$	38.40	0.6362
Shepp-Logan	Proposed	$\sigma = 5$, $s = 6$, $R = 20$	33.96	0.6360
Lena	Padded FBP		22.65	0.8417
Lena	Proposed	$\sigma = s = 3$, $R = 35$	35.89	0.9582
Lena	Proposed	$\sigma = 4$, $s = 6$, $R = 35$	33.80	0.9588
Pencil	Padded FBP		26.41	0.8542
Pencil	Proposed	$\sigma = s = 3$, $R = 10$	31.15	0.9840
Pencil	Proposed	$\sigma = 4$, $s = 6$, $R = 40$	31.91	0.9901
Pencil	Proposed	$\sigma = s = 3$, $R = 40$	34.22	0.9906

Table 4.2: Metrics of reconstruction quality for the three test images, computed inside the reconstructed ROI.

timization algorithm was used. The convergence is reached within 400 iterations. The prototype method available at [Pal16] takes the following execution times on a machine with a Intel Xeon CPU E5-1607 3.00GHz CPU, and a Nvidia GTX 750 Ti GPU : 7 seconds for a 260×260 extended slice, 32 seconds for a 520×520 extended slice, and 152 seconds for a 1040×1040 extended slice.

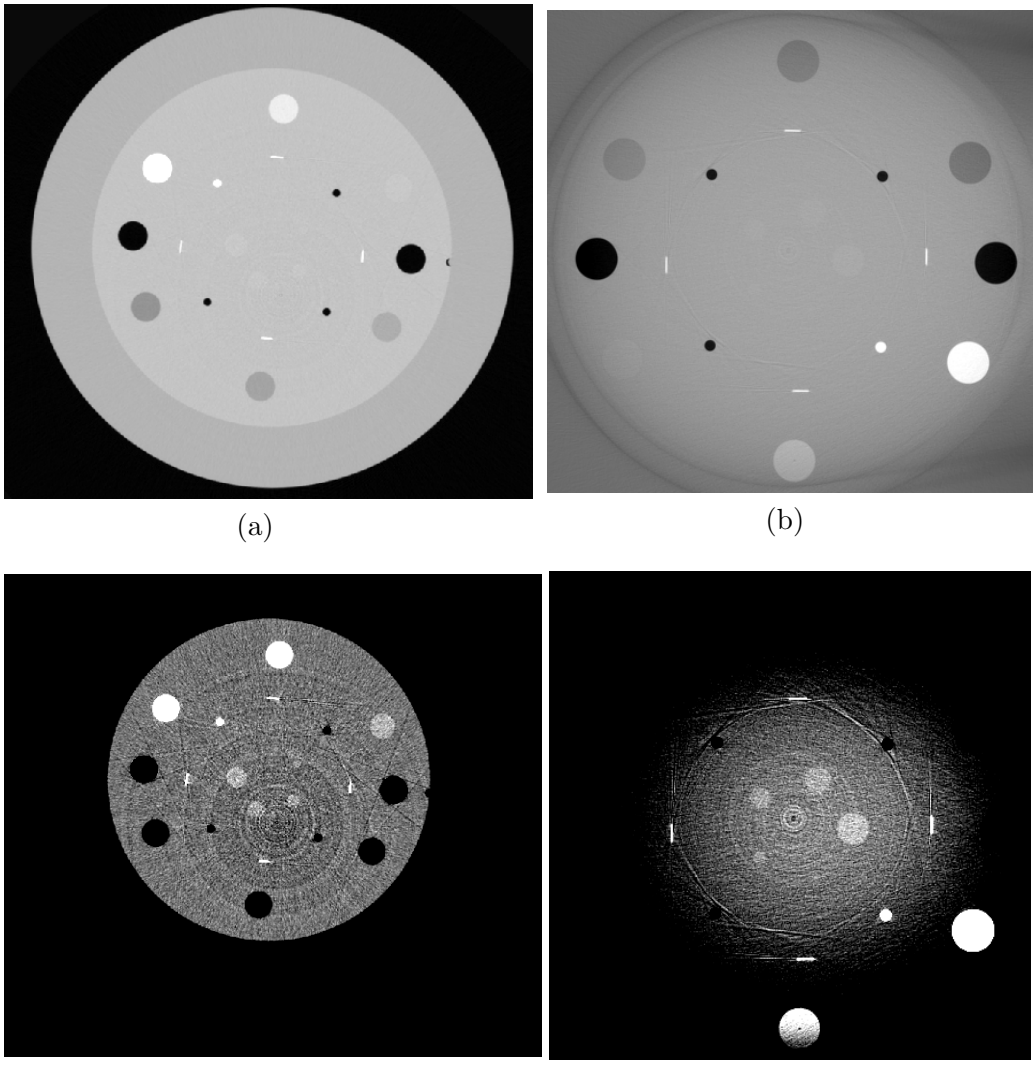
The proposed method depends on some parameters. The first is the size of the extended image, which should be big enough to model the contribution of the external part. The other parameters are the Gaussian standard deviation σ and the spacing s of the grid. Both are related in a way that the Gaussian functions should slightly overlap to approximate constant functions: if s value is high, then σ should also be high and conversely. These parameters essentially tune how coarse is the Gaussian basis: high values would yield fast convergence but coarse result, while small values would yield slow convergence and fine result.

Using a Gaussian basis does not yield an exact correction of the error, as Gaussian functions defined in Equation (4.2.9) do not form a basis. For example, Gaussian functions do not yield a partition of unity, although a very close approximation of this property can be achieved [Bal+02]. Thus, the final reconstruction cannot be *exact* due to the basis coarseness, but can provide results quite close to FBP with full data.

4.2.9 From simulated data to real data

In this subsection, we measure the quantitativeness of the proposed local tomography algorithm on a real dataset. We use the Catphan 504 medical imaging phantom [Lab13] and performed the scans at the ESRF beamline ID17, with a beam energy of 78 keV and a Germanium detector with direct detection. Two scans were performed: a non-local scan covering the entire field of view, and a scan were Plexiglas (PMMA) slabs were placed at some locations before the sample to simulate an additional absorbing material.

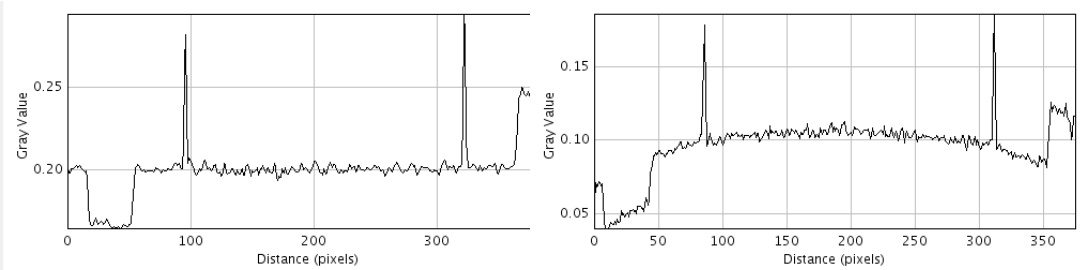
Figures 4.2.32 and 4.2.33 show the reconstruction of a slice of the phantom corresponding to the CTP404 module. On the right column, the cupping effect is clearly visible. This module contains materials of known linear attenuation coefficients at various energies, a table can be found in [Lab06].



(a) (b)

(c) (d)

Figure 4.2.32: Reconstructions of the Catphan 504 imaging phantom. (a, b) reconstructions in non-local and local settings, respectively. (c, d) enhancement of contrast of (a, b).



(a) (b)

Figure 4.2.33: Line profiles corresponding to Figure 4.2.32. (a) Reconstruction from global data. (b) Reconstruction from local data.

The line profiles on Figure 4.2.33 are crossing the following inserts:

- LDPE (Low-density polyethylene) between 0-50 pixels: $\mu_{78} = 0.168$, reconstruction gives $\hat{\mu_{78}} = 0.17$
- Two wire ramps made of teflon: $\mu_{78} = 0.346$, reconstruction gives $\hat{\mu_{78}} = 0.30$ for one ramp and 0.28 for the other. At this scan resolution, the ramps might might be too thin to give an accurate measurement of the linear attenuation coefficient.
- Delrin (Polyoxymethylene) from pixel 350: $\mu_{78} = 0.236$, reconstruction gives $\hat{\mu_{78}} = 0.23$

Figure 4.2.34 shows a line profile of the reconstruction using the proposed local tomography algorithm, using the Air inserts (black disks on Figure 4.2.32) as a known subregion. Arguably, the cupping effect is mostly removed, and the reconstruction quantitativeness is recovered by comparison to Figure 4.2.33.

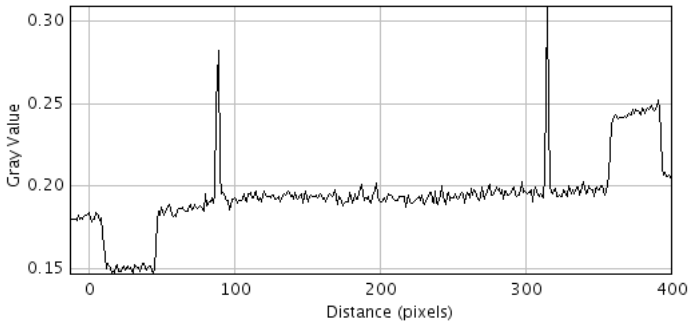


Figure 4.2.34: Line profile of a reconstructed slice of the CTP404 module, using the proposed local tomography algorithm.

Although performing quite well on previous datasets, the proposed method failed to remove the local tomography artefacts on some other datasets. A failure case example was an Archaeopteryx skull scan. The skull is very elongated, resulting in an almost total absorption in one direction.

In general, the local tomography artefacts are not the only artefacts, and several other effects are perturbing the reconstruction. The following parts give insights of the elements on the experimental side that could impact the reconstruction procedure.

Point Spread Function

An imaging system does not have an infinitely thin point spread function. In other words, an object having a width of exactly Δ will results in an image of this object with a width $\Delta_2 > \Delta$. The spread of objects supports by the imaging system can have different causes:

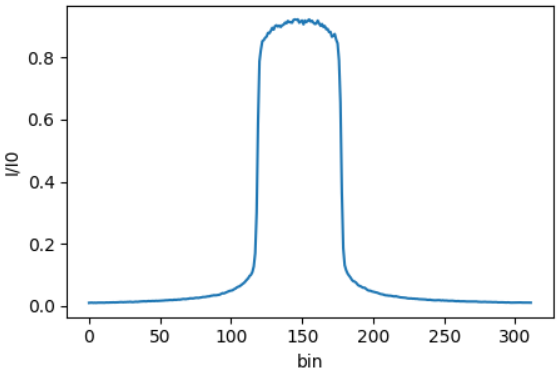
- The PSF of the detector, due to the response of its sensing elements.
- Light scattering in a scintillator, when one is used.
- Unwanted effects in the optics elements involved in the imaging chain.

All these effects can be combined in an equivalent PSF, which can be space-varying – although in general, a constant PSF is assumed in order to build a simple convolution-based model.

Figure 4.2.35 illustrates the PSF of a Frelon detector[5] with a Gadolinium Oxysulfide (Gadox) scintillator. This detector has a (horizontal) pixel size of 47 microns.

Apart from the non-ideal material used to measure the PSF (the slit is not perfectly sharp as a step function), it can be seen that the support is several tens of pixels. This large PSF is certainly essentially caused by the scintillator, as other measurements with a direct detector (without scintillator) led to a PSF of less than two pixels of 350 microns, i.e 0.7 mm (for the direct detector) to be compared to $0.94 - 1.13$ mm (for the detector with scintillator).

(a)



(b)

Figure 4.2.35: Measuring the point spread function of the (Frelon/Gadox) detector at 80 keV. (a) A Tungsten slab with a sharp slit is placed in front of the detector. (b) Relative intensity profile at the edges of the tungsten slit.

The deconvolution of the detector PSF has to be performed as a pre-processing step on the projections (not the sinogram) before computing the logarithm of the ratio $-\log(\boldsymbol{p}/\boldsymbol{p}_0)$ where $\boldsymbol{p}$ is a projection image and $\boldsymbol{p}_0$ is a measurement of the incoming beam (flat-field). An alternative could be to add the convolution in the forward model. Letting $\boldsymbol{C}$ be the convolution operator with the (known) PSF, the (unregularized) optimization problem becomes

$$\underset{\boldsymbol{x}}{\operatorname{argmin}} \left\{ \frac{1}{2} \left\| \boldsymbol{C}(\boldsymbol{p}_0 \odot \exp\left(-\boldsymbol{P}\boldsymbol{x}\right)) - \boldsymbol{p} \right\|_2^2 \right\} \tag{4.2.14}$$

where $\odot$ denotes an elementwise multiplication, and the exponential is computed elementwise. Here, the measured data $\boldsymbol{p}$ is the set of projections instead of a sinogram. The convolution takes place in the *projection domain*, (as the PSF is a detector feature) while $\boldsymbol{P}\boldsymbol{x}$ is a sinogram. Therefore, several slices have to be projected simultaneously in order to transpose the obtained stack of sinograms to form a subset of the projections (see Figure 1.2.2). The projection subset has to be large enough to contain the PSF support, i.e sufficiently many slices have to be reconstructed simultaneously. Thus, the vector $\boldsymbol{x}$ denotes a slab of slices (instead of a single slice) in this context.

Problem (4.2.14) is unfortunately not convex, so the algorithms reviewed in section 2.7 might not converge. We did not investigate further this model as it might be too cumbersome to solve whereas the gain might not be valuable. Besides, the acquisition is often followed by a phase retrieval procedure, as most tomography scans at ESRF

[5]Frelon is a designed at ESRF

exploit the phase contrast rather than the absorption contrast. In the case of the Paganin phase retrieval [Pag+02], a band-pass filter is applied to the projection data; thus, any deconvolution (roughly a high pass filter) will have limited effect.

Scintillator afterglow

Most scintillators are characterized by an *afterglow* effect, i.e they continue to scintillate light after the X-ray beam is stopped. The afterglow time ranges from a few to hundreds microseconds time scale [Nik06].

Figure 4.2.36, illustrates the afterglow of the Frelon/Gadox detector. When acquiring one image every 50 ms and closing the shutter, the intensity drops to the noise levels in one frame, i.e in less than 50 ms. Besides, many scans are *continuous*, meaning that instead of acquiring the projection images one after the other ("step by step mode"), the detector acquires the signal over a small angular range. This leads to a slight averaging effect which can be similar to the afterglow effect. Therefore, we believe that the afterglow is not a significant issue for most scan settings.

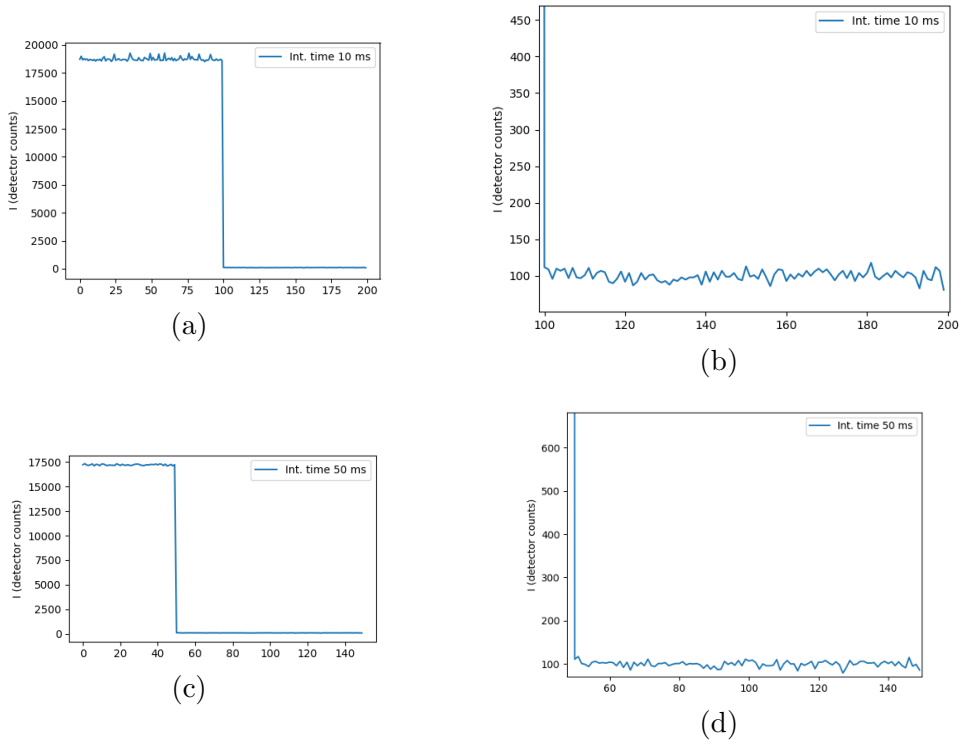


Figure 4.2.36: Measuring the remanence of the Frelon/Gadox detector. One image is acquired every 50 ms. (a) Integration time of 10 ms. (b) Zoom of (a) around the 100th frame. (c) Integration time of 50 ms. (d) Zoom of (c) around the 50th frame.

Other effects

Other effects due to the experimental setup can arise.

- Half tomography: consists in placing the rotation axis near an extremity of the sample and a performing a 360 degrees scan. The resulting sinogram is then "cut" in two half sinograms and merged to obtain a sinogram virtually covering twice

the original field of view. By doing so, each projection has an overlapping zone around the rotation axis (a zone at projection θ is found again at $\theta + 360$ degrees). This redundant zone should be weighted (for example by 1/2). In any case, half tomography can induce "transition artefacts".

- Strong absorption effects and phase retrieval when an object is strongly absorbing, artefacts arise after the phase retrieval. For example, the Paganin phase retrieval [Pag+02] assumes that the δ/β ratio is constant in the slice (where δ and β respectively denote the decrement of the real part and the imaginary part of the complex refractive index), which does not hold when objects have a strong absorption with respect to the others.

- Large field of view required: if the region of interest is very small with respect to the rest of the object, the extended slice size (N_2) has to be notably greater than the original ROI size N. This is problematic for performances when the slice size is already big (more than 4000×4000 pixels). Even with a smart projector and back-projector pair, the proposed algorithm starts to be impractical for 10000×10000 extended slices. Using a "too small" extended slice might invalidate the quantitativeness guarantee.

We believe that the two latter are the main responsible reasons for the failure of the proposed local tomography reconstruction algorithm on some datasets. As a possible improvement of this algorithm, an idea could be to first correct the cupping effect for a low resolution image (obtained with binned projections) and then applying the upsampled correction to the reconstruction. This could work because the correction only contains low frequencies, so it does not suffer significantly from downsampling or upsampling.

Conclusion

In this chapter, we showed how the mathematical framework for regularized tomographic reconstruction can be extended to take artefacts into account. More precisely, the forward model is transformed by the means of new operators to simulate either the rings or a local tomography setup, leading to a new optimization problem.

The iterative rings correction (section 4.1) combines the volume reconstruction with the ring artefacts removal and is shown to be efficient where other methods do not succeed, although finding a method successful in every case is still an unsolved problem. The resulting optimization problem is harder to solve than the usual ℓ_1-regularized problem, which motivated the development of the CSG algorithm (section 3.3).

The proposed local tomography reconstruction algorithm (section 4.2) addresses a difficult problem by actually solving a simpler related surrogate problem; yet, it yields promising results as quantitativeness can be recovered.

Bibliography

[Aar+16] Wim van Aarle et al. "Fast and flexible X-ray tomography using the ASTRA toolbox". In: *Opt. Express* 24.22 (Oct. 2016), pp. 25129–25147. DOI: 10.1364/OE.24.025129. URL: http://www.opticsexpress.org/abstract.cfm?URI=oe-24-22-25129.

[Bal+02] Richard A Bale et al. *Multidimensional partitions of unity and Gaussian terrains*. Tech. rep. CREWES, 2002.

[Bar+94] Richard Barrett et al. *Templates for the solution of linear systems: building blocks for iterative methods*. Vol. 43. Siam, 1994.

[Bil07] Anne Bilgot. "Méthodes locales d'identification de surfaces de discontinuité à partir de projections tronquées pour l'imagerie interventionnelle". PhD thesis. Université Joseph-Fourier-Grenoble I, 2007.

[Bla14] Justin Blair. *Code for doing tomographic reconstructions and image filtering*. https://github.com/gyronaut/tomography. Accessed: 2014-12-03. 2014.

[CD10] Rolf Clackdoyle and Michel Defrise. "Tomographic reconstruction in the 21st century". In: *Signal Processing Magazine, IEEE* 27.4 (2010), pp. 60–80.

[Cla+04] Rolf Clackdoyle et al. "Quantitative reconstruction from truncated projections in classical tomography". In: *Nuclear Science, IEEE Transactions on* 51.5 (2004), pp. 2570–2578.

[Cla+09] R. Clackdoyle et al. "Two-dimensional region-of-interest reconstruction: Analyzing the difference between virtual fanbeam and DBP-Hilbert reconstructions". In: *Nuclear Science Symposium Conference Record (NSS/MIC), 2009 IEEE*. Oct. 2009, pp. 3367–3371. DOI: 10.1109/NSSMIC.2009.5401760.

[Cou+08] M Courdurier et al. "Solving the interior problem of computed tomography using a priori knowledge". In: *Inverse problems* 24.6 (2008), p. 065001.

[Def+06] Michel Defrise et al. "Truncated Hilbert transform and image reconstruction from limited tomographic data". In: *Inverse problems* 22.3 (2006), p. 1037.

[EB14] F. O'Dowd E. X. Miqueles J. Rinkel and J. S. V. Bermúdez. "Generalized Titarenko's algorithm for ring artefacts reduction". In: *Journal of Synchrotron Radiation* 21 (2014), pp. 1333–1346. DOI: 10.1107/S1600577514016919.

[HO93] Per Christian Hansen and Dianne Prost O'Leary. "The Use of the L-Curve in the Regularization of Discrete Ill-Posed Problems". In: *SIAM Journal on Scientific Computing* 14.6 (1993), pp. 1487–1503. DOI: 10.1137/0914086. eprint: http://dx.doi.org/10.1137/0914086. URL: http://dx.doi.org/10.1137/0914086.

[KQR15] Esther Klann, Eric Todd Quinto, and Ronny Ramlau. "Wavelet methods for a weighted sparsity penalty for region of interest tomography". In: *Inverse Problems* 31.2 (2015), p. 025001. URL: http://stacks.iop.org/0266-5611/31/i=2/a=025001.

[KS92] Pl. Kannappan and P. K. Sahoo. "Rotation Invariant Separable Functions are Gaussian". In: *SIAM Journal on Mathematical Analysis* 23.5 (1992), pp. 1342–1351. DOI: 10.1137/0523076. eprint: http://dx.doi.org/10.1137/0523076. URL: http://dx.doi.org/10.1137/0523076.

[Kud+08] Hiroyuki Kudo et al. "Tiny a priori knowledge solves the interior problem in computed tomography". In: *Physics in medicine and biology* 53.9 (2008), p. 2207.

[Kyr+11] A Kyrieleis et al. "Region-of-interest tomography using filtered backprojection: assessing the practical limits". In: *Journal of microscopy* 241.1 (2011), pp. 69–82.

[Lab06] The Phantom Laboratory. *Catphan 500 and 600 manual*. http://www.uio.no/studier/emner/mat 600manual.pdf. 2006.

[Lab13] The Phantom Laboratory. *Catphan 504 manual*. https://github.com/jrkerns/pylinac/blob/mast %20Varian.pdf. 2013.

[Lee+15] Minji Lee et al. "Interior Tomography Using 1D Generalized Total Variation. Part II: Multiscale Implementation". In: *SIAM Journal on Imaging Sciences* 8.4 (2015), pp. 2452–2486. DOI: 10.1137/15M1015881. eprint: http://dx.doi.org/10.1137/15M1015881. URL: http://dx.doi.org/10.1137/15M1015881.

[Mir+14] Alessandro Mirone et al. "The PyHST2 hybrid distributed code for high speed tomographic reconstruction with iterative reconstruction and a priori knowledge capabilities". In: *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms* 324.0 (2014). 1st International Conference on Tomography of Materials and Structures, pp. 41–48. ISSN: 0168-583X. DOI: http://dx.doi.org/10.1016/j.nimb.2013.09.030. URL: http://www.sciencedirect.com/science/article/pii/S0168583X14000251.

[Moh+14] K.A. Mohan et al. "Model-based iterative reconstruction for synchrotron X-ray tomography". In: *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. May 2014, pp. 6909–6913. DOI: 10.1109/ICASSP.2014.6854939.

[Mün+09] Beat Münch et al. "Stripe and ring artifact removal with combined wavelet—Fourier filtering". In: *Opt. Express* 17.10 (May 2009), pp. 8567–8591. DOI: 10.1364/OE.17.008567. URL: http://www.opticsexpress.org/abstract.cfm?URI=oe-17-10-8567.

[NCP04] Frederic Noo, Rolf Clackdoyle, and Jed D Pack. "A two-step Hilbert transform method for 2D image reconstruction". In: *Physics in Medicine and Biology* 49.17 (2004), p. 3903.

[Nik06] Martin Nikl. "Scintillation detectors for x-rays". In: *Measurement Science and Technology* 17.4 (2006), R37.

[Noo+02] Frederic Noo et al. "Image reconstruction from fan-beam projections on less than a short scan". In: *Physics in Medicine and Biology* 47.14 (2002), p. 2525.

[NSK07] K Niinimäki, S Siltanen, and V Kolehmainen. "Bayesian multiresolution method for local tomography in dental x-ray imaging". In: *Physics in Medicine and Biology* 52.22 (2007), p. 6663.

[Pag+02] D. Paganin et al. "Simultaneous phase and amplitude extraction from a single defocused image of a homogeneous object". In: *Journal of Microscopy* 206.1 (2002), pp. 33–40. ISSN: 1365-2818. DOI: 10.1046/j.1365-2818.2002.01010.x. URL: http://dx.doi.org/10.1046/j.1365-2818.2002.01010.x.

[Pal16] Pierre Paleo. *Implementation of a new method for local tomography reconstruction*. https://github.com/pierrepaleo/localtomo. 2016.

[PC11] Thomas Pock and Antonin Chambolle. "Diagonal preconditioning for first order primal-dual algorithms in convex optimization". In: *2011 International Conference on Computer Vision*. IEEE. 2011, pp. 1762–1769.

[PDM17] Pierre Paleo, Michel Desvignes, and Alessandro Mirone. "A practical local tomography reconstruction algorithm based on a known sub-region". In: *Journal of Synchrotron Radiation* 24.1 (Jan. 2017), pp. 257–268. DOI: 10.1107/S1600577516016556. URL: https://doi.org/10.1107/S1600577516016556.

[PKK09] Daniel Prell, Yiannis Kyriakou, and Willi A Kalender. "Comparison of ring artifact correction methods for flat-detector CT". In: *Physics in Medicine and Biology* 54.12 (2009), p. 3881. URL: http://stacks.iop.org/0031-9155/54/i=12/a=018.

[PM15] Pierre Paleo and Alessandro Mirone. "Ring artifacts correction in compressed sensing tomographic reconstruction". In: *Journal of Synchrotron Radiation* 22.5 (Sept. 2015), pp. 1268–1278. DOI: 10.1107/S1600577515010176. URL: https://doi.org/10.1107/S1600577515010176.

[PM17] Pierre Paleo and Alessandro Mirone. "Efficient implementation of a local tomography reconstruction algorithm". In: *Advanced Structural and Chemical Imaging* 3.1 (2017), p. 5. ISSN: 2198-0926. DOI: 10.1186/s40679-017-0038-1. URL: http://dx.doi.org/10.1186/s40679-017-0038-1.

[Ras+97] Farrokh Rashid-Farrokhi et al. "Wavelet-based multiresolution local tomography". In: *Image Processing, IEEE Transactions on* 6.10 (1997), pp. 1412–1430.

[RK10] Essam A Rashed and Hiroyuki Kudo. "Recent advances in interior tomography". In: *mathematical programming in the 21st century: Algorithms and modeling* 1676 (2010), pp. 145–156.

[RLH12] Sabrina Rashid, Soo Lee, and Md Hasan. "An improved method for the removal of ring artifacts in high resolution CT imaging". In: *EURASIP Journal on Advances in Signal Processing* 2012.1 (2012), p. 93. ISSN: 1687-6180. DOI: 10.1186/1687-6180-2012-93. URL: http://asp.eurasipjournals.com/content/2012/1/93.

[SD05] Challa S Sastry and PC Das. "A convolution back projection algorithm for local tomography". In: *The ANZIAM Journal* 46.03 (2005), pp. 341–360.

[SY11] Kang Yang Shuangren Zhao and Xintie Yang. "Reconstruction from truncated projections using mixed extrapolations of exponential and quadratic functions". In: *Journal of X-ray Science and Technology* 19.2 (2011), p. 155.

[Tit+10] Valeriy Titarenko et al. *Regularization methods for inverse problems in x-ray tomography*. 2010. DOI: 10.1117/12.860260. URL: http://dx.doi.org/10.1117/12.860260.

[TYT12] Shaojie Tang, Yi Yang, and Xiangyang Tang. "Practical interior tomography with radial Hilbert filtering and a priori knowledge in a small round area". In: *Journal of X-ray science and technology* 20.4 (2012), p. 405.

[Van09] Gert Van Gompel. "Towards accurate image reconstruction from truncated X-ray CT projections". PhD thesis. Antwerp: University of Antwerp, 2009.

[VDV06] Gert Van Gompel, Michel Defrise, and Dirk Van Dyck. "Elliptical extrapolation of truncated 2D CT projections using Helgason-Ludwig consistency conditions". In: *Medical Imaging*. International Society for Optics and Photonics. 2006, 61424B–61424B.

[VRU08] Cédric Vonesch, Sathish Ramani, and Michael Unser. "Recursive risk estimation for non-linear image deconvolution with a wavelet-domain sparsity constraint". In: *2008 15th IEEE International Conference on Image Processing*. IEEE. 2008, pp. 665–668.

[WY13] Ge Wang and Hengyong Yu. "The meaning of interior tomography". In: *Physics in medicine and biology* 58.16 (2013), R161.

[Yan+10] Jiansheng Yang et al. "High-order total variation minimization for interior tomography". In: *Inverse problems* 26.3 (2010), p. 035013.

[Ye+07] Yangbo Ye et al. "A general local reconstruction approach based on a truncated Hilbert transform". In: *Journal of Biomedical Imaging* 2007.1 (2007), pp. 2–2.

[YW06] Hengyong Yu and Ge Wang. "A general formula for fan-beam lambda tomography". In: *International journal of biomedical imaging* 2006 (2006).

[YW09] Hengyong Yu and Ge Wang. "Compressed sensing based interior tomography". In: *Physics in medicine and biology* 54.9 (2009), p. 2791.

[Zen10] Gengsheng Lawrence Zeng. *Medical image reconstruction*. Springer, 2010.

[ZPS05] Yu Zou, Xiaochuan Pan, and Emil Y Sidky. "Image reconstruction in regions-of-interest from truncated projections in a reduced fan-beam scan". In: *Physics in Medicine and Biology* 50.1 (2005), p. 13.

Chapter 5

Conclusion

This thesis aimed at investigating how regularized tomography algorithms can address reconstruction challenges, in particular limited/noisy data and artefacts. These reconstruction algorithms amount to solve an optimization problem which depends on the noise model and prior knowledge on the volume. In this work, we brought contributions to three aspects of regularized reconstruction methods: modelling, algorithmic and computational sides.

On the modelling side, we developed a unified formalism for all the reconstruction methods, enabling to properly separate the forward model, operators and optimization algorithms. The design of a reconstruction method schematically follows three steps:

- A noise model and a prior knowledge on the volume are defined.
- The reconstruction problem is cast in the form of an minimization problem, where the data fidelity term depends on the noise model and the regularization term comes from the assumption that the volume can be sparsely represented in some basis.
- The best-suited optimization algorithm is designed and implemented to solve the resulting minimization problem.

We extended the standard Bayesian framework by incorporating artefacts as new variables in the corresponding optimization problem, which corresponds to modelling them as a structured noise. Two applications are given: rings artefacts removal, and local tomography reconstruction. The proposed rings removal method simultaneously reconstructs a volume with regularization and corrects for artefacts, and is shown to compete with standard correction methods. The proposed interior tomography algorithm, primarily dedicated at artefact removal, allows for quasi-exact reconstruction.

On the algorithmic side, we used and extended state-of-the art convex optimization algorithms dealing with non-differentiable objective functions for tomographic reconstruction. With the recent advances in proximal algorithms, the presence of the ℓ_1 term – necessary to benefit from the robustness to undersampling – is not an issue as it used to be. Depending on the volume prior (regularization), either Chambolle-Pock, ADMM or FISTA is a good choice.

- For Total Variation regularization (piecewise-constant images), we showed that the Chambolle-Pock algorithm is appealing for both its high versatility and the fact that it does not require to solve inner sub-problems.
- For Wavelets, FISTA and ADMM are simple and efficient, although ADMM requires to tune algorithm parameters for a better convergence. We showed that in our parallel geometry setting, the least-squares proximal operator can be efficiently computed, leading to an interesting speed-up of algorithms like ADMM.

- For Dictionary synthesis reconstruction combined with rings artefacts correction, we designed a new optimization algorithm tailored for ill-conditioned LASSO problems.

On the computational side, we implemented all the regularized methods in the tomography software used at ESRF, notably a high-performance Chambolle-Pock Total Variation solver and a GPU wavelets library. The implementation on GPU of data processing algorithm is not an option today, as modern detectors data throughput tend to overpace the grow of computing power. The software PyHST offers a toolbox of ready-to-use advanced reconstruction algorithms running on GPU. We also developed a GPU Wavelets library which is ready to be integrated in other projects of regularized linear inverse problems. These regularized methods enable to reconstruct highly subsampled/noisy data where traditional methods usually fail, while being reasonably fast. Although the computation time is one reason restraining users to adopt these methods, we believe that our high-performance implementation efforts make them more usable in practice. Regularized methods indeed start to be used at ESRF beamlines in some cases. However, these methods are more likely to be used if they are available in several reconstruction softwares. In this work, we tried to give enough details to implement them, and many prototypes and codes are publicly available.

Outlooks

Iterative methods, and therefore regularized iterative methods, are still little used in practice for two main reasons: computation time and usability. On the one hand, these methods are slower than the plain FBP, so datasets take more time to reconstruct; on the other hand they involve a parameter selection which can be tricky. Therefore, further efforts in regularized algorithms should address these two issues.

Computation time: algorithmic

Algorithmic work can be carried out to improve the convergence rate. On the one hand, it can be valuable to investigate fast converging algorithms like second-order Nesterov-Bregman methods in the context of tomographic reconstruction. On the other hand, existing algorithms could be "mixed" together to use a "warm restart" property. For example, ADMM or conjugate (sub)gradient can be used in the first stage to provide an initial solution. This solution is then used as a starting point by FISTA or Chambolle-Pock in a second stage.

Computation time: implementation

This work leaves some outlooks on the computational side. The CSG algorithm, although fast converging, uses several preconditioners which use a lot of memory. Transferring these arrays in the GPU kernels functions is actually a bottleneck in the current implementation. The preconditioners can be merged in one single array, since all of them are "binary" array– for example, a single "uint8" array is enough to store four binary preconditioners.

The interior tomography algorithm, although it has a GPU implementation, can still be accelerated. Since only the low-frequencies artefacts should be corrected, the "correction image" can be computed in a binned space in order to be expanded afterwards. This

would dramatically speed up the process, as it is computationally expensive to compute the projection of huge arrays, even with the current implementation.

The "prox-filter" should also be further investigated to see if the constraint to reconstruct in the inner circle can be alleviated.

Lastly, the projection and backprojection operators can be made more efficient by using the Fourier slice theorem. As detectors now produce images above 4096×4096 pixels, these operators become the bottleneck of the reconstruction process.

Usability

In order to be more used in practice, iterative methods could be made more interactive. For example, a coarse reconstruction or a reconstruction of a region of interest can be first carried on before the full reconstruction. The choice of the regularization parameter is also an issue: should it be automatically computed ? Some users might want to tune it depending on what feature they actually want to see, but others would just need a "plug-and-play" algorithm. To this end, we believe that building a fully Bayesian model is the most mathematically sound approach.

Better regularization and modelling

Lastly apart from the computation time and usability issues, we can think of making modelling and regularization even "better". Although the parallel geometry is computationally very convenient, three-dimensional regularization would certainly improve the reconstruction results, are correlation between adjacent voxels is taken into account. To this end, slices can be reconstructed by slabs instead of individually.

Other regularizations can also be considered; for example some works use jointly TV and Wavelets, or the Dual Tree Complex Wavelets Transform. Group sparsity priors can also be used in order to further reduce the thresholding artefacts. Constraints like positivity should always be enforced when possible as it leads to better results with a significant improvement in the convergence rate.

Operators can also be modified to account for physical effects. For example, the plain Radon transform assumes a Beer-Lambert transmission, which is true only for a monochromatic beam. The forward and adjoint operators could therefore be designed for a polychromatic beam and other effects like Compton scattering.

Lastly, regularization techniques can also be designed from new priors. In most regularized methods, the regularization term is in the form of the ℓ_1 norm (or a group norm) of the image coefficients, which promotes sparsity/compressibility in some basis. The design of new models can be done the other way around, in order to establish a clearer link between the prior on the image and the optimization problem.

Chapter 6

Appendix

This appendix contains material which is certainly known by the initiate reader, and therefore moved to the Appendix for an easier reading.

6.1 Mathematical proofs

6.1.1 First chapter

The following is a proof of Proposition 1.2.1.

Proof. The one dimensional Fourier Transform of the Radon Transform of $\boldsymbol{f}$ at angle θ_0 reads

$$\mathcal{F}_1[\mathcal{R}_{\theta_0}[\boldsymbol{f}]](\nu) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \boldsymbol{f}\begin{pmatrix} s\cos\theta_0 - t\sin\theta_0 \\ s\sin\theta_0 + t\cos\theta_0 \end{pmatrix} \mathrm{d}t\; e^{-\boldsymbol{\jmath} 2\pi\nu s}\,\mathrm{d}s$$

Let $\begin{pmatrix} x \\ y \end{pmatrix} = \begin{bmatrix} \cos\theta_0 & -\sin\theta_0 \\ \sin\theta_0 & \cos\theta_0 \end{bmatrix}\begin{pmatrix} s \\ t \end{pmatrix}$ be a variable substitution. The Jacobian is equal to 1 and $s = x\cos\theta_0 + y\sin\theta_0$, giving

$$\mathcal{F}_1[\mathcal{R}_{\theta_0}[\boldsymbol{f}]](\nu) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \boldsymbol{f}(x,y) e^{-\boldsymbol{\jmath} 2\pi\nu(x\cos\theta_0 + y\sin\theta_0)}\,\mathrm{d}x\,\mathrm{d}y$$

Which, after rearranging the complex exponential argument, yields the two dimensional Fourier Transform of $\boldsymbol{f}$ evaluated at frequencies $(\nu\cos\theta_0,\, \nu\sin\theta_0)$. The integrals could be switched as $\boldsymbol{f}$ belongs to $\ell_1(\mathbb{R}^2)$. □

The following is a proof of Proposition 1.2.2.

Proof. Let $\theta_0 \in [0, 2\pi]$. Applying the Fourier-Slice theorem 1.2.1 and the convolution theorem yields

$$\begin{aligned}
\mathcal{F}_1\mathcal{R}_{\theta_0}[\boldsymbol{f} * \boldsymbol{g}](\nu) &= \mathcal{F}_2(\boldsymbol{f} * \boldsymbol{g})(\nu\cos\theta_0,\, \nu\sin\theta_0) \\
&= (\hat{\boldsymbol{f}} \cdot \hat{\boldsymbol{g}})(\nu\cos\theta_0,\, \nu\sin\theta_0) \\
&= \hat{\boldsymbol{f}}(\nu\cos\theta_0,\, \nu\sin\theta_0) \cdot \hat{\boldsymbol{g}}(\nu\cos\theta_0,\, \nu\sin\theta_0) \\
&= \mathcal{F}_1[\mathcal{R}_{\theta_0}[\boldsymbol{f}]](\nu) \cdot \mathcal{F}_1[\mathcal{R}_{\theta_0}[\boldsymbol{g}]](\nu)
\end{aligned}$$

Applying the inverse Fourier Transform $\mathcal{F}_1^{-1}$ and the convolution theorem leads to Equation (1.2.5). □

The following is a proof of Proposition 1.4.1.

Proof. The function $\boldsymbol{f}(x,y)$ can be written as the inverse FT of its FT $\hat{\boldsymbol{f}}(\nu_x, \nu_y)$:

$$\boldsymbol{f}(x,y) = \iint_{\mathbb{R}^2} \hat{\boldsymbol{f}}(\nu_x,\nu_y) e^{+\boldsymbol{\jmath} 2\pi(\nu_x x + \nu_y y)}\,\mathrm{d}\nu_x\,\mathrm{d}\nu_y$$

Let $\boldsymbol{\varphi}(\theta,\nu) = \begin{pmatrix}\nu\cos\theta\\ \nu\sin\theta\end{pmatrix}$ be a polar variable substitution, so that $\begin{pmatrix}\nu_x\\ \nu_y\end{pmatrix} = \boldsymbol{\varphi}(\theta,\nu) = \begin{pmatrix}\nu\cos\theta\\ \nu\sin\theta\end{pmatrix}$. The Jacobian matrix of $\boldsymbol{\varphi}$ is $\begin{bmatrix}\frac{\partial\boldsymbol{\varphi}}{\partial\theta} & \frac{\partial\boldsymbol{\varphi}}{\partial s}\end{bmatrix} = \begin{bmatrix}\cos\theta & -\nu\sin\theta\\ \sin\theta & \nu\cos\theta\end{bmatrix}$ which has determinant $|\nu|$. Therefore, the previous integral can be written

$$\boldsymbol{f}(x,y) = \int_0^{2\pi}\int_{-\infty}^{\infty} \hat{\boldsymbol{f}}(\nu\cos\theta,\nu\sin\theta)e^{+\jmath 2\pi\nu(\cos\theta x+\sin\theta y)}|\nu|\,\mathrm{d}\nu\,\mathrm{d}\theta$$

Applying the Fourier-Slice theorem (Proposition 1.2.1), we have $\hat{\boldsymbol{f}}(\nu\cos\theta,\nu\sin\theta) = \mathcal{F}_1[\boldsymbol{p}](\theta,\nu)$, yielding

$$\boldsymbol{f}(x,y) = \int_0^{2\pi}\int_{-\infty}^{\infty} \mathcal{F}_1[\boldsymbol{p}](\theta,\nu)|\nu|e^{+\jmath 2\pi\nu(\cos\theta x+\sin\theta y)}\,\mathrm{d}\nu\,\mathrm{d}\theta$$

Which yields the second equality in Equation (1.4.1). The term $\int_{-\infty}^{\infty} \mathcal{F}_1[\boldsymbol{p}](\theta,\nu)|\nu|e^{+\jmath 2\pi\nu(\cos\theta x+\sin\theta y)}\,\mathrm{d}\nu$ is clearly the inverse 1D FT of $\mathcal{F}_1[\boldsymbol{p}](\theta,\nu)|\nu|$ evaluated in $s = \cos\theta x + \sin\theta y$. Applying the convolution theorem leads to first equality in Equation (1.4.1). □

The following is a proof of Proposition 1.4.2.

Proof. The vector $\boldsymbol{x}_m = \boldsymbol{A}^T\left(\boldsymbol{A}\boldsymbol{A}^T\right)^{-1}\boldsymbol{y}$ is clearly a solution of $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$. Let us show that $\boldsymbol{x}_m$ has the minimum norm among all the solutions of $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$. Let $\boldsymbol{x}_s$ be one of these solutions. Then: $\boldsymbol{A}(\boldsymbol{x}_s - \boldsymbol{x}_m) = 0$, so $\boldsymbol{x}_s - \boldsymbol{x}_m \in \mathrm{Ker}\,(\boldsymbol{A})$. From basic linear algebra, we also have $\mathrm{Ker}\,(\boldsymbol{A}) = \mathrm{Im}\left(\boldsymbol{A}^T\right)^{\perp}$ (the complement of the image of $\boldsymbol{A}$), so $\boldsymbol{x}_s - \boldsymbol{x}_m \in \mathrm{Im}\left(\boldsymbol{A}^T\right)^{\perp}$. As $\boldsymbol{x}_m \in \mathrm{Im}\left(\boldsymbol{A}^T\right)$, we have $\langle \boldsymbol{x}_s - \boldsymbol{x}_m\,,\,\boldsymbol{x}_m\rangle = 0$. Thus, applying the Pythagorean theorem:

$$\|\boldsymbol{x}_s\|_2 = \|\boldsymbol{x}_s - \boldsymbol{x}_m + \boldsymbol{x}_m\|_2 = \|\boldsymbol{x}_s - \boldsymbol{x}_m\|_2 + \|\boldsymbol{x}_m\|_2 \geq \|\boldsymbol{x}_m\|_2$$

□

6.1.2 Second chapter

The following is a proof of Equation (2.4.2)

Proof. Problem (2.4.2) will be derived from the beginning in a Bayesian framework. This derivation is the same for all probability distributions of the exponential family. The acquisition model is $\boldsymbol{d} = \boldsymbol{P}\boldsymbol{x} + \boldsymbol{n}$, where $\boldsymbol{n}$ is a zero-mean additive Gaussian noise of PDF

$$p_{\boldsymbol{n}}(\boldsymbol{n}) = |\det 2\pi\boldsymbol{\Sigma}|^{-1/2}\exp\left(-\frac{1}{2}\boldsymbol{n}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{n}\right)$$

The quantity of interest is the *probability that the latent signal is equal to* $\boldsymbol{x}$*, given that the observed data is* $\boldsymbol{d}$: $p_{\boldsymbol{x}}(\boldsymbol{x} \mid \boldsymbol{d})$. The Bayes formula gives this quantity as a posterior probability:

$$p_{\boldsymbol{x}}(\boldsymbol{x} \mid \boldsymbol{d}) = \frac{p_{\boldsymbol{d}}(\boldsymbol{d} \mid \boldsymbol{x})p_{\boldsymbol{x}}(\boldsymbol{x})}{p_{\boldsymbol{d}}(\boldsymbol{d})}$$

Since $\boldsymbol{d} = \boldsymbol{P}\boldsymbol{x} + \boldsymbol{n}$, a change of variable theorem states that the probability of observing the data $\boldsymbol{d}$ given $\boldsymbol{x}$, $p_{\boldsymbol{d}}(\boldsymbol{d} \mid \boldsymbol{x})$ (likelihood) can be computed from the PDF of $\boldsymbol{n}$ as

$$p_{\boldsymbol{d}}(\boldsymbol{d} \mid \boldsymbol{x}) = p_{\boldsymbol{n}}(\boldsymbol{d} - \boldsymbol{P}\boldsymbol{x} \mid \boldsymbol{x})$$

On the other hand, the *prior knowledge* on $\boldsymbol{x}$ is encoded as $\boldsymbol{x}$ being normally distributed with covariance matrix $\boldsymbol{\Gamma}^{-1}$ in the "$\boldsymbol{D}$ domain":

$$p_{\boldsymbol{x}}(\boldsymbol{D}\boldsymbol{x}) = \left|\det 2\pi\boldsymbol{\Gamma}^{-1}\right|^{-1/2} \exp\left(-\frac{1}{2}(\boldsymbol{D}\boldsymbol{x})^T\boldsymbol{\Gamma}(\boldsymbol{D}\boldsymbol{x})\right)$$

The posterior probability $p_{\boldsymbol{x}}(\boldsymbol{x} \mid \boldsymbol{d})$ is to be maximized, as we are looking for the most probable solution. Therefore, if the covariance matrices of the noise and of $\boldsymbol{D}\boldsymbol{x}$ are known, the multiplicative terms $|\det 2\pi\boldsymbol{\Sigma}|^{-1/2}$ and $\left|\det 2\pi\boldsymbol{\Gamma}^{-1}\right|^{-1/2}$ can be dropped. Then

$$p_{\boldsymbol{x}}(\boldsymbol{x} \mid \boldsymbol{d}) \sim \exp\left(-\frac{1}{2}(\boldsymbol{d}-\boldsymbol{P}\boldsymbol{x})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{d}-\boldsymbol{P}\boldsymbol{x})\right) \exp\left(-\frac{1}{2}(\boldsymbol{D}\boldsymbol{x})^T\boldsymbol{\Gamma}(\boldsymbol{D}\boldsymbol{x})\right)$$

Maximizing the posterior probability amounts to minimizing its negative logarithm. The MAP solution is thus given by

$$\underset{\boldsymbol{x}}{\operatorname{argmin}}\ \{-\log p_{\boldsymbol{x}}(\boldsymbol{x} \mid \boldsymbol{d})\} = \underset{\boldsymbol{x}}{\operatorname{argmin}}\ \left\{\frac{1}{2}\left\|\boldsymbol{P}\boldsymbol{x}-\boldsymbol{d}\right\|^2_{\boldsymbol{\Sigma}^{-1}} + \frac{1}{2}\left\|\boldsymbol{D}\boldsymbol{x}\right\|^2_{\boldsymbol{\Gamma}}\right\}$$

□

6.2 Definitions and properties

6.2.1 Spatial gradient and divergence

The following is a definition of the discrete spatial gradient.

Definition 15 (Spatial gradient)
Let `img` be an image with M columns and N lines. Let i, j denote zero-based integer indexes for the lines and columns, respectively. The gradient of `img`, without boundary handling, is a $2 \times N \times M$ image defined by

$$(\nabla\texttt{img})_y(i,j) = \begin{cases} \texttt{img}(i+1,j) - \texttt{img}(i,j) & \text{if } 0 \leq i < N \\ 0 & \text{otherwise} \end{cases}$$
$$(\nabla\texttt{img})_x(i,j) = \begin{cases} \texttt{img}(i,j+1) - \texttt{img}(i,j) & \text{if } 0 \leq j < M \\ 0 & \text{otherwise} \end{cases}$$

where $(\nabla\texttt{img})_y$ and $(\nabla\texttt{img})_x$ denote the $N \times M$ images corresponding to the gradient along lines and columns, respectively.
Extension to a volume is straightforward.

The following is a definition of the corresponding divergence operator.

Proposition 6.2.1 (Divergence operator)
The spatial gradient is a linear operator from $\mathbb{R}^{N\times M}$ to $\mathbb{R}^{2\times N\times M}$. Its adjoint is given by the opposite of the divergence operator div. Letting `gradimg`, be a "gradient img" (i.e `gradimg` $= \nabla$`img` for some `img`), the divergence of `gradimg` is defined by

$$\operatorname{div}\texttt{gradimg}(i,j) = \begin{cases} \texttt{gradimg}_y(i,j) - \texttt{gradimg}_y(i-1,j) & \text{if } 0 < i \\ \texttt{gradimg}_y(i,j) & \text{otherwise} \end{cases} + \begin{cases} \texttt{gradimg}_x(i,j) - \texttt{gradimg}_x(i,j-1) & \text{if } 0 < j \\ \texttt{gradimg}_x(i,j) & \text{otherwise} \end{cases}$$

6.2.2 Introduction to the wavelet transform

Wavelet transform is a signal processing tool which found a large range of applications since it was introduced decades ago. It basically came as a replacement for the Fourier transform when it comes to analyzing non-stationary signals. The one-dimensional Wavelet transform decomposes a time-dependent signal $\boldsymbol{s}(t)$ in a basis of signals $\left(\boldsymbol{\psi}_{(a,\tau)}(t)\right)_{(a,\tau)}$. Each basis function is a shifted and dilated version of a *mother wavelet* $\boldsymbol{\psi}$: $\boldsymbol{\psi}_{(a,\tau)}(t) = \frac{1}{\sqrt{a}}\boldsymbol{\psi}(\frac{t-\tau}{a})$. Designing a set of shifts τ and dilations $a > 0$ enables to analyze both the local discontinuities and long-term trend of a signal. In the time-frequency plane, the possible time shifts τ define how the time axis is tiled, while the possible dilations $a > 0$ define how the frequency axis is tiled. Representing the very low frequencies would entail to choose $s \to +\infty$; thus, in practice, a low-pass *scaling function* $\varphi(t)$ is designed to complete the time-frequency analysis.

The time-frequency analysis of $\boldsymbol{s}(t)$ is theoretically performed with the Continuous Wavelet Transform (CWT), defined as an Hermitian product in the space $\ell_2(\mathbb{R})$ of square integrable functions: $\langle \boldsymbol{s}(t)\,, \boldsymbol{\psi}_{a,\tau}(t)\rangle$. By designing a set of particular time shifts and dilations, and by and representing the set of inner products with a filter bank, an efficient implementation of the CWT named DWT is defined. The derivation of the Discrete Wavelet Transform from the Continuous Wavelet Transform is out of the scope of this paper, the reader can refer to [VH92], [She96], [Val99] and reference therein.

6.2.3 Common proximal mappings and Moreau identity

The Euclidean projection onto a convex set Ω is not straightforward in general. However, Propositions 6.2.2 and 6.2.3 give particular cases where such a projection is simple.

Proposition 6.2.2 (Euclidean projection onto a hyperrectangle)
The Euclidean projection of a vector $\boldsymbol{x} \in \mathbb{R}^n$ onto the hyperrectangle

$$\Omega_{\boldsymbol{a},\boldsymbol{b}} = \{\boldsymbol{x} \in \mathbb{R}^n,\ a_i \leq x_i \leq b_i\}$$

is given by

$$(P_{\Omega_{\boldsymbol{a},\boldsymbol{b}}}(\boldsymbol{x}))_i = \begin{cases} a_i & \text{if } x_i < a_i \\ x_i & \text{if } a_i \leq x_i \leq b_i \\ b_i & \text{if } b_i < x_i \end{cases}$$

Proposition 6.2.3 (Euclidean projection onto the intersection of hyperplanes)
Let $\boldsymbol{A} \int \mathbb{R}^{m\times n}$. The Euclidean projection of a vector $\boldsymbol{x} \in \mathbb{R}^n$ onto the intersection of hyperplanes

$$\Omega^{\boldsymbol{b}}_{\boldsymbol{A}} = \{\boldsymbol{x} \in \mathbb{R}^n,\ \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}\}$$

is given by

$$P_{\Omega^{\boldsymbol{b}}_{\boldsymbol{A}}}(\boldsymbol{x}) = \boldsymbol{x} + \boldsymbol{A}^*(\boldsymbol{A}\boldsymbol{A}^*)^{-1}(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x})$$

Proposition 6.2.3 is a direct application of the minimum norm solution of a system of equations (see Proposition 1.4.2).

Lastly, theorem (6.2.4) links the proximal of f and the proximal of its conjugate f^*.

Theorem 6.2.4 (Moreau identity)
Let f be a convex function and $\lambda > 0$. The following property holds

$$\text{prox}_{\gamma f^*}(\boldsymbol{x}) = \boldsymbol{x} - \gamma \,\text{prox}_{\gamma^{-1} f}\left(\gamma^{-1}\boldsymbol{x}\right) \tag{6.2.1}$$

This identity can be seen as a generalization of the projection onto orthogonal subspaces. Indeed, letting $f = i_\Omega$, then it can be shown that $f^* = i_{\Omega^\perp}$. Theorem (6.2.4) gives $\text{prox}(f) + \text{prox}(f^*) = \boldsymbol{I}$, i.e $P_\Omega + P_{\Omega^\perp} = \boldsymbol{I}$.

6.2.4 The power method

Algorithm 6.2.1 gives the outline of the power method to compute the largest eigenvalue of a linear operator $\boldsymbol{K}$.

Algorithm 6.2.1 Power method

$\boldsymbol{K}$: Operator
n : number of iterations

procedure PowerMethod(K, n)
  $\boldsymbol{x} = \boldsymbol{x}_0$ ▷ Initial vector
  for $k \leftarrow 1 \dots n$ **do**
    $\boldsymbol{x}_{k+1} = \boldsymbol{K}^T\boldsymbol{K}\boldsymbol{x}_k$ ▷ Apply operator $\boldsymbol{K}^T\boldsymbol{K}$
    $s = \|\boldsymbol{x}_{k+1}\|_2$
    $\boldsymbol{x}_{k+1} = \boldsymbol{x}_{k+1}/s$ ▷ Normalization step
  end for
  return $\sqrt{s}$
end procedure

Algorithm 6.2.1 involves $\boldsymbol{K}^T\boldsymbol{K}$ instead of $\boldsymbol{K}$, as $\boldsymbol{K}\boldsymbol{x}$ may not be in the same space as $\boldsymbol{x}$ (whereas $\boldsymbol{K}^T\boldsymbol{K}\boldsymbol{x}$ is). Fortunately, the largest singular value (in term of magnitude) of $\boldsymbol{K}$ is linked to the largest eigenvalue of $\boldsymbol{K}^T\boldsymbol{K}$ by $\sigma_{\max}(\boldsymbol{K}) = \sqrt{\lambda_{\max}(\boldsymbol{K}^T\boldsymbol{K})}$.

6.3 Derivations for section 3.4

This section gives more information and formal derivations for section 3.4 describing a filter-based proximal computation.

6.3.1 On the (non-)invertible property of $\boldsymbol{A}^T\boldsymbol{A}$

Let $\boldsymbol{A}$ be a $N_2 \times N$ matrix (N_2 lines, N columns). Let $\boldsymbol{a}_1, \dots, \boldsymbol{a}_N$ be the columns vectors of $\boldsymbol{A}$, so $\boldsymbol{A} = \begin{bmatrix}\boldsymbol{a}_1 \dots \boldsymbol{a}_N\end{bmatrix}$. The matrix $\boldsymbol{A}^T\boldsymbol{A}$ is called the Gramian matrix of the vectors $\boldsymbol{a}_1, \dots, \boldsymbol{a}_N$. It can be shown that $\det(\boldsymbol{A}^T\boldsymbol{A}) \neq 0$ if and only if these vectors are linearly independent. Therefore, conclusions can already be drawn depending on the shape of $\boldsymbol{A}$:

- If $N > N_2$ ($\boldsymbol{A}$ is "fat"), there are more vectors $\boldsymbol{a}_i$ than components of each vector; thus, the vectors $\boldsymbol{a}_i$ are linearly dependent. In this case, $\boldsymbol{A}^T\boldsymbol{A}$ is never invertible.
- If $N = N_2$ ($\boldsymbol{A}$ is square), $\boldsymbol{A}^T\boldsymbol{A}$ is invertible iff $\boldsymbol{A}$ is, and $\det(\boldsymbol{A}^T\boldsymbol{A}) = \det(\boldsymbol{A})^2$.

- If $N < N_2$ ($\boldsymbol{A}$ is "tall"), the vectors might be linearly independent, so $\boldsymbol{A}^T\boldsymbol{A}$ might be invertible.

In the latter case, as $\boldsymbol{A}$ maps vectors with N components to vectors with $N_2 > N$ components, it is said to be an *overcomplete transform* in some contexts. Thus, overcomplete mappings can be inverted provided that its column vectors are linearly independent.

In contrast, tomographic reconstruction from few projections is modelled by an operator $\boldsymbol{P}$ having more columns (number of pixels in the image) than rows (number of projection angles times number of detector bins horizontally). Therefore, $\boldsymbol{P}^T\boldsymbol{P}$ is never invertible in the context of tomographic reconstruction from few projections. In fact, numerical experiments suggest that even with $N_2 \geq N$ (i.e with a sufficient number of projections), $\boldsymbol{P}^T\boldsymbol{P}$ is not invertible; thus, $\boldsymbol{P}\boldsymbol{P}^T$ is not invertible either in the case of few projections.

6.3.2 Shift-invariance and convolution

Let $\mathcal{A}$ be an operator acting on continuous (1D) functions of $\mathbb{R} \to \mathbb{R}$. The operator $\mathcal{A}$ is said to be shift invariant if for all $\tau \in \mathbb{R}$,

$$\mathcal{A}[f_\tau](t) = \mathcal{A}[f](t - \tau)$$

where $f_\tau : t \mapsto f(t - \tau)$ is the function f delayed by τ. In other words, applying the operator $\mathcal{A}$ on a delayed function results on a delayed output (the "form" of the output stays the same). The operator therefore somewhat commutes with the "delay" operator[1]: `operator(delay) = delay(operator)`. This concept is the same in two or more dimensions: the time variable becomes spatial coordinates, and a time delay becomes a spatial shift. The fact that $\mathcal{A}$ commutes with the "delay operator" explains the name "shift invariant operator".

These operators can be used to model many physical systems. What make them appealing is that they can be characterized by a single function, the impulse response: for each linear time-invariant operator $\mathcal{A}$, there exists an impulse response $h_\mathcal{A}$ such that $\mathcal{A}[f] = h_\mathcal{A} * f$. In other words, these systems are entirely described by the convolution with a function. Knowing the impulse response $h_\mathcal{A}$ (which is a function) is knowing the entire system (modelled by an operator). It can be shown that $h_\mathcal{A} = \mathcal{A}[\delta]$ is the response to an "impulse" Dirac delta function.

Another important property of time-invariant operators is that they are diagonal in the Fourier basis, in other words, (co)sine functions are the eigenfunctions of such systems. Given an input function f, the response of the system $\mathcal{A}$ can be computed as $h_\mathcal{A} * f$, but also as $f_\mathcal{F} \odot h_{\mathcal{F}\mathcal{A}}$ – where $f_\mathcal{F}$ and $h_{\mathcal{F}\mathcal{A}}$ denote the functions f and $h_\mathcal{A}$ in the Fourier domain, respectively – which is a simple elementwise multiplication.

In the discrete setting, time/shift-invariant operators are also characterized by (discrete) convolution, and the (discrete) Fourier Transform diagonalizes these operators. In the (discrete) Fourier basis, the matrix-vector multiplication $\boldsymbol{A}\boldsymbol{x}$ becomes an elementwise product $\boldsymbol{h_{FA}} \odot \boldsymbol{x_F}$ with the same notations as previously. This property enables a major computational benefit: not only can be the operator $\boldsymbol{A}$ computed efficiently, but its (regularized) inverse can also be computed.

[1] the notion of delay operator can be formalized, for example, with distributions: $f_\tau(t) = f(t) * \delta(t - \tau)$

6.3.3 Regularized inversion and Wiener-Hunt deconvolution

This part formally derives the least-squares deconvolution with FT. The operator formalism used here only serves for the derivation; for example, "diagonal matrices" are never used in practice (they are replaced with vector-vector elementwise multiplications).

Let $\boldsymbol{A}$ be a shift invariant operator and $\boldsymbol{h_A} = \boldsymbol{h}$ its associated impulse response (see section 3.4.1). The DFT of a vector $\boldsymbol{x}$ is denoted $\boldsymbol{x_F} = \boldsymbol{F}\boldsymbol{x}$. The DFT of $\boldsymbol{h}$, named transfer function, is therefore $\boldsymbol{h_F}$.

Since $\boldsymbol{A}$ is diagonal in the Fourier basis, there exists a diagonal matrix $\boldsymbol{D_A}$ such that

$$\boldsymbol{A} = \boldsymbol{F}^* \boldsymbol{D_A} \boldsymbol{F} \tag{6.3.1}$$

The diagonal matrix $\boldsymbol{D_A}$ models the elementwise multiplication with the Fourier Transform of the impulse response $\boldsymbol{h}$: $\boldsymbol{h_F} = \mathrm{diag}(\boldsymbol{D_A})$. Since $\boldsymbol{x_F} = \boldsymbol{F}\boldsymbol{x}$, we have $\boldsymbol{x} = \boldsymbol{F}^*\boldsymbol{x_F}$, so the evaluation of $\boldsymbol{A}\boldsymbol{x}$ can therefore be computed as

$$\boldsymbol{A}\boldsymbol{x} = (\boldsymbol{F}^* \boldsymbol{D_A} \boldsymbol{F})(\boldsymbol{F}^* \boldsymbol{x_F}) = \boldsymbol{F}^* \boldsymbol{D_A} \boldsymbol{x_F} \tag{6.3.2}$$

in words: Fourier transform $\boldsymbol{x}$, multiply elementwise with the transfer function, and inverse Fourier transform the result. The pseudoinverse $(\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T$ can be therefore be computed as

$$\begin{aligned}
(\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T &= ((\boldsymbol{F}^* \boldsymbol{D_A} \boldsymbol{F})^* (\boldsymbol{F}^* \boldsymbol{D_A} \boldsymbol{F}))^{-1} (\boldsymbol{F}^* \boldsymbol{D_A} \boldsymbol{F})^* \\
&= \left(\boldsymbol{F}^* |\boldsymbol{D_A}|^2 \boldsymbol{F}\right)^{-1} (\boldsymbol{F}^* \boldsymbol{D_A} \boldsymbol{F})^* \\
&= \left(\boldsymbol{F}^* |\boldsymbol{D_A}|^{-2} \boldsymbol{F}\right) (\boldsymbol{F}^* \boldsymbol{D_A^*} \boldsymbol{F}) \\
&= \boldsymbol{F}^* |\boldsymbol{D_A}|^{-2} \boldsymbol{D_A^*} \boldsymbol{F} \\
&= \boldsymbol{F}^* \boldsymbol{D_A}^{-1} \boldsymbol{F}
\end{aligned} \tag{6.3.3}$$

where $|\boldsymbol{D_A}|^2 = \boldsymbol{D_A^*}\boldsymbol{D_A}$. The third line comes from the fact that $\boldsymbol{F}^* = \boldsymbol{F}^{-1}$ and $|\boldsymbol{D_A}|$ is symmetric (since diagonal) and real. Applying the pseudoinverse (6.3.3) on $\boldsymbol{x}$ consists, in words, in computing the DFT of $\boldsymbol{x}$, dividing by the FT of the impulse response, and retuning the inverse FT the result. Although it can be computed efficiently, this pseudoinverse has of course numerical issues: most convolution kernels $\boldsymbol{h}$ are such that the associated operator $\boldsymbol{A}^T\boldsymbol{A}$ (and therefore $\boldsymbol{D_A}$) is not invertible. This is why the approach "divide by the transfer function" is usually not a good approach.

The regularized pseudoinverse $(\boldsymbol{I} + \gamma\boldsymbol{A}^T\boldsymbol{A})^{-1}$ (where $\gamma > 0$) can be computed in the same fashion:

$$\begin{aligned}
(\boldsymbol{I} + \gamma\boldsymbol{A}^T\boldsymbol{A})^{-1} &= \left(\boldsymbol{I} + \gamma\boldsymbol{F}^* |\boldsymbol{D_A}|^2 \boldsymbol{F}\right)^{-1} \\
&= \left(\boldsymbol{F}^* (\boldsymbol{I} + \gamma |\boldsymbol{D_A}|^2) \boldsymbol{F}\right)^{-1} \\
&= \boldsymbol{F}^* \boldsymbol{H}^{-1} \boldsymbol{F}
\end{aligned} \tag{6.3.4}$$

where $\boldsymbol{H} = \boldsymbol{I} + \gamma|\boldsymbol{D_A}|^2$ is the diagonal matrix where the diagonal component k equals $1 + \gamma|\boldsymbol{h_F}(k)|^2$.

Lastly, we derive the inversion of the commonly used regularized least squares deblurring

$$\underset{\boldsymbol{x}}{\mathrm{argmin}} \left\{ \frac{1}{2} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 + \gamma \|\boldsymbol{R}\boldsymbol{x}\|_2^2 \right\} \tag{6.3.5}$$

where $\boldsymbol{R}$ is a regularization operator which is assumed to be circulant (i.e implementable as a convolution) – common choices are the spatial gradient or Laplacian. The optimality condition is

$$(\boldsymbol{A}^T\boldsymbol{A} + \gamma\boldsymbol{R}^T\boldsymbol{R})\boldsymbol{x} = \boldsymbol{A}^T\boldsymbol{b}$$

as $\boldsymbol{R}$ is assumed to be circulant, there exists a vector $\boldsymbol{h_R}$ such that for all $\boldsymbol{x}$: $\boldsymbol{Rx} = \boldsymbol{h_R} * \boldsymbol{x}$, and $\boldsymbol{R} = \boldsymbol{F}^* \boldsymbol{D_R} \boldsymbol{F}$ where $\boldsymbol{D_R} = \operatorname{diag}(\boldsymbol{F h_R})$. Therefore, the inverse filter can be computed:

$$\begin{aligned}(\boldsymbol{A}^T\boldsymbol{A} + \gamma \boldsymbol{R}^T\boldsymbol{R})^{-1}\boldsymbol{A}^T &= \left(\boldsymbol{F}^*|\boldsymbol{D_A}|^2\boldsymbol{F} + \gamma \boldsymbol{F}^*|\boldsymbol{D_R}|^2\boldsymbol{F}\right)^{-1}\boldsymbol{A}^T \\ &= \left(\boldsymbol{F}^*(|\boldsymbol{D_A}|^2 + \gamma|\boldsymbol{D_R}|^2)\boldsymbol{F}\right)^{-1}\boldsymbol{A}^T \\ &= \boldsymbol{F}^*\boldsymbol{H}^{-1}\boldsymbol{D_A}^*\boldsymbol{F}\end{aligned} \tag{6.3.6}$$

where

$$\boldsymbol{H} = |\boldsymbol{D_A}|^2 + \gamma|\boldsymbol{D_R}|^2 \tag{6.3.7}$$

is a diagonal matrix. In the context of image deblurring, the computation

$$\hat{\boldsymbol{x}} = \boldsymbol{F}^*\boldsymbol{H}^{-1}\boldsymbol{D_A}^*\boldsymbol{F} \tag{6.3.8}$$

is called *Wiener-Hunt deconvolution*. It consists in Fourier transforming $\boldsymbol{b}$, multiplying with the *Wiener filter* $\boldsymbol{H}^{-1}\boldsymbol{D_A}^*$, and taking the inverse FT. This method is implemented, for example, in the scikit-image Python module [Wal+14].

6.3.4 Convergence to the pseudoinverse

Let $f(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{Ax} - \boldsymbol{b}\|_2^2$ where $\boldsymbol{A}^T\boldsymbol{A}$ is invertible. The minimizer of f is obtained with the pseudoinverse of $\boldsymbol{A}$: $\boldsymbol{x}_{\mathrm{ML}} = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{b}$. The gradient-descent iteration

$$\begin{aligned}\boldsymbol{x}_{k+1} &= \boldsymbol{x}_k - \alpha\nabla f(\boldsymbol{x}_k) \\ &= \boldsymbol{x}_k - \alpha\boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{x}_k - \boldsymbol{b})\end{aligned} \tag{6.3.9}$$

converges to $\boldsymbol{x}_{\mathrm{ML}}$ if the gradient step size α is less than $1/\left\|\boldsymbol{A}^T\boldsymbol{A}\right\|$. The latter norm is equal to $\sqrt{\lambda_{\max}((\boldsymbol{A}^T\boldsymbol{A})^T(\boldsymbol{A}^T\boldsymbol{A})}} = \lambda_{\max}(\boldsymbol{A}^T\boldsymbol{A})$ (the last equality comes from the fact that $\boldsymbol{A}^T\boldsymbol{A}$ is symmetric and positive semi-definite). Iteration (6.3.9) can be rewritten as

$$\boldsymbol{x}_{k+1} = (\boldsymbol{I} - \alpha\boldsymbol{A}^T\boldsymbol{A})\boldsymbol{x}_k + \alpha\boldsymbol{A}^T\boldsymbol{b} \tag{6.3.10}$$

which defines an arithmetico geometric sequence; thus, is $\boldsymbol{x}_0 = \boldsymbol{0}$, we have

$$\boldsymbol{x}_n = \left[\alpha\sum_{k=0}^{n-1}(\boldsymbol{I} - \alpha\boldsymbol{A}^T\boldsymbol{A})^k\right]\boldsymbol{A}^T\boldsymbol{d} \tag{6.3.11}$$

In general, given a symmetric positive semi-definite matrix $\boldsymbol{U}$ such that $\|\boldsymbol{U}\| < 1$, it is easy to show[2] that

$$\sum_{k=0}^{\infty}\boldsymbol{U}^k = (\boldsymbol{I} - \boldsymbol{U})^{-1} \tag{6.3.12}$$

Therefore, Equation (6.3.12) can be applied to iterate (6.3.11) with $\boldsymbol{U} = \boldsymbol{I} - \alpha\boldsymbol{A}^T\boldsymbol{A}$: the term between square brackets converges to $\alpha(\boldsymbol{I} - (\boldsymbol{I} - \alpha\boldsymbol{A}\boldsymbol{P}^T\boldsymbol{A}))^{-1} = \alpha(\alpha\boldsymbol{A}^T\boldsymbol{A})^{-1} = (\boldsymbol{A}^T\boldsymbol{A})^{-1}$, (which is independent from α). Indeed, as the gradient step size α is precisely chosen so that $\alpha < 1/\left\|\boldsymbol{A}^T\boldsymbol{A}\right\|$, we have $\left\|\alpha\boldsymbol{A}^T\boldsymbol{A}\right\| < 1$. Now, $\left\|\boldsymbol{I} - \alpha\boldsymbol{A}^T\boldsymbol{A}\right\| = \lambda_{\max}((\boldsymbol{I} -$

[2] let $\boldsymbol{S}_n = \sum_{k=0}^{n-1}\boldsymbol{U}^k = \boldsymbol{I} + \boldsymbol{U} + \ldots + \boldsymbol{U}^{n-1}$, then $\boldsymbol{U}\boldsymbol{S}_n = \boldsymbol{U} + \ldots + \boldsymbol{U}^n$, so $\boldsymbol{S}_n - \boldsymbol{U}\boldsymbol{S}_n = \boldsymbol{I} - \boldsymbol{U}^n$, i.e $(\boldsymbol{I} - \boldsymbol{U})\boldsymbol{S}_n = \boldsymbol{I} - \boldsymbol{U}^n$. As $\|\boldsymbol{U}\| < 1$ and $\boldsymbol{U}$ is symmetric, we have $\|\boldsymbol{I} - \boldsymbol{U}\| = \lambda_{\max}(\boldsymbol{I} - \boldsymbol{U})$. On the other hand, $\lambda_{\max}(\boldsymbol{I} - \boldsymbol{U}) = 1 - \lambda_{\min}(\boldsymbol{U})$. As $\|\boldsymbol{U}\| < 1$, the last quantity is strictly positive, so $\boldsymbol{I} - \boldsymbol{U}$ is invertible. Therefore, $\boldsymbol{S}_n = (\boldsymbol{I} - \boldsymbol{U})^{-1}(\boldsymbol{I} - \boldsymbol{U}^n)$. As $\|\boldsymbol{U}\| < 1$, $\boldsymbol{U}^n \to \boldsymbol{0}$ when $n \to \infty$.

$\alpha \boldsymbol{A}^T\boldsymbol{A})^T(\boldsymbol{I}-\alpha\boldsymbol{A}^T\boldsymbol{A})) = \lambda_{\max}(\boldsymbol{I}-\alpha\boldsymbol{A}^T\boldsymbol{A})$[3], which in turn is equal to $1-\alpha\lambda_{\min}(\boldsymbol{A}^T\boldsymbol{A})$. This quantity $\left\|\boldsymbol{I}-\alpha\boldsymbol{A}^T\boldsymbol{A}\right\| = 1-\alpha\lambda_{\min}(\boldsymbol{A}^T\boldsymbol{A})$ must be less than one in order for $\sum_{k=0}^{\infty}(\boldsymbol{I}-\alpha\boldsymbol{A}^T\boldsymbol{A})^k$ to converge. This means that we must have $\lambda_{\min}(\boldsymbol{A}^T\boldsymbol{A}) > 0$, i.e $\boldsymbol{A}^T\boldsymbol{A}$ invertible, which was assumed to be the case.

In the context of tomographic reconstruction, $\boldsymbol{A}=\boldsymbol{P}$, and $\boldsymbol{P}^T\boldsymbol{P}$ is not invertible. This means that any iterative least squares algorithm is an unstable process ($\sum_{k=0}^{n}(\boldsymbol{I}-\alpha\boldsymbol{P}^T\boldsymbol{P})^k$ does not converge as $\lambda_{\min}(\boldsymbol{P}^T\boldsymbol{P})=0$), so iterations should be stopped early to avoid numerical issues (eg. noise amplification).

In order to compute the inversion of $f(\boldsymbol{x}) = \frac{\gamma}{2}\left\|\boldsymbol{P}\boldsymbol{x}-\boldsymbol{d}\right\| + \frac{1}{2}\left\|\boldsymbol{x}\right\|_2^2$, we can apply Equation (6.3.12) to $\boldsymbol{U} = (1-\alpha)\boldsymbol{I} - \alpha\gamma\boldsymbol{P}^T\boldsymbol{P}$. Indeed, writing the gradient descent iteration $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha\nabla f(\boldsymbol{x}_k)$ leads to $\boldsymbol{x}_{k+1} = ((1-\alpha)\boldsymbol{I}-\alpha\gamma\boldsymbol{P}^T\boldsymbol{P})\boldsymbol{x}_k + \alpha\boldsymbol{P}^T\boldsymbol{d}$. To do so, we must have

$$\begin{cases} \boldsymbol{I}-\boldsymbol{U} \text{ invertible} \\ \left\|\boldsymbol{U}\right\| < 1 \end{cases}$$

The first condition entails that $\boldsymbol{I}+\gamma\boldsymbol{P}^T\boldsymbol{P}$ is invertible, which is true for all $\gamma > 0$[4]. The second condition entails that $\left\|\boldsymbol{I}-\alpha(\boldsymbol{I}+\gamma\boldsymbol{P}^T\boldsymbol{P})\right\| < 1$. If α is chosen such that

$$\alpha < \frac{1}{1+\gamma\left\|\boldsymbol{P}^T\boldsymbol{P}\right\|} \tag{6.3.13}$$

then $\boldsymbol{U} = \boldsymbol{I}-\alpha(\boldsymbol{I}+\gamma\boldsymbol{P}^T\boldsymbol{P})$ is (symmetric) positive semi-definite. Therefore, $\left\|\boldsymbol{U}\right\| = 1-\alpha\lambda_{\min}(\boldsymbol{I}+\gamma\boldsymbol{P}^T\boldsymbol{P})$ which is less than one since $\alpha<1$ and $\boldsymbol{I}+\gamma\boldsymbol{P}^T\boldsymbol{P}$ is invertible. The series $\alpha\sum_{k=0}^{n}\boldsymbol{U}^k$ of Equation (6.3.11) then converges to $\alpha(\boldsymbol{I}-\boldsymbol{U})^{-1} = (\boldsymbol{I}+\gamma\boldsymbol{P}^T\boldsymbol{P})^{-1}$, which is independent from α (the only purpose of α is the gradient descent, the limit should not depend from it).

6.4 Supplementary material

6.4.1 The GSURE method for ISTA

This subsection derives the generalized Stein Unbiaised Risk Estimate (SURE) when solving the *synthesis* inverse problem

$$\underset{\boldsymbol{w}}{\operatorname{argmin}} \left\{ \frac{1}{2}\left\|\boldsymbol{A}\boldsymbol{D}^*\boldsymbol{w}-\boldsymbol{y}\right\|_2^2 + \lambda\left\|\boldsymbol{w}\right\|_1 \right\} \tag{6.4.1}$$

with the ISTA. We follow a simplified derivation of [GEE08]. The SURE method aims at finding the optimal λ so that the risk $\mathbb{E}\left[\boldsymbol{w}_\lambda - \boldsymbol{w}^\sharp\right]$ is minimized. Although the ground-truth value $\boldsymbol{w}^\sharp$ is unknown in practice, Skein showed that the risk can be estimated with only known data.

Let $\Phi_\lambda(\boldsymbol{y})$ be the abstract reconstruction process with parameter λ, i.e a solution of (6.4.1). This solution only depends on the data $\boldsymbol{y}$ and the regularization parameter λ. In [GEE08], the variable substitution $\boldsymbol{u} = \boldsymbol{A}^T\boldsymbol{y}$ is done invoking the Rao-Blackwell theorem (the MSE is smaller or equal for a sufficient statistics of $\boldsymbol{y}$). The (G)SURE estimate reads [Eld09]

$$\left\|\Phi_\lambda(\boldsymbol{u})\right\|_2^2 - 2\boldsymbol{x}_{ML}^T\Phi_\lambda(\boldsymbol{u}) + 2\,\mathrm{div}\,\Phi_\lambda(\boldsymbol{u}) \tag{6.4.2}$$

[3] since $\left\|\alpha\boldsymbol{A}^T\boldsymbol{A}\right\| < \left\|\boldsymbol{I}\right\| = 1$, so $\boldsymbol{I}-\alpha\boldsymbol{A}^T\boldsymbol{A}$ is (symmetric) positive semi-definite

[4] if $\lambda_1,\dots,\lambda_p$ denote the eigenvalues of $\boldsymbol{P}^T\boldsymbol{P}$, then $\det(\boldsymbol{I}+\gamma\boldsymbol{P}^T\boldsymbol{P}) = (1+\gamma\lambda_1)\dots(1+\gamma\lambda_p)$ which cannot be zero since $\lambda_i \geq 0$ ($\boldsymbol{P}^T\boldsymbol{P}$ is symmetric positive semi-definite).

where $\boldsymbol{x}_{ML} = \boldsymbol{D}^* \boldsymbol{w}_{ML}$ is the maximum likelihood solution of Equation (6.4.1). The div term is not a spatial divergence, it is a *temporal* divergence, i.e the derivative with respect $\boldsymbol{u}$, which makes the computation of (6.4.1) cumbersome.

The divergence is estimated on the fly with the ISTA iteration. Letting S_λ denote the soft thresholding operator with threshold λ, one step of ISTA is

$$\boldsymbol{w}_{n+1} = S_\lambda \left((\boldsymbol{I} - \gamma \boldsymbol{D}\boldsymbol{A}^T \boldsymbol{A}\boldsymbol{D}^*) \boldsymbol{w}_n + \gamma \boldsymbol{A}^T \boldsymbol{u} \right) \tag{6.4.3}$$

$$= S_\lambda \left(\boldsymbol{K}\boldsymbol{w}_n + \gamma \boldsymbol{A}^T \boldsymbol{u} \right) \tag{6.4.4}$$

where $\boldsymbol{K} = \boldsymbol{I} - \gamma \boldsymbol{D}\boldsymbol{A}^T \boldsymbol{A}\boldsymbol{D}^*$. After n iterations, the reconstruction process is $\Psi_\lambda(\boldsymbol{u}) = \boldsymbol{D}^* \boldsymbol{w}_n$, so

$$\operatorname{div} \Psi_\lambda(\boldsymbol{u}) = \operatorname{tr} \left(\boldsymbol{D}^* \frac{\mathrm{d}\boldsymbol{w}_n}{\mathrm{d}\boldsymbol{u}} \right) \tag{6.4.5}$$

Keeping the Jacobian is impracticable: if $\boldsymbol{u}$ has N components (pixels), and $\boldsymbol{w}_n$ has N_2 components (coefficients), then $\frac{\mathrm{d}\boldsymbol{w}_n}{\mathrm{d}\boldsymbol{u}}$ has $N \times N_2$ components, which cannot be stored. Instead, a stochastic estimator can be used: $\mathbb{E}\left[\boldsymbol{b}^T \boldsymbol{D}^* \frac{\mathrm{d}\boldsymbol{w}_n}{\mathrm{d}\boldsymbol{u}} \boldsymbol{b}\right]$ where $\boldsymbol{b}$ is a standard Gaussian noise. The update equation enabling to compute the trace is :

$$\frac{\mathrm{d}\boldsymbol{w}_{k+1}}{\mathrm{d}\boldsymbol{u}} \cdot \boldsymbol{b} = S'_\lambda(\boldsymbol{K}\boldsymbol{w}_k + \gamma \boldsymbol{A}^T \boldsymbol{u}) \left[\boldsymbol{K} \frac{\mathrm{d}\boldsymbol{w}_k}{\mathrm{d}\boldsymbol{u}} \cdot \boldsymbol{b} + \gamma \boldsymbol{A}^T \cdot \boldsymbol{b} \right] \tag{6.4.6}$$

where S'_λ is the derivative of the soft thresholding operator – which raise problems at $\pm\lambda$.

The work [GEE08] proposes a greedy algorithm for minimizing the Stein risk (6.4.2) using Equations (6.4.5) and (6.4.6). Although this method seems promising, we were not able to successfully apply this method for tomographic reconstruction. This is likely to come from a bad estimation of the Jacobian with the stochastic estimate. Nevertheless, we did not further investigate this problem, as automatic parameter selection was not our priority considering the discussion of section 2.4.6.

6.4.2 Pseudocode of optimized FISTA

This section provides a pseudocode of an accelerated version of FISTA described in [KF16]. Our numerical experiments showed that it is consistently twice faster than any other version of FISTA (original, relaxed, continuation method) for tomographic reconstruction. As analysed in 2.7, this optimization algorithm is especially interesting for a quadratic data fidelity term and a penalty which has a simple prox; which is the case for orthogonal wavelets regularization. As usual, $\boldsymbol{P}$ denotes the projection operator ($\boldsymbol{P}^T$ is the backprojection), and $\boldsymbol{W}$ is the forward DWT ($\boldsymbol{W}^T$ is its inverse).

Algorithm 6.4.1 FISTA with optimized step size

n : number of iterations
$\boldsymbol{d}$: data (sinogram)
λ: regularization parameter
L : largest eigenvalue of $\boldsymbol{P}^T\boldsymbol{P}$

procedure FISTA(n, τ, σ, θ)
  ▷ Initialize y_1 (zeros or initial reconstruction)
  for $k \leftarrow 1 \dots n$ **do**
    ▷ Update the gradient of smooth part at $\boldsymbol{y}_k$
    $\boldsymbol{g}_k = \boldsymbol{P}^T(\boldsymbol{P}\boldsymbol{y}_k - \boldsymbol{d})$
    ▷ Compute the prox of non-smooth part
    $\boldsymbol{w}_k = \boldsymbol{W}(\boldsymbol{y}_k - \frac{1}{L}\boldsymbol{g}_k)$
    $\boldsymbol{x}_{k+1} = \boldsymbol{W}^T\left(\mathcal{S}_{\lambda/L}\boldsymbol{w}_k\right)$ ▷ $\mathcal{S}$ is the soft thresholding operator
    $t = \frac{1}{2}\left(1 + \sqrt{1+4t^2}\right)$
    $\boldsymbol{y}_{k+1} = \boldsymbol{x}_{k+1} + \frac{t_k - 1}{t_{k+1}}(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k) + \frac{t_k}{t_{k+1}}(\boldsymbol{x}_{k+1} - \boldsymbol{y}_k)$
  end for
  return $\boldsymbol{x}_n$
end procedure

6.4.3 List of software packages used for tomographic reconstruction

The following is a list of softwares used for tomographic reconstruction at various institutes. This list is by no means a complete survey. Additional information can be found in [Aar+16] and at the website `tomopedia.github.io/software`.

Table 6.1: Tomography software packages

Software	**Geometries**	**Methods**	**License**	**Institutes**
tomopy	Parallel	Gridrec, ASTRA algorithms	GPLv3	APS, PSI, Elettra
ASTRA	Any	FBP, FDK, SIRT, SART, CGLS, EM	GPLv3	APS, PSI, Soleil
PyHST2	Parallel, conic	FBP, Gridrec, SIRT, TV, Wavelets, Dictionary	LGPL	ESRF
MMX-I	Parallel	FBP, ART	?	Soleil
TOFU	Parallel	FBP, Gridrec, SART, SIRT, TV, POCS	LGPL-3	KIT
CCPI ITR	Parallel	FBP, Landweber, CGLS, MLEM, OSEM, FISTA, Primal-Dual	?	Manchester X-ray facility
RTK	Parallel, conic	FDK, TV, Wavelets	Apache	Creatis, ?

Résumé de la thèse

Cette section résume en Français ce travail de manière plus détaillée.

Chapitre 1 - Tomographie continue et numérique

Dans ce premier chapitre, nous modélisons l'acquisition d'un *scan* tomographique par un opérateur mathématique appelé transformée de Radon $\mathcal{R}$. Cette modélisation est valable sous des conditions relativement contraignantes (faisceau monochromatique, gémoétrie parallèle) qui sont satisfaites dans notre environnement de travail, le Synchrotron. Le modèle intégral obtenu s'applique aussi bien pour la tomographie de transmission ou de contraste de phase.

Comprendre les propriétés de cet opérateur est le point de départ de la conception des méthodes de reconstruction. Nous donnons ainsi les principales propriétés de la transformée de Radon en quittant rapidement la modélisation continue pour un formalisme discret, compatible avec l'algèbre linéaire numérique et l'implémentation sur machine. La transformée de Radon continue $\mathcal{R}$ devient l'opérateur de projection $\boldsymbol{P}$ qui conserve la plupart des propriétés, dont la linéarité. Les théorèmes de coupe-projection et de rétroprojection filtrée sont énoncés de manière concise, et les difficultés pratiques d'implémentation sont soulignées.

La dernière partie est consacrée à l'opérateur de Calderón Λ, dont l'inverse correspond à un filtrage passe-haut via la fonction $\nu \mapsto |\nu|$ dans le domaine fréquentiel. La relation $\mathcal{R}^*\mathcal{R} = \Lambda^{-1}$, qui suggère que l'opérateur $\mathcal{R}^*\mathcal{R}$ a une structure particulière, est notamment exploitée dans le chapitre 3.

Chapitre 2 - Méthodes itératives régularisées

Méthodes itératives et problème inverse

Nous présentons d'abord les méthodes itératives classiques. Ces méthodes se sont révélées nécessaires pour pallier les limitations des méthodes directes (rétroprojection filtrée, coupe-projection) lorsque les données sont limitées et/ou bruitées, ou lorsque la géométrie ne permet pas l'utilisation d'une méthode directe. Dans ce cadre, la reconstruction est traitée comme un problème inverse $\boldsymbol{y} = \boldsymbol{P}\boldsymbol{x}$ avec $\boldsymbol{P}$ l'opérateur (discret) de projection, $\boldsymbol{y}$ les données acquises (sinogramme) et $\boldsymbol{x}$ l'image (ou volume) à reconstruire. La présence explicite de l'opérateur $\boldsymbol{P}$ permet de modéliser plus fidèlement le processus d'acquisition, ce qui n'était pas possible avec les méthodes directes.

Le cadre bayésien est ensuite utilisé pour unifier la présentation des méthodes de reconstruction. De manière générale, le problème $\boldsymbol{y} = \boldsymbol{P}\boldsymbol{x}$ étant mal posé (au sens de Hadamard), il s'agit de minimiser une fonction de coût $f(\boldsymbol{y}, \boldsymbol{x})$ mesurant la distance entre la projection de la solution $\boldsymbol{P}\boldsymbol{x}$ et les données acquises $\boldsymbol{y}$. Il est montré que cette fonction de coût est en fait associée au modèle de bruit, via la minimisation de l'opposé du log-vraissemblance : c'est l'approche *maximum de vraissemblance*.

Par exemple, pour un bruit blanc additif gaussien, la fonction de coût f est quadratique : $f(\boldsymbol{y}, \boldsymbol{x}) \sim \|\boldsymbol{P}\boldsymbol{x} - \boldsymbol{y}\|_2^2$, ce qui conduit à un algorithme de reconstruction aux moindres carrés. Dans le cas d'un bruit additif gaussien caractérisé par une matrice de covariance diagonale $\boldsymbol{\Sigma}$ dont chaque composante est égale à l'inverse de la somme de $\boldsymbol{P}$ le long des lignes, la fonction des coût est $f(\boldsymbol{y}, \boldsymbol{x}) \sim \|\boldsymbol{P}\boldsymbol{x} - \boldsymbol{y}\|_{\boldsymbol{\Sigma}}^2$ et l'algorithme SIRT converge vers son minimum. Dans le cas d'un modèle de bruit poissonien, minimiser l'opposé du log-vraissemblance conduit à l'algorithme EM.

Méthodes régularisées

Les méthodes itératives classiques revues précédemment souffrent d'un problème de stabilité. Par exemple, il est connu qu'une reconstruction aux moindres carrés sera "bruitée" si le nombre d'itérations est trop élevé. Ce comportement est lié au fait que la minimisation de la fonction de coût f est un problème mal conditionné : le processus d'optimisation amplifie le bruit qui peut être présent dans les données. Afin de "stabiliser" la procédure, une technique courante consiste à imposer des propriétés de la solution en modifiant la fonction de coût. Il s'agit alors d'optimiser la somme de deux termes : $f(\boldsymbol{y}, \boldsymbol{x}) + g(\boldsymbol{x})$ avec $g(\boldsymbol{x})$ le terme qui traduit les connaissances a priori sur la solution. Par exemple, la reconstruction aux moindres carrés avec une régularisation de Tikhonov vise à minimiser $\|\boldsymbol{P}\boldsymbol{x} - \boldsymbol{d}\|_2^2 + \rho \|\boldsymbol{x}\|_2^2$ avec $\rho > 0$ le paramètre de régularisation.

La méthode de régularisation correspond, dans le cadre bayésien, à l'approche "maximum a posteriori". La fonction g qui encode les propriétés de la solution correspond en fait à un modèle statistique sur la solution. Outre la régularisation de Tikhonov avec une "norme 2" ($\|\boldsymbol{x}\|_2^2$), un choix populaire est la "norme 1" ($\|\boldsymbol{D}\boldsymbol{x}\|_1$) pour un certain opérateur $\boldsymbol{D}$. Le succès de la reconstruction tomographique par minimisation de la norme 1 du gradient (variation totale $\|\nabla \boldsymbol{x}\|_1$) est expliqué par le cadre de l'acquisition comprimée.

Dans ce cadre, la régularisation par norme 1 permet une reconstruction exacte à partir de données très limitées/bruitées, fournissant de nouvelles conditions d'échantillonnage plus "efficaces" que le critère de Nyquist. De manière générale, un terme $g(\boldsymbol{x}) = \lambda \|\boldsymbol{D}\boldsymbol{x}\|_1$ va promouvoir une solution parcimonieuse (*sparse*) dans le domaine de la transformation $\boldsymbol{D}$. Trois représentations parcimonieuses sont revues dans ce travail: le gradient spatial utilisé pour la variation totale, les ondelettes et les *frames* (dictionnaires). Les deux ingrédients principaux de l'acquisition comprimée sont le choix d'une représentation parcimonieuse $\boldsymbol{D}$ et la résolution d'un problème non différentiable impliquant une "norme 1" $\|\boldsymbol{D}\boldsymbol{x}\|_1$.

Algorithmes d'optimisation convexe

Les méthodes régularisées qui nous intéressent dans ce travail, fondées sur le cadre de l'acquisition comprimée, nécessitent la minimisation d'une fonction non différentiable. Cette partie fait une revue des algorithmes récents utilisés à cet effet. Le formalisme proximal (mettant en jeu l'opérateur proximal $\text{prox}_f(\hat{\boldsymbol{x}}) = \underset{\boldsymbol{x}}{\text{argmin}} \left\{1/2 \|\boldsymbol{x} - \hat{\boldsymbol{x}}\|_2^2 + f(\boldsymbol{x})\right\}$) unifie les algorithmes de l'état de l'art tout en donnant un cadre très adaptable pour de nombreuses "formes" de fonctions de coût. Trois classes d'algorithmes sont considérées pour minimiser $f + g$: *forward-backward* (FB), Lagrangien augmenté (LA) et Primal-Dual (PD). Chacune a des conditions d'applications "privilégiées": pour FB, il est nécessaire que f soit différentiable et préférable que g soit à proximal simple ; pour (LA) il est préférable que f et g soient toutes deux à proximal simple. La classe (PD) est plus adaptable, mais le problème de minimisation doit être ré-écrit.

Chapitre 3 - Implémentation efficace de méthodes régularisées de reconstruction

Ce chapitre décrit l'implémentation efficace de méthodes régularisées de reconstruction.

Reconstruction rapide régularisée par variation totale

Cette partie discute de l'implémentation efficace de la méthode de reconstruction régularisée par variation totale (Total Variation). Le problème d'optimisation associé est

$$\underset{\boldsymbol{x}}{\operatorname{argmin}} \left\{ \frac{1}{2} \|\boldsymbol{P}\boldsymbol{x} - \boldsymbol{d}\|_2^2 + \lambda \|\nabla \boldsymbol{x}\|_1 \right\}$$

Afin de résoudre ce problème sans lissage de la norme 1, trois algorithmes d'optimisation sont analysés: FISTA, ADMM et C-P. L'algorithme C-P a une vitesse de convergence plus faible que les deux autres, mais il a l'avantage de ne pas nécessiter de sous-itérations pour résoudre un sous-problème. Cette caractéristique est intéressante car le temps d'exécution est dominé par le nombre d'opérations "matrice-vecteur", en particulier projection-rétroprojection. De plus, nous exploitons la propriété $(\mathcal{R}^*\mathcal{R})^{-1} = \Lambda$ en géométrie parallèle, ce qui permet d'accélérer la vitesse de convergence en utilisant à chaque itération la rétroprojection filtrée (FBP) au lieu de la simple rétroprojection. Nous retenons donc cet algorithme pour une implémentation sur cible parallèle (GPU), et nous montrons sa versatilité par l'incorporation de la contrainte de positivité.

La comparaison avec une implémentation de référence de FISTA sur GPU valide la supériorité de l'algorithme C-P pour la reconstruction par variation totale. La vitesse obtenue permet de reconstruire des jeux de données standard (volume de 2000^3 voxels) en un temps de l'ordre d'une heure. Des exemples d'application sur des données réelles sont donnés.

Reconstruction régularisée par ondelettes

L'implémentation de la reconstruction régularisée par ondelettes vient du constant suivant : d'une part, la reconstruction par TV est rapide mais surtout adaptée aux images constantes par morceaux ; d'autre part la reconstruction par dictionnaire (déjà présente dans PyHST) est plus lente et nécessite "d'apprendre" un dictionnaire à partir d'une reconstruction de bonne qualitée. Cependant, une telle reconstruction n'est pas toujours disponible. Ainsi, la régularisation par ondelettes se présente comme un compromis entre ces deux régularisations. En effet, la transformée en ondelettes est connue pour être une "transformée parcimonieuse" pour les images naturelles ; autrement dit, les images naturelles ont peu de grands coefficients dans une base d'ondelettes du fait de leur corrélation intrinsèque.

Des expériences numériques montrent que la transformée en ondelettes discrète (DWT), en particulier la transformée de Haar, est effectivement une bonne transformée pour la reconstruction régularisée. Dans le cadre de l'acquisition comprimée, la DWT présente une bonne incohérence avec l'opérateur de projection, et "compresse" les images de manière intéressante. Dans le cadre calculatoire, la DWT est l'une des transformées les plus rapides en traitement d'image.

Nous présentons l'implémentation d'une bibliothèque de transformée en ondelettes discrète sur GPU. Le but est de fournir une transformée en ondelettes 2D rapide, adaptable à toutes les tailles d'images et tous les types de transformées (directe, inverse, décimée, stationnaire), et facile à intégrer dans un projet. Une telle bibliothèque n'était pas présente au moment de ce travail, c'est pourquoi il a été entrepris de la développer. L'implémentation est validée et présente une vitesse d'exécution compatible avec un traitement à haut débit.

Cette bibliothèque est ensuite utilisée pour la tomographie régularisée par DWT en prenant en compte des considérations pratiques comme les artefacts de seuillage. Des exemples d'applications sur des données réelles sont donnés. La DWT est aussi utilisée pour une méthode de correction d'artefacts en anneaux par pré-traitement du sinogramme.

Algorithme de sous-gradient conjugué

La régularisation par variation totale et par ondelettes correspondent à des formulations *d'analyse*, i.e la variable du problème d'optimisation est une image. La reconstruction par dictionnaire est quant à elle fondée sur une formulation de *synthèse*, i.e la variable du problème d'optimisation est un ensemble de coefficients. Nous présentons ici un nouvel algorithme dédié au problème "LASSO"

$$\underset{\boldsymbol{w}}{\operatorname{argmin}} \left\{ \frac{1}{2} \left\| \boldsymbol{P}\boldsymbol{D}^T\boldsymbol{w} - \boldsymbol{d} \right\|_2^2 + \lambda \left\| \boldsymbol{w} \right\|_1 \right\}$$

où $\boldsymbol{D}^T$ (resp. $\boldsymbol{D}$) est l'opérateur de synthèse (resp. analyse) du dictionnaire, et $\boldsymbol{w}$ est un ensemble de coefficients tel que $\boldsymbol{D}^T\boldsymbol{w}$ est une image. Dans certains cas, en particulier dans le cas de la correction d'anneaux combinée à la reconstruction (chapitre 4), on remarque en effet que les algorithmes usuels (FISTA) convergent relativement lentement pour un tel problème.

L'idée de base de ce nouvel algorithme est d'adapter la méthode du gradient conjugué, réputée rapide pour un problème de moindres carrés linéaires, au problème LASSO. Nous utilisons le sous-gradient, la généralisation du gradient, avec une règle délicate en zéro pour faciliter la convergence vers une solution parcimonieuse. En utilisant cette règle et des préconditioneurs, nous parvenons à construire un ensemble de directions conjuguées.

Les essais numériques sur un problème très mal conditionné ainsi qu'en tomographie suggèrent que l'algorithme proposé dépasse les algorithmes de l'état de l'art. Cet algorithme est notamment utilisé pour la méthode de correction des artefacts en anneaux combinée avec la reconstruction présentée au chapitre 4.

Calcul rapide du proximal de $1/2 \left\| \boldsymbol{P}\boldsymbol{x} - \boldsymbol{d} \right\|_2^2$

L'algorithme ADMM présente la meilleure vitesse de convergence parmi les algorithmes testés pour la reconstruction régularisée. Cependant, le coût par itération est élevée car il doit calculer le proximal de $\boldsymbol{x} \mapsto \gamma/2 \left\| \boldsymbol{P}\boldsymbol{x} - \boldsymbol{d} \right\|_2^2$, ce qui revient à évaluer l'inverse de $(\boldsymbol{I} + \gamma \boldsymbol{P}^T\boldsymbol{P})$ à chaque itération, ce qui est fait de manière itérative. Dans cette partie, nous montrons que la propriété $(\mathcal{R}^*\mathcal{R})^{-1} = \Lambda$ conduit à un calcul rapide de ce proximal.

L'idée est que $\boldsymbol{P}^T\boldsymbol{P}$ est (presque) invariant par translation, ce qui permet de caractériser cet opérateur par sa seule réponse impulsionnelle. L'inversion de l'opérateur $(\boldsymbol{I} + \gamma \boldsymbol{P}^T\boldsymbol{P})$ se réduit ainsi à une succession d'évaluations de cet opérateur sur un Dirac. Nous montrons que cette approche permet effectivement d'approcher le calcul du vrai proximal, et donc d'accélérer la minimisation par ADMM.

Chapitre 4 - Correction d'artefacts et tomographie locale

Dans ce chapitre, nous montrons que le formalisme de reconstruction régularisée peut être étendu pour prendre en compte les artefacts de reconstruction.

Correction d'artefacts en anneaux

Les artefacts en anneaux sont courants en tomographie et rendent les traitement post-reconstruction difficiles. Quelles que soient leurs causes, ces artefacts apparaissent dans le sinogramme comme un bruit structuré en lignes verticales quasi-constantes. Dans cette partie, nous développons une méthode de reconstruction simultanée à la correction de ces

artefacts. Plus précisément, nous étendons le formalisme de reconstruction régularisée

$$\underset{\boldsymbol{x},\boldsymbol{r}}{\operatorname{argmin}} \left\{ \frac{1}{2} \left\| \boldsymbol{P}\boldsymbol{x} + \boldsymbol{U}\boldsymbol{r} - \boldsymbol{d} \right\|_2^2 + \lambda \left\| \boldsymbol{D}\boldsymbol{x} \right\|_1 + \lambda_r \left\| \boldsymbol{r} \right\|_1 \right\}$$

avec $\boldsymbol{r}$ une nouvelle variable dont le but est de "capturer" les artefacts en lignes, $\boldsymbol{U}$ l'opérateur qui ajoute une valeur sur chaque ligne du sinogramme, et $\boldsymbol{D}$ une régularisation spatiale.

Nous implémentons cette méthode pour les trois régularisations présentées dans ce travail (variation totale, ondelettes et synthèse par dictionnaire). Des expériences sur des données simulées et réelles montrent que cette méthode est compétitive avec les méthodes de l'état de l'art.

Tomographie locale

En tomographie locale, le champ de vue du détecteur n'est pas assez large pour couvrir tout l'objet scanné, résultant en des données "incomplètes". Des résultats théoriques montrent cependant que sous certaines hypothèses, la région d'intérêt peut être reconstruite. La FBP avec extension du sinogramme fournit en général des résultats satisfaisants mais souffre de deux inconvénients : la reconstruction n'est pas quantitative (ce qui est un problème fondamental de la tomographie locale) et présente un artefact de "coupe" (*cupping*). Dans cette partie, nous nous focalisons sur une méthode visant à atténuer l'artefact de *cupping* caractéristique d'une reconstruction de données locales avec la méthode FBP.

La méthode proposée part de d'une reconstruction FBP $\boldsymbol{r}_0$ et corrige les basses fréquences. Il s'agit donc de calculer une image $\boldsymbol{c}$ telle que $\boldsymbol{r}_0 + \boldsymbol{c}$ corrige l'artefact de cupping. L'image à reconstruire est étendue afin d'estimer grossièrement l'extérieur de la région d'intérêt. Afin de réduire le temps de calcul, et comme seule la correction des basses fréquences nous intéresse, l'image $\boldsymbol{c}$ est représentée dans une base de *blobs* gaussiens. D'autre part, pour projeter et rétroprojeter les coefficients de cette représentation, nous utilisons une représentation des opérateurs de projection et rétroprojection sous forme de matrice creuse au format CSR.

Nous utilisons une hypothèse de zone connue, peu contraignante en pratique, pour contraindre la correction des basses fréquences. Des expériences numériques montrent que l'artefact de *cupping* est effectivement fortement atténué, et nous retrouvons l'aspect quantitatif de la reconstruction. La méthode a été également validée sur des données expérimentales en tomographie d'absorption.