SMAI-JCM
SMAI Journal of Computational Mathematics

Multiresolution greedy algorithm dedicated to reflective tomography

Jean-Baptiste Bellet

Volume 4 (2018), p. 259-296.
<http://smai-jcm.cedram.org/item?id=SMAI-JCM_2018__4__259_0>

SMAI Journal of Computational Mathematics
Vol. 4, 259-296 (2018)

Multiresolution greedy algorithm dedicated to reflective tomography

Jean-Baptiste Bellet [1]

[1] Université de Lorraine, CNRS, IECL, F-57000 Metz, France
E-mail address: jean-baptiste.bellet@univ-lorraine.fr.

Abstract. Reflective tomography recovers the surfaces of a scene to be imaged, from optical images: a tomographic algorithm computes a full volumic reconstruction and then the surfaces are extracted from this reconstruction. For better performance, we would like to avoid computing accurately the full reconstruction, and we want to focus computations on the sought surfaces. For that purpose we propose an iterative multiresolution process. The initialization computes a coarse reconstruction, and the iterations refines it. To identify the voxels to be refined, we take advantage of the asymptotic behaviour of the reconstruction, with respect to its cut-off frequency: it discriminates the surfaces to be extracted. By the way the proposed algorithm is greedy: each iteration maximizes the accumulated intensity of the selected voxels, with prescribed volume. The combination of the complexity analysis and the numerical results shows that this novel approach succeeds in reconstructing surfaces and is relatively efficient compared with the standard method. These works pave the way towards accelerated algorithms in reflective tomography. They can be extended to a general class of problems concerning the determination of asymptotically discriminated sets, what is related to the computation of singular support of distributions.

2010 Mathematics Subject Classification. 78A97, 94A12, 65B99, 65Y20.

Keywords. Computational optics, reconstruction, acceleration, complexity.

1. Introduction

1.1. Reflective Tomography

In optics, reflective tomography reconstructs a scene from images of scattered intensity [8]. The idea is to apply a tomographic algorithm on reflective projections [13, 11, 6, 2]: a filtered backprojection. It offers several ways of representing the scene [7, 3, 5, 9] in three-dimensional optical imaging. The most intense values of the reconstruction are located near the original surfaces, up to artifacts. The surfaces are extracted using this information, for example by thresholding [4].

1.2. Goal

This article is part of a search for improvement of the algorithms of reflective tomography. Here we would especially like to tackle the computational efforts. The observation is that the technique computes a filtered backprojection, and thus it computes a full volume, whereas we look for surfaces only. To gain performance, we would like to concentrate fine calculations close to surfaces, and limit calculations far from surfaces. The chicken and the egg problem: compute only the voxels close to surfaces that are unknown before computing all the voxels. In this article we design a new algorithm that helps to solve this problem. This is a first attempt to accelerate the solver of reflective tomography using a sequential approach.

1.3. Proposed strategy

To achieve this, we add *a priori* about the reconstruction. Surfacic extraction by thresholding assumes essentially that the intense values of the reconstructed volume constitute the surfaces. We have got

recently more quantitative results about the reconstruction [1]. Assuming essentially that the input is piecewise smooth, these results formulate the reconstruction as a function of the cut-off frequency of the filtering, when this frequency plays the role of an asymptotic parameter. We propose to take advantage of this dependence to compute a reconstruction which is refined near the surfaces, iteration after iteration. This approach will compute reconstructions for several cut-off frequencies, and thus several resolutions, trying to focus the highest resolutions on surfaces. Such a multiresolution refinement has similarities with an algorithm derived in [15]; but the framework and the core of the solver are different.

1.4. **Main results**

The most important result of this article is a new algorithm dedicated to reflective tomography: Algorithm 6 page 272. It aims at answering efficiently to the following question: compute a small set of voxels which represents the surfaces of the scene, and whose cardinality is set in advance. It is a multiresolution greedy algorithm, guided by the asymptotic behaviour of the reconstruction under the assumptions of [1]. We also derive complexity bounds: Table 3.1; they show that the method is hoped to be competitive, at least for advantageous occurrences. And we extend these two-dimensional results to a three-dimensional set-up: Algorithm 9 and Table 5.1. Numerical examples show that the method is relevant on synthesis images with specular reflection and diffusion, simulated by the Gouraud model [10], even if the images are corrupted with quite a strong speckle noise. By the way the principle of the multiresolution method concerns a much more general class of problems that we introduce: it deals with the efficient determination of a set from asymptotic discrimination.

1.5. **Organization**

This paper is organized as follows. We first recall the principle of reflective tomography; we formulate the algorithm that is the reference method in the field. Then we design a new algorithm in a two dimensional set-up, and we analyze its complexity. We test this algorithm on cases of imaging with occlusions. To finish with, we extend these results in three dimensions, and we show the relevance of the new algorithm, by comparing it with the standard algorithm on simulated optical images, with and without noise.

2. **Surface extraction in reflective tomography**

2.1. **Reflective tomography**

We recall what 2D reflective tomography is.

2.1.1. *Record: reflectogram*

We consider a scene whose objects are opaque; we collect reflective projections similarly as [2]. We assume that we measure in this way a *reflectogram* $P : (\theta, s) \in \Theta \times [-R, R] \mapsto P(\theta, s)$, where $\Theta = [0, |\Theta|] \subset [0, 2\pi]$ is an interval of angles, and $R > 0$ is the half-width of a screen. More precisely let $\partial O = \partial W \cup \partial S$, where ∂W denotes the (boundary of the) wall and ∂S denotes the (boundary of the) objects of the scene. As in Figure 2.1, for every angle $\theta \in \Theta$, and for every radial value $s \in [-R, R]$, we consider the ray $x \cdot \boldsymbol{\theta} = s$, where $\boldsymbol{\theta} = (\cos\theta, \sin\theta)$; the scene is projected along this ray, along the direction $-\boldsymbol{\theta}^\perp = (\sin\theta, -\cos\theta)$. We record a *reflective projection* on a screen which is parallel to $\boldsymbol{\theta}$, and between the objects and the wall:

$$P(\theta, s) = f(\theta, y(\theta, s)), s \in [-R, R], \quad y(\theta, s) \in \{x \in \partial O : x \cdot \boldsymbol{\theta} = s\}, \quad f : (\Theta, \partial O) \to \mathbb{R}, \tag{2.1}$$

where $y(\theta, s)$ denotes the visible point, and f is the "physics" of the problem. The opacity assumption means that for the ray $x \cdot \boldsymbol{\theta} = s$, the visible point is the last point of ∂O when we travel along the ray in the direction $-\boldsymbol{\theta}^\perp$. The reflectogram P is the collection of the reflective projections: $P \equiv (P(\theta, \cdot))_{\theta \in \Theta}$. The function f can take different forms depending on the application. In optics, the measurement is the power per unit area incident on the sensor: the irradiance [12]. So $P(\theta, s) = f(\theta, y(\theta, s))$ represents the amount of light that is reflected from $y(\theta, s)$ to the sensor. Other example: if $f|_{\partial S} = 1$ and $f|_{\partial W} = 0$, we measure 1 if the ray $x \cdot \boldsymbol{\theta} = s$ intersects the scene ∂S, and 0 otherwise, and then P represents the silhouettes of ∂S [14]. Silhouettes can be obtained by binarising optical data. For the numerical examples of this article, we will always assume that the wall is *black*: $f|_{\partial W} = 0$.

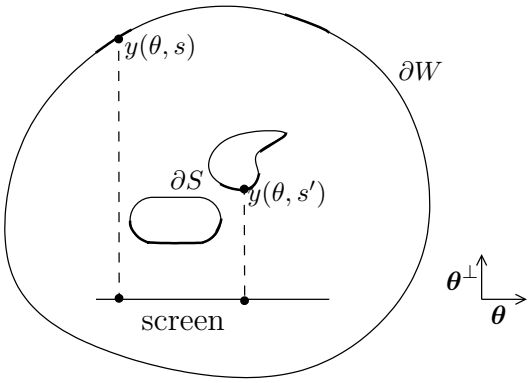

Figure 2.1. Reflective projection for an angle $\theta \in \Theta$: the wall ∂W and the scene ∂S are projected in the direction $-\boldsymbol{\theta}^\perp$ on a screen, along the rays $x \cdot \boldsymbol{\theta} = s, s \in [-R, R]$. The set of the projected visible points, $y(\theta, s), s \in [-R, R]$, is represented in bold. The dashed lines represent two rays of projection, from the visible points $y(\theta, s)$ and $y(\theta, s')$, to the screen.

2.1.2. *Imaging by tomography*

Then we consider the following imaging problem.

Problem 1. Reconstruct the scene ∂S, from the knowledge of the reflectogram P on $\Theta \times [-R, R]$.

The acquisition geometry recalls the acquisition of the Radon transform in transmission tomography: the scene is projected along parallel rays. Thus the filtered backprojection, which is a Radon inversion, was introduced to solve this problem [13, 2]. The denomination "reflective tomography" comes from that point: the method applies a tomographic solver on reflective data. More precisely for $\psi(\sigma) = |\sigma|$ and $\Omega > 0$, we define a filter whose frequency window is contained in $[-\Omega, \Omega]$:

$$\psi_\Omega(s) := \frac{1}{2\Omega} \int_{-\Omega}^{\Omega} \psi(\frac{\sigma}{\Omega}) \mathrm{e}^{\mathrm{i}\sigma s} \mathrm{d}\sigma = \int_0^1 \sigma \cos(\Omega \sigma s) \mathrm{d}\sigma.$$

We represent the scene ∂S using the ψ_Ω-filtered backprojection of P:

Definition 2.1. For every $\Omega > 0$, the ψ_Ω-filtered backprojection of P is the two-dimensional function:

$$\mathcal{R}^*[P \star \psi_\Omega] : x \in \mathbb{R}^2 \mapsto \mathcal{R}^*[P \star \psi_\Omega](x) = \int_\Theta P(\theta, \cdot) \star \psi_\Omega(x \cdot \boldsymbol{\theta}) \mathrm{d}\theta,$$

where $\star$ denotes the convolution with respect to the radial variable:

$$P(\theta, \cdot) \star \psi_\Omega(t) = \int_{-R}^{R} P(\theta, s) \psi_\Omega(t - s) \mathrm{d}s.$$

Of course, the *reconstruction* $\mathcal{R}^*[P \star \psi_\Omega]$ is only a heuristic representation of ∂S, because P is not a Radon transform. The half-width Ω of the frequency window plays the role of a resolution parameter. The reconstruction depends on that parameter. We will use this dependency later in the article.

2.2. **Discretization**

We recall a discrete version of the tomographic algorithm.

2.2.1. *Records*

We assume that the angular interval Θ is discretized on the grid of the $\theta_j = (j-1)\Delta\theta, 1 \leqslant j \leqslant m$, with $\Delta\theta = \frac{|\Theta|}{m}$. We assume that the radial interval $[-R, R]$ is discretized on the grid of the $t_j = -R+(j-1)\Delta t, 1 \leqslant j \leqslant n$, with $\Delta t = \frac{2R}{n}$. We assume that the reflectogram $P(\theta, t)$ is only known on the grid of the (θ_j, t_k); the record is thus a uniform discretization of the reflectogram P. We approximate the filtered backprojection $\mathcal{R}^*[P \star \psi_\Omega](x)$, for x scanning the grid of the $(t_i, t_j), 1 \leqslant i, j \leqslant n$.

2.2.2. *Discretization of the filtering*

First the filtering is computed for every angle θ of the grid. Let $\Delta\omega = \frac{\pi}{R}$, $\alpha = \lceil \frac{-n}{2} \rceil \Delta\omega, \beta = \lceil \frac{n}{2} \rceil \Delta\omega$; we assume $[-\Omega, \Omega) \subset [\alpha, \beta)$, i.e. $\Omega \leqslant -\alpha = \lfloor \frac{n}{2} \rfloor \Delta\omega$. Let $\omega_k = \alpha + (k-1)\Delta\omega, 1 \leqslant k \leqslant n$. Then for $t = t_l, 1 \leqslant l \leqslant n$,

$$P(\theta, \cdot) \star \psi_\Omega(t) = \frac{\pi}{\Omega} \int_{-R}^{R} P(\theta, s) \frac{1}{2\pi} \int_{\alpha}^{\beta} \left(\psi \mathbb{1}_{[-1,1)} \right) \left(\frac{\sigma}{\Omega} \right) \mathrm{e}^{\mathrm{i}\sigma(t-s)} \mathrm{d}\sigma \mathrm{d}s.$$

Notation. For every vector u, whose size is n, the forward, resp. inverse, discrete Fourier transform of u is: $\mathrm{FFT}\, u := \left[\mathrm{e}^{-\mathrm{i}\frac{2\pi}{n}(j-1)(k-1)} \right]_{1 \leqslant k,j \leqslant n} u, \quad \mathrm{IFFT}\, u := \frac{1}{n} \left[\mathrm{e}^{\mathrm{i}\frac{2\pi}{n}(j-1)(k-1)} \right]_{1 \leqslant j,k \leqslant n} u.$

These notations emphasize that such operations are computed using fast implementations: Fast Fourier Transform and Inverse Fast Fourier Transform.

Notation. Let $\hat{\star}$ denote a product component by component, composed by a shift: for all u, v, of size n,

$$u \hat{\star} v := (u_1 v_{\lfloor \frac{n}{2} \rfloor + 1}, u_2 v_{\lfloor \frac{n}{2} \rfloor + 2}, \ldots, u_{\lceil \frac{n}{2} \rceil} v_n, u_{\lceil \frac{n}{2} \rceil + 1} v_1, u_{\lceil \frac{n}{2} \rceil + 2} v_2, \ldots, u_n v_{\lfloor \frac{n}{2} \rfloor}).$$

By the left Riemann sum, the filtering is approximated by a discrete Fourier inversion of a discrete spectral product:

$$[P(\theta, \cdot) \star \psi_\Omega(t_l)]_{1 \leqslant l \leqslant n} \approx \frac{\pi}{\Omega} \mathrm{IFFT} \left[\mathrm{FFT}[P(\theta, t_j)]_{1 \leqslant j \leqslant n} \hat{\star} \left[\left(\psi \mathbb{1}_{[-1,1)} \right) \left(\frac{\omega_k}{\Omega} \right) \right]_{1 \leqslant k \leqslant n} \right]. \tag{2.2}$$

This first step approximates the $\tilde{P}_\Omega(\theta_j, t_l) = P(\theta_j, \cdot) \star \psi_\Omega(t_l)$, $1 \leqslant j \leqslant m$, $1 \leqslant l \leqslant n$, for $\Omega \leqslant \lfloor \frac{n}{2} \rfloor \Delta\omega$. The `filtering` function, written hereafter in pseudo-code, realizes this step for $\Omega = \lfloor \frac{n}{2} \rfloor \frac{\pi}{R}$.

2.2.3. *Discretization of the backprojection*

Then we discretize the backprojection. By the left Riemann sum:

$$\mathcal{R}^*[\tilde{P}_\Omega](x) \approx \Delta\theta \sum_{j=1}^{m} \tilde{P}_\Omega(\theta_j, x \cdot \boldsymbol{\theta}_j).$$

The $\tilde{P}_\Omega(\theta_j, x\cdot\boldsymbol{\theta}_j)$ must be interpolated on the grid of the t_l. We assume that $t_1 < x\cdot\boldsymbol{\theta}_j \leqslant t_n, 1 \leqslant j \leqslant m$; this assumption is satisfied if $|x| \leqslant R - \Delta t$. Then by linear interpolation:

$$\tilde{P}_\Omega(\theta_j, x\cdot\boldsymbol{\theta}_j) \approx \tilde{P}_\Omega(\theta_j, t_l)\frac{x\cdot\boldsymbol{\theta}_j - t_{l+1}}{t_l - t_{l+1}} + \tilde{P}_\Omega(\theta_j, t_{l+1})\frac{x\cdot\boldsymbol{\theta}_j - t_l}{t_{l+1} - t_l}, \quad 1 \leqslant l := \left\lceil \frac{x\cdot\boldsymbol{\theta}_j - a}{\Delta t} \right\rceil \leqslant n-1.$$

Hence this second step computes:

$$\mathcal{R}^*[P \star \psi_\Omega](x) \approx \frac{\Delta\theta}{\Delta t}\sum_{j=1}^{m} -\tilde{P}_\Omega(\theta_j, t_l)(x\cdot\boldsymbol{\theta}_j - t_{l+1}) + \tilde{P}_\Omega(\theta_j, t_{l+1})(x\cdot\boldsymbol{\theta}_j - t_l). \tag{2.3}$$

This is done using the `backprojection` function.

2.2.4. *Reconstruction*

Combining the filtering and the backprojection, we compute and we plot the reconstruction:

$$\mathcal{H}(i,j) := \mathbb{1}_{|x|\leqslant R-\Delta t}\mathcal{R}^*[P\star\psi_\Omega](x), x = (t_i, t_j), 1 \leqslant i,j \leqslant n, \quad \Omega = \left\lfloor \frac{n}{2} \right\rfloor \Delta\omega.$$

This represents the scene under the form of an image $\mathcal{H}$ of n^2 pixels.

2.3. **Surface extraction**

The following problem formulates the question of surface extraction.

Problem 2. We know the $P(\theta_j, t_k)$. Let $\alpha \in (0,1]$. Represent the scene ∂S using αn^2 pixels.

We use as *a priori* that the most intense pixels of $\mathcal{H}$ represent the surfaces, up to artifacts. Hence among the n^2 pixels of $\mathcal{H}$, we select the αn^2 most intense. A way is to sort the set of pixels, by decreasing intensity $|\mathcal{H}(i,j)|$; then we keep the first αn^2 pixels. In this article sorting will be realized by a fast algorithm: merge sort. At the end we get the Algorithm 3. We will consider that it is the reference method to solve Problem 2.

2.4. **Costs**

We give complexity indicators of the reference algorithm.

2.4.1. *Filtering*

The size of the filtered array $\tilde{P}_\Omega$ is the size of the data array P: mn. For every $\theta = \theta_j, 1 \leqslant j \leqslant m$, we compute $\mathrm{FFT}[P(\theta_j, \cdot)]$ with cost $\mathcal{O}(n\log n)$, we multiply in Fourier space with cost $\mathcal{O}(n)$, then we compute $\frac{\pi}{\Omega}\,\mathrm{IFFT}[\cdot]$ with cost $\mathcal{O}(n\log n)$. In total, computing the array $\tilde{P}_\Omega$ costs $\mathcal{O}(2mn\log n)$ operations.

2.4.2. *Backprojection*

The array $\mathcal{H}$ contains n^2 elements. Backprojections must be computed for the (t_i, t_j) inside the disk $|x| \leqslant R - \Delta t$. Dividing the area of the disk $|x| < R$ by the area of the square $[-R, R]^2$, we get $\frac{\pi}{4}$; thus the number of computed backprojections is about $\frac{\pi}{4}n^2$. Every backprojection costs $\mathcal{O}(m)$ operations. In total, $\mathcal{O}(m\frac{\pi}{4}n^2)$ operations are needed to compute $\mathcal{H}$.

2.4.3. *Selection*

Lastly the cost of surface extraction is predominated by the cost of a merge sort of n^2 elements: $\mathcal{O}\left(n^2 \log n^2\right)$.

2.4.4. *Synthesis*

The costs are summarized in the first column of Table 3.1. We will choose m and n of the same order. Hence the total cost is dominated by the cost of backprojection: $\mathcal{O}\left(m\frac{\pi}{4}n^2\right)$. That is the reason why we would like to tackle this cost in the sequel: by decreasing the number of backprojections, and by reducing the cost of some backprojections.

Algorithm 1 Function $[\tilde{P}_\Omega(\theta_j, t_l)] = \texttt{filtering}([P(\theta_j, t_l)], R)$

Inputs: $P(\theta_j, t_l), 1 \leqslant j \leqslant m, 1 \leqslant l \leqslant n$, with $\theta_j = (j-1)\Delta\theta$, $t_l = -R + (l-1)\Delta t$, $\Delta t = \frac{2R}{n}$.

$\Omega = \lfloor \frac{n}{2} \rfloor \frac{\pi}{R}$, $\omega_k = (\lceil \frac{-n}{2} \rceil + (k-1))\frac{\pi}{R}, 1 \leqslant k \leqslant n$, $\psi(\sigma) = |\sigma|$.
For every $1 \leqslant j \leqslant m$,

$$\tilde{P}_\Omega(\theta_j, t.) = \frac{\pi}{\Omega} \,\mathrm{IFFT}\left[\mathrm{FFT}[P(\theta_j, t_l)]_{1\leqslant l\leqslant n} \hat{\star} \left[\left(\psi \mathbb{1}_{[-1,1)}\right)\left(\tfrac{\omega_k}{\Omega}\right)\right]_{1\leqslant k\leqslant n}\right].$$

Output: filtered data $\tilde{P}_\Omega(\theta_j, t_l), 1 \leqslant j \leqslant m, 1 \leqslant l \leqslant n$, for $\Omega = \lfloor \frac{n}{2} \rfloor \frac{\pi}{R}$.

Algorithm 2 Function $\mathcal{H}$=$\texttt{backprojection}(x, [\tilde{P}(\theta_j, t_l)], \Delta\theta, R, \Delta t)$

Inputs: $x \in \mathbb{R}^2$ and $\tilde{P}(\theta_j, t_l), 1 \leqslant j \leqslant m, 1 \leqslant l \leqslant n$, where $\theta_j = (j-1)\Delta\theta$, $t_l = -R + (l-1)\Delta t$, $\Delta t = \frac{2R}{n}$.

If $|x| > R - \Delta t$, $\mathcal{H} = 0$;
else,

$$\mathcal{H} = \frac{\Delta\theta}{\Delta t} \sum_{j=1}^{m} -\tilde{P}(\theta_j, t_l)(x \cdot \boldsymbol{\theta}_j - t_{l+1}) + \tilde{P}(\theta_j, t_{l+1})(x \cdot \boldsymbol{\theta}_j - t_l), \quad l = \left\lceil \tfrac{x\cdot\boldsymbol{\theta}_j + R}{\Delta t} \right\rceil.$$

Output: backprojection of $\tilde{P}$, evaluated at x : $\mathbb{1}_{|x|\leqslant R-\Delta t}\mathcal{R}^*[\tilde{P}](x)$.

Algorithm 3 Reference of reconstruction-extraction in reflective tomography.

Inputs: reflective projections $P(\theta_j, t_l), 1 \leqslant j \leqslant m, 1 \leqslant l \leqslant n$, with $\theta_j = (j-1)\Delta\theta$, $t_l = -R + (l-1)\Delta t$, $\Delta t = \frac{2R}{n}$; rate α of wished pixels.

Filtering: $[\tilde{P}_\Omega(\theta_j, t_l)] = \texttt{filtering}([P(\theta_j, t_l)], R)$
Backprojection: computation of $\mathcal{R}^*\tilde{P}_\Omega$.
For every $1 \leqslant i, k \leqslant n$,
$x = (t_i, t_k)$, $\mathcal{H}(i,k) =$$\texttt{backprojection}(x, [\tilde{P}(\theta_j, t_l)], \Delta\theta, R, \Delta t)$.
Selection: sort the $(i, j, \mathcal{H}(i,j))$ by decreasing $|\mathcal{H}(i,j)|$, then select the first αn^2.

Output: the αn^2 most intense pixels of the reconstruction $\mathcal{H}$ (sorted by decreasing intensity).

Figure 3.1. Principle of the multiresolution algorithm: compute an image with coarse pixels, then refine sets of intense pixels, iteratively.

3. Multiresolution reflective tomography

3.1. Asymptotic result

An asymptotic expansion of the reconstruction $\mathcal{R}^*[P \star \psi_\Omega]$ is derived in [1], the asymptotic parameter being the frequency parameter: $\Omega \to \infty$. Assuming essentially that the record P is piecewise smooth, the reconstruction satisfies:

$$\Omega^{1.5}\mathcal{R}^*[P \star \psi_\Omega](x) = \mathcal{O}(1), \mathcal{O}\left(\Omega^{-0.5}\log\Omega\right), \mathcal{O}\left(\Omega^{-0.5}\right).$$

The intensity of the reconstructed surfacic points is $\mathcal{O}(1)$, or $\mathcal{O}(\Omega^{-0.5}\log\Omega)$. The intensity of the artifacts is $\mathcal{O}(\Omega^{-0.5}\log\Omega)$, whereas $\mathcal{O}(\Omega^{-0.5})$ represents a noise. This result is consistent with the numerical observations: the highest values represent the surfaces of the original scene.

But this result is stronger: the heuristic reconstruction is quantified as a function of the resolution parameter Ω. Let us compute the normalized reconstruction $\Omega^{1.5}\mathcal{R}^*[P \star \psi_\Omega]$ for several values of Ω, large enough. If the asymptotics at x is $o(1)$, the result is expected to decrease when Ω increases. But if the asymptotics at x is $\mathcal{O}(1)$, the result should not really change. We propose to take advantage of this model for better performance: injecting this *a priori* into the solver should increase the efficiency of the reconstruction-extraction process.

3.2. Principle

The main idea that we will develop is described below; see Figure 3.1.

3.2.1. *Iterative refining*

First we compute a normalized reconstruction $\Omega^{1.5}\mathcal{R}^*[P \star \psi_\Omega](x)$, for x scanning a coarse grid, and Ω being associated with the mesh size. Then we iteratively refine this reconstruction as follows. The intense values are expected to represent surfaces, and the low values are expected to represent noise. Thus we select a set of pixels whose intensity is high. We divide them into four (sub-)pixels. For all of the new pixels, the normalized reconstruction $\Omega^{1.5}\mathcal{R}^*[P \star \psi_\Omega]$ is computed with an adequate value of Ω: from a pixel to a sub-pixel, Ω is doubled. Then we iterate. This process is multiresolution: the reconstruction is computed for several values of the resolution parameter Ω. The idea is to increase iteratively the resolution of the computed intense pixels.

Let us describe the sets of pixels that we refine. We can indeed obtain several variants of the method, depending on this choice. A way is to refine a single pixel at each iteration: the most intense. In this paper we will use the following rule. At the highest resolution, the sought thin pixels fill a surface whose area is A units area (proportional to αn^2). For every iteration, we refine the most intense pixels, among the pixels whose resolution is not maximal; we constrain their area: adding these pixels with

the pixels at maximal resolution (already computed) must fill an area A. This process is illustrated on Figure 3.1, with A representing the area of the 12 smallest pixels.

3.2.2. *Greedy algorithm*

The proposed algorithm can be interpreted as a greedy algorithm for the following problem:

Problem 3. Find a surface S, whose area is A, divided into pixels, and whose accumulated intensity is maximal. Here, the accumulated intensity is defined as the sum of the absolute values of the normalized reconstructions, computed at the center of the pixels.

Each iteration of the algorithm solves a similar problem, from pixels whose resolution is not maximal, and then refines the pixels of the optimal surface S. The iterations are stopped when the pixels whose resolution is maximal fill an area A. Each iteration increases (or does not change) the number of computed pixels at maximal resolution. The area of the surface, that is to be determined such that the thin pixels fill an area A, is a decreasing sequence (of the number of iteration).

In comparison solving Problem 2 by the reference algorithm computes all the pixels at maximal resolution and then solves Problem 3 in a single step. It cannot be proved that the greedy algorithm yields the same result; but it is designed to give a similar result with a lower cost.

3.2.3. *Comments*

According to the asymptotic result, refining the most intense pixels should first refine the $\mathcal{O}(1)$, and then the $\mathcal{O}(\Omega^{-0.5}\log\Omega)$. This is desired because we want to first refine the surfaces of the scene. Such refinements should yield reconstructions of the same order near the surfaces, even if they are computed for different Ω. By the way, for every wrong alarm, i.e. selection of a non-surfacic pixel: the result is $o(1)$ and thus the refinement should decrease the intensity. This is requested in order to reduce the noise and also to avoid refining more and more the noise in future iterations. As a result selecting and refining a pixel produces a positive effect for both cases (true and wrong alarms).

Let us notice that it makes sense to compare reconstructions thanks to the normalization factor $\Omega^{1.5}$: even on different grids, $\mathcal{O}(1)$ values can be compared, and they are expected to be larger than $o(1)$ values. On the contrary, without the normalization factor $\Omega^{1.5}$, the comments of the last paragraph may become false. In particular comparing reconstructions computed for several values Ω would not make sense. Knowing the asymptotic behaviour of the reconstruction has been crucial to adjust the proposed method. It was used to find a reasonable normalization; and it is now used to understand the behaviour of the proposed algorithm.

Finally the method does not need to eliminate pixels during the process. The algorithm decides itself which zones must be refined, and it has a desired behaviour for noise (see the wrong alarms above). For every computed pixel, there may exist a future iteration which refines it. This has two advantages. This avoids eliminating prematurely pixels that should be preserved. Secondly, if the aimed area is the area of the full image, then the algorithm converges to the reference reconstruction.

3.2.4. *Extension*

In this paper, the multiresolution greedy principle is derived and is studied in the context of reflective tomography. But the ideas can be immediately extended to a general class of problems, concerning the efficient computation of a set which is discriminated thanks to the high-frequency asymptotics of a band-limited function:

Problem 4. Let $\mathcal{I}_\Omega : \mathbb{R}^d \to \mathbb{R}$ be a band-limited function, where the resolution parameter $\Omega > 0$ is a bound over the frequencies: the Fourier transform $\mathcal{F}[\mathcal{I}_\Omega] : \xi \in \mathbb{R}^d \mapsto \int_{\mathbb{R}^d} \mathcal{I}_\Omega(x) \mathrm{e}^{-\mathrm{i}x\cdot\xi} \mathrm{d}x$ is supported in the ball $\{\xi \in \mathbb{R}^d : |\xi| \leqslant \Omega\}$. We assume the following asymptotics, when $\Omega \to \infty$:

$$\mathcal{I}_\Omega(x) = \begin{cases} \mathcal{O}(1), & \text{if } x \in S, \\ o(1), & \text{otherwise,} \end{cases}$$

where $S \subset \mathbb{R}^d$ is a *small* set (*small* Lebesgue measure). We assume that we are able to compute, or to approximate:

$$\mathcal{I}_\omega(x), x \in \mathbb{R}^d, 0 < \omega \leqslant \Omega,$$

where Ω is *large* and fixed. Question: compute S.

The brute force method computes $\mathcal{I}_\Omega$ on a full grid associated to the largest resolution Ω and then selects the highest values. The multiresolution method starts the computation at a lower resolution on a coarser grid, and then refines iteratively sets of meshes. If the cost of evaluation of $\mathcal{I}_\Omega$ is high, if d is large, if S is small, and if Ω is large, then the multiresolution greedy process may accelerate the computations.

This class of problems is linked to the numerical determination of the singular support (or support) of a distribution that would be known only through a low-pass filter. For example, up to the normalization, $\mathcal{I}_\Omega(x_1, \ldots, x_d) = \prod_{1 \leqslant i \leqslant d} \operatorname{sinc} \Omega x_i$ is a low-pass version of the Dirac distribution δ_0 in $\mathbb{R}^d$, with $S = \{0\}$. Let $d = 10$, and $\Omega = 10^6$, and let us imagine that we have a black box whose input is $(x, \omega), x \in \mathbb{R}^d, \omega \leqslant \Omega$, and whose output is $\mathcal{I}_\omega(x)$. It is clear that the multiresolution process would outperform the brute force for determining the set S.

3.3. **Computations at several resolutions**

We precise the computational formulas, as function of the resolution.

3.3.1. *Normalized reconstructions*

We assume that $n = 2^p$, where $p \geqslant 1$ is an integer. Let $0 \leqslant k \leqslant p$. At scale k, the square $[-R - \frac{\Delta t}{2}, R - \frac{\Delta t}{2}]^2$ is divided into square meshes with sides of length $\Delta t_k = 2^{p-k}\Delta t$, and whose center is a point of the grid:

$$G_k = \{x_{ij}^k := (-R - \tfrac{\Delta t}{2}, -R - \tfrac{\Delta t}{2}) + \Delta t_k (i - \tfrac{1}{2}, j - \tfrac{1}{2}), 1 \leqslant i, j \leqslant 2^k\}.$$

At the coarser scale, $k = 0$, we get a single mesh which is the full square. At the finer scale, $k = p$, we get n^2 meshes, whose centers are the $x_{ij}^p = (t_i, t_j)$. Let the unit area be the area of a mesh at scale $k = p$; thus the area of a mesh at scale k is $\Delta t_k^2 = 4^{p-k}$ units area.

For the reconstruction at scale k, at $x = x_{ij}^k \in G_k$, we use the frequency $\Omega_k = 2^{k-p} \lfloor \frac{n}{2} \rfloor \Delta\omega$. The associated normalized reconstruction is:

$$\mathcal{H}_k(i, j) := \mathbb{1}_{|x_{ij}^k| \leqslant R - \Delta t} \Omega_k^{1.5} \mathcal{R}^*[P \star \psi_{\Omega_k}](x_{ij}^k).$$

We could use this reconstruction, combining (2.2) for the filtering and (2.3) for the backprojection. But in the sequel we will use a more efficient scheme.

3.3.2. *Multiresolution filtering*

We propose indeed to downsample the filtered data $P \star \psi_{\Omega_k}(\theta_j, t_l)$, $1 \leqslant j \leqslant m, 1 \leqslant l \leqslant n$. We have in mind to speed up the computations for temporary scales $k < p$, and to keep precision only for the final scale $k = p$.

Thus we keep only the l such that $t_l \in -R + \Delta t_k \mathbb{N}$ and the j such that $\theta_j \in 2^{p-k}\Delta\theta\mathbb{N}$. For the scale $k \geqslant 1$, let $m_k = 1 + \left\lfloor \frac{m-1}{2^{p-k}} \right\rfloor$, $\theta_j^{(k)} = (j-1)2^{p-k}\Delta\theta = \theta_{1+(j-1)2^{p-k}}$, and $t_l^{(k)} = -R + (l-1)\Delta t_k = t_{1+(l-1)2^{p-k}}$. Hence the filtered data are:

$$\tilde{P}_{\Omega_k}(\theta_j^{(k)}, t_l^{(k)}), 1 \leqslant j \leqslant m_k, 1 \leqslant l \leqslant 2^k.$$

Notation. Let $\mathrm{cut}_p[x_h] = \mathrm{cuts}_p[x_h] = [x_h]$, and for every $k < p$:

$$\mathrm{cut}_k[x_h]_{1\leqslant h\leqslant n} = [0, x_{\frac{n}{2}-2^{k-1}+2}, x_{\frac{n}{2}-2^{k-1}+3}, \ldots, x_{\frac{n}{2}+2^{k-1}}],$$
$$\mathrm{cuts}_k[f_i]_{1\leqslant i\leqslant n} = [f_1, \ldots, f_{2^{k-1}}, 0, f_{n-2^{k-1}+2}, \ldots, f_n].$$

Hence for every angle $\theta_j^{(k)}$:

$$\tilde{P}_{\Omega_k}(\theta_j^{(k)}, t_{\cdot}^{(k)}) \approx \frac{2^{k-p}\pi}{\Omega_k} \mathrm{IFFT}\left[\mathrm{cuts}_k \mathrm{FFT}[P(\theta_j^{(k)}, t_l)]_{1\leqslant l\leqslant n} \hat{\star} \mathrm{cut}_k\left[\left(\psi \mathbb{1}_{[-1,1)}\right)\left(\tfrac{\omega_h}{\Omega_k}\right)\right]_{1\leqslant h\leqslant n}\right].$$

In comparison with (2.2) we cut the frequency range. This is a way to downsample with respect to t, and also to reduce the computational cost. It is indeed sufficient to keep the frequencies $\omega_h \in [-\Omega_k, \Omega_k)$ due to the support of $\psi\mathbb{1}_{[-1,1)}$. The cut is operated using cut_k and cuts_k. In order to obtain a hermitian spectral signal and a real signal, we cancelled the Fourier coefficients associated with the frequency $-\Omega_k$, $k < p$ (see the definition of cut_k and cuts_k). Due to these cuts, the IFFT is applied on a vector whose size is 2^k; at the end, the points of the spatial grid are the $t_l^{(k)}, 1 \leqslant l \leqslant 2^k$. The `filteringMultiResol` function computes the downsampled filtering for the different resolutions.

3.3.3. *Efficient normalized reconstructions*

Then for all $1 \leqslant \imath, \jmath \leqslant 2^k$, let:

$$\mathcal{H}_k^{\downarrow}(\imath, \jmath) := \mathbb{1}_{|x_{ij}^k| \leqslant R - \Delta t_k} \Omega_k^{1.5} \mathcal{R}^*[P \star \psi_{\Omega_k}](x_{ij}^k),$$

where the notation $\cdot^{\downarrow}$ remembers that $\mathcal{R}^*[P \star \psi_{\Omega_k}]$ is computed from a downsampling. In a similar manner as (2.3), for $x = x_{ij}^k$ such that $|x| \leqslant R - \Delta t_k$:

$$\mathcal{H}_k^{\downarrow}(\imath, \jmath) \approx \Omega_k^{1.5} \frac{\Delta\theta}{\Delta t} \sum_{j=1}^{m_k} -\tilde{P}_{\Omega_k}(\theta_j^{(k)}, t_l^{(k)})(x \cdot \boldsymbol{\theta}_j^{(k)} - t_{l+1}^{(k)}) + \tilde{P}_{\Omega_k}(\theta_j^{(k)}, t_{l+1}^{(k)})(x \cdot \boldsymbol{\theta}_j^{(k)} - t_l^{(k)}), \quad l = \left\lceil \frac{x \cdot \boldsymbol{\theta}_j^{(k)} + R}{\Delta t_k} \right\rceil.$$

These computations are now accelerated due to the angle downsampling. The `backprojection` function is still adequate for such computations.

To summarize, at a scale $1 \leqslant k < p$ the reconstruction $\mathcal{H}_k^{\downarrow}$ is considered on a grid G_k coarser than G_p, with a frequency Ω_k lower than Ω_p; downsampling speeds up the computations, and reduces the precision. At the finer scale $k = p$, there is no downsampling, the full precision is kept; the reconstruction coincides with the reference reconstruction, up to the normalization factor: $\mathcal{H}_p^{\downarrow} = \Omega_p^{1.5}\mathcal{H}$.

3.4. **Multiresolution algorithm**

The proposed multiresolution algorithm is the Algorithm 6 page 272. We now describe its steps.

3.4.1. *Initialization*

At the beginning we compute all the filtering: $\tilde{P}_{\Omega_k}(\theta_j^{(k)}, t_l^{(k)})$, $k_0 \leqslant k \leqslant p$, $1 \leqslant j \leqslant m_k$, $1 \leqslant l \leqslant 2^k$. Then we compute the full reconstruction at scale k_0: $\mathcal{H}_{k_0}^{\downarrow}(i, j)$, for $x_{ij}^{k_0}$ scanning the grid G_{k_0}; the $(k_0, i, j, \mathcal{H}_{k_0}^{\downarrow}(i, j))$ are stored in *cells*:

Definition 3.1. A *cell* (or a *pixel*) is a quadruplet $(k, i, j, \mathcal{H})$, where $\mathcal{H} = \mathcal{H}_k^{\downarrow}(i, j)$ is the normalized reconstruction, computed at scale k at $x_{ij}^k \in G_k$. The *intensity* of the cell is $|\mathcal{H}|$. The associated mesh is a square with side of length Δt_k and with centre x_{ij}^k. The *area* of the cell is the area of this mesh: 4^{p-k} units area.

The computed cells are stored in a list L. Then we sort L by decreasing intensity $|\mathcal{H}|$. We define a list L_p, which contains the cells computed with scale p. At the moment L_p is void. To finish the task we must insert $A = \alpha n^2$ cells in L_p; this defines an area A to be filled.

3.4.2. *Iteration*

Then we iterate the following process. At the beginning of the iteration, L contains cells with scale $< p$, sorted by decreasing intensity, L_p contains cells with scale p. We would like L_p to contain αn^2 cells. $A > 0$ cells are missing; they represent an area to be filled. Hence we refine the most intense cells of L, such that their area fill (at least) the area A. We update L, L_p, A. The iteration decreases the area A to be filled as much as it increases L_p.

We first define an intermediate function: `refine`. This function refines the most intense cell of L, and returns its area. Since L is assumed to be sorted, the refined cell is the first cell of L. Writing this cell as $(k, i, j, \mathcal{H}_k^{\downarrow}(i, j))$, the returned area is $a = 4^{p-k}$. We divide this cell into four cells with scale $k + 1 \leqslant p$: $(k + 1, I, J, \mathcal{H}_{k+1}^{\downarrow}(I, J))$, $I = 2i, 2i - 1$, $J = 2j, 2j - 1$. If $k + 1 = p$, then the resolution of the computed cells is maximal, and thus they are inserted into L_p; else, we insert these cells into a temporary list $\tilde{L}$. The original cell $(k, i, j, \mathcal{H}_k^{\downarrow}(i, j))$ is obsolete after refinement; hence it is deleted from L.

Using this intermediate function, an iteration is defined as follows. We refine several cells using an inner loop. The accumulated area of the refined cells is stored in a counter a. The inner loop stops as soon as the accumulated area a reaches the aimed area A. After this loop, the list L does not contain anymore the refined cells, the list $\tilde{L}$ contains the new cells with scale $< p$, and the list L_p has been completed with the new ones with scale p. To update the sorted list L of cells with scale $< p$, $\tilde{L}$ is sorted by decreasing intensity $|\mathcal{H}|$; then $\tilde{L}$ and L are merged into L. We want $|L_p| \geqslant \alpha n^2$; the area to be filled for this purpose is now $A = \max(0, \alpha n^2 - |L_p|)$.

Notation. We will sometimes index objects, using the number of the current iteration. At the beginning of the iteration $I \geqslant 1$, the thinnest cells (scale $k = p$) are stored in the list $L_p^{(I)}$, the list of cells with scale $k < p$ is $L^{(I)}$, and the area to be filled is $A^{(I)} = \alpha n^2 - |L_p^{(I)}| > 0$; at the end, we get respectively $L_p^{(I+1)}$, $L^{(I+1)}$ and $A^{(I+1)}$.

3.4.3. *Stopping criterion*

The natural stopping criterion is: $A = 0$. The area $A = \max(0, \alpha n^2 - |L_p|)$ represents indeed the number of missing thin cells; thus we iterate while $A > 0$.

Remark. We could add a maximum number of iterations $I_{\max}$ for safety purposes; we did not write it in Algorithm 6 for simplicity.

We now justify that the algorithm stops, and we count the final number of cells with maximal resolution.

Proposition 3.2. *The Algorithm 6 stops after N iterations, where N is a finite integer.*

Proof. At the beginning and at the end of every iteration, adding the area of the cells of L_p with the area of the cells of L gives always the total area n^2. Hence if we enter in the outer loop with a

missing area $A = \alpha n^2 - |L_p| > 0$, then the area $n^2 - |L_p|$ of L satisfies: $n^2 - |L_p| \geqslant A$. Thus the inner loop will stop: it is possible to find and refine a set of cells of L, whose area is at least A. By the way the area of L, $n^2 - |L_p|$, is filled by at most $\frac{1}{4}(n^2 - |L_p|)$ cells with scale $< p$. Thus:

$$|L| \leqslant \frac{1}{4}(n^2 - |L_p|). \tag{3.1}$$

Concerning the outer loop, if the iterations do not stop, we consider the successive missing areas: $A = \alpha n^2 - |L_p| > 0$. This sequence decreases and takes its values in a finite set. Thus it is constant after a finite number of iterations; so is the number of thin cells $|L_p|$. But the number of cells of $L \cup L_p$ is $|L| + |L_p|$ and strictly increases. As a result $|L|$ indefinitely increases. This is in contradiction with the bound (3.1). This proves that the algorithm stops. ∎

Proposition 3.3. *For the output of Algorithm 6, the number of cells with maximal resolution is:*

$$|L_p| = 4 \left\lceil \frac{\alpha n^2}{4} \right\rceil. \tag{3.2}$$

Proof. The stopping criterion implies $|L_p| \geqslant \alpha n^2$. Since L_p is built by adding 4 cells by 4 cells, $|L_p|$ is a multiple of 4. It remains to prove that $|L_p| < \alpha n^2 + 4$.
Let us consider the last iteration, whose number is N:

$$A^{(N)} = \alpha n^2 - |L_p^{(N)}| > 0 = A^{(N+1)} \geqslant \alpha n^2 - |L_p^{(N+1)}|.$$

We first prove that the scale of the last four computed cells is p. If we enter only once inside the inner loop, then the last four computed cells strictly increase $|L_p|$ because $A^{(N+1)} < A^{(N)}$. Else we enter several times inside the inner loop. In that case, if the scale of the last four computed cells is $< p$, we consider the penultimate iteration of the inner loop. At the end of this iteration, the final list $L_p^{(N+1)}$ has already been obtained; thus the accumulated area a satisfies: $a \geqslant |L_p^{(N+1)}| - |L_p^{(N)}|$. Then $a \geqslant A^{(N)} - (\alpha n^2 - |L_p^{(N+1)}|) \geqslant A^{(N)}$, and thus the inner loop must stop. This is a contradiction: the penultimate iteration is not the last one.
We now prove that $|L_p^{(N+1)}| - 4 < \alpha n^2$. If the inner loop contains only one iteration, it is clear: $|L_p^{(N+1)}| - 4 = |L_p^{(N)}| < \alpha n^2$. Else, at the end of the penultimate iteration of the inner loop, if $|L_p^{(N+1)}| - 4 \geqslant \alpha n^2$, then $a \geqslant |L_p^{(N+1)}| - 4 - |L_p^{(N)}| \geqslant A^{(N)} - (\alpha n^2 - |L_p^{(N+1)}| + 4) \geqslant A^{(N)}$, which is a contradiction as before. ∎

3.4.4. *Solution to the main problem*

The Algorithm 6 computes a set of cells: $L \cup L_p$. Displaying these cells, using their meshes and values, provides a multiresolution image. We can also display only the thinnest cells: L_p. The obtained image is the new solution that we propose to solve the Problem 2.

Algorithm 4 Function $[\tilde{P}_{\Omega_k}(\theta_j^{(k)}, t_l^{(k)})]_{k,j,l}$=`filteringMultiResol`$([P(\theta_j, t_l)], k_0, R)$

Inputs: $P(\theta_j, t_l), 1 \leqslant j \leqslant m, 1 \leqslant l \leqslant n = 2^p$, $1 \leqslant k_0 < p$, where $\theta_j = (j-1)\Delta\theta$, $t_l = -R + (l-1)\Delta t$, $\Delta t = \frac{2R}{n}$.

$\omega_h = (\lceil \frac{-n}{2} \rceil + (h-1))\frac{\pi}{R}, 1 \leqslant h \leqslant n, \qquad \psi(\sigma) = |\sigma|$.
For every $1 \leqslant j \leqslant m$, $\hat{P}(\theta_j, t_\cdot) = \mathrm{FFT}[P(\theta_j, t_l)]_{1 \leqslant l \leqslant n}$.
For every $k_0 \leqslant k \leqslant p$,
$m_k = 1 + \left\lfloor \frac{m-1}{2^{p-k}} \right\rfloor, \qquad \Omega_k = 2^{k-1}\frac{\pi}{R}, \qquad (t_l^{(k)} = t_{1+(l-1)2^{p-k}}, 1 \leqslant l \leqslant 2^k)$
for every $1 \leqslant j \leqslant m_k, \qquad (\theta_j^{(k)} = \theta_{1+(j-1)2^{p-k}})$

$$\tilde{P}_{\Omega_k}(\theta_j^{(k)}, t_\cdot^{(k)}) = \frac{2^{k-p}\pi}{\Omega_k} \mathrm{IFFT}\left[\mathrm{cuts}_k\, \hat{P}(\theta_j^{(k)}, t_\cdot) \hat{\star}\, \mathrm{cut}_k\left[\left(\psi \mathbb{1}_{[-1,1)}\right)\left(\tfrac{\omega_h}{\Omega_k}\right)\right]_{1 \leqslant h \leqslant n}\right].$$

Output: downsampled filtered data $\tilde{P}_{\Omega_k}(\theta_j^{(k)}, t_l^{(k)})$, $k_0 \leqslant k \leqslant p$, $1 \leqslant j \leqslant m_k, 1 \leqslant l \leqslant 2^k$.

Algorithm 5 Function a =`refine`$(L_p, L, \tilde{L}, [\tilde{P}_{\Omega_k}(\theta_j^{(k)}, t_l^{(k)})], \Delta\theta, R, \Delta t)$

Inputs: $L_p, L, \tilde{L}$: lists of cells with scale p, $< p$, $< p$; L is sorted...

From the first cell $(k, i, j, \mathcal{H}_k^{\downarrow}(i,j))$ of L to four cells with scale $k+1$.
Delete the first cell of L.
The area of the refined cell is $a = 4^{p-k}$.
$K = k+1, \qquad \Omega_K = 2^{K-1}\frac{\pi}{R}, \qquad \Delta t_K = 2^{p-K}\Delta t, \qquad \Delta\theta_K = 2^{p-K}\Delta\theta$.
For all $(I, J) \in \{2i, 2i-1\} \times \{2j, 2j-1\}$,
$x = (-R - \frac{\Delta t}{2}, -R - \frac{\Delta t}{2}) + \Delta t_K(I - \frac{1}{2}, J - \frac{1}{2}), \qquad (= x_{IJ}^K)$
$\mathcal{H} = \Omega_K^{1.5}$`backprojection`$(x, [\tilde{P}_{\Omega_K}(\theta_j^{(K)}, t_l^{(K)})]_{j,l}, \Delta\theta_K, R, \Delta t_K), \qquad (= \mathcal{H}_K^{\downarrow}(I,J))$
insert the new cell $(K, I, J, \mathcal{H})$ into L_p if $K = p$, into $\tilde{L}$ otherwise.

Output: area a of the refined cell (and updated lists).

Algorithm 6 Multiresolution greedy algorithm dedicated to reflective tomography.

Inputs: reflective projections $P(\theta_j, t_l), 1 \leqslant j \leqslant m, 1 \leqslant l \leqslant n = 2^p$, $\theta_j = (j-1)\Delta\theta$, $t_l = -R + (l-1)\Delta t$, $\Delta t = \frac{2R}{n}$; rate $\alpha \in (0,1]$ of wished cells; initial resolution $1 \leqslant k_0 < p$.

Initialization

List of cells: the lists L and L_p are void.

Filtering: computation of the downsampled filtered data $\tilde{P}_{\Omega_k}(\theta_j^{(k)}, t_l^{(k)})$.

$[\tilde{P}_{\Omega_k}(\theta_j^{(k)}, t_l^{(k)})]_{k,j,l}$=`filteringMultiResol`$([P(\theta_j, t_l)], k_0, R)$

Backprojection: computation of a coarse reconstruction at scale k_0, $\mathcal{H}_{k_0}^{\downarrow}$.

$k = k_0, \qquad \Omega_k = 2^{k-1}\frac{\pi}{R}, \qquad \Delta t_k = 2^{p-k}\Delta t, \qquad \Delta\theta_k = 2^{p-k}\Delta\theta.$

For all $1 \leqslant \imath, \jmath \leqslant 2^k$,

$x = (-R - \frac{\Delta t}{2}, -R - \frac{\Delta t}{2}) + \Delta t_k(\imath - \frac{1}{2}, \jmath - \frac{1}{2}), \qquad (= x_{\imath\jmath}^k)$

$\mathcal{H} = \Omega_k^{1.5}$`backprojection`$(x, [\tilde{P}_{\Omega_k}(\theta_j^{(k)}, t_l^{(k)})]_{j,l}, \Delta\theta_k, R, \Delta t_k), \qquad (= \mathcal{H}_k^{\downarrow}(\imath, \jmath))$

insert the cell $(k, \imath, \jmath, \mathcal{H})$ into L.

Sort: sort the list L by decreasing intensity $|\mathcal{H}|$.

Area to be filled: $A = \alpha n^2$.

Iterations

While the number of thin cells is too small, i.e. $A > 0$.

$a = 0$, $\tilde{L}$ is void.

While the accumulated area of refined cells is too small, i.e. $a < A$,

refine the most intense cell of L:

$a = a +$ `refine`$(L_p, L, \tilde{L}, [\tilde{P}_{\Omega_k}(\theta_j^{(k)}, t_l^{(k)})], \Delta\theta, R, \Delta t)$.

EndWhile

Sorted list: sort $\tilde{L}$ by decreasing $|\mathcal{H}|$, then merge L and $\tilde{L}$ into L.

Area to be filled: $A = \max(0, \alpha n^2 - |L_p|)$.

EndWhile

Output: multiresolution reconstruction $L \cup L_p$; L_p contains $4\lceil\frac{\alpha n^2}{4}\rceil$ cells at full resolution.

3.5. **Analysis of the algorithm**

We study some properties of the multiresolution algorithm in order to establish guaranties and to estimate its performance. See Table 3.1 page 277 for a synthetic view.

3.5.1. *Evolution and conservation laws*

Let $1 \leqslant I \leqslant N+1$. For every $k_0 \leqslant k \leqslant p-1$, let $C_k^{(I)}$ denote the number of cells with scale k in $L^{(I)}$, and let $C_p^{(I)}$ the number of cells (with scale p) in $L_p^{(I)}$. In particular, after the initialization:

$$C_{k_0}^{(1)} = \frac{n^2}{4^{p-k_0}}, \quad C_{k_0+1}^{(1)} = \cdots = C_p^{(1)} = 0,$$

and the number of cells with scale $< p$ is:

$$|L^{(I)}| = \sum_{k=k_0}^{p-1} C_k^{(I)}. \tag{3.3}$$

For every $k \leqslant p-1$, let $S_k^{(I)}$ denote the number of cells with scale k which are selected and refined. The inner loop builds $\tilde{L}^{(I)}$, with $\tilde{L}^{(I)} = 4\sum_{k=k_0}^{p-2} S_k^{(I)}$ cells, and builds $L_p^{(I+1)}$ by inserting $4S_{p-1}^{(I)}$ cells into $L_p^{(I)}$. After the iteration $1 \leqslant I \leqslant N$, the sizes of $L_p^{(I+1)}$ and $L^{(I+1)}$ are:

$$|L_p^{(I+1)}| = |L_p^{(I)}| + 4S_{p-1}^{(I)}, \quad |L^{(I+1)}| = |L^{(I)}| - S_{p-1}^{(I)} + 3\sum_{k=k_0}^{p-2} S_k^{(I)}. \tag{3.4}$$

The following schemes describe the evolution of the number of cells of the different scales, during the iterations: for every $1 \leqslant I \leqslant N$,

$$C_{k_0}^{(I+1)} - C_{k_0}^{(I)} = -S_{k_0}^{(I)}, \tag{3.5}$$

$$C_k^{(I+1)} - C_k^{(I)} = -S_k^{(I)} + 4S_{k-1}^{(I)}, k_0 < k < p, \tag{3.6}$$

$$C_p^{(I+1)} - C_p^{(I)} = 4S_{p-1}^{(I)}. \tag{3.7}$$

Using (3.2), the final number of thin cells is:

$$|L_p^{(N+1)}| = C_p^{(N+1)} = 4\sum_{I=1}^{N} S_{p-1}^{(I)} = 4\left\lceil \frac{\alpha n^2}{4} \right\rceil. \tag{3.8}$$

Furthermore the area of the cells of $L^{(I)} \cup L_p^{(I)}$ is conserved during the iterations:

$$\sum_{k=k_0}^{p} 4^{p-k} C_k^{(I)} = n^2, \quad 1 \leqslant I \leqslant N+1. \tag{3.9}$$

The $S_k^{(I)}$ are constrained: $0 \leqslant S_k^{(I)} \leqslant C_k^{(I)}$. During the iteration I the refined cells reach the area $A^{(I)}$, but if we delete the last refined cell, whose scale is some $k_I \in [k_0, p-1]$, the area does not reach $A^{(I)}$:

$$A^{(I)} \leqslant \sum_{k=k_0}^{p-1} 4^{p-k} S_k^{(I)} < A^{(I)} + 4^{p-k_I}, \quad 1 \leqslant I \leqslant N.$$

To finish with, the area to be filled satisfies: $A^{(1)} = \alpha n^2$, $A^{(N+1)} = 0$, and a recurrence relation: $A^{(I+1)} = A^{(I)} - 4S_{p-1}^{(I)}, 1 \leqslant I \leqslant N-1$.

3.5.2. *Filtering*

Proposition 3.4. *The filtering of Algorithm 6 costs* $\mathcal{O}\left(\frac{7}{3}mn\log n\right)$ *operations. The filtered data are stored in an array of size* $\mathcal{O}\left(\frac{4}{3}mn\right)$.

Proof. The filtering starts by m FFT of vectors of size n: $\hat{P}(\theta_j,\cdot) = \mathrm{FFT}[P(\theta_j,\cdot)]$, $1 \leqslant j \leqslant m$. This costs $\mathcal{O}(mn\log n)$ operations. Then for each frequency $\Omega_k, k_0 \leqslant k \leqslant p$, the main cost comes from $m_k = \mathcal{O}\left(\frac{m}{2^{p-k}}\right)$ IFFT of vectors of size $2^k = \frac{n}{2^{p-k}}$: $\mathcal{O}\left(\frac{m}{2^{p-k}}\frac{n}{2^{p-k}}\log\frac{n}{2^{p-k}}\right)$. The total cost is:

$$\mathcal{O}(mn\log n) + \mathcal{O}(mn\log n) + \mathcal{O}\left(\frac{m}{2}\frac{n}{2}\log\frac{n}{2}\right) + \cdots + \mathcal{O}\left(\frac{m}{2^{p-k_0}}\frac{n}{2^{p-k_0}}\log\frac{n}{2^{p-k_0}}\right)$$
$$= \mathcal{O}\left(\left(1+1+\frac{1}{4}+\cdots+\frac{1}{4^{p-k_0}}\right)mn\log n\right) = \mathcal{O}\left(\frac{7}{3}mn\log n\right);$$

the constants of the $\mathcal{O}$ are of the same order than the constant of the cost of FFT.
After the computations, we save: an array of size $m_k \times 2^k = \mathcal{O}\left(\frac{m}{2^{p-k}}\frac{n}{2^{p-k}}\right)$, for each $k_0 \leqslant k \leqslant p$. The storage cost is:

$$\mathcal{O}(mn) + \mathcal{O}\left(\frac{m}{2}\frac{n}{2}\right) + \cdots + \mathcal{O}\left(\frac{m}{2^{p-k_0}}\frac{n}{2^{p-k_0}}\right) = \mathcal{O}\left(\left(1+\frac{1}{4}+\cdots+\frac{1}{4^{p-k_0}}\right)mn\right) = \mathcal{O}\left(\frac{4}{3}mn\right).$$

■

3.5.3. *Backprojection*

The backprojection cost is not easy to evaluate because it depends on the data. We will derive bounds, based on bounds for the number of computed cells:

Proposition 3.5. *Let* C_k *be the number of computed cells at scale* k *during Algorithm 6.*

(i) $C_p = 4\lceil\frac{\alpha n^2}{4}\rceil$ *and the number of cells computed during the iterations satisfies:*

$$\frac{4}{3}(1-4^{k_0-p})C_p \leqslant C_p + C_{p-1} + \cdots + C_{k_0+1} \leqslant C_p + \frac{1}{3}(1-4^{k_0+1-p})n^2. \tag{3.10}$$

(ii) *The lower bound is reached if, and only if, each cell computed at a scale* $k_0 < k < p$ *was refined.*

(iii) *The upper bound is reached if, and only if, all the cells of all the intermediate scales* $k_0 < k < p$ *were computed.*

Proof. At the end of the algorithm, for the scale $k = p$: $C_p = C_p^{(N+1)} = |L_p^{(N+1)}|$. But more generally, for all $k \leqslant p$ and $I \leqslant N+1$, we have only $C_k \geqslant C_k^{(I)}$, due to deletions. The size of the full image at scale k is $\left(\frac{n}{2^{p-k}}\right)^2$, thus $C_k \leqslant \frac{n^2}{4^{p-k}}$. Then, due to the refinement strategy, for every $k_0 < k \leqslant p$, C_k is a multiple of 4 and $C_{k-1} \geqslant \frac{1}{4}C_k$. So $C_k \geqslant \frac{1}{4^{p-k}}C_p, k_0 < k \leqslant p$. Hence the number of pixels computed during the iterations satisfies:

$$\left(1+\frac{1}{4}+\cdots+\frac{1}{4^{p-k_0-1}}\right)C_p \leqslant C_p + C_{p-1} + \cdots + C_{k_0+1} \leqslant C_p + \left(\frac{1}{4}+\cdots+\frac{1}{4^{p-k_0-1}}\right)n^2.$$

This gives (i). Then (ii) and (iii) are deduced by studying the conditions of equality. ■

The following proposition estimate the cost of backprojection; see its proof for estimations on the number of computed backprojections.

Proposition 3.6. *The total cost of backprojection for the Algorithm 6 is*

(i) *at least* $\mathcal{O}\left(\frac{\pi}{4}\frac{m}{2^{p-k_0}}\frac{n^2}{4^{p-k_0}}\right) + \mathcal{O}\left(\frac{8}{7}(1-8^{k_0-p})\alpha mn^2\right)$;

(ii) *at most* $\mathcal{O}\left(\left(\alpha + \frac{\pi}{4}\frac{1}{7}(1-8^{k_0-p})\right)mn^2\right)$.

Proof. The initialization computes a reconstruction on the grid G_{k_0} of size $\frac{n^2}{4^{p-k_0}}$, from m_{k_0} angles. This costs $\mathcal{O}\left(\frac{\pi}{4}\frac{n^2}{4^{p-k_0}}\right)$ backprojections of cost $\mathcal{O}\left(\frac{m}{2^{p-k_0}}\right)$ operations. Hence the cost is $\mathcal{O}\left(\frac{\pi}{4}\frac{m}{2^{p-k_0}}\frac{n^2}{4^{p-k_0}}\right)$ operations for the initial reconstruction. We must now add the costs of the iterations.
We prove (i) by considering the case where the number of computed reconstructions would be minimal. The number of cells that we compute during the iterations is at least the lower bound of Proposition 3.5. Reaching this bound would mean that the algorithm would converge without useless computations: every computed cell with scale $k_0 < k < p$ would be utilized (refined) to get the final result. In that case, for every scale $k_0 < k \leqslant p$, we would compute $C_k = \frac{C_p}{4^{p-k}}$ reconstructions of cost $\mathcal{O}\left(\frac{m}{2^{p-k}}\right)$ operations, with $C_p = \mathcal{O}(\alpha n^2)$; the total cost of backprojection during the iterations would be:

$$\mathcal{O}(mC_p) + \mathcal{O}\left(\frac{m}{2}\frac{C_p}{4}\right) + \cdots + \mathcal{O}\left(\frac{m}{2^{p-k_0-1}}\frac{C_p}{4^{p-k_0-1}}\right) = \mathcal{O}\left(\left(1+\frac{1}{8}+\cdots+\frac{1}{8^{p-k_0-1}}\right)\alpha mn^2\right).$$

We now show (ii) by maximizing the number of computed cells; this means reaching the upper bound of Proposition 3.5. The algorithm would compute C_p cells at scale p, and the full reconstruction, at every intermediate scale $k_0 < k < p$. In that case, the number of backprojections would be:

$$\mathcal{O}\left(\left(\alpha + \frac{\pi}{4}\frac{1}{3}(1-4^{k_0+1-p})\right)n^2\right).$$

The corresponding cost would be: $\mathcal{O}\left(\alpha mn^2 + \frac{m}{2}\frac{\pi}{4}\frac{n^2}{4} + \cdots + \frac{m}{2^{p-k_0-1}}\frac{\pi}{4}\frac{n^2}{4^{p-k_0-1}}\right)$. ∎

The following result establish guaranties concerning the use of memory.

Proposition 3.7. *The total number of cells* $|L \cup L_p|$ *satisfies:*

$$|L \cup L_p| \leqslant \frac{1}{4}n^2 + 3\left\lceil\frac{\alpha n^2}{4}\right\rceil = \mathcal{O}\left((1+3\alpha)\frac{n^2}{4}\right).$$

Proof. By combining the inequality (3.1) and $|L_p| \leqslant C_p = 4\lceil\frac{\alpha n^2}{4}\rceil$, the number of cells in memory always satisfies: $|L| + |L_p| \leqslant \frac{1}{4}n^2 + \frac{3}{4}C_p$ (also true during the iterations). ∎

3.5.4. *Number of iterations*

The following Proposition gives bounds for the number of iterations; if we do not use the structure of the data and of the solver, they are "optimal":

Proposition 3.8.

(i) *Algorithm 6 stops after* N *iterations, with*

$$p - k_0 \leqslant N \leqslant \left\lceil\frac{\alpha n^2}{4}\right\rceil + \frac{n^2}{4\cdot 3}(1-4^{k_0+1-p}). \tag{3.11}$$

(ii) *If each iteration* $1 \leqslant I \leqslant p-k_0$ *refines* $\lceil\frac{\alpha n^2}{4^{p-k_0-I+1}}\rceil$ *cells with scale* k_0+I-1, *then* $N = p-k_0$.

(iii) *If each iteration refines a single cell, and if the number of computed cells is maximal (upper bound of* (3.10)*), then* $N = \lceil \frac{\alpha n^2}{4} \rceil + \frac{n^2}{4 \cdot 3}(1 - 4^{k_0+1-p})$.

Proof. We easily see the cases of equality (ii) and (iii). We prove (i). For every iteration, the increment of the largest scale of the cells of L is at most one. Starting with cells with scale k_0, we need at least $p - k_0$ iterations to obtain cells with scale p: $N \geqslant p - k_0$. By the way, we compute at least 4 cells at every iteration; thus $4N$ is lesser or equal to the number of computed cells during the iterations, which is itself dominated using (3.10). ∎

3.5.5. *Selection*

We count the main costs related to the selection of intense values: merge sorts and (extra-)merges.

Proposition 3.9. *The merge sorts of Algorithm 6 cost:*

$$\mathcal{O}\left(\frac{n^2}{4^{p-k_0}} \log \frac{n^2}{4^{p-k_0}}\right) + \mathcal{O}\left(\frac{n^2}{3}(1 - 4^{k_0+1-p}) \log \frac{n^2}{3}(1 - 4^{k_0+1-p})\right).$$

Proof. The initialization sorts a list of size $(\frac{n}{2^{p-k_0}})^2$, with cost $\mathcal{O}\left(\frac{n^2}{4^{p-k_0}} \log \frac{n^2}{4^{p-k_0}}\right)$. If $k_0 = p - 1$, no additional sort. Else, $k_0 < p-1$, each iteration sorts the list $\tilde{L}$ of refined cells with scale $< p$, with cost $\mathcal{O}\left(|\tilde{L}| \log |\tilde{L}|\right)$. Summing these costs over the iteration number $1 \leqslant I \leqslant N$, the cost to be estimated is $\mathcal{O}\left(\sum_{I=1}^{N} \left(4 \sum_{k=k_0}^{p-2} S_k^{(I)}\right) \log \left(4 \sum_{k=k_0}^{p-2} S_k^{(I)}\right)\right)$. Summing (3.4) over I, with (3.7), we get:

$$3 \sum_{I=1}^{N} \sum_{k=k_0}^{p-2} S_k^{(I)} = |L^{(N+1)}| - |L^{(1)}| + \frac{1}{4}(C_p^{(N+1)} - C_p^{(1)}).$$

After the initialization $|L^{(1)}| = \frac{n^2}{4^{p-k_0}}$, and $C_p^{(1)} = 0$. And at the end, from (3.1), $|L^{(N+1)}| \leqslant \frac{n^2 - C_p^{(N+1)}}{4}$, with $C_p^{(N+1)} = 4\lceil \frac{\alpha n^2}{4} \rceil$. Therefore:

$$3 \sum_{I=1}^{N} \sum_{k=k_0}^{p-2} S_k^{(I)} \leqslant \frac{n^2}{4}(1 - 4^{k_0+1-p}),$$

and we conclude:

$$\begin{aligned} \sum_{I=1}^{N} \left(4 \sum_{k=k_0}^{p-2} S_k^{(I)}\right) \log \left(4 \sum_{k=k_0}^{p-2} S_k^{(I)}\right) &\leqslant \left(4 \sum_{I=1}^{N} \sum_{k=k_0}^{p-2} S_k^{(I)}\right) \log \left(4 \sum_{J=1}^{N} \sum_{k=k_0}^{p-2} S_k^{(J)}\right) \\ &\leqslant \frac{n^2}{3}(1 - 4^{k_0+1-p}) \log \frac{n^2}{3}(1 - 4^{k_0+1-p}). \end{aligned}$$

∎

Proposition 3.10. *The merges of Algorithm 6 cost*

$$\mathcal{O}\left(\frac{Nn^2}{4}\gamma\right),$$

with $\gamma \in [0, 1]$*, and the number of iterations* N *satisfies* (3.11)*. (This cost is null if* $k_0 = p - 1$*).*

Proof. If $k_0 = p - 1$, no additional merge to be considered. Else, $k_0 < p - 1$, each iteration sorts $\tilde{L}$, and merges $\tilde{L}$ with L. This additional merge costs $\mathcal{O}\left(|\tilde{L}| + |L|\right)$ (the cost of the merges of the

sort is counted in the sort cost). Summing over $1 \leqslant I \leqslant N$, the merge cost to be estimated is: $\mathcal{O}\left(\sum_{I=1}^{N} |L^{(I+1)}|\right)$. Combining the area conservation (3.9), and a mean value formula:

$$\sum_{k=k_0}^{p-1} 4^{p-k} C_k^{(I+1)} = n^2 - C_p^{(I+1)} = \frac{1}{\gamma_1^{(I+1)}} \sum_{k=k_0}^{p-1} C_k^{(I+1)}, \quad \text{with } \frac{1}{\gamma_1^{(I+1)}} \in [4, 4^{p-k_0}].$$

We deduce $|L^{(I+1)}| = \gamma_1^{(I+1)}(n^2 - C_p^{(I+1)})$ from the decomposition (3.3). Therefore:

$$\sum_{I=1}^{N} |L^{(I+1)}| = \sum_{I=1}^{N} \gamma_1^{(I+1)}(n^2 - C_p^{(I+1)}) = \gamma_2 N n^2 - \gamma_3 \sum_{I=1}^{N} C_p^{(I+1)}, \quad \text{with } \gamma_2, \gamma_3 \in [4^{k_0-p}, 4^{-1}].$$

We apply $\sum_{1 \leqslant J \leqslant N} \sum_{1 \leqslant I \leqslant J}$ on (3.7), with $C_p^{(1)} = 0$. We get:

$$\sum_{1 \leqslant J \leqslant N} C_p^{(J+1)} = 4 \sum_{1 \leqslant I \leqslant N} (N - I + 1) S_{p-1}^{(I)} = (1 + (N-1)\gamma_4) 4 \sum_{1 \leqslant I \leqslant N} S_{p-1}^{(I)},$$

with $\gamma_4 \in [0, 1]$. We now obtain from (3.8):

$$\sum_{I=1}^{N} |L^{(I+1)}| = \gamma_2 N n^2 - \gamma_3 (1 + (N-1)\gamma_4) 4 \lceil \frac{\alpha n^2}{4} \rceil,$$

which proves the writing: $\mathcal{O}\left(\frac{N n^2}{4}\gamma\right)$, $\gamma \in [0, 1]$. ∎

3.6. **Discussion**

We discuss the interest of the new method, in comparison with the reference method, concerning the computational effort. The discussion is based on Table 3.1.

	Algorithm 3	**Algorithm 6**
Filtering		
- operations	$\mathcal{O}\left(2mn \log n\right)$	$\mathcal{O}\left(\frac{7}{3} mn \log n\right)$
- storage	$\mathcal{O}\left(mn\right)$	$\mathcal{O}\left(\frac{4}{3} mn\right)$
Backprojection		
- number	$\mathcal{O}\left(\frac{\pi}{4} n^2\right)$	at least $\mathcal{O}\left(\frac{\pi}{4} \frac{n^2}{4^{p-k_0}}\right) + \mathcal{O}\left(\alpha n^2 \frac{4}{3}(1 - \frac{1}{4^{p-k_0}})\right)$, at most $\mathcal{O}\left(\alpha n^2 + \frac{\pi}{4}\frac{n^2}{3}(1 - \frac{1}{4^{p-k_0}})\right)$
- operations	$\mathcal{O}\left(\frac{\pi}{4} mn^2\right)$	at least $\mathcal{O}\left(\frac{\pi}{4} \frac{mn^2}{8^{p-k_0}}\right) + \mathcal{O}\left(\alpha mn^2 \frac{8}{7}(1 - \frac{1}{8^{p-k_0}})\right)$, at most $\mathcal{O}\left(\alpha mn^2 + \frac{\pi}{4}\frac{mn^2}{7}(1 - \frac{1}{8^{p-k_0}})\right)$
- storage	$\mathcal{O}\left(n^2\right)$	$\mathcal{O}\left(\frac{n^2}{4}(1 + 3\alpha)\right)$
Selection		
- operations	$\mathcal{O}\left(n^2 \log n^2\right)$	$\mathcal{O}\left(\frac{n^2}{4^{p-k_0}} \log \frac{n^2}{4^{p-k_0}}\right) \ldots + \mathbb{1}_{k_0 < p-1} \left[\mathcal{O}\left(\frac{n^2}{3} \log \frac{n^2}{3}\right) + \mathcal{O}\left(\frac{Nn^2}{4}\right)\right]$, with $p - k_0 \leqslant N \leqslant \left\lceil \frac{\alpha n^2}{4} \right\rceil + \frac{n^2}{4 \cdot 3}(1 - \frac{1}{4^{p-k_0-1}})$

Table 3.1. Costs: reference method (Algorithm 3) versus multiresolution greedy method (Algorithm 6).

3.6.1. *Filtering*

The additional cost due to downsampled filtering at several scales is at most 33%, for both computation and storage. This step does not determine what method is the most efficient.

3.6.2. *Backprojection*

We need to compute at least αn^2 backprojections, with cost $\mathcal{O}\left(\alpha m n^2\right)$, in order to solve Problem 2 using tomography. We compare the two methods with these lower bounds.

The reference method computes $100(\frac{\pi}{4\alpha} - 1)\%$ additional pixels; the corresponding extra cost is $100(\frac{\pi}{4\alpha} - 1)\%$. Concerning the new method, for the most advantageous cases (lower bound reached), with a small k_0, the number of computed backprojections would be close to $\frac{4}{3}\alpha n^2$, i.e. 33% additional backprojections; the extra cost would be close to $\frac{100}{7} \approx 14\%$. For the most disadvantageous cases (upper bound reached) the extra cost would be close to $100\frac{\pi}{4 \cdot 7\alpha}\%$.

Concerning the backprojection cost, if α is small then the new method is always better than the reference one. But if α becomes larger than $\frac{6\pi}{4 \cdot 7} \approx 0.67$, then the new method becomes more expensive for the most disadvantageous cases; if α becomes larger than $\frac{7\pi}{32} \approx 0.69$, this stands true even for the most favourable cases.

The multiresolution method is designed to diminish the backprojection cost when we want to compute only a small portion of the full volume at full resolution. This analysis shows that the proposed method achieves its goal. It also reveals that if we want to compute a large portion of the full volume (let us say more than 67%), it is better to directly use the reference method.

3.6.3. *Selection*

For the main costs of selection (merges and sorts), let us first consider the case $k_0 = p - 1$. The cost is dominated by the cost of the sort of the coarse reconstruction: $\mathcal{O}\left(\frac{n^2}{2} \log \frac{n}{2}\right)$, which is at least four times smaller than the selection cost for the reference. If α is small, then the new method should reduce the total execution time.

It is not so obvious for $k_0 < p - 1$. If the number of iterations is reasonable, i.e. $N = \mathcal{O}\left(\log n^2\right)$, then the cost is reasonable; its order is roughly the order of the sort cost of the reference method. But if N becomes large, in comparison with $\log n^2$, then the main cost comes from the merges, and is $\mathcal{O}\left(\frac{Nn^2}{4}\right)$. In particular, if N is close to $4m$, then the merge cost is close to the global cost mn^2 of the reference method. In such a case the multiresolution algorithm would not really decrease the execution time. Worse still, if N is close to its upper bound, then the merge cost may be close to a n^4 term; this is prohibitive.

Unfortunately the number of iterations is not known in advance. From a pessimistic point of view, we do not know in advance if the multiresolution process will outperform the reference one (for $k_0 < p-1$). From a more optimistic point of view, the proposed analysis does not take into account the structure of the data set, nor the kind of solver. But in practice the algorithm is not used on arbitrary data, but on reflectograms. And the solver (normalized filtered backprojection) has been adjusted for such data. According to the expected behaviour of the refinement process, as discussed in the subsubsection 3.2.3, we hope that the costs will be in practice close to the lower bounds (and not the upper bounds).

3.6.4. *Performance indicators*

The multiresolution algorithm is designed to focus iteratively on the surfaces of the scene, from a coarse initial reconstruction, and to reach a set of thin pixels, whose number is set in advance. The efficiency of the algorithm to achieve its goal appears under several forms in the complexity analysis.

Firstly, by the number of computed intermediate cells; this is related with the backprojection cost. For the most advantageous cases, the algorithm would focus on the parts to be extracted and would compute only the necessary cells. For the most disadvantageous ones, the algorithm would spread out the computations, by computing all the cells of all the intermediate scales. We define an indicator which measures this phenomenon:

Definition 3.11. For the Algorithm 6, the number of intermediate cells $S = C_{p-1} + \cdots + C_{k_0+1}$ is between the lower bound $S_0 = \frac{1}{3}(1 - 4^{k_0+1-p})4\left\lceil \frac{\alpha n^2}{4} \right\rceil$ and the upper bound $S_1 = \frac{1}{3}(1 - 4^{k_0+1-p})n^2$. The focus F is defined by:

$$F := \frac{S_1 - S}{S_1 - S_0} \in [0, 1].$$

Thus: F close to 1 means that few useless cells were computed, and that the algorithm efficiently focused on the parts to be extracted. On the contrary, F close to 0 means that many useless cells were computed. In practice S is computed by incrementing a counter during the iterations, and F is computed at the end of the execution. (Of course, F makes sense only if $k_0 < p - 1$).

Secondly, by the number of iterations N; this is related with the selection cost, which conditions the interest of the multiresolution algorithm. For the optimal case, $N = p - k_0$. If N is a few multiples of $2p$, the method is still relevant. But if N becomes very large, in comparison with p, then the new method is not interesting concerning the execution time.

In the numerical tests, we will check the efficiency of the Algorithm 6 by checking *a posteriori* the values of F and N.

3.6.5. *Modified algorithm*

We can define variations of the Algorithm 6, without changing the main principle: refining the most intense pixels. For example, downsampling the angles is not mandatory. This would need more computational effort, but would increase the precision of the intermediate computations. This may improve the coarse reconstruction, the selections, and the final result.

Other example, if we want to be sure that the selection cost is not $\mathcal{O}(n^4)$, we can add a lower threshold $\beta n^2 \in (0, \alpha n^2]$, set in advance, for the area of the refined zone. The condition of the inner loop, $a < A$, is replaced by: $a < A$ or $a < \beta n^2$. After N iterations, the accumulation of the refined areas is at least $N\beta n^2$. But this cannot exceed $n^2(p - k_0 + 1)$, and thus $N \leqslant \frac{p-k_0+1}{\beta}$. The disadvantage of this version: we refine more cells than the strict necessary when $A < \beta n^2$; this may increase the backprojection cost.

4. **Numerical results**

4.1. **Implementation**

The multiresolution algorithm was implemented in `Fortran`. The computations are performed using double precision. The Fourier transforms are computed using the library `fftw3`. For the sorts, we implemented a merge sort, by decreasing absolute value.

Concerning the data structures, we defined a type `cell` for the cells. The lists of cells are stored in arrays which are dynamically allocated. L_p is an array of C_p cells. L and $\tilde{L}$ are stored respectively at the beginning and at the end of a single array, whose size is large enough. To locate the beginning and the end of $L_p, L, \tilde{L}$ in these large arrays, we define sentinels: the insertions or deletions, at the top or at the queue of the lists, update the sentinels (increment or decrement).

The code is compiled with `gfortran-4.6`. The execution takes place on a workstation HP Z820, processors Intel Xeon E5-2609, 2.40GHz. During the execution, we measure time dedicated to: initial filtering, initial backprojection, initial sort, and iterations.

4.2. **Full example**

We consider the silhouettes of a scene containing two circles: in model (2.1), $f = 1$ on the circles, and $f = 0$ on the wall: $P(\theta, s) = 1$ if the ray $x \cdot \boldsymbol{\theta} = s$ intersects at least one of the circles, $P(\theta, s) = 0$ otherwise. See Figure 4.1 for an image of the scene, and the associated reflectogram P of size $m \times n = 805 \times 256$ (θ is the abscissa, and s is the ordinate).

Figure 4.1. Scene with white circle (left) and its reflectogram 805×256 (right).

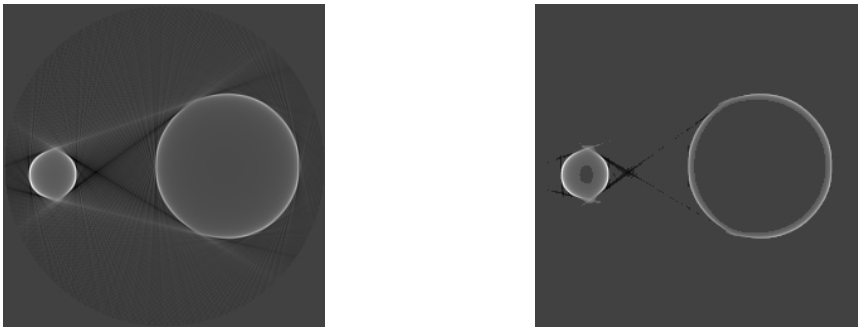

Figure 4.2. Reference for the reflectogram of Figure 4.1: full reconstruction of size 256^2 (left), and after extraction of 5% of the pixels (right).

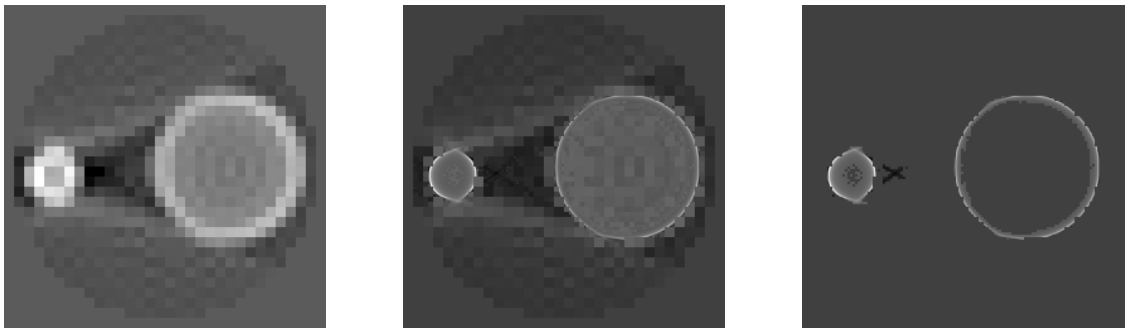

Figure 4.3. Multiresolution reconstruction for Figure 4.1: initialization of size 32^2 (left), multiresolution image (middle), and the 0.05×256^2 thinnest pixels (right).

On Figure 4.2, we represent the full reconstruction of size $n^2 = 256^2$, and the reconstruction after extraction of the most intense pixels, $\alpha = 5\%$. Computing the reconstruction and the extraction took

3.61 seconds. Qualitatively we find what was already known: the most intense pixels correspond to the surfaces of the scene, up to artifacts (the artifacts being the lines that are tangent to both circles).

By the way we applied the multiresolution algorithm. On Figure 4.3, we represent the initial coarse reconstruction of size 32^2 ($k_0 = 5$). We represent the final multiresolution reconstruction, and the 0.05×256^2 thinnest pixels. This multiresolution computation took 0.364 second. Qualitatively we observe what was announced: the pixels are refined near the surfaces (and artifacts) and they stay coarse elsewhere. The algorithm succeeds in focusing the computations near the relevant places. The indicators show the efficiency: we got $F = 0.951$ for the focus, and $N = 25 = 1.5 \cdot 2p + 1$ iterations.

At the end, the extracted reconstruction is not as nice as the reference reconstruction, but it looks similar, and the computation time was divided by 10.

4.3. **Influence of the initial grid**

We consider the scene and the reflectogram of size $m \times n = 1609 \times 512$, of Figure 4.4. Here, $f = 0.77$ on the large circle, and $f = 1$ on the small circle. We set $\alpha = 0.01$ and we look at the behaviour of the method for several sizes $(2^{k_0})^2$ of the initial grid. The produced images are presented on Figure 4.5. The last line is the reference, considered as an extreme multiresolution case, where $k_0 = p$, $N = 0$, and the pre-extraction reconstruction is the initial image.

We observe some convergence of the reconstructions to the reference reconstruction. We notice that for $k_0 = p - 1, p - 2$ (where $p = \log_2 n = 9$), the results after extraction are relatively close to the reference. For the smallest values of k_0, i.e. for the coarsest initial grids, the result contains relevant information, but with a degradation (holes). We grant that k_0 cannot become too small: the algorithm is based on the high-resolution asymptotics of the reconstruction.

The performance indicators are in Table 4.1. The focus is always very high. The number of iterations N is reasonable, in comparison with $2p = 18$. The measured times are consistent with these indicators: the multiresolution method is clearly faster than the reference. Of course, for $k_0 = p - 1$, the method converges after a single iteration.

Figure 4.4. Scene (left) and its reflectogram 1609×512 (right).

size 2^{k_0}	focus F	iterations N	time (s)
16	0.984	45	0.728
32	0.981	43	0.752
64	0.984	24	0.784
128	0.988	13	1.12
256	×	1	4.14
512	×	0	31.8

Table 4.1. Performances as a function of the initial grid, for the reconstructions of Figure 4.5.

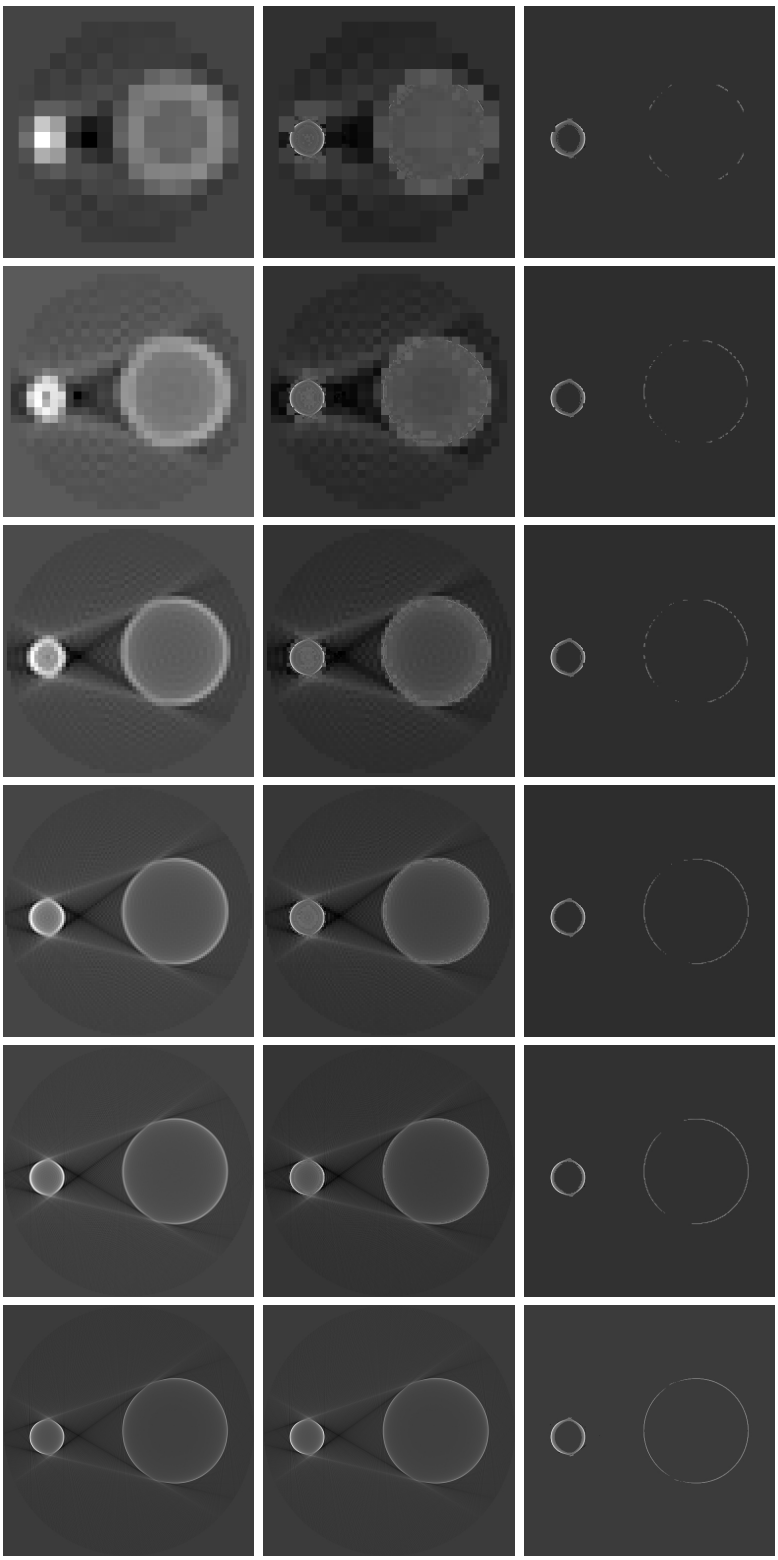

Figure 4.5. Reconstructions for the Figure 4.4: coarse initialization of size $(2^{k_0})^2$ (left), multiresolution reconstruction (middle), and the thinnest pixels (right). From top to bottom: $k_0 = 4, 5, 6, 7, 8$; on the last line: reference reconstruction.

To conclude this example, the case $k_0 = 7 = p - 2$ is a good compromise between computation time and quality of the extracted reconstruction. We propose more generally to set k_0 such that $p - k_0$ is a small integer.

4.4. **Influence of the rate**

We consider the scene and the reflectogram of size $m \times n = 1609 \times 512$, of Figure 4.4. Let $k_0 = 7 = p-2$; we study the behaviour of the multiresolution method with respect to the rate $\alpha = 0.32 \cdot 2^\beta$, $-8 \leqslant \beta \leqslant 0$. The initial reconstruction has already been represented on Figure 4.5.

On Figure 4.6, we represent the extracted reconstructions. We observe that the algorithm reconstructs in priority the circles, then the straight artifacts, then the noise. This is explained by the asymptotics: $\Omega^{1.5}\mathcal{R}^*[P \star \psi_\Omega](x)$ is $\mathcal{O}(1)$ on the circles, $\mathcal{O}(\Omega^{-0.5}\log\Omega)$ on the lines, and $\mathcal{O}(\Omega^{-0.5})$ elsewhere. This sets the priority order when we select the most intense pixels. That was the idea of the algorithm. We also observe that α must be appropriate: if α is too small, the reconstructed part contains too few points; increasing α increases the number of reconstructed points, but it also increases the artifacts and the noise. In practice we could imagine several steps: start with a small α, then launch the iterations again to refine more cells, and so on.

In Table 4.2, we reported the performance indicators. The execution time increases lightly with α when α is very small; then it increases sublinearly (from $\beta = -4$). In particular we cannot hope

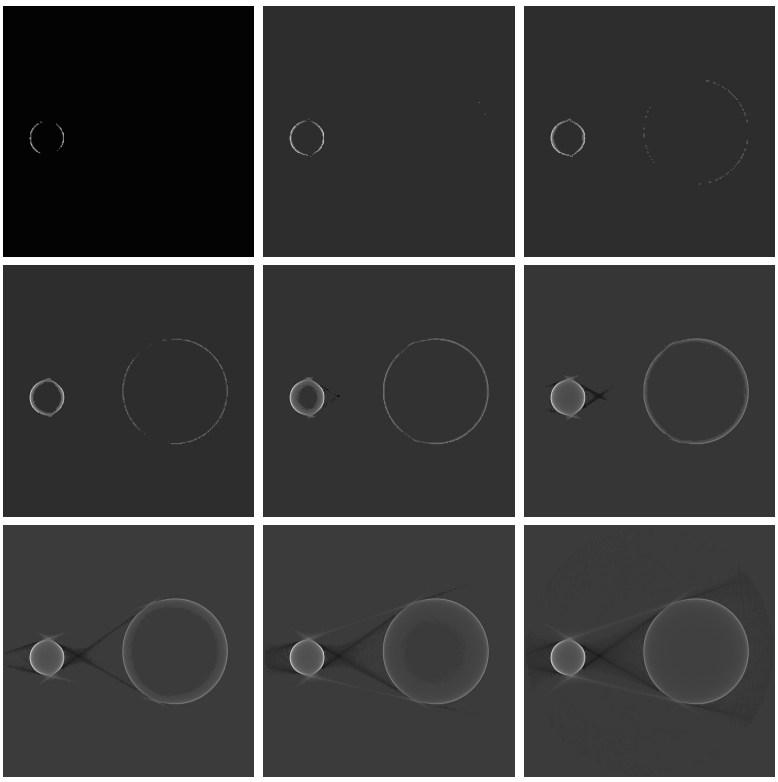

Figure 4.6. Reconstructions of $\alpha 512^2$ pixels for the scene of Figure 4.4, with initialization of size 128^2 and $\alpha = 0.32 \cdot 2^\beta$; $-8 \leqslant \beta \leqslant 0$ increases in the usual reading order (left to right, top to bottom).

$\beta = \log \frac{\alpha}{0.32}$	focus F	iterations N	time (s)
-8	0.998	9	0.676
-7	0.997	11	0.736
-6	0.994	10	0.860
-5	0.988	13	1.12
-4	0.981	12	1.61
-3	0.973	13	2.57
-2	0.936	14	4.62
-1	0.907	7	8.41
0	0.770	15	16.2

Table 4.2. Performances as a function of the rate of pixels α, for the reconstructions of Figure 4.6.

vanishing this time, even if α is very small: the method contains incompressible costs (including the initialization).

The other important conclusions of the table concern the focus. We observe that the focus F decreases when α increases. The main decreases appear at the end of the table. We must link this with the Figure 4.6. The algorithm focuses on some parts of the scene, by taking advantage of the asymptotic behaviour when Ω increases. As soon as the parts corresponding to the right behaviour are reconstructed at full resolution, the algorithm works on pixels which contains noise, and not on signals containing the sought behaviour. Thus the algorithm cannot focus anymore on well targeted regions, and so F decreases.

We have already mentioned that the focus F gives information on the efficiency of the algorithm. We now see that it gives indications about the quantity of useful signal in the final image. If F is very close to 1, increasing α (or relaunching iterations) may improve the quality of the reconstruction. On the contrary, if F is small, the computed reconstruction may be already corrupted by noise, and it may be preferable to keep only the most intense computed pixels; or we can alternately start again with a smaller α. These comments also suggest another variation of the algorithm, taking into account the focus in the stopping criterion: continue while the focus is above some fixed threshold.

4.5. **Complex scene**

We add an example which is more difficult: see Figure 4.7. The size of the reflectogram is $m \times n = 3217 \times 1024$. The scene contains: a small circle with $f = 1$, a large circle with $f = 0.77$, a polygon with $f = 0.63$, a croissant with $f = 85$ on the convex portion, and two parts for the concavity, $f = 0.63$ and $f = 0.73$. The reconstruction $\Omega^{1.5}\mathcal{R}^*[P \star \psi_\Omega](x)$ is essentially expected to be $\mathcal{O}(1)$ on convex portions, and $\mathcal{O}(\Omega^{-0.5} \log \Omega)$ on lines and isolated points.

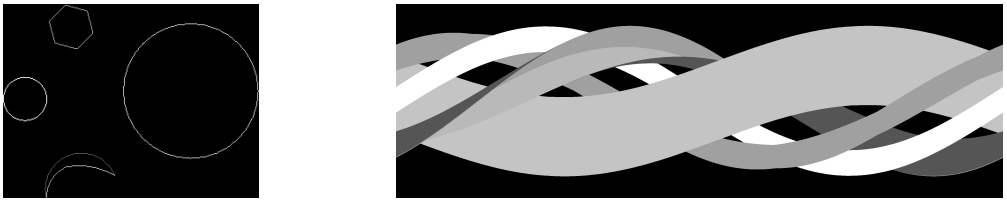

Figure 4.7. Scene (left) and its reflectogram 3217×1024 (right).

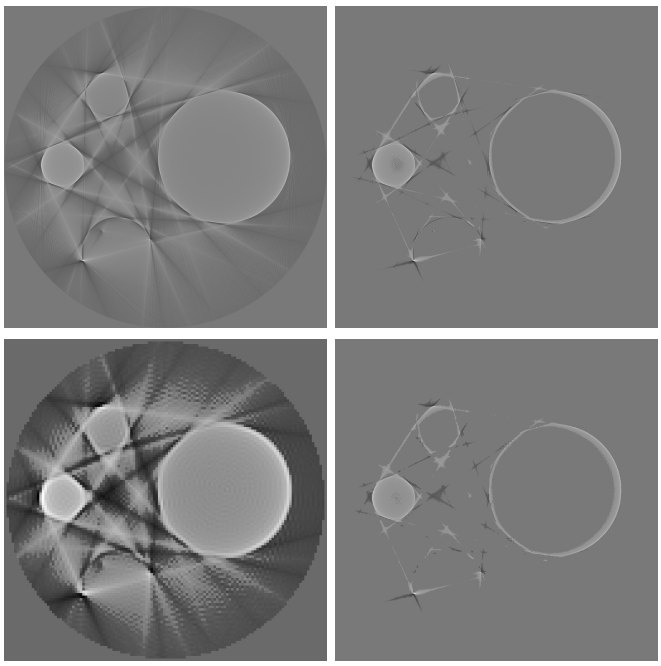

Figure 4.8. Reconstructions for the Figure 4.7. Top: reference before (left) and after extraction (right). Bottom: initialization (left) and final reconstruction (right) by the multiresolution method.

On Figure 4.8, we represent the full reference reconstruction, of size $n^2 = 1024^2$, and the extraction of the 0.05×1024^2 most intense pixels. The reconstruction and the extraction took 286 seconds. On the second line of Figure 4.8, we represent the reconstructions of the multiresolution method: for the initial grid of size 128^2 ($k_0 = 7$), and the final 0.05×1024^2 thinnest pixels. For this test the focus is $F = 0.948$, and the number of iterations is $N = 33$; the test took 25.9 seconds. By zooming we can see that the reference is slightly better; but it took 11 times more time.

Remark. For clarity, the images of Figure 4.8 do not represent $\mathcal{H}$, but $\sqrt{|\mathcal{H}|}\,\mathrm{sign}(\mathcal{H})$.

5. Three-dimensional extension

We extend the multiresolution greedy algorithm to a problem in three dimensions, for an acquisition by slice.

5.1. Reconstruction by slices

We measure a three-dimensional reflectogram, by slice: the collected reflective projections are the $P(\theta, s, z)$, where for every altitude z, $P(\cdot, \cdot, z)$ is a reflectogram in the horizontal plane $x_3 = z$. The full reconstruction is obtained horizontal slice by horizontal slice: for every altitude z, the reconstruction, restricted to the plane $x_3 = z$, is $(x_1, x_2) \mapsto \mathcal{R}^*[P(\cdot, \cdot, z) \star \psi_\Omega](x_1, x_2)$. We assume that the parameters θ, s, x_1, x_2 are discretized as before. Concerning the altitude, we assume that z scans the values $z_l, 1 \leqslant l \leqslant n$, with constant step $\Delta z = z_{l+1} - z_l$.

Similarly as the two dimensional problem, the question is: represent the 3D scene using at least αn^3 thin voxels, where $\alpha \in (0, 1]$. The reference method consists in computing a full reconstruction of

size n^3, by computing the reconstruction slice by slice, i.e. for every $z_l, 1 \leqslant l \leqslant n$. Then the voxels are sorted by decreasing intensity, and we select the first αn^3 voxels.

We would like to take advantage of the principle of Algorithm 6 in order to outperform this 3D process. The easiest way is the following: we directly apply Algorithm 6, for every slice of altitude $z_l, 1 \leqslant l \leqslant n$. Once Algorithm 6 is implemented, this is easy: we have only to add a loop over l. But this approach has a disadvantage: the 3D structure of the problem is not utilized. We will prefer operating selection on the whole volume, and not slice by slice. This is what we are going to do.

5.2. **Multiresolution computations**

We assume: $n = 2^p$ with $p \geqslant 1$ an integer. Let $0 \leqslant k \leqslant p$. At scale k, we divide the volume $[-R - \frac{\Delta t}{2}, R - \frac{\Delta t}{2}]^2 \times [z_1 - \frac{\Delta z}{2}, z_n + \frac{\Delta z}{2}]$ into parallelepiped voxels of size $\Delta t_k \times \Delta t_k \times \Delta z$, with $\Delta t_k = 2^{-k} 2R$, and whose center is a point of the grid:

$$G_k = \{(x_{ij}^k, z_\ell) = (-R - \tfrac{\Delta t}{2}, -R - \tfrac{\Delta t}{2}, z_\ell) + \Delta t_k (i - \tfrac{1}{2}, j - \tfrac{1}{2}, 0), 1 \leqslant i, j \leqslant 2^k, 1 \leqslant \ell \leqslant n\}.$$

Thus a refinement (increment of scale) is operated in the two horizontal directions; vertically, the discretization is always the thinnest possible. At the coarser scale, $k = 0$, the volume contains n slices with width Δz. At the thinner scale, $k = p$, the volume contains n^3 thin voxels, with $(x_{ij}^p, z_\ell) = (t_i, t_j, z_\ell)$. The volume unit is chosen such that the volume of a parallelepiped with scale k is $\Delta t_k^2 \Delta z = 4^{p-k}$ volume units.

The altitude z_ℓ of every point of G_k belongs to the altitudes where P is known. Thus, at scale k, at $x = (x_{ij}^k, z_\ell) \in G_k$, we compute using the 2D formula, with $\Omega_k = 2^{k-p} \lfloor \frac{n}{2} \rfloor \Delta\omega$:

$$\mathcal{H}_k^{\downarrow}(i, j, \ell) := \mathbb{1}_{|x_{ij}^k| \leqslant R - \Delta t_k} \Omega_k^{1.5} \mathcal{R}^*[P(\cdot, \cdot, z_\ell) \star \psi_{\Omega_k}](x_{ij}^k);$$

each $P(\cdot, \cdot, z_\ell) \star \psi_{\Omega_k}, 1 \leqslant \ell \leqslant n$, is downsampled, as in 2D.

5.3. **Multiresolution greedy algorithm**

5.3.1. *Principle*

We use the same principle as before: we start from a coarse reconstruction, and we iteratively refine the most intense voxels. At each iteration, we refine a set of voxels whose volume fills the volume to be filled. Essentially, in comparison with 2D, we must add an altitude coordinate to the objects.

5.3.2. *Filtering*

The filtering is operated angle by angle, scale by scale, altitude by altitude, and with a downsampling in t and θ. Here we take care of memory usage, so we combine reading of the data and filtering. We avoid the simultaneous storage of several full volumes: see the function `filteringMultiResol3D`. For every angle θ_j, the image with angle θ_j is loaded and filtered in Fourier space; then, for every scale k such that the angle θ_j is selected by the downsampling in θ, for every altitude z_ℓ, we invert the cut Fourier data.

5.3.3. *Iterative refining of cells*

The cells that we now define contain also an altitude. And now, refining a voxel with scale k, whose volume is 4^{p-k}, produces 4 sub-voxels. We get at the end the Algorithm 9, using the function `refine3D`.

Algorithm 7 Function $(R, \Delta t, \Delta\theta, [\tilde{P}_{\Omega_k}(\theta_j^{(k)}, t_l^{(k)}, z_\ell)]_{k,j,l,\ell}$=`filteringMultiResol3D`$(k_0, n, m)$

Inputs: initial scale k_0, sizes $n = 2^p$ and m.

Read Δt, $\Delta\theta$.
For every angle $\theta_j, 1 \leqslant j \leqslant m$,
  for every altitude $z_\ell, 1 \leqslant \ell \leqslant n$,
    read $[P(\theta_j, t_l, z_\ell)]_{1\leqslant l\leqslant n}$,
    compute $\hat{P}_\psi(\theta_j, t_\cdot, z_\ell) = \mathrm{FFT}[P(\theta_j, t_l, z_\ell)]_{1\leqslant l\leqslant n} \hat{\star} \frac{\pi}{R}[\frac{n}{2}, \ldots, 1, 0, 1, \ldots, \frac{n}{2} - 1]$;
  for every scale $k_0 \leqslant k \leqslant p$, $\quad \Omega_k = 2^{k-1}\frac{\pi}{R}$,
    if $\bar{j} = 1 + (j-1)2^{k-p}$ is integer, $\quad (\theta_{\bar{j}}^{(k)} = \theta_j)$
      for every altitude $z_\ell, 1 \leqslant \ell \leqslant n$,
        invert the cut frequency data:

$$\tilde{P}_{\Omega_k}(\theta_{\bar{j}}^{(k)}, t_\cdot^{(k)}, z_\ell) = \frac{\Delta t}{\Omega_k} \,\mathrm{IFFT}\,\mathrm{cuts}_k\, \hat{P}_\psi(\theta_{\bar{j}}^{(k)}, t_\cdot, z_\ell).$$

Output: parameters R, Δt, $\Delta\theta$ and downsampled filtered data $\tilde{P}_{\Omega_k}(\theta_j^{(k)}, t_l^{(k)}, z_\ell)$, $k_0 \leqslant k \leqslant p$, $1 \leqslant j \leqslant m_k, 1 \leqslant l \leqslant 2^k$, $1 \leqslant \ell \leqslant n$.

Algorithm 8 Function $v =$`refine3D`$(L_p, L, \tilde{L}, [\tilde{P}_{\Omega_k}(\theta_j^{(k)}, t_l^{(k)}, z_\ell)], \Delta\theta, R, \Delta t)$

Inputs: $L_p, L, \tilde{L}$: lists of cells with scale p, $< p$, $< p$; L is sorted;...

From the first cell $(k, i, j, \ell, \mathcal{H}_k^\downarrow(i, j, \ell))$ of L to four cells with scale $k+1$.
  Delete the first cell of L.
  The volume of the refined cell is $v = 4^{p-k}$.
  $K = k + 1, \quad \Omega_K = 2^{K-1}\frac{\pi}{R}, \quad \Delta t_K = 2^{p-K}\Delta t, \quad \Delta\theta_K = 2^{p-K}\Delta\theta.$
  For all $(I, J) \in \{2i, 2i-1\} \times \{2j, 2j-1\}$,
    $x = (-R - \frac{\Delta t}{2}, -R - \frac{\Delta t}{2}) + \Delta t_K(I - \frac{1}{2}, J - \frac{1}{2}), \quad (= x_{IJ}^K)$
    $\mathcal{H} = \Omega_K^{1.5}$`backprojection`$(x, [\tilde{P}_{\Omega_K}(\theta_j^{(K)}, t_l^{(K)}, z_\ell)]_{j,l}, \Delta\theta_K, R, \Delta t_K)$, $(= \mathcal{H}_K^\downarrow(I, J, \ell))$
    insert the new cell $(K, I, J, \ell, \mathcal{H})$ into L_p if $K = p$, into $\tilde{L}$ otherwise.

Output: volume v of the refined voxel (and updated lists).

Algorithm 9 Multiresolution greedy algorithm for 3D reflective tomography.

Inputs: Sizes $n = 2^p, m$; rate $\alpha \in (0, 1]$; initial scale $1 \leqslant k_0 < p$.

Initialization

Lists of cells: the lists L and L_p are void.

Reading and filtering: computation of the downsampled filtered data $\tilde{P}_{\Omega_k}(\theta_j^{(k)}, t_l^{(k)}, z_\ell)$.

$(R, \Delta t, \Delta\theta, [\tilde{P}_{\Omega_k}(\theta_j^{(k)}, t_l^{(k)}, z_\ell)]_{k,j,l,\ell})$=`filteringMultiResol3D`$(k_0, n, m)$

Backprojection: computation of a coarse reconstruction at scale k_0, $\mathcal{H}_{k_0}^{\downarrow}$.

$k = k_0, \qquad \Omega_k = 2^{k-1}\frac{\pi}{R}, \qquad \Delta t_k = 2^{p-k}\Delta t, \qquad \Delta\theta_k = 2^{p-k}\Delta\theta.$

For all $1 \leqslant \imath, \jmath \leqslant 2^k$ and $1 \leqslant \ell \leqslant n$,

$x = (-R - \frac{\Delta t}{2}, -R - \frac{\Delta t}{2}) + \Delta t_k(\imath - \frac{1}{2}, \jmath - \frac{1}{2}), \qquad (= x_{\imath\jmath}^k)$

$\mathcal{H} = \Omega_k^{1.5}$`backprojection`$(x, [\tilde{P}_{\Omega_k}(\theta_j^{(k)}, t_l^{(k)}, z_\ell)]_{j,l}, \Delta\theta_k, R, \Delta t_k), \qquad (= \mathcal{H}_k^{\downarrow}(\imath, \jmath, \ell))$

insert the cell $(k, \imath, \jmath, \ell, \mathcal{H})$ into L.

Sort: sort the list L by decreasing intensity $|\mathcal{H}|$.

Volume to be filled: $V = \alpha n^3$.

Iterations

While the number of thin voxels is too small, i.e. $V > 0$.

$v = 0$, $\tilde{L}$ is void.

While the accumulated volume of the refined cells is too small, i.e. $v < V$,

refine the most intense cell of L :

$v = v +$ `refine3D`$(L_p, L, \tilde{L}, [\tilde{P}_{\Omega_k}(\theta_j^{(k)}, t_l^{(k)}, z_\ell)], \Delta\theta, R, \Delta t)$.

EndWhile

Sorted list: sort $\tilde{L}$ by decreasing $|\mathcal{H}|$, then merge L and $\tilde{L}$ into L.

Volume to be filled: $V = \max(0, \alpha n^3 - |L_p|)$

EndWhile

Output: multiresolution reconstruction $L \cup L_p$; L_p contains $4\lceil\frac{\alpha n^3}{4}\rceil$ voxels at full resolution.

5.3.4. *Costs*

Similarly as the 2D analysis, we get Table 5.1 for the costs of Algorithm 9. Here again we will check *a posteriori* the value of N and we will compute the focus F, based on the number S of voxels computed at scales $k_0 < k < p$:

$$F = \frac{S_1 - S}{S_1 - S_0} \in [0,1], \quad \frac{1}{3}(1 - 4^{k_0+1-p})4\left\lceil \frac{\alpha n^3}{4} \right\rceil =: S_0 \leqslant S \leqslant S_1 := \frac{1}{3}(1 - 4^{k_0+1-p})n^3.$$

	Reference	**Multiresolution greedy**
Filtering		
- storage	$\mathcal{O}\left(mn^2\right)$	$\mathcal{O}\left(\frac{4}{3}mn^2\right)$
- operations	$\mathcal{O}\left(2mn^2 \log n\right)$	$\mathcal{O}\left(\frac{7}{3}mn^2 \log n\right)$
Backprojection		
- number	$\mathcal{O}\left(\frac{\pi}{4}n^3\right)$	at least $\mathcal{O}\left(\frac{\pi}{4}\frac{n^3}{4^{p-k_0}}\right) + \mathcal{O}\left(\alpha n^3 \frac{4}{3}(1 - \frac{1}{4^{p-k_0}})\right)$, at most $\mathcal{O}\left(\alpha n^3 + \frac{\pi}{4}\frac{n^3}{3}(1 - \frac{1}{4^{p-k_0}})\right)$
- operations	$\mathcal{O}\left(\frac{\pi}{4}mn^3\right)$	at least $\mathcal{O}\left(\frac{\pi}{4}\frac{mn^3}{8^{p-k_0}}\right) + \mathcal{O}\left(\alpha mn^3 \frac{8}{7}(1 - \frac{1}{8^{p-k_0}})\right)$, at most $\mathcal{O}\left(\alpha mn^3 + \frac{\pi}{4}\frac{mn^3}{7}(1 - \frac{1}{8^{p-k_0}})\right)$
- storage	$\mathcal{O}\left(n^3\right)$	$\mathcal{O}\left(\frac{n^3}{4}(1 + 3\alpha)\right)$
Selection		
- operations	$\mathcal{O}\left(n^3 \log n^3\right)$	$\mathcal{O}\left(\frac{n^3}{4^{p-k_0}} \log \frac{n^3}{4^{p-k_0}}\right) \ldots + \mathbb{1}_{k_0<p-1}\left[\mathcal{O}\left(\frac{n^3}{3} \log \frac{n^3}{3}\right) + \mathcal{O}\left(\frac{Nn^3}{4}\right)\right]$, $p - k_0 \leqslant N \leqslant \left\lceil \frac{\alpha n^3}{4} \right\rceil + \frac{n^3}{4 \cdot 3}(1 - \frac{1}{4^{p-k_0-1}})$

Table 5.1. Costs for 3D reflective tomography: reference method versus multiresolution greedy algorithm (Algorithm 9).

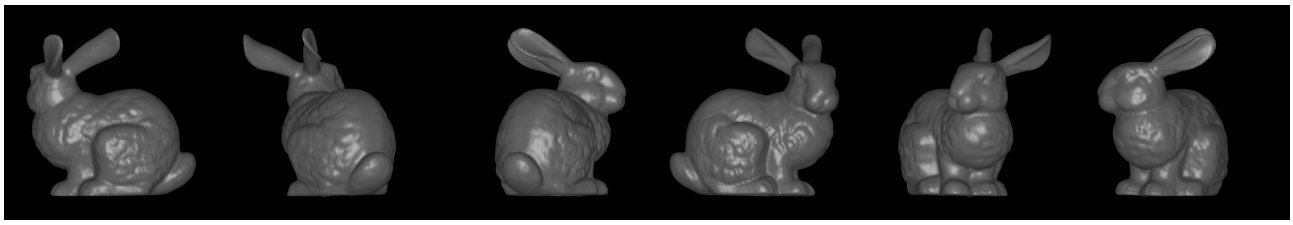

Figure 5.1. Simulated images of the Stanford Bunny, using the Gouraud model.

5.4. **Numerical tests**

5.4.1. *Simulation of optical images*

Using the triangulation of the Stanford Bunny [16], we simulated a sequence of $m = 805$ images of size $n \times n = 512 \times 512$, by turning around the bunny; the angular step is $\frac{2\pi}{m}$. These images are computed using the Gouraud model of `Matlab`, for orthographic projections (parallel rays). These synthesis images contain a component of scattering, and a component of specular reflection. See Figure 5.1 for a few images of the sequence. The pixels values depend particularly on the light source position, the pixel location and the normal to the surface at the visible point.

5.4.2. *Computation of the reconstructed images*

Concerning the reconstruction, we implemented the Algorithm 9 by adjusting the `Fortran` code of the 2D case. For the visualization, we project the thinnest voxels on three orthogonal planes, using the Maximum Intensity Projection (MIP - see for example [17]): for each plane of projection, along every line orthogonal to the plane, we project the intensity of the most intense positive voxel (or 0 if all the voxels of the line are negative). For clarity we apply the square root at each image, before printing.

5.4.3. *Reconstructions at 5%*

On Figure 5.3, we represent the reference reconstruction, before extraction, and after extraction of $\alpha = 5\%$ of the voxels. The computation took 8680 seconds. On Figure 5.4, we represent the greedy reconstruction, with initialization of size $128^2 \cdot 512$, and $\alpha = 5\%$. The computation needed 24 iterations, for a total time of 1060 seconds; this is 8 times smaller than the reference time. The focus is 0.962. To observe the differences between the extracted reconstructions, we subtracted the MIPs of the greedy reconstruction from the MIPs of the reference: see Figure 5.2. The renderings of the two methods are finally similar.

5.4.4. *Reconstructions at 1%*

We now compare the reference with the greedy method for $\alpha = 1\%$ (and initial size $128^2 \cdot 512$): see Figure 5.5. The computation took 8730 seconds for the reference, while it took 416 seconds for the greedy method; the ratio of computation times is 21. The greedy method converges in 27 iterations, with a focus 0.986.

5.4.5. *Speckle noise*

We would like to test the algorithm on a more challenging case with speckle noise. We simulate Gouraud images of the Stanford Bunny as before, but with a striped pattern on the surface: see Figure 5.6.

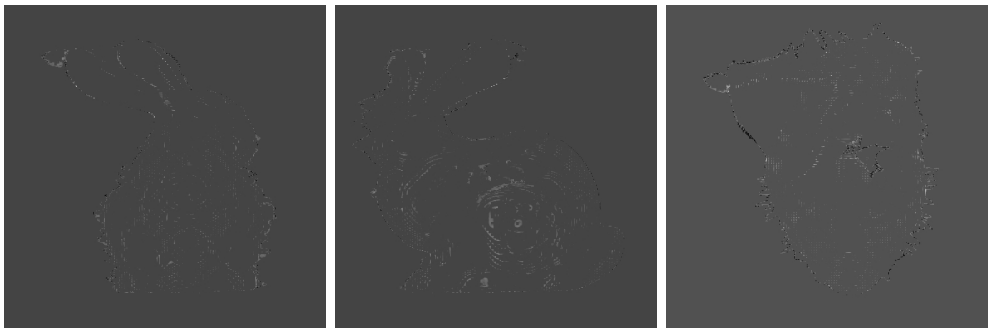

FIGURE 5.2. Greedy MIP subtracted from the reference MIP, for $\alpha = 5\%$.

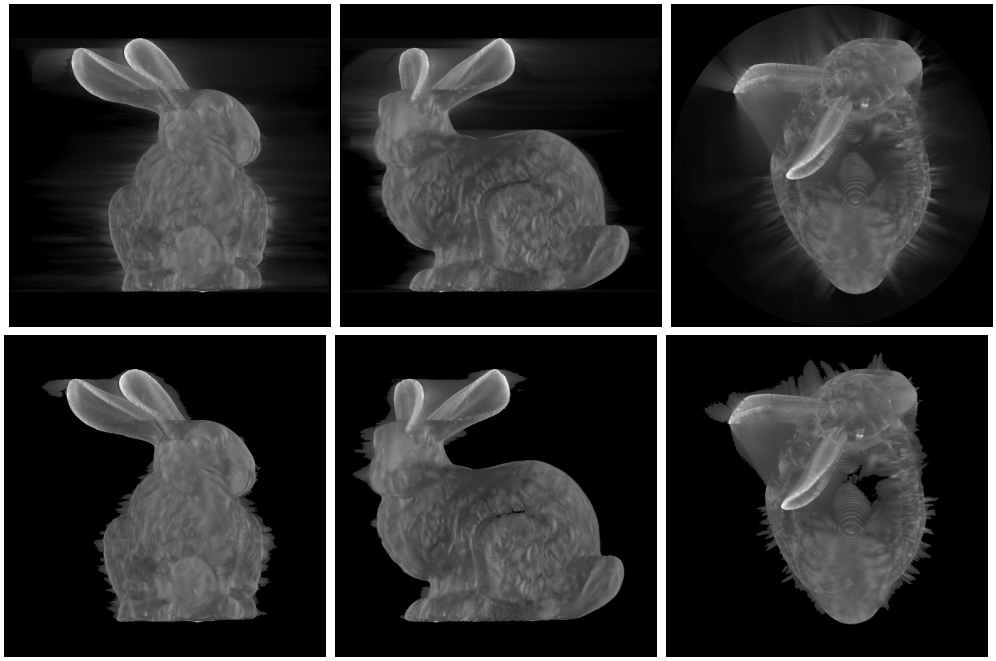

FIGURE 5.3. Reference reconstruction: full volume of size 512^3 (top), and extraction of $0.05 \cdot 512^3$ voxels (bottom).

After the rescaling $P := 1 + \frac{P}{\max(P)}$, the range of the values of P is $[1, 2]$. Then we add a speckle noise: each value $P(\theta, s, z)$ is replaced by $P(\theta, s, z) := P(\theta, s, z)(1 + \varepsilon(\theta, s, z))$, where ε contains independent realizations of the gaussian $\mathcal{N}(0, 1)$. After these changes, the first line of Figure 5.6 becomes the second one. Reconstructing the surface with such a level of noise is quite challenging.

We compare the reference with the greedy method for $\alpha = 0.001$, and initial size $64^2 \cdot 512$: see Figure 5.7. The computation took 8760 seconds for the reference, while it took 237 seconds for the greedy method (the ratio of time is 37). The greedy method converges in 96 iterations, with focus 0.996. The initial reconstruction of the greedy method is regularized thanks to the frequency cut; this helps the method to concentrate the computations on the surfaces. At the end, the cloud of the greedy reconstruction looks less diffuse than the reference one.

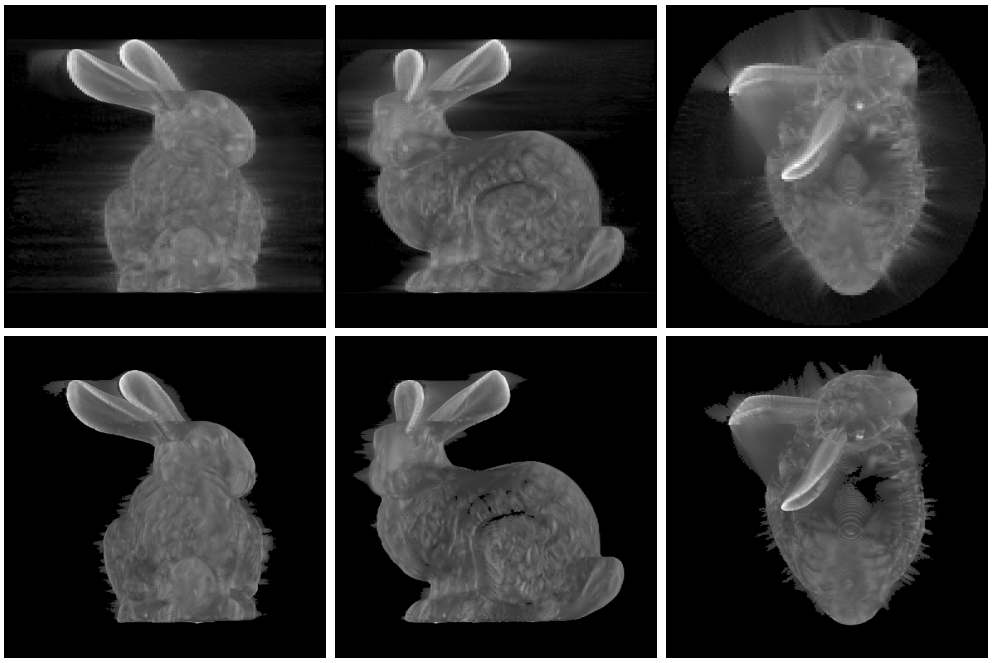

Figure 5.4. Greedy reconstruction: initialization of size $128^2 \cdot 512$ (top), and reconstruction at 5% (bottom).

6. Conclusion

6.1. Synthesis

These works provide an original solution to the following problem: extract the surfaces of a scene by reflective tomography.

6.1.1. *Multiresolution greedy algorithm*

This solution is a multiresolution greedy algorithm, derived from the high-frequency asymptotics of the reflective filtered backprojection. The idea: start from a coarse reconstruction on a coarse grid, then refine iteratively sets of voxels. Each iteration selects and refines a set of voxels maximizing the accumulated normalized filtered backprojection, with a volume constraint. In comparison with the reference, this approach does not compute the full volume at full resolution; the computations focus themselves on the regions of interest. The principles of the algorithm can be extended for a general class of problems, concerning the efficient determination of a set fixed by asymptotics.

6.1.2. *Performances*

The expected consequence is a reduction of the computation time. Concerning the complexity, we derived bounds for the costs. The lower bounds show that the method can be competitive for the most advantageous occurrences. The upper bounds are prohibitive, but they do not take into account the special structures of the data and of the solver. We measure *a posteriori* the efficiency of the algorithm, by looking at the number of iterations and a focus indicator. The observed values in the numerical tests reveal that the algorithm quickly converges and efficiently focuses.

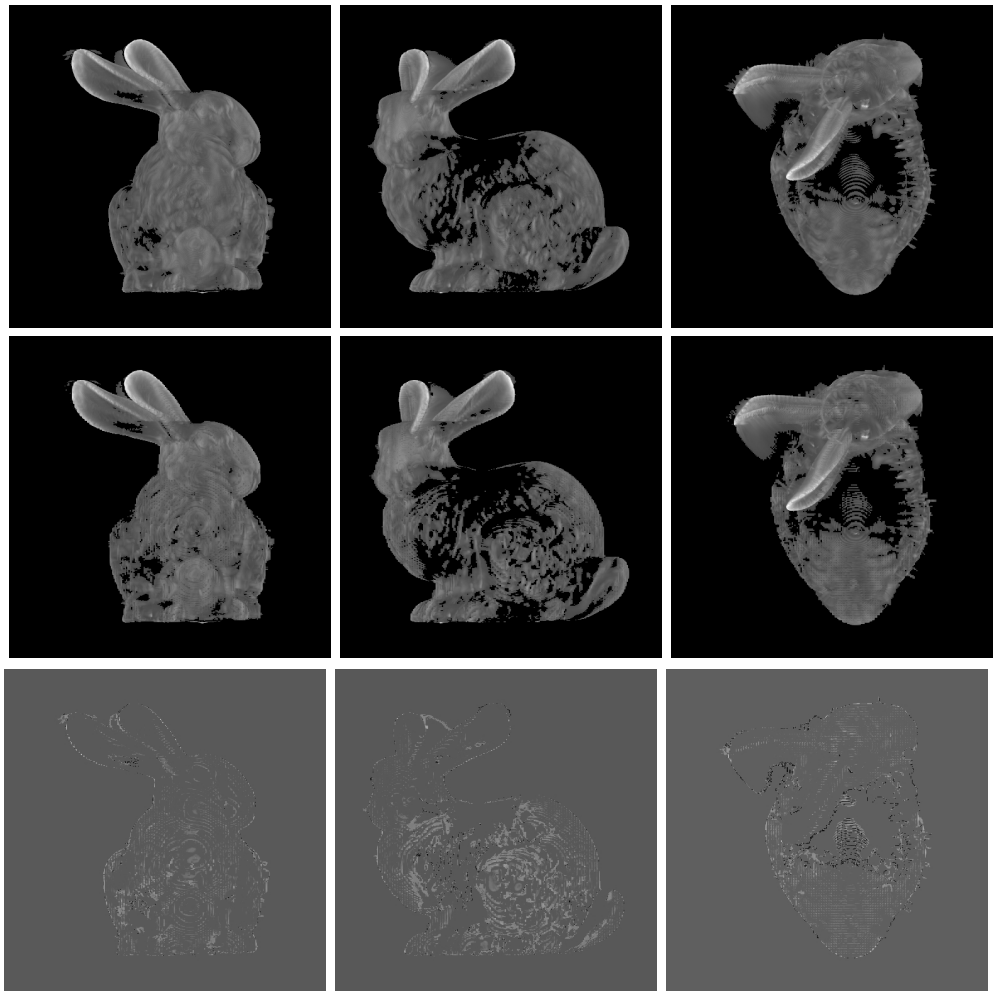

Figure 5.5. Reference reconstruction (top), greedy reconstruction with initialization $128^2 \cdot 512$ (middle), and difference between the two (bottom); $\alpha = 1\%$.

6.1.3. *Settings*

There is a price to be paid; two parameters must chosen in advance: initial resolution and rate of wished voxels. If they are too small, the reconstruction might be too sparse: pieces of surfaces not reconstructed. Based on empirical observations, a good compromise between computational effort and quality of the result: reconstruct a few percents of the full volume, from an initial resolution that is near the full resolution. In such a case, the reconstruction is expected to be close to the reference, but with a significant reduction of computation time.

6.1.4. *Accelerated solver*

At the end, the proposed method is an alternative to the reference method, and especially for large volumes of data. It accelerates the reference solver and it offers new opportunities to develop an algorithm which is as fast as possible.

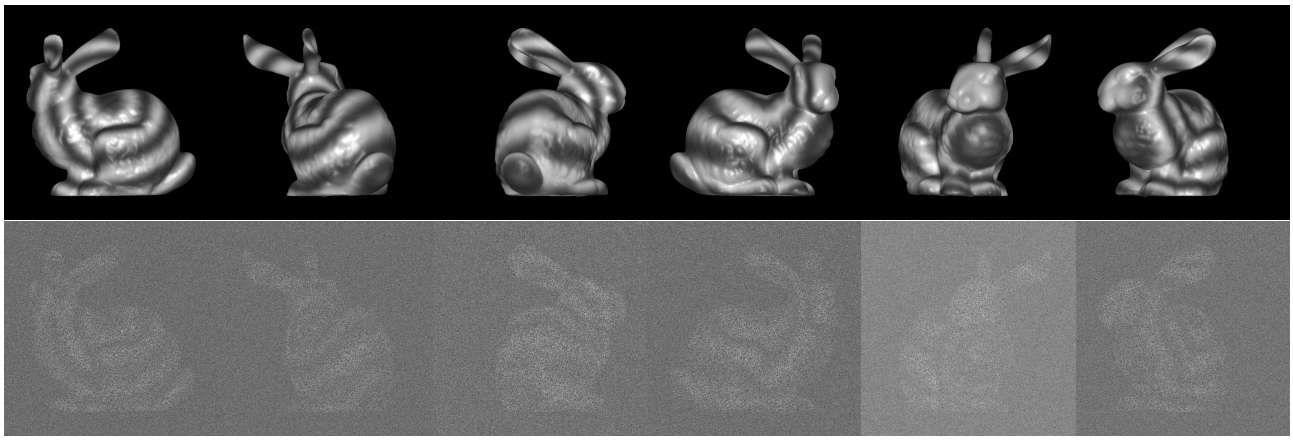

Figure 5.6. Gouraud images of a striped bunny: without noise (top) and with speckle noise (bottom).

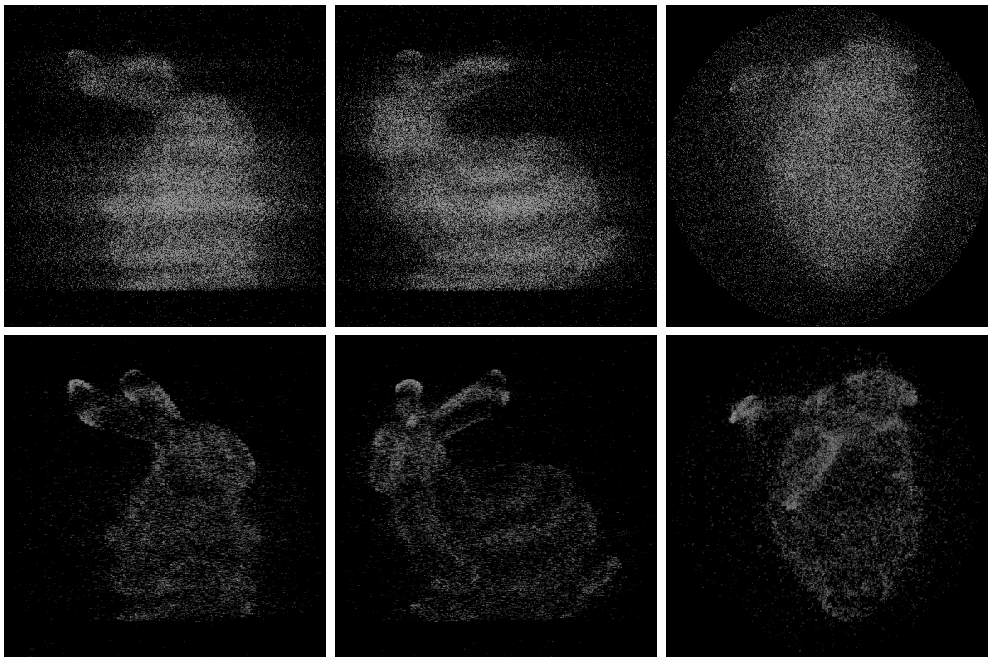

Figure 5.7. Reconstructions for the noisy striped bunny. Reference reconstruction (top), greedy reconstruction with initialization $64^2 \cdot 512$ (bottom); $\alpha = 0.001$.

6.2. **Perspectives**

We suggest several options to be explored.

6.2.1. *Complexity*

The theoretical analysis of complexity must be deepen. The bounds that we derived do not explore the special structure of the data, nor the form of the solver. It would be interesting to refine the bounds, in a framework to be determined. This question is difficult. Another point to be studied is the average-case complexity.

6.2.2. *Filtering*

We used the Ram-Lak filter: the method stands on the asymptotics of the corresponding reflective filtered backprojection. We can tune the process for other filters. If we want another filter, we must first study the associated asymptotics to find the right normalization, and to check that the algorithm will have the expected behaviour.

6.2.3. *Parallel computing*

By the way, in tomography, a standard way of gaining in performance is parallel computing. Here we introduced an acceleration principle in a sequential algorithm. It could be interesting to derive an algorithm which combines such a principle with parallel computing.

6.2.4. *Extension*

To finish with, the extended problem about the determination of an asymptotically discriminated set is relatively general, and thus it should be possible to apply the multiresolution greedy principles to other fields. It would be worth checking the relevancy of the concept on other problems that involve asymptotic methods such as stationary phase analysis.

References

[1] Jean-Baptiste Bellet. Analyse asymptotique et géométrique de la tomographie réflective. <hal-01571707>, 2017.

[2] Jean-Baptiste Bellet and Gérard Berginc. Reflective filtered backprojection. *Comptes rendus - Mathématique*, 354:960–964, 2016.

[3] I. Berechet and G. Berginc. Advanced algorithms for identifying targets from a three-dimensional reconstruction of sparse 3D Ladar data. In G. Berginc, editor, *Optical Complex Systems: OCS11, 81720Z*, volume 8172 of *Proc. of SPIE*, 2011.

[4] Stefan Berechet, Ion Berechet, Jean-Baptiste Bellet, and Gérard Berginc. Procédé de discrimination et d'identification par imagerie 3D d'objets d'une scène. Patent WO2016097168 A1, 2015.

[5] G. Berginc, J.-B. Bellet, I. Berechet, and S. Berechet. Optical 3D imaging and visualization of concealed objects. In *Proc. SPIE*, volume 9961, page 99610Q, 2016.

[6] G. Berginc and M. Jouffroy. Simulation of 3D laser systems. In *Geoscience and Remote Sensing Symposium, 2009 IEEE International, IGARSS 2009*, volume 2, pages 440–444. IEEE, 2009.

[7] G. Berginc and M. Jouffroy. Simulation of 3D laser imaging. *PIERS Online*, 6(5):415–419, 2010.

[8] Gérard Berginc. Scattering models for 1-D–2-D–3-D laser imagery. *Optical Engineering*, 56(3):031207, 2016.

[9] David T. Gering and W.M. Wells. Object modeling using tomography and photography. In *Multi-View Modeling and Analysis of Visual Scenes, 1999.(MVIEW'99) Proceedings. IEEE Workshop on*, pages 11–18. IEEE, 1999.

[10] Henri Gouraud. Continuous shading of curved surfaces. *IEEE transactions on computers*, 100(6):623–629, 1971.

[11] Markus Henriksson, Tomas Olofsson, Christina Grönwall, Carl Brännlund, and Lars Sjöqvist. Optical reflectance tomography using TCSPC laser radar. In *Proc. SPIE*, volume 8542, page 85420E, 2012.

[12] Berthold Horn. *Robot vision*. MIT press, 1986.

[13] F.K. Knight, S.R. Kulkarni, R.M. Marino, and J.K. Parker. Tomographic Techniques Applied to Laser Radar Reflective Measurements. *Lincoln Laboratory Journal*, 2(2):143–160, 1989.

[14] Aldo Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE Transactions on pattern analysis and machine intelligence*, 16(2):150–162, 1994.

[15] Charles Soussen and Jérôme Idier. 3D reconstruction of localized objects from radiographs and based on multiresolution and sparsity. In *IEEE International Conference on Image Processing*, volume 3, pages 744–747. IEEE, 2005.

[16] Greg Turk and Marc Levoy. Zippered polygon meshes from range images. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 311–318. ACM, 1994.

[17] J.W. Wallis and T.R. Miller. Three-Dimensional Display in Nuclear Medicine and Radiology. *The Journal of Nuclear Medicine*, pages 534–546, 1991.