A new linogram algorithm for computerized tomography

Daniel Potts
Medical University of Lübeck
Institute of Mathematics
Wallstr. 40
D–23560 Lübeck
potts@math.mu-luebeck.de

and

Gabriele Steidl
University of Mannheim
Institute of Computer Science
D–68131 Mannheim
steidl@math.uni-mannheim.de

Abstract. In this paper, we propose a new linogram algorithm for the high quality Fourier reconstruction of digital $N \times N$ images from their Radon transform. The algorithm is based on univariate fast Fourier transforms for nonequispaced data in the time domain and in the frequency domain. The algorithm requires only $\mathcal{O}(N^2 \log N)$ arithmetic operations and preserves the good reconstruction quality of the filtered backprojection.

1 Introduction

We are interested in efficient and high quality reconstructions of digital $N \times N$ medical images from their Radon transform. The standard reconstruction algorithm, the filtered backprojection, ensures a good quality of the images at the expense of $O(N^3)$ arithmetic operations. Fourier reconstruction methods reduce the number of arithmetic operations to $\mathcal{O}(N^2 \log N)$, a feature which will be of particular interest for future three–dimensional image processing. Unfortunately, the straightforward Fourier reconstruction algorithm suffers from unacceptable artifacts so that it is useless in practice. A better quality of the reconstructed images can be achieved by linogram algorithms [5, 21], the gridding algorithm [16, 20, 19], the unified Fourier reconstruction algorithm (UFR–algorithm) [12, 13] or by recently developed algorithms by K. Fourmont [8], J. Walden [25] and D. Gottlieb et al. [9].
The consideration of other algorithms than Fourier reconstruction algorithms for the inversion of the Radon transform, e.g. iterative algorithms, is beyond the scope of this paper. Here we refer to [15].
In this paper, we propose a new Fourier reconstruction algorithm having better or at least the same quality as the above algorithms. Our algorithm is based on recent developments in connection with the efficient computation of discrete Fourier transforms for nonequispaced data (NFFTs). By using the linogram geometry, our algorithm requires only univariate NFFTs.

Consequently, we can avoid the bivariate gridding step contained in the gridding algorithm and in the UFR–algorithm.

This paper is organized as follows: In Section 2, we provide fast approximative algorithms for the computation of discrete Fourier transforms for nonequispaced data recently developed in [4, 1, 23]. We apply these algorithms for the numerical inversion of the Radon transform in Section 3. Special attention is paid to the comparison of the NFFT and the chirp z–transform, which can be used instead of the NFFT in the first step of our reconstruction algorithm. Finally, Section 4 presents a numerical example.

2 Fast Fourier Transforms for nonequispaced data

While gridding methods in connection with efficient computations of discrete Fourier transforms for nonequispaced data were applied in digital signal processing for a long time [22, 16], the theoretical foundations for these methods, in particular the relation between the speed of the algorithm and the approximation error were developed only recently [4, 1, 23, 2]. On the other hand, the theoretical examinations lead to a couple of improved and modified fast Fourier transform algorithms for nonequispaced data [6, 17, 8, 26]. In the following, we shortly describe their basic idea.

Let $\Pi^d := [-\frac{1}{2}, \frac{1}{2})^d$ and $I_N := \{k \in \mathbb{Z}^d : -\frac{N}{2} \leq k < \frac{N}{2}\}$, where the inequalities hold componentwise. For $v_j \in N\Pi^d$, and $f_k \in \mathbb{C}$, we are interested in the fast and robust computation of the discrete Fourier transforms

$$f(v_j) = \sum_{k \in I_N} f_k \, \mathrm{e}^{-2\pi \mathrm{i} k v_j / N} \qquad (j \in I_M) \tag{2.1}$$

and

$$h(k) := \sum_{j \in I_M} f_j \, \mathrm{e}^{-2\pi \mathrm{i} k v_j / N} \qquad (k \in I_N)\,, \tag{2.2}$$

i.e. either the nodes in time or frequency domain are equispaced. It is easy to check that once we have an algorithm for the efficient computation of (2.1), we can simply design an efficient algorithm for (2.2) which we will call the "transposed" algorithm. Therefore, we restrict our attention to (2.1). For an algorithm with both nonequispaced nodes in time and frequency domain see [7]. Straightforward computation of (2.1) requires $\mathcal{O}(N^d M^d)$ arithmetic operations, too much for the applications we have in mind. Only in case of equispaced nodes $v_j := j$ ($j \in I_N$), the above values can be evaluated by the well–known *fast Fourier transform* (FFT) with only $\mathcal{O}(N^d \log N)$ arithmetic operations. To speed up the computation of (2.1), we suggest the following approximate procedure:

Instead of evaluating the 1–periodic trigonometric polynomial

$$f(v) := \sum_{k \in I_N} f_k \, \mathrm{e}^{-2\pi \mathrm{i} k v} \tag{2.3}$$

at the nodes $w_j := v_j / N \in \Pi^d$ ($j \in I_M$), we intend to evaluate a function of the form

$$s_1(v) := \sum_{l \in I_n} g_l \, \varphi(v - \frac{l}{n})\,.$$

2

Here φ is an 1–periodic function which we will specify later and $n := \alpha N$ with an *oversampling factor* $\alpha > 1$. Switching to the frequency domain, we obtain

$$s_1(v) = \sum_{k \in I_n} \hat{g}_k \, c_k(\varphi) \, \mathrm{e}^{-2\pi \mathrm{i} k v} + \sum_{r \in \mathbb{Z}^d \setminus \{0\}} \sum_{k \in I_n} \hat{g}_k \, c_{k+nr}(\varphi) \, \mathrm{e}^{-2\pi \mathrm{i}(k+nr)v} \tag{2.4}$$

with

$$\hat{g}_k := \sum_{l \in I_n} g_l \, \mathrm{e}^{2\pi \mathrm{i} k l / n}, \tag{2.5}$$

$$c_k(\varphi) := \int_{\Pi^d} \varphi(v) \mathrm{e}^{2\pi \mathrm{i} k v} \, \mathrm{d}v \quad (k \in \mathbb{Z}^d).$$

Let $c_k(\varphi) \neq 0$ $(k \in I_N)$. Since s_1 should be a good approximation of f, we suggest by comparing (2.3) with (2.4) to set

$$\hat{g}_k := \begin{cases} f_k / c_k(\varphi) & k \in I_N, \\ 0 & k \in I_n \setminus I_N. \end{cases} \tag{2.6}$$

Then the values g_l can be obtained from (2.5) by the reduced inverse d–variate FFT of size n. If φ is well–localized in time domain such that it can be approximated by a 1–periodic function ψ with $\operatorname{supp}\psi \cap \Pi^d \subseteq \frac{m}{n}\Pi^d$ $(m \ll n)$, then

$$f(w_j) \approx s_1(w_j) \approx s(w_j) = \sum_{l \in I_{n,m}(w_j)} g_l \, \psi\left(w_j - \frac{l}{n}\right), \tag{2.7}$$

where $I_{n,m}(w_j) := \{l \in I_N : n w_j - m \leq l \leq n w_j + m\}$. For fixed $w_j \in \Pi^d$, the above sum contains at most $(2m+1)^d$ nonzero summands.

In summary, we obtain the following algorithm for the fast computation of (2.1) with $\mathcal{O}((\alpha N)^d \log(\alpha N) + (2m+1)^d M^d)$ arithmetic operations:

Algorithm 2.1 (NFFT)
Input: $N \in \mathbb{N}$, $\alpha > 1$, $n := \alpha N$, $w_j \in \Pi^d$, $f_k \in \mathbb{C}$ $(j \in I_M, k \in I_N)$.
Precomputation: $c_k(\varphi)$ $(k \in I_N)$, $\psi(w_j - \frac{l}{n})$ $(j \in I_M, l \in I_{n,m}(w_j))$.

1. Form $\hat{g}_k := f_k / c_k(\varphi)$ $(k \in I_N)$.

2. Compute by d–variate reduced FFT

$$g_l := n^{-d} \sum_{k \in I_N} \hat{g}_k \, \mathrm{e}^{-2\pi \mathrm{i} k l / n} \quad (l \in I_n).$$

3. Set

$$s(w_j) := \sum_{l \in I_{n,m}(w_j)} g_l \, \psi\left(w_j - \frac{l}{n}\right) \quad (j \in I_M).$$

3

Output: $s(w_j)$ approximate value of $f(w_j)$ $(j \in I_M)$.

The corresponding "transposed" algorithm for the fast computation of (2.2) reads as follows:

Algorithm 2.2 (NFFT$^\mathbf{T}$)

Input: $N \in \mathbb{N}$, $\alpha > 1$, $n := \alpha N$, $w_j \in \Pi^d$, $f_j \in \mathbb{C}$ $(j \in I_M)$.
Precomputation: $c_k(\varphi)$ $(k \in I_N)$, $\psi(w_j - \frac{l}{n})$ $(l \in I_n, j \in J_{n,m}(l))$.

1. Set
$$\tilde{g}_l := \sum_{j \in J_{n,m}(l)} f_j \, \psi(w_j - \frac{l}{n}) \quad (l \in I_n),$$
   where $J_{n,m} := \{j \in I_M : l - m \leq nw_j \leq l + m\}$ $(l \in I_n)$.

2. Compute by d–variate reduced FFT
$$\tilde{c}_k(g) := n^{-d} \sum_{l \in I_n} \tilde{g}_l \, e^{-2\pi i k l / n} \quad (k \in I_N).$$

3. Form $\tilde{h}(k) := \tilde{c}_k(g)/c_k(\varphi)$ $(k \in I_N)$.

Output: $\tilde{h}(k)$ approximate value of $h(k)$ $(k \in I_N)$.

Step 3 of Algorithm 2.1 and Step 1 of Algorithm 2.2 are called "*gridding steps*".
Both algorithms introduce the same approximation error (cf. [7, 23]). For the NFFT this error is given by
$$E(w_j) := |f(w_j) - s(w_j)| \leq E_a(w_j) + E_t(w_j)$$

and splits by (2.4) and (2.7) into the *aliasing error* $E_a(w_j) := |f(w_j) - s_1(w_j)|$ and the *truncation error* $E_t(w_j) := |s_1(w_j) - s(w_j)|$. By (2.4) and (2.6), the aliasing error can be estimated by
$$E_a(w_j) \leq ||f||_1 \max_{k \in I_N} \sum_{r \in Z^d \setminus \{0\}} \left| \frac{c_{k+nr}(\varphi)}{c_k(\varphi)} \right|,$$

where $||f||_1 := \sum_{k \in I_N} |f_k|$. By (2.7), (2.6) and (2.5), the truncation error fulfills

$$E_t(w_j) \leq ||f||_1 \, n^{-d} \left(\max_{k \in I_N} |c_k(\varphi)|^{-1} \right) \sum_{l \in I_n} |\varphi(w_j - \frac{l}{n}) - \psi(w_j - \frac{l}{n})|.$$

Note that the truncation error may be zero, i.e. $\varphi = \psi$, if φ has compact support.
Thus, the whole approximation error $E(w_j)$ depends on the localization of the function φ in time and frequency domain. Clearly, by Heisenberg's uncertainty principle, there doesn't exist a window function φ with arbitrary good localization in both time and frequency domain. However, for various functions φ, it was proved that the approximation error $E(w_j)$ decays exponentially as a function of the "support width" m of ψ. In particular, we refer to

- [4, 23, 6, 2] for estimates with (tensor products of) Gaussian bells, Gaussian bells tapered with Hanning windows or with sinc–kernels,

- [1, 18] for estimates with (tensor products of) B–splines and to [6] for estimates with three–directional Box–splines,

- [10, 8] for estimates with (tensor products of) Kaiser–Bessel functions.

Note that other candidates for φ with good localization in time and frequency domain as for example prolate spheroidal functions are not suited for our algorithms since their evaluation at various points $w_j - \frac{l}{n}$ in the precomputation step is rather expensive. See also [2].
In the following, we apply the Algorithms 2.1 and 2.2 with $d = 1$, the dilated periodized Gaussian bell

$$\varphi(v) = (\pi b)^{-1/2} \sum_{r \in \mathbb{Z}} e^{-(n(v+r))^2/b} \tag{2.8}$$

and its truncated version

$$\psi(v) = (\pi b)^{-1/2} \sum_{r \in \mathbb{Z}} e^{-(n(v+r))^2/b} \chi_{[-m,m]}(n(v+r)), \tag{2.9}$$

where $b := \frac{2\alpha m}{(2\alpha-1)\pi}$ and where $\chi_{[-m,m]}$ denotes the characteristic function of $[-m, m]$. By [7], it is sufficient to choose $m = 5$ to obtain an approximation error $\leq 10^{-5}$ (single precision). Finally, it is remarkable that similar to the classical FFT the NFFT is more robust with respect to roundoff errors introduced by the finite arithmetic of the computer than the straightforward summation of (2.1) or (2.2) [18].

3 Application of NFFT and NFFT$^{\mathrm{T}}$ in computerized tomography

In this section, we propose a new Fourier reconstruction algorithm for computerized tomography, where we restrict our attention to the standard parallel scanning geometry. More precisely, we are interested in the inversion of the *Radon transform*
$R : L_2(\Omega) \to L_2([-1, 1] \times S^1; (1 - s^2)^{-1/2})$,

$$Rf(s, \varphi) := \int\limits_{(x,\theta)=s} f(x)\, \mathrm{d}x \qquad (\theta := (\cos \varphi, \sin \varphi)^T)$$

based on the *Fourier Slice Theorem*

$$\hat{f}(\sigma\theta) = \int\limits_{-\infty}^{\infty} Rf(s, \varphi)e^{-2\pi \mathrm{i}s\sigma}\, \mathrm{d}s = \hat{R}f(\sigma, \varphi). \tag{3.1}$$

We suppose that $\operatorname{supp} f \subseteq \Omega := \{x \in \mathbb{R}^2 : ||x|| \leq 1\}$. We want to reconstruct f on the grid

$$(x_j, y_k) := \left(j\frac{2}{N}, k\frac{2}{N}\right) \qquad (j, k = -\frac{N}{2}, \dots \frac{N}{2} - 1).$$

Let Rf be given at the grid points

$$(s_r, \varphi_t) := \left(r\frac{2}{R}, t\frac{\pi}{T}\right) \qquad (t = 0, \dots, T - 1; r = -\frac{R}{2}, \dots, \frac{R}{2} - 1),$$
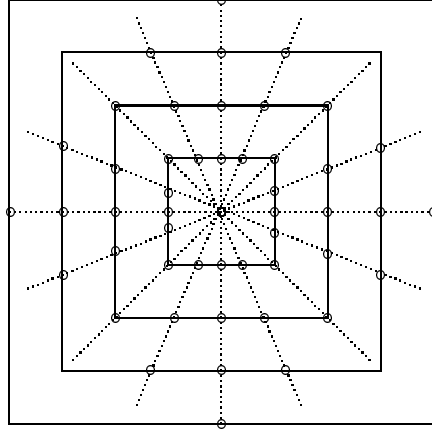
5

Figure 1: Linogram of Algorithm 3.2

where, by [3], $R \geq N$ and $T \geq \frac{\pi R}{2}$.

The standard Fourier reconstruction method follows directly from (3.1) and consists of the following three steps:

Algorithm 3.1

1. Computation of

$$\hat{f}(\frac{m}{\gamma}\theta_t) = \hat{R}f(\frac{m}{\gamma}, \varphi_t) := \frac{2}{R} \sum_{r=-\frac{R}{2}}^{\frac{R}{2}-1} Rf(r\frac{2}{R}, \varphi_t)\mathrm{e}^{-2\pi irm/(\frac{R\gamma}{2})}$$

   for $m = -\frac{R\gamma}{4}, \ldots, \frac{R\gamma}{4} - 1; t = 0, \ldots T - 1$ by T univariate FFT's of length $\frac{R\gamma}{2}$. Here $\frac{\gamma}{2} \geq 1$ is an oversampling factor.

2. Interpolation from the polar grid to the cartesian grid.

3. Computation of $f(x_j, y_k)$ $(j, k = -\frac{N}{2}, \ldots, \frac{N}{2} - 1)$ by bivariate FFT of size $\frac{\gamma}{2}N$.

The above algorithm produces essential artifacts so that it is useless in practice. In [14], F. Natterer proved that most of these artifacts result from the interpolation in the radial direction in Step 2. This justifies a couple of higher quality Fourier reconstruction algorithms as:

- *Angular interpolation algorithm* [8, 25]
  By applying T univariate NFFTs of length N in Step 1, the algorithm requires only linear interpolations in angular directions in Step 2.

- *Gridding algorithm / UFR-algorithm* [16, 20, 12, 13, 21, 19]
  Here Step 1 of Algorithm 3.1 is computed with oversampling factor $\frac{\gamma}{2} > 1$, i.e., $\frac{\gamma}{2} = 4$ [21]. Instead of Step 2, a bivariate gridding step is performed which approximates the values of $\hat{f}$ on the cartesian grid. Step 3 coincides with the corresponding step of Algorithm 3.1.

  Note that [19] presents a new gridding algorithm based on bivariate NFFT^{T}s. A similar algorithm is also in progress by K. Fourmont and F. Natterer.

- *Linogram algorithms* [5, 21]

  By T chirp–z transforms of length $\frac{R\gamma}{2}$ in Step 1, the algorithm requires only linear interpolations in x–direction and in y–direction, respectively, i.e. in "nearly" angular directions in Step 2. Another version of the Linogram algorithm [5] computes linear interpolations in the Radon domain (sinogram). Then the linear interpolations in Step 2 can be avoided by using chirp–z transforms in Step 3, too.

In the following, we propose a Fourier reconstruction algorithm which is based on the linogram geometry, but avoids linear interpolations by using NFFTs and NFFT^{T}s. The algorithm includes only univariate transforms so that a bivariate gridding step as in the gridding algorithm or in the UFR–algorithm is not necessary.

We sketch the algorithm first and give some explanations later.

Let T be divisible by 4.

Algorithm 3.2 (NFFT/NFFT$^\mathbf{T}$)

1. Computation of

$$\hat{f}(\frac{m}{\gamma}\frac{1}{\cos\varphi_t}\theta_t) = \hat{R}f(\frac{m}{\gamma}\frac{1}{\cos\varphi_t},\varphi_t) := \frac{2}{R}\sum_{r=-\frac{R}{2}}^{\frac{R}{2}-1} Rf(r\frac{2}{R},\varphi_t)\mathrm{e}^{-2\pi\mathrm{i}rm\frac{1}{\cos\varphi_t}/(\frac{R\gamma}{2})}$$

   for $t = 0,\dots,\frac{T}{4},\frac{3T}{4},\dots,T-1$ and $m = \lceil-\frac{R\gamma}{4}\cos\varphi_t\rceil,\dots,\lceil\frac{R\gamma}{4}\cos\varphi_t\rceil-1$,

$$\hat{f}(\frac{m}{\gamma}\frac{1}{\sin\varphi_t}\theta_t) = \hat{R}f(\frac{m}{\gamma}\frac{1}{\sin\varphi_t},\varphi_t) := \frac{2}{R}\sum_{r=-\frac{R}{2}}^{\frac{R}{2}-1} Rf(r\frac{2}{R},\varphi_t)\mathrm{e}^{-2\pi\mathrm{i}rm\frac{1}{\sin\varphi_t}/(\frac{R\gamma}{2})}$$

   for $t = \frac{T}{4}+1,\dots,\frac{3T}{4}-1$ and $m = \lceil-\frac{R\gamma}{4}\sin\varphi_t\rceil,\dots,\lceil\frac{R\gamma}{4}\sin\varphi_t\rceil-1$ by univariate NFFTs. Let the other values $\hat{f}(\frac{m}{\gamma}\frac{1}{\cos\varphi_t}\theta_t)$ and $\hat{f}(\frac{m}{\gamma}\frac{1}{\sin\varphi_t}\theta_t)$ $m \in (-\frac{R\gamma}{4},\dots,\frac{R\gamma}{4}-1)$ be zero.

2. Computation of

$$f_1(x_j,y_k) := \frac{\pi}{\gamma^2 T}\sum_{m=-\frac{R\gamma}{4}}^{\frac{R\gamma}{4}-1}\nu_m\sum_{t=-\frac{T}{4}}^{\frac{T}{4}-1}\frac{1}{\cos^2\varphi_t}\hat{f}(\frac{m}{\gamma},\frac{m}{\gamma}\frac{\sin\varphi_t}{\cos\varphi_t})\mathrm{e}^{2\pi\mathrm{i}\frac{\sin\varphi_t}{\cos\varphi_t}mk/(\frac{N\gamma}{2})}\mathrm{e}^{2\pi\mathrm{i}jm/(\frac{N\gamma}{2})}$$

$$f_2(x_j,y_k) := \frac{\pi}{\gamma^2 T}\sum_{m=-\frac{R\gamma}{4}}^{\frac{R\gamma}{4}-1}\nu_m\sum_{t=-\frac{T}{4}+1}^{\frac{T}{4}}\frac{1}{\cos^2\varphi_t}\hat{f}(\frac{m}{\gamma}\frac{\sin\varphi_t}{\cos\varphi_t},\frac{m}{\gamma})\mathrm{e}^{2\pi\mathrm{i}\frac{\sin\varphi_t}{\cos\varphi_t}mj/(\frac{N\gamma}{2})}\mathrm{e}^{2\pi\mathrm{i}km/(\frac{N\gamma}{2})}$$

   $(j,k = -\frac{N}{2},\dots,\frac{N}{2}-1)$ by $\frac{R\gamma}{2}$ univariate NFFT^{T}s of length $\frac{N\gamma}{2}$ for the inner sums and N univariate FFTs of length $\frac{N\gamma}{2}$ for the outer sum, where

$$\nu_m := \begin{cases} -m & m = -\frac{R\gamma}{4},\dots,-1\,, \\ \frac{1}{6\gamma^2} & m = 0\,, \\ m & m = 1,\dots,\frac{R\gamma}{4}-1\,. \end{cases}$$

   Set

$$f(x_j,y_k) := \frac{1}{2}\mathrm{Re}(f_1(x_j,y_k) + f_2(x_j,y_k))\,.$$

The algorithms uses the fact that

$$\hat{f}(-\sigma\theta) = \overline{\hat{f}(\sigma\theta)}\,. \tag{3.2}$$

Based on (3.2) further improvements are possible which are incorporated in our implementation but will not be described in detail here. Further, as usual a "filter step" in the Fourier domain will be added after Step 1. Here we apply sinc filter.

Let us give some more comments concerning our algorithm.

First step
For arbitrary fixed φ_t, let $h(s) := Rf(s, \varphi_t)$ and $\hat{h}(\sigma) := \hat{R}f(\sigma, \varphi_t)$. In the first step of our algorithm, we discretize the integral

$$\hat{h}(\sigma) = \int\limits_{-1}^{1} h(s) \mathrm{e}^{-2\pi \mathrm{i} s\sigma} \, \mathrm{d}s.$$

By Poisson's summation formula, we obtain

$$\hat{h}(\sigma) + \sum_{\substack{n \in \mathbb{Z} \\ n \neq 0}} \hat{h}(\sigma + n\frac{R}{2}) = \frac{2}{R} \sum_{r=-\frac{R}{2}}^{\frac{R}{2}-1} h(r\frac{2}{R}) \mathrm{e}^{-2\pi \mathrm{i} r\sigma/(\frac{R}{2})}\,. \tag{3.3}$$

Since we want to reconstruct only details of f of size $\geq \frac{2}{N}$ and $R \geq N$, we can assume by Shannon's sampling theorem, that $\hat{h}(\sigma)$ is negligible small for $|\sigma| > \frac{R}{4}$. Thus, the right–hand side of (3.3) is a good approximation of $\hat{h}(\sigma)$ for $\sigma \in [-\frac{R}{4}, \frac{R}{4}]$.

Second step
In the second step of our algorithm, we compute a discretized form of the integral

$$
\begin{aligned}
f(x, y) &= \int\limits_{-\infty}^{\infty} \int\limits_{-\infty}^{\infty} \hat{f}(u, v) \mathrm{e}^{2\pi \mathrm{i}(ux+vy)} \, \mathrm{d}u \mathrm{d}v \\
&= \int\limits_{0}^{\infty} \sigma \int\limits_{-\pi}^{\pi} \hat{f}(\sigma \cos\varphi, \sigma \sin\varphi) \mathrm{e}^{2\pi \mathrm{i}\sigma(\cos\varphi\, x + \sin\varphi\, y)} \, \mathrm{d}\sigma \mathrm{d}\varphi.
\end{aligned}
$$

Since the inner integral considered as function of σ is even, the above formula can be rewritten as

$$f(x, y) = \frac{1}{2} \int\limits_{-\pi}^{\pi} \int\limits_{-\infty}^{\infty} |\sigma| \hat{f}(\sigma \cos\varphi, \sigma \sin\varphi) \mathrm{e}^{2\pi \mathrm{i}\sigma(\cos\varphi\, x + \sin\varphi\, y)} \, \mathrm{d}\varphi \mathrm{d}\sigma\,. \tag{3.4}$$

We consider the inner integral. For arbitrary fixed $(x, y) \in [-1, 1]^2$ and $\varphi \in [-\pi, \pi]$, we set

$$
\begin{aligned}
g(\sigma) &:= |\sigma| \hat{f}(\sigma \cos\varphi, \sigma \sin\varphi) \mathrm{e}^{2\pi \mathrm{i}\sigma(\cos\varphi\, x + \sin\varphi\, y)}, \\
\hat{g}(v) &:= \int\limits_{-\infty}^{\infty} g(\sigma) \mathrm{e}^{-2\pi \mathrm{i} v\sigma} \, \mathrm{d}\sigma.
\end{aligned}
$$

Then we obtain by Poisson's summation formula

$$\hat{g}(0) + \sum_{\substack{n \in \mathbb{Z} \\ n \neq 0}} \hat{g}(\gamma \cos\varphi\, n) = \frac{1}{\gamma \cos\varphi} \sum_{m \in \mathbb{Z}} g\left(\frac{m}{\gamma \cos\varphi}\right) \qquad (\varphi \in [-\tfrac{\pi}{4}, \tfrac{\pi}{4}])$$

and since $\hat{f}(\sigma \cos\varphi, \sigma \sin\varphi)$ is negligible small for $|\sigma| > \frac{R}{4}$,

$$\hat{g}(0) + \sum_{\substack{n \in \mathbb{Z} \\ n \neq 0}} \hat{g}(\gamma \cos\varphi\, n) \approx \frac{1}{\gamma \cos\varphi} \sum_{m=\lceil -\frac{R\gamma}{4}\cos\varphi\rceil}^{\lceil \frac{R\gamma}{4}\cos\varphi\rceil - 1} g\left(\frac{m}{\gamma \cos\varphi}\right) \qquad (\varphi \in [-\tfrac{\pi}{4}, \tfrac{\pi}{4}]).$$

The aliasing error on the left–hand side becomes smaller with increasing γ. For example, since $\cos\varphi \geq \frac{\sqrt{2}}{2}$ ($\varphi \in [-\frac{\pi}{4}, \frac{\pi}{4}]$)., the right–hand side is a good approximation of $\hat{g}(0)$ if $\gamma = \sqrt{2}$ and if $\hat{g}(n)$ ($n \in \mathbb{Z}\backslash\{0\}$) is negligible small.

Under the assumption that $\tilde{g}(\sigma) := \sigma \hat{f}(\sigma \cos\varphi, \sigma \sin\varphi) e^{2\pi i\sigma(\cos\varphi\, x + \sin\varphi\, y)}$ has "essential" band-width $\ll \gamma \cos\varphi$, the aliasing error can be estimated as in [15] by

$$\sum_{\substack{n \in \mathbb{Z} \\ n \neq 0}} \hat{g}(\gamma \cos\varphi n) \approx \frac{1}{6\gamma^2 \cos^2\varphi} \hat{f}(0,0). \tag{3.5}$$

Since we only want to reconstruct details of f of size $\geq \frac{2}{N}$, we can discretize the outer integral in (3.4) with small aliasing error by the trapezoidal rule at the nodes $\varphi_t = t\frac{\pi}{T}$ ($t = -T, \dots, T-1$) if $T \geq \pi\frac{R}{2}$. This results in Step 2 of our Algorithms 3.2. In particular, (3.5) explains the coefficient ν_0 of $\hat{f}(0,0)$.

In Step 1 of Algorithm 3.2, the NFFTs can be replaced by chirp z–transforms as follows: We want to compute

$$\hat{f}_m = \sum_{r=-\frac{R}{2}}^{\frac{R}{2}-1} f_r e^{-2\pi i r m c/R} \quad (m = -\frac{R}{2}, \dots, \frac{R}{2} - 1), \tag{3.6}$$

where $\hat{f}_m := \hat{f}(\frac{m}{\gamma}\frac{1}{\cos\varphi_t}\theta_t)$, $f_r := \frac{2}{R} Rf(r\frac{2}{R}, \varphi_t)$, $c := 1/\cos(\varphi_t)$ and $\gamma = 2$. Using

$$rm = -\frac{(m-r)^2}{2} + \frac{r^2}{2} + \frac{m^2}{2},$$

and setting

$$g_r := f_r e^{-2\pi i r^2 c/(2R)}, \tag{3.7}$$
$$\tilde{g}_m := \hat{f}_m e^{2\pi i m^2 c/(2R)}, \tag{3.8}$$

(3.6) can be rewritten as

$$\tilde{g}_m = \sum_{r=-\frac{R}{2}}^{\frac{R}{2}-1} g_r e^{2\pi i(m-r)^2 c/(2R)} \quad (m = -\frac{R}{2}, \dots, \frac{R}{2} - 1). \tag{3.9}$$

9

Now (3.9) can be computed by the cyclic convolution $(\tilde{g}_m)_{m=-R}^{R-1} := \boldsymbol{g} * \boldsymbol{w}$ of length $2R$ of the vectors $\boldsymbol{w} := \left(\mathrm{e}^{-2\pi\mathrm{i}r^2 c/(2R)}\right)_{r=-R}^{R-1}$ and $\boldsymbol{g} := (g_r)_{r=-R}^{R-1}$, where $g_r := 0$ if $r \notin \{-\frac{R}{2}, \frac{R}{2} - 1\}$. Note that we only need the inner R components of $\tilde{\boldsymbol{g}}$. As usual, we can use FFTs for the efficient computation of cyclic convolutions.

In summary, the fast computation of (3.6) by the chirp z–transform requires

1. R premultiplications (real number – complex number) (see (3.7))

2. Cyclic convolution via

    2.1. reduced FFT of length $2R$ of the complex vector $\boldsymbol{g}$
    2.2. FFT of length $2R$ of the complex, even vector $\boldsymbol{w}$
    2.3. $2R$ multiplication of complex numbers
    2.4. reduced inverse FFT of length $2R$ of the complex vector $\widehat{\boldsymbol{g} * \boldsymbol{w}}$

3. R postmultiplications of complex numbers (see (3.8))

In contrast, if we compute (3.6) by the NFFT with $\alpha = 2$, (2.8), (2.9) and $m = 5$, we need

1. R premultiplications (real number – complex number)

2. reduced FFT of length $2R$ of a complex vector

3. $R(2m + 2) = 12\,R$ multiplications (real number – complex number) and $R(2m + 1)$ additions of complex numbers, i.e., $2R(2m + 2)$ multiplications of real numbers and $2R(2m + 1)$ additions of real numbers

The arithmetic complexity of the Steps 1 and 2 of the NFFT is the same as the arithmetic complexity of the Steps 1 and 2.4 of the chirp z–transform. If we assume that the FFT of length R for complex data can be computed with $\approx R \log R - 3R$ multiplications (of real data) and $\approx 3R \log R - 3R$ additions (of real data), then the Steps 2.2 – 3 of the chirp–z transform require $\approx 4R \log(2R)$ multiplications and $\approx 12R \log(2R)$ additions. These are significantly more arithmetic operations than those needed in Step 3 of the NFFT for $R \geq 32$. Our numerical results (Table 1) confirm these considerations.

4 Numerical examples

A commonly examined model in computerized tomography is the Shepp–Logan Phantom of the brain. This model consists of several ellipses so that its Radon transform can be evaluated analytically. In order to get a sampled version of the phantom and its Radon transform we have used the software packages "RadonAna" [24]. The algorithms were implemented in C on a Sun Ultrasparc–II 248MHz.

The original image (Figure 2 (left)) is of size $N \times N = 180 \times 180$ and its sinogram of size $R \times T = 180 \times 600$. Figure 2 (right) presents the reconstructed image obtained by the filtered backprojection. Here we have used the software package "iradon" [24].

The reconstructed image in (Figure 3 (left)) was computed by the linogram algorithm with linear interpolation in x and y directions (NFFTL) [21], where we have used the filter sinc^3 in the Fourier domain.

Figure 3 (right) shows that our Algorithm 3.2 leads to higher quality images than the above convenient linogram algorithm. Here Algorithm 3.2 was applied with the oversampling factor
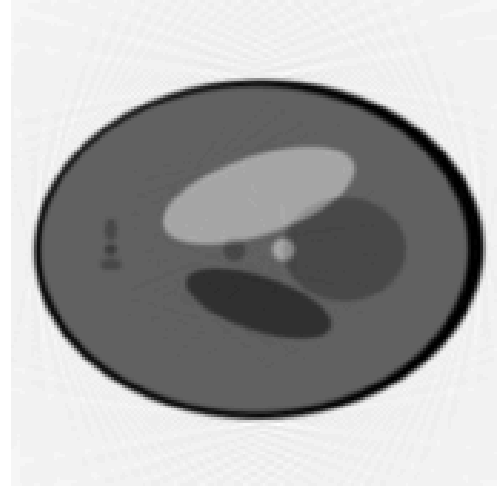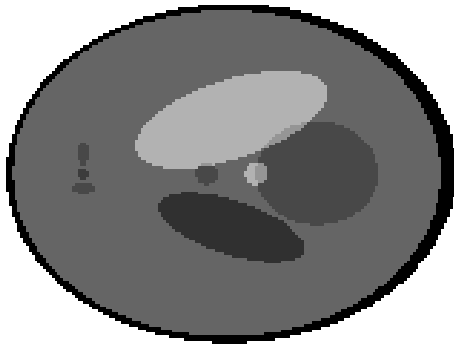
Figure 2: Shepp–Logan phantom, original (left) and reconstructed image by filtered backprojection (right).
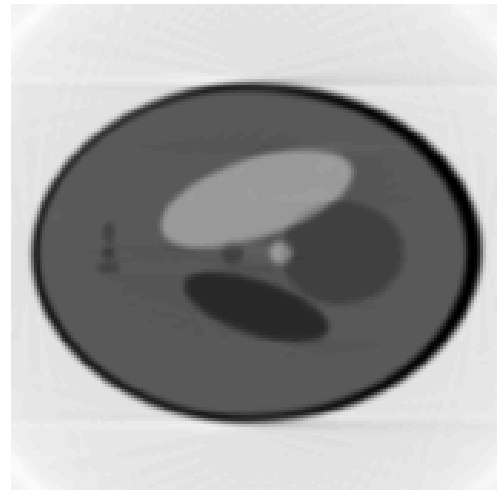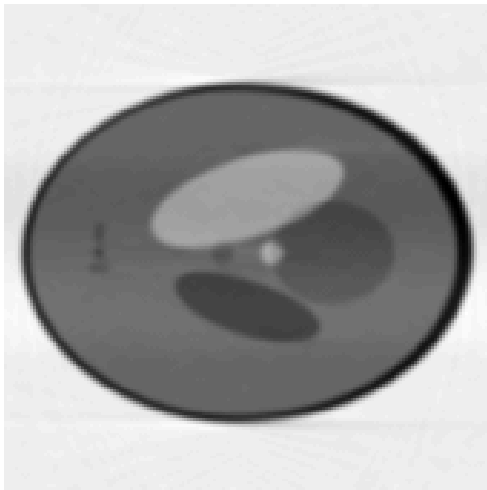

Figure 3: Reconstructed image by Algorithm NFFTL (left) and by Algorithm 3.2 (right).

$\frac{\gamma}{2} = \frac{256}{180}$ ($\approx \sqrt{2}$). Oversampling seems to be only necessary for artificial images containing high frequencies. Step 1 of Algorithm 3.2 was realized by Algorithm 2.1 with $d = 1$, oversampling factor $\alpha = 2$, $m = 5$ and $b = \frac{20}{3\pi}$. We have chosen φ as dilated periodized Gaussian bell (2.8) and ψ as its truncated version (2.9). Step 2 of Algorithm 3.2 was computed by Algorithm 2.2 with the same parameters as in Step 1. Further we have used the sinc filter in the Fourier domain.

Next we compare the computation time of the filtered backprojection and of different linogram algorithms. Note that our FFT algorithms are not fully optimized, i.e. the computation time may be further improved e.g. by using the FFTW–library or reduced FFTs.

11

We compare the following algorithms:

FB Filtered backprojection (implementation by [24])

ChirpL Linogram algorithm with chirp z–transforms in the first step, linear interpolation in x and y direction, respectively, and Step 3 of Algorithm 3.1 (twodimensional FFT)

NFFTL Linogram algorithm with NFFTs ($\alpha = 2$, $m = 5$, $b = \frac{20}{3\pi}$) in the first step, linear interpolation in x and y direction, respectively, and Step 3 of Algorithm 3.1 (twodimensional FFT)

Chirp/NFFT^T Linogram algorithm with chirp z–transforms in the first step and NFFT^{T}s in the second step

NFFT/NFFT^T Linogram algorithm with NFFTs in the first step and NFFT^{T}s in the second step, i.e. Algorithm 3.2

The second column and third column of Table 1 show the size $R \times T$ of the given sinogram and the forth column the size of the reconstructed image. Note that we have used oversampling factors $\gamma = \frac{256}{180}$ and $\gamma = \frac{512}{362}$ ($\approx \sqrt{2}$), respectively. The fifth column contains the computation time in seconds. The last column presents the contribution of FFT algorithms to the total computation time. As expected, the algorithms with NFFTs in the first step (NFFTL, NFFT/NFFT^T) are much faster than the algorithms with chirp–z transforms in the first step (ChirpL, Chirp/NFFT^T). The quality of the reconstructed images is the same for ChirpL, NFFTL and for Chirp/NFFT^T, NFFT/NFFT^T, respectively.

	R	T	N	time in s	% FFT
FB	180	600	180	20.2	11.7
ChirpL	180	600	180	4.22	80.3
NFFTL	180	600	180	2.08	43.6
Chirp/NFFT^T	180	600	180	5.63	66.7
NFFT/NFFT^T	180	600	180	3.5	32.6
FB	362	900	362	127.81	3.49
ChirpL	362	900	362	15.32	84.1
NFFTL	362	900	362	8.44	45.6
Chirp/NFFT^T	362	900	362	19.15	70.4
NFFT/NFFT^T	362	900	362	10.59	36.3

Table 1: Computation time of the filtered backprojection and of different linogram algorithms

Finally, note that we can detect differences in the quality of the reconstructed images much better if we are given colored images. For this we refer to

http : //*www.math.mu* − *luebeck.de*/*potts*/*radon*/*ima*.

References

[1] G. Beylkin. On the fast Fourier transform of functions with singularities. *Appl. Comput. Harmon. Anal.*, 2:363 − 381, 1995.

[2] A. J. W. Duijndam and A. M. Schonewille. Nonuniform fast Fourier transform. *Preprint*, 1997.

[3] R. N. Bracewell and A. C. Riddle. Strip integration in radio astronomy. *Australian J. Phys.*,9:198 − 217, 1956.

[4] A. Dutt and V. Rokhlin. Fast Fourier transforms for nonequispaced data. *SIAM J. Sci. Statist. Comput.*, 14:1368 − 1393, 1993.

[5] P. Edholm and G. Herman. Linograms in image reconstruction from projections. IEEE *Trans. Med. Imag.*, 6:301 − 307, 1987.

[6] B. Elbel. Mehrdimensionale Fouriertransformation für nichtäquidistante Daten. Master's thesis, Technical University Darmstadt, 1998.

[7] B. Elbel and G. Steidl. Fast Fourier transform for nonequispaced data. In C. K. Chui and L. L. Schumaker, editors, *Approximation Theory IX*, Vanderbilt University Press, Nashville, 1998.

[8] K. Fourmont. *Schnelle Fourier–Transformation bei nichtäquidistanten Gittern und tomographische Anwendungen*. PhD thesis, University of Münster, 1999.

[9] D. Gottlieb, B. Gustafsson and P. Forssen. On the direct Fourier method for computer tomography. *IEEE Trans. Med. Imag.*, 9:223 − 232, 2000.

[10] J. I. Jackson. Selection of a convolution function for Fourier inversion using gridding. IEEE *Trans. Med. Imag.*, 10:473 − 478, 1991.

[11] A. J. Jerry. The Shannon sampling theorem − its various extensions and applications: a tutorial review. *Proc. IEEE*, 1965.

[12] M. Kaveh and M. Soumekh. Computer–assisted diffraction tomography. In H. Stark, editor, *Image Recovery: Theory and Applications*, pages 369 − 413, Academic Press, Orlando, 1987.

[13] M. Kaveh, M. Soumekh, and J. Greenleaf. Signal processing for diffraction tomography. IEEE *Transactions on Sonics and Ultrasonics*, SU–31, 1984.

[14] F. Natterer. Fourier reconstruction in tomography. *Numer. Math.*, 47:343 − 353, 1985.

[15] F. Natterer and F. Wübbeling. Mathematical Methods in Image Reconstruction. 2000.

[16] J. O'Sullivan. A fast sinc function gridding algorithm for Fourier inversion in computer tomography. IEEE *Trans. Med. Imag.*, 4:200 − 207, 1985.

[17] J. Pelt. Fast computation of trigonometric sums with application to frequency analysis of astronomical data. *Preprint, University of Delft*, 1998.

[18] D. Potts, G. Steidl, and M. Tasche. Fast Fourier transforms for nonequispaced data: A tutorial. In J. J. Benedetto and P. Ferreira, editors, *Modern Sampling Theory: Mathematics and Application*, chapter 12, pages 253 – 274. Birkhäuser, 2000.

[19] D. Potts and G. Steidl. New Fourier reconstruction algorithms for computerized tomography. *Proc. SPIE Conf.*, San Diego, 2000 (in print).

[20] H. Schomberg and J. Timmer. The gridding method for image reconstruction by Fourier transformation. IEEE *Trans. Med. Imag.*, MI–14:596 – 607, 1995.

[21] J. Schulte. *Fourier–Rekonstruktionen in der Computer–Tomographie*. Master's thesis, University of Münster, 1994.
http://www.math.uni-muenster.de/inst/num/Abschlussarbeiten/Schulte/.

[22] R. A. Scramek and F. R. Schwab. Imaging. In F. R. S. R. Perley and A. Bridle, editors, *Astronomical Society of the Pacific Conference, Vol 6*, pages 117 – 138. 1988.

[23] G. Steidl. A note on fast Fourier transforms for nonequispaced grids. *Adv. Comp. Math*, 9:337 – 352, 1998.

[24] P. Toft. *The Radon Transform - Theory and Implementation*. PhD thesis, Technical University of Denmark, 1996. `http://www.sslug.dk/~pto`.

[25] J. Walden. Analysis of the direct Fourier method for computer tomography. *IEEE Trans. Med. Imag.*, 9:211 – 222, 2000.

[26] A. Ware. Fast approximate Fourier transforms for irregularly spaced data. SIAM *Review*, 40:838 – 859, 1998.