FAST ALGORITHMS AND EFFICIENT GPU IMPLEMENTATIONS FOR THE RADON TRANSFORM AND THE BACK-PROJECTION OPERATOR REPRESENTED AS CONVOLUTION OPERATORS

FREDRIK ANDERSSON, MARCUS CARLSSON, AND VIKTOR V. NIKITIN

AMS subject classifications. 65R32, 65R10

Key words. Radon transform, Fast algorithms, FFT, GPU.

Abstract. The Radon transform and its adjoint, the back-projection operator, can both be expressed as convolutions in log-polar coordinates. Hence, fast algorithms for the application of these operators can be constructed by using FFT, if data is resampled at log-polar coordinates. Radon data is typically measured on an equally spaced grid in polar coordinates, and reconstructions are represented (as images) in Cartesian coordinates. Therefore, in addition to FFT, several steps of interpolation have to be conducted in order to apply the Radon transform and the back-projection operator by means of convolutions. However, in comparison to the interpolation conducted in Fourier-based gridding methods, the interpolation performed in the Radon and image domains will typically deal with functions that are substantially less oscillatory. Reasonable reconstruction results can thus be expected using interpolation schemes of moderate order. It also provides better control over the artifacts that can appear due to measurement errors.

Both the interpolation and the FFT operations can be efficiently implemented on Graphical Processor Units (GPUs). For the interpolation, it is possible to make use of the fact that linear interpolation is hard-wired on GPUs, meaning that it has the same computational cost as direct memory access. Cubic order interpolation schemes can be constructed by combining linear interpolation steps and this provides important computation speedup.

We provide details about how the Radon transform and the back-projection can be implemented efficiently as convolution operators on GPUs. For large data sizes, these algorithms are several times faster than those of other software packages based on GPU implementations of the Radon transform and the back-projection operator. Moreover, the gain in computational speed is substantially higher when comparing against other CPU based algorithms.

1. Introduction. The two-dimensional Radon transform is the mapping of functions to their line integrals, i.e., a mapping $\mathcal{R} : \mathbb{R}^2 \to S^1 \times \mathbb{R}$ where $S^1$ denotes the unit circle, defined by

\begin{equation}
\mathcal{R}f(\theta, s) = \int f(x)\delta(x \cdot \theta - s)dx.
\end{equation}

The parameter $\theta$ represents the (normal) direction of the lines, and the parameter $s$ denotes the (signed) distance of the line to the origin. It is customary to use $\theta$ both as a point on the unit sphere and as an angle, i.e., the notation $x \cdot \theta$ is used to parameterize lines in $\mathbb{R}^2$ by the relation $x \cdot \theta = x_1 \cos(\theta) + x_2 \sin(\theta)$. Note that each line is defined twice in this definition, since $s$ can take both positive and negative, and since $\theta \in S^1$.

A schematic illustration of the Radon transform is depicted in Figure 1, where beams are propagating through an object and after absorption are measured by receivers. The Radon transform appears for instance in computational tomography (CT). The tomographic inversion problem lies in recovering an unknown function $f$ given knowledge of $\mathcal{R}f$. For more details about CT, see [18, 21, 26, 30].

One of the most popular methods of inverting the usual Radon transform is by means of the filtered back-projection (FBP) method [30]. It uses the inversion formula

\begin{equation}
f = \mathcal{R}^\# W\mathcal{R}f.
\end{equation}

*Center for Mathematical Sciences, Lund University. Box 118, 22100 Lund, Sweden. This work was supported by the Crafoord Foundation (20140633) and the Swedish Research Council (2011-5589).
where $W$ is a convolution operator acting only on the $s$-variable, and where $\mathcal{R}^\# : S^1 \times \mathbb{R} \to \mathbb{R}^2$ denotes the back-projection operator, which integrates over all lines through a point, i.e.,

$$\mathcal{R}^\# g(x) = \int_{S^1} g(\theta, x \cdot \theta).$$

The back-projection operator is adjoint to the Radon transform. The convolution operator $W$ can either be described as a Hilbert transform followed by a derivation, both with respect to the variable $s$; or as a convolution operator with a transfer function being a suitably scaled version of $|\sigma|$, where $\sigma$ denotes the conjugate variable of $s$.

A direct implementation of the filtered back-projection algorithm has a time complexity of $N^3$, if we assume that reconstructions are made on an $N \times N$ lattice and that the numbers of samples in $s$ and $\theta$ are both $O(N)$. This is because for each point $x$, integration has to be made over all lines ($N$ directions) passing through that point. However, there are fast methods for computing the back-projection. For instance, the method of fast hierarchical filtered back-projection (FHBP) introduced by Bresler et al. [7, 22] is based on a hierarchical decomposition of the Radon transform. In this way the total time can be reduced, but at a cost of decreased accuracy, cf. [7]. The technique proposed in [12, 13] is also based on a hierarchical scheme. A speedup is obtained in this case by viewing the back-projection as a line integration in the Radon domain, where longer line integrals are recursively formed from shorter ones.

Another class of fast methods for inversion of Radon data goes via the Fourier-slice theorem. These methods are based on the Fourier slice theorem which relates one-dimensional Fourier transform of the Radon data with a two-dimensional Fourier transform of the reconstruction image. The sampling of the Radon data gives rise to a non-standard two-dimensional sampling in the frequency domain that can not be computed directly by using standard FFT. To circumvent this problem, algorithms for unequally spaced FFT (USFFT, also known under the acronyms NFFT and NUFFT) can be used [8, 9, 20, 40]). In the standard parallel beam case, the Radon data can be mapped onto data on a polar grid. As an alternative, pseudo-polar sampling can
be used to construct fast Fourier reconstructions by making use of chirp-Z transforms, see for instance [3, 4, 5].

The data in the frequency domain will typically be oscillatory, as seen in Figure 2d) and (at in increased resolution) in the lower right panel of Figure 2c). Hence, in order to interpolate data of this type high interpolation order is required. In comparison, the data in the Radon domain will not be particularly oscillatory. This is illustrated in Figure 2b) and the upper left panel of Figure 2c). This fact implies that interpolation methods of moderate order can be expected to produce reasonable results. This means in turn that less time can be spent on conducting interpolation. It also gives more control over the interpolation errors, as local errors will be kept local in the Radon domain (and hence more easily distinguishable as artifacts in the reconstruction).

In this paper, we discuss how to design fast algorithms for the application of the Radon transform and the back-projection operator by using the fact that they can be expressed in terms of convolutions when represented in log-polar coordinates [2, 17, 25, 43]. In particular, we follow the approach suggested in [2]. This formulation turns out to be particularly well-suited for implementation on GPUs. A major advantage with using GPUs is that the routines for linear interpolation are fast. In fact, the cost for computing linear interpolation is the same as reading directly from memory [23]. This feature can be utilized for constructing fast interpolation schemes. In particular, in this paper we will work with cubic interpolation on GPU [35, 39].

For the sake of comparison, we will provide performance and accuracy tests of the proposed method along with comparisons against other software packages for tomographic computations. We also conduct a performance comparison between the different methods as utilized in iterative reconstruction techniques. The iterative methods rely on applying the Radon transform and the back-projection operator several times. An advantage of keeping all computations on the GPU is that the needed time for CPU-GPU memory transfer can be reduced.

2. The Radon transform and the back-projection expressed as convolutions. We recapitulate some of the main ideas of the method described in [2]. A key part there is the usage of log-polar coordinates, i.e.,

\[
\begin{align*}
    x_1 &= e^\rho \cos(\theta), \\
    x_2 &= e^\rho \sin(\theta),
\end{align*}
\]

where \(-\pi < \theta < \pi\). To simplify the presentation, we identify \(f(\theta, \rho)\) with \(f(x_1, x_2)\) if \((\theta, \rho)\) in the log-polar coordinate system corresponds to the point \(x = (x_1, x_2)\) of the Cartesian coordinate
system, and similarly for other coordinate transformations.

By representing the distance between lines and the origin $s = e^\rho$, and by a change of variables in (1.1) from Cartesian to log-polar coordinates the log-polar Radon transform can be expressed as

$$
R_{lp} f(\theta, \rho) = \int_{-\pi}^{\pi} \int_{-\infty}^{\infty} f(\theta', \rho') e^{\rho'} \delta \left( \cos(\theta - \theta') - e^{\rho - \rho'} \right) d\rho' d\theta',
$$

(2.1)

where $\zeta(\theta, \rho) = \delta(\cos(\theta) - e^\rho)$. In particular, $R f(\theta, s) = R_{lp} f(\theta, \log(s))$ for $s > 0$ (which is sufficient since the information for $s < 0$ is redundant).

We briefly mention how to put this formula in a theoretical framework. Set $S = (-\pi, \pi) \times \mathbb{R}$ and note that for a compactly supported smooth function $h$ on $S$, we have

$$
\int_{-\pi}^{\pi} \int_{-\infty}^{\infty} h(\theta, \rho) \zeta(\theta, \rho) d\rho d\theta = \int_{-\pi/2}^{\pi/2} h(\theta, \log(\cos(\theta))) \cos(\theta) d\theta,
$$

which can be written as $\int_S h d\mu$ where $\mu$ is an infinite measure on $S$. Hence, the formula extends by continuity to, e.g., all continuous compactly supported functions $h$ in $S$. It follows that (2.1) is well-defined whenever $f$ is a continuous compactly supported function which is zero in a neighborhood of 0.

As the Radon transform in the coordinate system $(\theta, \rho)$ is essentially a convolution between $f$ and the distribution $\zeta(\theta, \rho)$, it can be rapidly computed by means of Fourier transforms. Special care has to be taken to the distribution $\zeta$, an issue we will return to in what follows. Ignoring possible difficulties with the distribution $\zeta$, let us discuss how (2.1) can be realized by using fast Fourier transforms. It is natural to assume that the function $f$ has compact support (the object that is measured has to fit in the device that is measuring it). The compact support also implies that the Radon transform of $f$ will have compact support in the $s$-variable. However, this is not true in the log-polar setting, since $\rho \to -\infty$ as $s \to 0$.

Note also that for any point $x$ in the plane there is a direction for which there is a line passing through $x$ and the origin. This implies that it is not possible to approximate the values of the Radon transform by using a finite convolution in log-polar coordinates if it is to be computed for all possible line directions. However, by restricting the values of $\theta$, and by making a translation so that the support of $f$ is moved away from the origin, it is in fact possible to describe the partial Radon transform as a finite convolution, and then recover the full Radon transform by adding the contributions from various partial Radon transforms. The setup is illustrated in Figure 3.

Figure 3a) schematically shows the parameters used by the proposed method. Assume that the function we are considering has support in the unit circle. Let $\beta$ be a fixed angle and translate and scale $f$ according to Figure 3a). The new function is now supported in a circle with radius

$$
a_R = \frac{\sin \left( \frac{\theta}{2} \right)}{1 + \sin \left( \frac{\theta}{2} \right)}.
$$

Moreover, the tangent lines $L_1$ and $L_2$ are chosen to be perpendicular to the straight lines determining the (symmetric) angle span. Let $a_r$ be the shortest distance from the center $O_0$ (in the translated coordinates) to the tangent lines. From the geometry, it is readily verified that

$$
a_r = \frac{\cos \left( \frac{\theta}{2} \right) - \sin \left( \frac{\theta}{2} \right)}{1 + \sin \left( \frac{\theta}{2} \right)}. $$

4
Assume for the moment that $f$ has support in the gray circle indicated in Figure 3a). In log-polar coordinates $(\theta, \rho)$ it then has support inside of $[-\frac{\beta}{2}, \frac{\beta}{2}] \times [\log(1 - 2a_R), 0]$. If we restrict our attention to values of $R_{lp}(f)$ in the same sector $\theta \in [-\frac{\beta}{2}, \frac{\beta}{2}]$, then the only nonzero values of $R_{lp}(f)$ will be for $\rho$ in the interval $[\log a_r, 0]$. This means that $R_{lp}(f)(\theta, \rho)$ values can be computed by the finite convolution

\[
\int_{-\beta/2}^{\beta/2} \int_{\log(1-2a_R)}^{0} f(\theta', \rho') e^{\rho'} \zeta'(\theta - \theta', \rho - \rho') d\theta' d\rho',
\]

where $(\theta, \rho) \in \left[-\frac{\beta}{2}, \frac{\beta}{2}\right] \times [\log a_r, 0]$. We now replace the integral (2.2) by the periodic convolution

\[
R_{lp}^p f(\theta, \rho) = \int_{-\beta}^{\beta} \int_{\log(a_r)}^{0} f(\theta', \rho') e^{\rho'} \zeta_{\text{per}}(\theta - \theta', \rho - \rho') d\theta' d\rho',
\]

where $\zeta_{\text{per}}$ is the periodic extension of $\zeta$ defined on $[-\beta, \beta] \times [\log(a_r), 0]$. It is readily verified that for $(\theta, \rho) \in \left[-\frac{\beta}{2}, \frac{\beta}{2}\right] \times [\log a_r, 0]$, it thus holds that

\[R_{lp}^p f(\theta, \rho) = R f(\theta, e^\rho).\]

We refer to $R_{lp}^p$ as the partial log-polar Radon transform.

Note that, in analogy with the argument following (2.1), the formula (2.2) can be written as a convolution between $f(\theta', \rho') e^{\rho'}$ and a finite measure, whereas (2.3) can be written as a convolution with a locally finite periodic measure. The above formulas are thus well defined as long as $f$ is continuous (or even piecewise continuous) in the log-polar coordinates. We refer to [46, Chapter 11], for basic results about convolution between functions and periodic distributions.

The convolution setup for the Radon transform is depicted in Figure 4a). The rightmost black solid curve $C$ shows $\rho = \log(\cos(\theta))$, $(-\beta \leq \theta \leq \beta)$, which is the support of $\zeta$ in $[-\beta, \beta] \times [\log a_r, 0]$. 

Fig. 3: (a) Tangent lines to the circle to determine the support of the log-polar Radon transform function; (b) three angle spans to compute partial Radon transforms.
Let $D$ denote the circle,

$$D = \{(x_1, x_2) : (x_1 - 1 + a_R)^2 + x_2^2 < a_R^2\}.$$

The black dots show the perimeter of $D$ (in log-polar coordinates), and the gray curves indicate translation of the curve $C$ associated with the black dots on the circle, within the interval $[-\beta, \beta]$. There is a difference in grayscale for the points with $\theta$ inside the range $[-\frac{\beta}{2}, \frac{\beta}{2}]$, as only these values are of interest to us. Note that the smallest $\rho$-value of the contributing part in this interval is $\log(a_r)$. Moreover, the red lines indicate parts of the translations of $C$ outside $[-\beta, \beta] \times [\log(a_r), 0]$, and the blue lines show how these curves are wrapped back into the domain $[-\beta, \beta] \times [\log(a_r), 0]$ by the periodic extension of $\zeta$. We see that these effects do not have any influence on the domain $[-\frac{\beta}{2}, \frac{\beta}{2}] \times [\log(a_r), 0]$.

We now describe how $\mathcal{R}_p^\beta$ can be used to recover $\mathcal{R}f$ for a function $f$ with support in the unit circle. We split the angular variable into $M$ different parts, and choose

(2.4) \[ \beta = \frac{\pi}{M}. \]

For $m = 0, 1, \ldots, M - 1$ let $T_m : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ denote the change of coordinates

$$T_m(x) = a_R \begin{pmatrix} \cos(m\beta) & \sin(m\beta) \\ -\sin(m\beta) & \cos(m\beta) \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 1 - a_R \\ 0 \end{pmatrix},$$

and let $T_m f = f(T_m^{-1} x)$. Note that

(2.5) \[ \mathcal{R}f(\theta, s) = a_R^{-1} \mathcal{R}_p^\beta (T_m f) \left( \theta - m\beta, a_R s + (1 - a_R) \cos(\theta - m\beta) \right). \]

The Radon transform can thus be computed for arbitrary $\theta$ and $0 < s < 1$, by using the relation

(2.6) \[ \mathcal{R}f(\theta, s) = a_R^{-1} \mathcal{R}_p^\beta (T_m f) \left( \theta - m\beta, \log(a_R s + (1 - a_R) \cos(\theta - m\beta)) \right), \]

where $m = \lfloor \theta / \beta \rfloor$ and $[x]$ denotes the rounding operator to the closest integer to $x$, and where mod denotes the modulus operator.
We denote the change of coordinates above by

\[ S_m(\theta, s) = \left( \theta - m\beta, \log (a_R s + (1 - a_R) \cos(\theta - m\beta)) \right), \]

and we then have that

\[ R f(\theta, s) = a_R^{-1} \mathcal{R}_{lp} (T_m f) (S_m(\theta, s)). \]

We remark that for fixed \( \theta \), the connection (2.5) between the Radon data in the different domains has an affine dependence on \( s \). Since the filter operator \( \mathcal{W} \) acts as a convolution operator with regard to \( s \), its action will in principle be the same regardless if the coordinate transformation \( T_m \) is used or not. We use the notation \( \mathcal{W}_{lp} \) to denote the action of the filter operator in log-polar coordinates.

The adjoint operator (back-projection) associated with the Radon transform (1.1) can be written as

\[ \mathcal{R}^# g(x) = \int_{-\infty}^{\infty} \int_{S^1} g(\theta, s) \delta(x \cdot \theta - s) d\theta ds, \]

cf. [30]. It is a weighted integral of \( g \) over lines passing through the point \( x \), and just as for the Radon transform it can be expressed as a convolution in log-polar coordinates. We define

\[ \mathcal{R}^#_{lp} g(\theta, \rho) = \int_{-\infty}^{\infty} \int_{-\pi}^{\pi} g(\theta', \rho') \delta \left( e^{\rho - \rho'} \cos(\theta - \theta') - 1 \right) d\theta' d\rho' \]

\[ = \int_{-\infty}^{\infty} \int_{-\pi}^{\pi} g(\theta', \rho') \zeta^# (\rho - \rho', \theta - \theta') d\theta' d\rho'. \]

It is shown in [2] that \( \mathcal{R}^# f(x) = 2 \mathcal{R}^#_{lp} f(\theta, \rho) \), where the factor two comes from the fact that the corresponding integration in the polar representation \( (\theta, s) \) is only done in the half-plane \( s > 0 \). The log-polar back-projection operator has the same problem as the log-polar Radon transform in dealing with \( s = 0 \), and in a similar fashion we make use of partial back-projections in order to avoid this problem. Because of the relation (2.6) and the fact that the filter operator \( \mathcal{W} \) can be applied to each partial Radon data individually, it will be enough to consider partial back-projections for Radon data with \( \theta \in [-\frac{\beta}{2} + m\beta, \frac{\beta}{2} + m\beta] \) according to the setup of Figure 3. By applying \( T_m^{-1} \) to each of the partial back-projection, and summing up the results, we will recover the original function. For detailed calculations, we refer to [2].

The idea is thus to split the Radon data into \( M \) parts, where each part is transformed according to Figure 3a). For each part, the filtered data is back-projected according to

\[ \int_{-\infty}^{\infty} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} g(\theta', \rho') \zeta^# (\rho - \rho', \theta - \theta') d\theta' d\rho'. \]

Since we assumed that our original function had support inside the unit circle, we are only interested in the contributions inside the disc \( D \). Since only lines with \( \rho \in [\log(a_r), 0] \) go through this circle, the integration in the \( \rho \) variable above can be limited to \( \rho \in [\log(a_r), 0] \). Similarly as for the Radon transform, we now want to write this (finite) convolution as a periodic convolution. Figure 4b) illustrates how this can be achieved. The black solid lines show translations of the curve \( \rho = -\log(\cos(\theta)) \) representing the back-projection integral in the log-polar coordinates. The black dots now show the perimeter of a support of the Radon data, indicated by dark gray in the left illustration of Figure 4. Here, the dark gray curves of the back-projection illustration
show the translations of $\rho = -\log(\cos(\theta))$ that will give a contribution inside the disc $D$. The light gray curves illustrate contributions that fall outside the support of $D$. The red curves show contributions that will fall outside the range $[-\beta, \beta] \times [\log(a_r), 0]$, and the blue curves show the effect when these lines are wrapped back into the domain $[-\beta, \beta] \times [\log(a_r), 0]$. We note that the blue curves do not intersect the circle. Hence, we define the partial log-polar back-projection operator as

$$R_p^\# g(\theta, \rho) = \int_0^\beta \int_{-\frac{\beta}{2}}^{\frac{\beta}{2}} g(\theta', \rho') \zeta_{\text{per}}^\#(\rho - \rho', \theta - \theta') \, d\theta' \, d\rho'.$$

where $\zeta_{\text{per}}^\#$ is the periodic extension of $\zeta^\#$ defined on $[-\beta, \beta] \times [\log(a_r), 0]$, and note that for $(\theta, \rho)$ corresponding to points inside the domain $D$, it holds that

$$R_p^\# g(\theta, \rho) = R_p^\# g(\theta, \rho).$$

Formula (1.2) can now be recast as

$$2 \sum_{m=0}^{M-1} T_m R_p^\# W_p R_p^T_m f(x) = f(x)$$

for all $x$ in the unit disc.

3. Fast evaluation of the log-polar Radon transform and the log-polar back-projection. Let $h$ be a continuous function on some rectangle $R$ in $\mathbb{R}^2$ with area $A_R$, and let $\mu$ be a finite measure on $R$, and let $\mu_{\text{per}}$ be its periodic extension. Along the same lines as [46, Theorem 11.6-3], it is easy to see that the (normalized) Fourier coefficients of their periodic convolution satisfy

$$h * \mu_{\text{per}} = A_R \hat{h} \hat{\mu},$$

where $\hat{h} \hat{\mu}$ is the pointwise multiplication of the respective Fourier coefficients with respect to $R$. We will use this formula and FFT to fast evaluate $R_p^\# (2.3)$ and $R_p^\# (2.7)$. Note that the area of $[-\beta, \beta] \times [\log(a_r), 0]$ is $-2\beta \log a_r$. For $(\theta, \rho)$ in this rectangle we then have

$$f(\theta, \rho) = \sum_{k_\theta, k_\rho} \hat{f}_{k_\theta, k_\rho} e^{2\pi i \left( \frac{\alpha_{k_\theta}}{\log a_r} + \frac{\rho k_\rho}{a_r} \right)},$$

$$g(\theta, \rho) = \sum_{k_\theta, k_\rho} \hat{g}_{k_\theta, k_\rho} e^{2\pi i \left( \frac{\alpha_{k_\theta}}{\log a_r} + \frac{\rho k_\rho}{a_r} \right)},$$

$$\zeta(\theta, \rho) = \sum_{k_\theta, k_\rho} \hat{\zeta}_{k_\theta, k_\rho} e^{2\pi i \left( \frac{\alpha_{k_\theta}}{\log a_r} + \frac{\rho k_\rho}{a_r} \right)},$$

$$\zeta^\#(\theta, \rho) = \sum_{k_\theta, k_\rho} \hat{\zeta}_{k_\theta, k_\rho} e^{2\pi i \left( \frac{\alpha_{k_\theta}}{\log a_r} + \frac{\rho k_\rho}{a_r} \right)}.$$
The Fourier coefficients for the two distributions $\zeta$ and $\zeta^#$ are given by

$$
\hat{\zeta}_{k_\alpha,k_\beta} = \frac{1}{-2\beta \log a_r} \int_{-\beta}^{\beta} \delta(\cos(\theta) - e^{\rho}) e^{-2\pi i \frac{k_\alpha \theta}{\log(a_r)}} e^{-2\pi i \frac{k_\beta}{\log(a_r)}} d\rho d\theta =
$$

$$
= \frac{1}{-2\beta \log a_r} \int_{-\beta}^{\beta} \delta(\cos(\theta) - e^{\rho}) e^{-2\pi i \frac{k_\alpha \theta}{\log(a_r)}} e^{-2\pi i \frac{k_\beta}{\log(a_r)}} d\rho d\theta =
$$

$$
= \frac{1}{-2\beta \log a_r} \int_{-\beta}^{\beta} e^{-2\pi i \frac{k_\alpha \theta}{\log(a_r)}} (\cos(\theta) - 1) e^{-2\pi i \frac{k_\beta}{\log(a_r)}} d\rho d\theta =
$$

$$
= \frac{1}{-2\beta \log a_r} \int_{-\beta}^{\beta} e^{-2\pi i \frac{k_\alpha \theta}{\log(a_r)}} (\cos(\theta) - 1) e^{-2\pi i \frac{k_\beta}{\log(a_r)}} d\rho d\theta =
$$

$$
(3.3)
$$

$$
\hat{\zeta}^#_{k_\alpha,k_\beta} = \frac{1}{-2\beta \log a_r} \int_{-\beta}^{\beta} \delta(\cos(\theta) - e^{\rho}) e^{-2\pi i \frac{k_\alpha \theta}{\log(a_r)}} e^{-2\pi i \frac{k_\beta}{\log(a_r)}} d\rho d\theta =
$$

$$
= \frac{1}{-2\beta \log a_r} \int_{-\beta}^{\beta} \delta(\cos(\theta) - e^{\rho}) e^{-2\pi i \frac{k_\alpha \theta}{\log(a_r)}} e^{-2\pi i \frac{k_\beta}{\log(a_r)}} d\rho d\theta =
$$

$$
= \frac{1}{-2\beta \log a_r} \int_{-\beta}^{\beta} e^{-2\pi i \frac{k_\alpha \theta}{\log(a_r)}} (\cos(\theta) - 1) e^{-2\pi i \frac{k_\beta}{\log(a_r)}} d\rho d\theta =
$$

$$
= \frac{1}{-2\beta \log a_r} \int_{-\beta}^{\beta} e^{-2\pi i \frac{k_\alpha \theta}{\log(a_r)}} (\cos(\theta) - 1) e^{-2\pi i \frac{k_\beta}{\log(a_r)}} d\rho d\theta =
$$

$$
(3.4)
$$

Both the integrals on the right hand sides of (3.3) and (3.4) are of the form

$$
P(\mu, \alpha, \beta) = \int_{-\beta}^{\beta} e^{i\mu \theta} \cos(\theta)^\alpha d\theta,
$$

where $\mu = -2\pi k_\alpha/\beta$, $\alpha = -2\pi i \frac{k_\beta}{\log(a_r)} - 1$ for (3.3) and $\alpha = -2\pi i \frac{k_\beta}{\log(a_r)}$ for (3.4), respectively.

We briefly describe how to evaluate (3.5) numerically. The integral is well suited for evaluation by FFT, since for a fixed value of $k_\rho$ (fixed $\alpha$) we can obtain $\hat{\zeta}_{k_\alpha,k_\rho}$ and $\hat{\zeta}^#_{k_\alpha,k_\rho}$ for all integers $k_\rho$ in a given range by evaluating the integral of (3.5) by the trapezoidal rule by means of FFT. However, for this procedure to be accurate, we need to oversample the integral and make use of end-point corrections. The function $\cos(\theta)^\alpha$ will be oscillatory, but the oscillation is determined by the fixed parameter $\alpha$. Neglecting boundary effects, we can therefore expect the trapezoidal rule to be efficient, provided that sufficient oversampling is used.

For the boundary effects, we can make use of end-point correction schemes [1]. For the computations used in this paper, we have used an oversampling factor of 8 and an 8-order end-point correction with weights

$$
1 + \frac{1}{120960} \left[ -23681, 55688, -66109, 57024, -31523, 9976, -1375 \right].
$$

Suppose next that we only know values of $f$ and $g$ in (2.3) and (2.7), respectively, on an equally spaced sampling covering $[-\beta, \beta] \times [\log(a_r), 0]$, i.e., $f$ and $g$ are known on the lattice

$$
\left\{ 2\pi \frac{j_\theta}{N_\theta}, \log(a_r) \left( 1 - \frac{j_\theta}{N_\theta} \right) \right\}, \quad -\frac{N_\theta}{2M} \leq j_\theta < \frac{N_\theta}{2M}, \quad 0 \leq j_\rho < N_\rho,
$$

(recall that $\beta = \pi/M$ by (2.4)). We denote these values by $f_{j_\theta,j_\rho}$ and $g_{j_\theta,j_\rho}$, respectively. In order for (2.3) and (2.7) to be meaningful, we need to have continuous representations of $f$ and $g$. This is particularly important as $\zeta$ and $\zeta^#$ are distributions. A natural way to do this is, to define $f_{k_\alpha,k_\rho}$ and $g_{k_\alpha,k_\rho}$ by the discrete Fourier transform of $f_{j_\theta,j_\rho}$ and $g_{j_\theta,j_\rho}$, i.e., let

$$
\tilde{f}_{k_\alpha,k_\rho} = \begin{cases} 
\frac{M}{N_\rho N_\theta} \sum_{j_\theta=-\frac{N_\theta}{2M}}^{\frac{N_\theta}{2M}-1} \sum_{j_\rho=0}^{N_\rho-1} f_{j_\theta,j_\rho} e^{-2\pi i \left( \frac{jk_\alpha}{N_\theta} + \frac{j_\rho k_\beta}{N_\rho} \right)}, & \text{if } -\frac{N_\theta}{2M} \leq k_\theta < \frac{N_\theta}{2M}, \quad 0 \leq k_\rho < N_\rho, \\
0 & \text{otherwise},
\end{cases}
$$
and

\[ \hat{g}_{k_x,k_y} = \begin{cases} \frac{M}{N_p N_\theta} \sum_{j_\theta=-N_\theta/2}^{N_\theta/2-1} \sum_{j_\rho=0}^{N_\rho-1} g_{j_\theta,j_\rho} e^{-2\pi i \left( \frac{j_\theta k_x}{N_\theta} + \frac{j_\rho k_y}{N_\rho} \right)}, & \text{if } -N_\theta/2M \leq k_\theta < N_\theta/2M, \\ 0, & \text{otherwise.} \end{cases} \]

We can then use (3.1) and (3.2) to define continuous representations of \( f \) and \( g \). Using this approach, the values of (2.3) and (2.7) are also well defined, namely

\[ \mathcal{R}_p^f(\theta,\rho) = -2\beta \log \alpha_r \sum_{k_x,k_y} \hat{f}_{k_x,k_y} \hat{\xi}_{k_x,k_y} e^{2\pi i \left( \frac{k_x \rho}{N_\rho} + \frac{k_y \theta}{N_\theta} \right)}, \]

and

\[ \mathcal{R}_p^g(\theta,\rho) = -2\beta \log \alpha_r \sum_{k_x,k_y} \hat{g}_{k_x,k_y} \hat{\xi}_{k_x,k_y} e^{2\pi i \left( \frac{k_x \rho}{N_\rho} + \frac{k_y \theta}{N_\theta} \right)}. \]

In the case where the two transforms are to be evaluated at \((\theta,\rho)\) on the lattice (3.6), the corresponding sums above can be rapidly evaluated by using FFT.

4. Sampling rates. There are three different cases for which it is natural to use equally spaced discretization: For the representation of \( f \) in Cartesian coordinates \((x_1,x_2)\) covering the unit circle \( S^1 \); for the polar representation \((\theta,s)\) of the sinograms \( \mathcal{R}(f) \); and for the log-polar coordinates \((\theta,\rho)\) for evaluation of \( \mathcal{R}_p \) and \( \mathcal{R}_p^\# \). We will use a rectangular grid in all three coordinate systems and interpolate data between them as we work with the different domains.

In this section we derive guidelines for how to choose discretization parameters. These will be based on the assumption that \( \hat{f} \) is “essentially supported” in a disc with radius \( N/2 \), in the sense that contributions from the complement of this disc can be ignored without affecting the computational accuracy. Due to the uncertainty principle, \( \hat{f} \) cannot have its support included in this disc, since \( f \) itself is supported in a disc with radius 1/2, but it may work quite well in practice and is, therefore, still convenient to use for deriving sampling rates.\(^1\) We also base our arguments on refinements of the Nyquist sampling rate, more precisely the Paley-Wiener-Levinson theorem.

In the spatial variables \((x_1,x_2)\), the Nyquist sampling rate (corresponding to the assumption on the support of \( \hat{f} \)) is 1/\(N\). This leads us to cover the unit circle with a grid of size \( N \times N \). We use

\[ X = \left\{ \frac{j_1}{N}, \frac{j_2}{N} \right\}, \quad -N/2 \leq j_1, j_2 < N/2. \]

To derive recommendations for the sampling in the polar coordinates \((\theta,s)\) (for the entire \( \mathcal{R} f \)), recall the Fourier slice theorem, which states that

\[
\int_{-\infty}^{\infty} \mathcal{R} f(s,\theta) e^{-2\pi i s \sigma} \, ds = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(s \theta + t \hat{\theta}) e^{-2\pi i \sigma \theta} \, ds \, dt = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x) e^{-2\pi i \sigma x} \, ds \, dt = \hat{f}(\sigma \theta).
\]

\(^1\)More strict results can be obtained by introducing notions such as numerical support. Recall that any compactly supported \( L^1 \) function has compact numerical support in both the spatial and the frequency domain. The presentation will quickly become substantially more technical and we therefore follow the signal processing practice and use Nyquist sample rates despite not having infinitely long samples.
Fig. 5: (a) Log-polar grid. Samples in the $\rho$ variable are chosen in order not to lose the accuracy of measuring data. (b) The bandwidth of a partial back-projection function.

Thus $Rf(\theta, s) = F^{-1}_{\sigma \rightarrow s}(\hat{f}(\sigma \theta))$, and hence the Nyquist sampling rate for $s$ is $\triangle s = \frac{1}{N}$, yielding that $N_s = N$. Let $\Delta \theta_p$ denote the angular sampling rate in the polar coordinate system. Since $f$ is supported in the unit disc, the Nyquist sampling rate in the frequency domain $\xi$ is 1 (on a rectangular infinite grid). However, the multidimensional version of the Paley-Wiener-Levinson theorem roughly says that it is sufficient to consider an irregular set of sample points whose maximal internal distance between neighbors is 1. Since we have assumed that the values of $\hat{f}(\xi)$ for $|\xi| > \frac{N}{2}$ are negligible, this leads us to choose $\Delta \theta_p$ and $\Delta \sigma$ (the latter will not be used) so that the polar grid points inside this circle has a maximum distance of 1. It follows that

$$\Delta \theta_p = \frac{2}{N}.$$ 

Since we want to cover an angle span of $[0, \pi]$, this leads to $N_\theta \approx \frac{\pi}{2}N$. We denote the polar grid by

$$\Sigma = \{ j_\theta \Delta \theta_p, j_s \Delta s \}, \quad 0 \leq j_\theta < N_\theta, \quad -\frac{N}{2} \leq j_s < \frac{N}{2}. \tag{4.2}$$

Practical tomographic measurements can therefore typically have a ratio $\frac{N_s}{N_\theta} \approx 1.5$ between the sampling rates in the $\theta$- and $s$-variables. We refer to [26, 30] for more details, and proceed to discuss the sampling in the log-polar coordinates.

To distinguish between the coordinate sampling parameters we use $\Delta \theta_{lp}$ for angular sampling rate in the log-polar coordinates. Recall that we wish to accurately represent functions of the form $T_m f$, which is a rotation, dilation (with factor $2a_R$) and translation of $f$. The support of $T_m f$ lies in the gray circle of Figure 3 and its essential frequency support is then inside the disc of radius $\frac{N}{4\pi a_R}$.

Figure 5 illustrates the setup where the black lines indicate equally spaced samples of $\theta_{lp}$; the blue curves indicate equally spaced sampling in $s_{lp}$; and where the red dashed curves indicate
equally spaced sampling in $\rho$. Note that the maximum distance between points in the $x_1$ direction occurs when $s = 1$ or $\rho = 0$. It is thus clear that $\Delta s_{lp}$ should equal $\Delta x_1 = \frac{2a_R}{N}$, whereas $\Delta \rho$ is determined by

$$
\max_{\rho \in [\log(a_r), 0]} e^\rho - e^{\rho - \Delta \rho} = \max_{\rho \in [\log(a_r), 0]} e^\rho (1 - e^{-\Delta \rho}) \leq \Delta x_1.
$$

As the largest distance occurs when $\rho = 0$ it follows that

$$
\Delta \rho \leq - \log \left( 1 - \frac{2a_R}{N} \right),
$$

and consequently, since the total distance that is to be covered is $- \log(a_r)$, that

$$
N_{\rho} \geq \left\lceil \frac{\log(a_r)}{\log(1 - \frac{2a_R}{N})} \right\rceil.
$$

(4.3)

where the notation $\lceil x \rceil$ denotes the nearest integer greater than or equal to $x$.

For the determination of sample rate in the angular variable for the representation of $T_m f$ in the disc $D$, we have that

$$
\Delta \theta_{lp} \approx \sin(\Delta \theta_{lp}) \leq \frac{2a_R}{N},
$$

(see Figure 5). Hence, we introduce

$$
\Omega_{lp} = \{j\theta \Delta \theta_{lp}, j\rho \Delta \rho\}, \quad -\frac{\beta}{2 \Delta \theta_{lp}} \leq j\theta < \frac{\beta}{2 \Delta \theta_{lp}}, \quad -N_{\rho} \leq j\rho \leq 0,
$$

(4.4)

for the representation of $T_m f$ in $D$. The representation of the partial Radon transform of $T_m f$ needs a reduced sample rate compared to (4.4), since the original Radon transform is sampled at $\Delta \theta_{lp}$. As the partial Radon transform is applied as a convolution on a log-polar grid (i.e. by a Fourier multiplier), the higher frequencies in the $\theta$-direction will not be needed. Hence, we can apply a low-pass filter in the $\theta$-direction, and apply the FFT operation the grid

$$
\Omega_p = \{j\theta \Delta \theta_p, j\rho \Delta \rho\}, \quad -\frac{N_{\theta}}{2M} \leq j\theta < \frac{N_{\theta}}{2M}, \quad -N_{\rho} \leq j\rho \leq 0,
$$

(4.5)

for the computation of the partial Radon transform of $T_m f$. The grid $\Omega_p$ will also be useful when resampling Radon data in log-polar coordinates.

5. Interpolation. As mentioned in the previous section it will be natural to use equally spaced sampling in the Cartesian, polar and log-polar coordinate systems, respectively. We typically want to reconstruct data on an equally spaced Cartesian grid; the tomographic data is sampled in equally spaced polar coordinates; and both the Radon transform and the backprojection can be rapidly evaluated by FFT when sampled on an equally spaced log-polar grid. There are several ways to interpolate between these coordinate systems. In this work we will make use of cubic (cardinal) B-spline interpolation. We will discuss how to incorporate some of the interpolation steps in the FFT operations, and also discuss how the B-spline interpolation can be efficiently implemented on GPUs.

The cubic cardinal B-spline is defined as

$$
B(x) = \begin{cases} 
(3|x|^3 - 6|x|^2 + 4)/6, & 0 \leq |x| < 1, \\
(-|x|^3 + 6|x|^2 - 12x + 8)/6, & 1 \leq |x| < 2, \\
0, & |x| \geq 2.
\end{cases}
$$
This function is designed so that it is non-negative, piecewise smooth, and \( C^2 \) at \(|x| = 0, 1, 2\). Note that at integer points, it holds that

\[
B(j) = \begin{cases} 
\frac{2}{3} & \text{if } j = 0, \\
\frac{1}{6} & \text{if } |j| = 1, \\
0 & \text{otherwise}.
\end{cases}
\]  

(5.1)

When used as a filter, it will smooth out information and is thus acting as a low-pass filter. Consequently, it can not be used directly for interpolation.

The Fourier series given by the coefficients in (5.1), which we denote by \( \hat{B} \), is given by

\[
\hat{B}(\xi) = \sum_j B(j) e^{-2\pi ij \xi} = \frac{1}{6} \left( e^{2\pi i x \xi} + 4 + e^{-2\pi i x \xi} \right) = \frac{2}{3} + \frac{1}{3} \cos(2\pi \xi).
\]  

(5.2)

Suppose that equally spaced samples \( f_k \) of a function \( f \) (in one variable) are available. We want to recover values of \( f \) at arbitrary points \( x \) using

\[
f(x) = \sum_k (Qf)_k B \left( x - \frac{k}{N} \right),
\]  

(5.3)

where the operator \( Q \) is a pre-filter operation, that is compensating for the fact that convolution with \( B \) suppresses high frequencies. Since \( B \) has short support, only values of \( (Qf)_k \) for \( k \approx xN \) will contribute in this sum. The pre-filter operation, which is boosting high frequencies in the samples \( f_k \), can be computed in different ways. Perhaps the most direct way is to define \( Q \) in the Fourier domain (by the discrete Fourier transform), where it essentially becomes division by \( \hat{B} \) (upon scaling and sampling). In this case, it is easy to see that the convolution with \( B \) and the pre-filter operation will cancel each other at points \( x = jN \), i.e., the original function is recovered at the sample points; which is a requisite for any interpolation scheme.

As we will compute the Radon transform and the back-projection by means of FFT in log-polar coordinates, we can in some steps incorporate the pre-filter step \( 1/\hat{B} \) in the Fourier domain, at virtually no additional cost. However, not all of the pre-filter operations can be incorporated in this way. While the pre-filter easily can be applied by separate FFT operations, we want to limit the total number of FFT operations, as these will be the most time-consuming part in the implementations we propose.

As an alternative to applying the pre-filter in the Fourier domain, it can be applied by recursive filters. In [36] these operations are derived by using the Z-transform. It turns out that if we define an auxiliary sequence \( (g_k) \) by

\[
g_k = 6f_1(\sqrt{3} - 2)g_{k-1},
\]  

then

\[
(Qf)_k = (\sqrt{3} - 2)((Qf)_{k+1} - g_k).
\]  

(5.4)

cf. [36, equations (12,13)], where the boundary condition on \( (Qf)_k \) and \( (g_k) \) are given in [36, equations (14,15)]. These filters can be efficiently implemented on GPUs.

Two-dimensional prefiltration can be done in two steps, one in each dimension. We will use the notation \( Qf \) for the pre-filtering also in this case. On a GPU this implies doing operations on rows and columns separately. However, there are highly optimized routines for transposing data, which means that the prefiltration can be made to act only on column data. This is done to
improve the so-called memory coalescing and GPU cache performance [44]. Memory coalescing refers to combining multiple memory accesses into a single transaction. In this way, the GPU threads run simultaneously and substantially increased cache hit ratios are obtained.

Let us now turn our focus to the convolution step in (5.3). Let
\[ k = [N x], \]
\[ \alpha = x - [x]. \]

The sum (5.3) then reduces to
\[ f(x) = w_0(\alpha)(Qf)_{k-1} + w_1(\alpha)(Qf)_k + w_2(\alpha)(Qf)_{k+1} + w_3(\alpha)(Qf)_{k+2}, \]
where
\[ w_0(\alpha) = B(\alpha - 2), \quad w_1(\alpha) = B(\alpha - 1), \quad w_2(\alpha) = B(\alpha), \quad w_3(\alpha) = B(\alpha + 1). \]

We now discuss how this sum can be evaluated using linear interpolators. As mentioned previously, linear interpolation is executed fast on GPUs. In [39] it is shown how this can be utilized to conduct efficient cubic interpolation. The cubic interpolation is expressed as two weighted linear interpolations, instead of four weighted nearest neighbor look-ups, yielding \( 2^d \) operations instead of \( 4^d \) for conducting cubic interpolation in \( d \) dimensions.

We briefly recapitulate the approach taken in [35, 39]. Given coefficients \((Qf)_k\), let \( Q_{f\text{lin}} \) be the linear interpolator
\[ Q_{f\text{lin}}(x) = (1 - (\alpha))(Qf)_k + \alpha(Qf)_{k+1} \]
\[ = (1 - (x - [x])(Qf)_{[N x]} + (x - [x])(Qf)_{[N x]+1}. \]

The sum (5.5) can then be written as
\[ f(x) = (w_0(\alpha) + w_1(\alpha))Q_{f\text{lin}} \left( k - 1 + \frac{w_1(\alpha)}{w_0(\alpha) + w_1(\alpha)} \right) \]
\[ + (w_2(\alpha) + w_3(\alpha))Q_{f\text{lin}} \left( k + 1 + \frac{w_3(\alpha)}{w_2(\alpha) + w_3(\alpha)} \right). \]

The evaluation of the function \( Q_{f\text{lin}}(x) \) can be performed by hard-wired linear interpolation on GPU. In modern GPU architecture, the so-called texture memory (cached on a chip) provide effective bandwidth by reducing memory requests to the off-chip DRAM. The two most useful features of this kind of memory with regards to conducting B-spline interpolation are:

1. The texture cache is optimized for the 2D spatial locality, giving the best performance to GPU threads that read texture addresses that are close together.
2. Linear interpolation of neighboring values can be performed directly in the GPUs texture hardware, meaning that the cost for computing the interpolation is the same as reading data directly from memory.

This implies that the cost for memory access in (5.6) will be two instead of four as only two function calls of \( Q_{f\text{lin}} \) are made in (5.6), and in two dimensions the corresponding reduction from 16 memory access operations to 4 will give significant improvement in computational speed.

6. Algorithms. We now have the necessary ingredients to present detailed descriptions on how to rapidly evaluate the Radon transform and the back-projection operator by FFT in log-polar coordinates. In the algorithms below, we let \( \hat{B}_{\theta,\rho} \) denote the values of the two-dimensional counterpart of (5.2), scaled to represent the sampling on \((\theta, \rho) \in \left[ -\frac{\beta}{2}, \frac{\beta}{2} \right] \times [\log a_r, 0]. \)

A few remarks on time complexity. The most time-consuming part of both algorithms are the convolutions that are implemented by FFT. In total, \( 2M \) FFT operations need to be computed...
Algorithm 1 Fast Radon transform

1: Given \( f \) sampled at \( X \) (4.1) compute \( Qf \) by (5.4)
2: for \( m = 0, \ldots, M-1 \) do
3: Resample \( T_m f \) at \( \Omega_p \) (4.4) by (5.3)
4: Downsample from \( \Omega_p \) (4.4) to \( \Omega_p \) (4.5)
5: Multiply result by \( e^{i\theta} \)
6: Apply the log-polar Radon transform with pre-filtering incorporated, i.e., compute \( \hat{f}_{k_\theta,k_\rho} \) from (3.7), and evaluate
\[
\sum_{k_\theta,k_\rho} \hat{f}_{k_\theta,k_\rho} \frac{\hat{Q}_{k_\theta,k_\rho}}{B_{k_\theta,k_\rho}} e^{2\pi i \left( j_\theta k_\theta \frac{\rho_\theta}{\Delta \theta} + j_\rho k_\rho \frac{\rho_\rho}{\Delta \rho} \right)}
\]
by using FFT.
7: Resample from \( S_m^{-1} \Omega_p \) (4.5) to \( \Sigma \) (4.2) by (5.3)
8: end for

Algorithm 2 Fast back-projection

1: Given \( g \) sampled at \( \Sigma \) (4.2) compute \( Qg \) by (5.4)
2: for \( m = 0, \ldots, M-1 \) do
3: Resample \( g(S_m) \) at \( \Omega_p \) (4.5) by (5.3)
4: Apply the log-polar back-projection with pre-filtering incorporated, i.e., compute \( \hat{g}_{k_\theta,k_\rho} \) from (3.8) and evaluate
\[
\sum_{k_\theta,k_\rho} \hat{g}_{k_\theta,k_\rho} \frac{\hat{Q}_{k_\theta,k_\rho}}{B_{k_\theta,k_\rho}} e^{2\pi i \left( j_\theta k_\theta \frac{\rho_\theta}{\Delta \theta} + j_\rho k_\rho \frac{\rho_\rho}{\Delta \rho} \right)}
\]
by using FFT.
5: Resample from \( T_m^{-1} \Omega_p \) (4.5) to \( X \) (4.1) by (5.3) to obtain partially back-projected data
6: end for
7: Sum up the \( M \) partial back-projections

(including forward and backward FFT’s), and each operation will be done on a grid of size \( \frac{2N_\theta}{M} \times N_\rho \). For \( M = 3 \), it follows from (4.3) that
\[
N_\rho \approx -\frac{N \log(a_r)}{2a_R} \approx 2.1N,
\]
and \( N_\rho \) is monotonically decreasing for increasing \( M \), (since \( a_r \) and \( a_R \) depend on \( \beta \) (recall Figure 3) which equals \( 2\pi/M \) by (2.4)). For \( M = 3 \) we thus need roughly twice as many samples in \( \rho \) compared to that originally used for the sampling of \( s \) (\( N_s = N \)), and in the angular variable we also need to sample with twice as many parameters in order to avoid aliasing effects. In total, we need to use an oversampling of about 4 in the FFT operations. Note that this is the same oversampling that is generally needed for computing the convolution of two functions if aliasing is to be avoided. The total time spent on FFT is thus \( M \) times the time it takes to do an FFT of size \( \frac{2N_\theta}{M} \times N_\rho \) which is comparable to the time it takes to do one FFT operation of size \( 2N_\theta N_\rho \). The FFT cost for applying the Radon transform and the back-projection operator is thus about the same as would be expected for a generic convolution. In addition to the FFT time, interpolation
also has to be conducted. For the back-projection case, the obtained result after computing a partial back-projection needs to be interpolated onto a $N \times N$ lattice, yielding an interpolation complexity of $O(MN^2)$. A similar result holds for the (forward) Radon transform. This gives a total computational cost that has $O(N^2 \log N + MN^2)$ cost. In practice the parameter $M$ should be chosen small and such that the data sizes that FFT is applied to are powers of two. In our simulations we use $M = 3$ which fits well for the typical setup where $N$ is a power of two and $N_\theta = 3N/2$ directions are used.

7. Numerical experiments. Let us start by briefly discussing the filters used in the filtered back-projection (1.2). This discussion is included in order to better interpret the errors obtained when comparing different methods. For theoretically perfect reconstruction (with infinitely dense sampling), the filter $W$ in (1.2) is given by

\begin{equation}
\hat{w}_{\text{ramp}}(\sigma) = |\sigma|.
\end{equation}

This filter is sometimes referred to as the ramp filter. If the sampling rate is insufficient in relation to the frequency content of $f$ (or the object upon which the measurements is conducted on), it can be desirable to suppress the highest frequencies in order to localize the effects of the insufficient sampling. There is a relation between one-dimensional filtering in the $s$-direction of Radon data, and a two-dimensional convolution in the spatial domain, see the computation following (4.1) or [30, p.102]. It can be explained by the Fourier slice theorem, which describes how Radon data can be converted to a polar sampling in the frequency domain by taking one-dimensional Fourier transforms in the $s$-direction. The application of the one-dimensional filter will then correspond to a change in amplitude (and possible phase) along the lines (indicated by dots with the same angles) in Figure 5b). As there is no information outside the circle with radius $N/2$, the action of the ramp filter will be equivalent of applying a two-dimensional convolution to the original function $f$ using a two-dimensional filter with Fourier transform

\begin{equation}
\hat{W}_{\text{ramp}}(\xi) = \begin{cases} 
1 & \text{if } |\xi| \leq \frac{N}{2}, \\
0 & \text{otherwise}.
\end{cases}
\end{equation}

We see that if $f$ contains more high frequent information than the one prescribed by the sampling rate $N$, then the sharp cutoff in (7.2) can yield artifacts, and in the presence of noise in the Radon sampling the high-frequency boosting of (7.1) will boost the noise. Replacing the ramp filter (7.1) with a filter that goes smoothly to zero at the highest (discrete) frequencies, will thus yield an image that is slightly smoother, but on the other hand, an image with suppressed high-frequency
noise and artifacts due to incomplete sampling. Sometimes, the ramp filter is modified so that it does not reach zero but only reduces the high-frequency amplitudes. Two common choices of filters are the cosine and the Shepp-Logan filters, defined by

\[
\hat{w}_{\cos}(\sigma) = \begin{cases} 
|\sigma| \cos \left( \frac{\pi |\sigma|}{N} \right) & \text{if } |\sigma| \leq \frac{N}{2}, \\
0 & \text{otherwise,}
\end{cases}
\]

\[
\hat{w}_{\text{SL}}(\sigma) = \begin{cases} 
|\sigma| \text{sinc} \left( \frac{\sigma}{N} \right) & \text{if } |\sigma| \leq \frac{N}{2}, \\
0 & \text{otherwise.}
\end{cases}
\]

The cutoff is made above the sampling bandwidth, as it will illustrate the practical effect that the filters have on measured data. The three filters are illustrated in the frequency domain in Figure 6. The two-dimensional filters associated with these filters have Fourier representations

\[
\hat{W}_{\cos}(\xi) = \begin{cases} 
\cos \left( \frac{\pi |\xi|}{N} \right) & \text{if } |\xi| \leq \frac{N}{2}, \\\n0 & \text{otherwise,}
\end{cases}
\]

and

\[
\hat{W}_{\text{SL}}(\xi) = \begin{cases} 
\text{sinc} \left( \frac{\xi}{N} \right) & \text{if } |\xi| \leq \frac{N}{2}, \\\n0 & \text{otherwise,}
\end{cases}
\]

respectively. As the \( \hat{w}_{\cos} \) goes to zero at the highest sampling rate, we can expect smaller errors when using this filter compared to the others. On the other hand, the highest frequencies are suppressed, and the results reconstructions will look slightly less sharp.

To illustrate the accuracy of the suggested implementation, we conduct some examples on the Shepp-Logan [37] phantom. We use the modified version introduced in [41, Appendix B.2]. The function used \((f)\) is illustrated in the left panel of Figure 2. The phantom consists of linear combinations of characteristic functions of ellipses, and its support is inscribed in the unit circle. Since the Radon transform is a linear operator, \(Rf\) is a linear combination of Radon transforms of characteristic functions of ellipses. Since the Radon transform of the characteristic function of a circle can be computed analytically, analytic expressions are available for the Radon transform of the Shepp-Logan phantom by applying transform properties of shifting, scaling, and rotation. This Radon transform is depicted in the right panel of figure 2.

The presence of a high-frequency discontinuity caused by the filter can cause artifacts, but also the discontinuity of the derivative at \( \sigma = 0 \). To avoid artifacts from this part, we apply end-point trapezoidal corrections. The effect of this correction can be seen around \( \sigma = 0 \) in Figure 6. Our aim is to eliminate as much of the errors as possible to distinguish the errors that the resampling between the different coordinate systems used in the proposed methods introduces.

In Figure 7 we show some reconstruction results from the filtered back-projection using the log-polar-based method. For these tests we use exact expressions for Shepp-Logan phantom and its Radon data, when applying the ramp, Shepp-Logan and cosine filters. The tests are performed for Radon data of sizes \((N_{\vartheta} \times N_{\varphi}) = (\frac{3}{2}N \times N)\) where \((N \times N)\) is the size of the reconstructed image. For the log-polar reconstruction, we used \(M = 3\) partial reconstructions, and \(N_{\varphi} = 2N\). The reconstructed panels have inscribed normalized \(\ell^2\)-errors against the filtered versions of the phantom. The tests are performed for \(N = 512\).

For the sake of quality comparison, we use the results of the back-projection from the following packages:
Fig. 7: Computational errors of the filtered back-projection for the log-polar-based method, with the ramp, Shepp-Logan and cosine filters.

- MATLAB Image Processing Toolbox
- ASTRA Tomography Toolbox
- NiftyRec Tomography Toolbox
- SPIRAL back-projection package
- Image Reconstruction Toolbox (IRT)
- NFFT3

In the reconstruction results that follow, we use the Shepp-Logan filter for accuracy comparisons. The different packages have different interfaces and different sets of parameters, and therefore some code modifications have to be done for making suitable comparisons. We have tried to keep the code modifications at a minimum and measure time only for the computation of the back-projection. The times for the pre-computational steps (initialization of grids, computing basis functions, etc.) are excluded from the total time as these can be computed off-line. All the packages contain examples on how to perform back-projection or reconstruction of different types.

**MATLAB.** The MATLAB functions *radon* and *iradon* are used as reference methods for comparing reconstruction accuracy of the back-projection performed with linear-, cubic- and spline-based interpolation. As suggested during the review, the computational times for the MATLAB reconstructions are not included in the speed comparison tables.

**ASTRA.** The ASTRA Toolbox [31, 42] is a MATLAB toolbox of GPU primitives for 2D and 3D tomography. In the simulations presented in this section, ASTRA v1.6 is used, and results are based on the routine *s014_FBP*. The iterative tests in the next section are based on the script *s003_gpu_reconstruction* where the parameter *EM_CUDA* is used for reconstruction.

**NiftyRec.** The NiftyRec Toolbox [32, 33] comes with both GPU and CPU implementations of Radon transforms. The results presented here uses NiftyRec v2.0 and make use of the back-projection part of the demo *tt_demo_mlem_parallel* for parallel beam transmission tomography with zero attenuation.

**SPIRAL.** The SPIRAL back-projection package implements the fast hierarchical back-projection method (FHBP) [7, 34]. The code was compiled using the Intel compiler (icc version 15.0.3) with the flag *-fast*. Two parameters are used to control the hierarchical scales: the parameter *B* is the base case size, at which the program stops recursion; and the parameter *D* which is the size of the image at which the program starts (angularly) to down-sample the filtered sinograms. Varying these parameters one can control accuracy and computational time. Following the setup in [34, Figure 3], the simulation results presented here use *B = N/4, D = N* (hierarchical approach), and *B = N, D = N* (which reduces the algorithm to the classic filter back-projection algorithm).
IRT. The Image Reconstruction Toolbox (IRT)\cite{19} includes a parallel implementation of the filtered back-projection (using OpenMP), as well as the Fourier based approach. In this implementation the USFFT algorithms use Kaiser-Bessel bell functions with an additional min-max criterion, cf. [20]. The simulations conducted are based on the examples fbp/fbp2_example and systems/tests/Gtomo_nufft_test. The USFFT tests are based on the MATLAB script systems/tests/Gtomo_nufft_test.

NFFT3. The C library NFFT 3.0 \cite{27} provides algorithms for unequally spaced fast Fourier transforms. The library also contains routines for computing Radon transforms either using the Fourier slice theorem or using the pseudo-polar Fourier transform. For our tests, we adapt the MATLAB script radon/radon to use only one back-projection step (it originally uses an iterative method).

For the pseudo-polar grids, it is assumed that the data is sampled in a non-standard fashion, i.e., not equally spaced in the $\theta$ and $s$-variables. This is an effect of the fact that the directions of the lines, as well as the corresponding spacing in the radial variable, have to be adapted so that they agree with the pseudo-polar frequency lattice. As the pseudo-polar grids are not orientation invariant, the Shepp-Logan filter should be slightly modified to include angular variations. The effect seems negligible, and we use the standard filter in what follows. The simulations make use of the fact that there are exact formulas for the Radon data of the Shepp-Logan phantom in order produce samples that agree with the pseudo-polar format, and we, therefore, sample the data at these locations. In practice, an interpolation step would be needed here.

For the tests a standard desktop computer with an Intel Core i7-3820 processor and an NVIDIA GeForce GTX 770 graphic card is used. The packages have been tested with MATLAB 2015b via MEX interfaces or via execution of binary files. All packages are tested in single precision. Figure 8 displays accuracy results. The reconstructions as before were made on a $512 \times 512$ grid using 768 samples in the angular direction.

The top three panels of Figure 8 show the Fourier based methods, using high order interpolation in the frequency domain. The next four panels (the second row and the left panel in the third row) show results using third order interpolation in the Radon domain, and the following four panels show the results using linear interpolation in the Radon domain. The last (bottom right) panel shows the hierarchical reconstruction.

Table 1 contains computation times for the different methods. Since the methods include both algorithmic and hardware accelerated methods, we also include times for a CPU version of the proposed method. This version makes use of the Intel libraries MKL (for FFT) and IPP (for interpolation), which contain highly optimized standard routines. The proposed algorithm had to be modified slightly in order to use the interpolation methods available for the IPP function remap. Specifically, the cubic interpolation method is used (giving a result comparable to that of cubic filtered back-projection).

To exclude initialization effects and assure high GPU load, the times in Table 1 are obtained by executing batches of reconstructions (to fill up the GPU memory) in one function call. Table 2 presents corresponding slowdown coefficients compared to the log-polar implementation. We note that the proposed program on GPU is about 2-6 times faster than ASTRA, substantially faster than the GPU version of NiftyRec, and about 2 or more magnitudes faster than the other CPU based methods.

8. Performance analysis of the log-polar implementation. For the GPU implementation of the proposed method, version 7.5 of the CUDA Toolkit \cite{16} is being used. The computations were performed in single precision.

Table 3 shows a breakdown of the computational times for the log-polar Radon transform. This table shows that most computational time is due to CPU-GPU data transfers. The transfers can be performed in parallel with computations in the case of processing big 3D data cubes: while
Fig. 8: Computational errors of the filtered back-projection for the Shepp-Logan filter.
one portion of data is computed, another portion can be transferred to/from GPU memory. CUDA Streams can be used to overlap data transfers and computations. In this way, operations in different streams can be interleaved and, when possible, they can run concurrently. This will reduce the impact of the CPU-GPU copy operations on the total execution time.

9. Iterative methods for tomographic reconstruction. In some situations it is preferable to use an iterative method for doing reconstruction from tomographic data. This could, for instance, be because of missing data, e.g., that data for some angles are missing; suppression of artifacts [29]; or that additional information about the noise contamination can be used to improve the reconstruction results compared to direction filtered back-projections (1.2). Iterative reconstruction methods rely on applying the forward and back-projection operators several times. Iterative algorithms can be computationally expensive when a large number of iterations is required for the algorithm to converge. For that reason, fast algorithms for computing the Radon transform and the associated back-projection can play an important role.

Iterative algebraic reconstruction techniques (ART) are popular tools for reconstruction from incomplete measurements [24]. It aims at solving the set of linear equations determined by the

Table 1: Computational time (in seconds) of the back-projection for sizes \((N_\theta \times N_s) = (\frac{3}{2}N \times N)\) (excluding initialization time).

<table>
<thead>
<tr>
<th>Package</th>
<th>Method</th>
<th>PU</th>
<th>128</th>
<th>256</th>
<th>512</th>
<th>1024</th>
<th>2048</th>
</tr>
</thead>
<tbody>
<tr>
<td>LP</td>
<td>log-polar</td>
<td>GPU</td>
<td>1.8e-04</td>
<td>7.4e-04</td>
<td>2.9e-03</td>
<td>1.2e-02</td>
<td>4.5e-02</td>
</tr>
<tr>
<td>Astra</td>
<td>FBP</td>
<td>GPU</td>
<td>3.3e-04</td>
<td>1.4e-03</td>
<td>7.1e-03</td>
<td>4.3e-02</td>
<td>2.8e-01</td>
</tr>
<tr>
<td>LP</td>
<td>log-polar</td>
<td>CPU</td>
<td>2.5e-03</td>
<td>5.8e-03</td>
<td>2.6e-02</td>
<td>1.1e-01</td>
<td>4.0e-01</td>
</tr>
<tr>
<td>NFFT3</td>
<td>Fourier</td>
<td>CPU</td>
<td>6.1e-03</td>
<td>3.6e-02</td>
<td>1.7e-01</td>
<td>6.9e-01</td>
<td>3.2e+00</td>
</tr>
<tr>
<td>NFFT3</td>
<td>Fourier(pseudo-polar)</td>
<td>CPU</td>
<td>1.3e-02</td>
<td>5.6e-02</td>
<td>2.2e-01</td>
<td>9.2e-01</td>
<td>4.6e+00</td>
</tr>
<tr>
<td>IRT</td>
<td>Fourier</td>
<td>CPU</td>
<td>3.3e-02</td>
<td>8.7e-02</td>
<td>3.7e-01</td>
<td>1.5e+00</td>
<td>6.3e+00</td>
</tr>
<tr>
<td>IRT</td>
<td>FBP</td>
<td>CPU</td>
<td>1.1e-02</td>
<td>4.1e-02</td>
<td>3.9e-01</td>
<td>1.7e+00</td>
<td>1.2e+01</td>
</tr>
<tr>
<td>NiftyRec</td>
<td>FBP</td>
<td>CPU</td>
<td>3.7e-03</td>
<td>2.9e-02</td>
<td>2.4e-01</td>
<td>1.7e+00</td>
<td>1.4e+01</td>
</tr>
<tr>
<td>SPIRAL</td>
<td>Hierarchical</td>
<td>CPU</td>
<td>6.7e-02</td>
<td>3.8e-01</td>
<td>2.5e+00</td>
<td>2.1e+01</td>
<td>1.9e+02</td>
</tr>
<tr>
<td>SPIRAL</td>
<td>FBP</td>
<td>CPU</td>
<td>8.0e-02</td>
<td>3.3e-01</td>
<td>5.2e+00</td>
<td>4.9e+01</td>
<td>3.9e+02</td>
</tr>
</tbody>
</table>

Table 2: Slowdown compared to the log-polar implementation on GPU for the back-projection for sizes \((N_\theta \times N_s) = (\frac{3}{2}N \times N)\) (excluding initialization time).

<table>
<thead>
<tr>
<th>Package</th>
<th>Method</th>
<th>PU</th>
<th>128</th>
<th>256</th>
<th>512</th>
<th>1024</th>
<th>2048</th>
</tr>
</thead>
<tbody>
<tr>
<td>LP</td>
<td>log-polar</td>
<td>GPU</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Astra</td>
<td>FBP</td>
<td>GPU</td>
<td>1.8</td>
<td>1.9</td>
<td>2.4</td>
<td>3.6</td>
<td>6.2</td>
</tr>
<tr>
<td>LP</td>
<td>log-polar</td>
<td>CPU</td>
<td>13.7</td>
<td>7.9</td>
<td>8.7</td>
<td>9.5</td>
<td>8.9</td>
</tr>
<tr>
<td>NFFT3</td>
<td>Fourier</td>
<td>CPU</td>
<td>33.1</td>
<td>49.4</td>
<td>56.9</td>
<td>57.5</td>
<td>70.1</td>
</tr>
<tr>
<td>NFFT3</td>
<td>Fourier(pseudo-polar)</td>
<td>CPU</td>
<td>70.6</td>
<td>75.6</td>
<td>75.7</td>
<td>77.4</td>
<td>100.8</td>
</tr>
<tr>
<td>IRT</td>
<td>Fourier</td>
<td>CPU</td>
<td>179.2</td>
<td>118.5</td>
<td>123.9</td>
<td>128.1</td>
<td>139.8</td>
</tr>
<tr>
<td>IRT</td>
<td>FBP</td>
<td>CPU</td>
<td>60.1</td>
<td>55.8</td>
<td>132.3</td>
<td>140.9</td>
<td>270.2</td>
</tr>
<tr>
<td>NiftyRec</td>
<td>FBP</td>
<td>GPU</td>
<td>20.4</td>
<td>39.3</td>
<td>81.2</td>
<td>143.4</td>
<td>305.5</td>
</tr>
<tr>
<td>SPIRAL</td>
<td>Hierarchical</td>
<td>CPU</td>
<td>364.5</td>
<td>511.2</td>
<td>839.9</td>
<td>1763.3</td>
<td>4082.4</td>
</tr>
<tr>
<td>SPIRAL</td>
<td>FBP</td>
<td>CPU</td>
<td>437.4</td>
<td>441.1</td>
<td>1760.1</td>
<td>4151.5</td>
<td>8648.1</td>
</tr>
</tbody>
</table>
Table 3: Time (in seconds) of main computational parts for the log-polar Radon transform.

<table>
<thead>
<tr>
<th>N</th>
<th>Prefilter</th>
<th>Interp</th>
<th>FFT</th>
<th>other'</th>
<th>copyCPU-GPU</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>1.8e-05</td>
<td>3.2e-05</td>
<td>3.0e-05</td>
<td>3.9e-05</td>
<td>5.4e-05</td>
<td>1.7e-04</td>
</tr>
<tr>
<td>256</td>
<td>5.1e-05</td>
<td>1.3e-04</td>
<td>1.7e-04</td>
<td>1.6e-04</td>
<td>2.1e-04</td>
<td>7.1e-04</td>
</tr>
<tr>
<td>512</td>
<td>1.8e-04</td>
<td>4.8e-04</td>
<td>7.2e-04</td>
<td>6.3e-04</td>
<td>8.3e-04</td>
<td>2.8e-03</td>
</tr>
<tr>
<td>1024</td>
<td>8.5e-04</td>
<td>1.9e-03</td>
<td>2.9e-03</td>
<td>2.5e-03</td>
<td>3.3e-03</td>
<td>1.1e-02</td>
</tr>
<tr>
<td>2048</td>
<td>4.9e-03</td>
<td>7.6e-03</td>
<td>1.2e-02</td>
<td>5.8e-03</td>
<td>1.4e-02</td>
<td>4.4e-02</td>
</tr>
</tbody>
</table>

' - small GPU kernels, GPU-GPU copy.

projection data. Transmission-based tomographic measurements measure the absorption of a media along a line. This puts a sign constraint on the function we wish to recover. It is clearly not ideal to assume the data is contaminated by normally distributed noise (for which the best estimate is given by the least squares estimates). A more reasonable assumption is that the added noise has a Poisson distribution [6, 45]. The simplest iterative method for solving the estimation/reconstruction problem under this noise assumption is by the Expectation-Maximization (EM) algorithm. The EM-algorithm is well-suited to reconstruct tomography data with non-Gaussian noise [15, 28, 38]. Alternative techniques include, for instance, the Row Action Maximum Likelihood Algorithm (RAMLA) [10, 14].

More details about the EM-algorithm can be found for instance in [11, 45]. In our notation, it can (given tomographic data $g$) be expressed as the iterative computations of

$$f^{k+1} = f^k \frac{R^\# \left( \frac{g}{R^k f} \right)}{R^\# \chi_C},$$

where the function $\chi_C(\theta, s)$ is one if the line parameterized by $(\theta, s)$ passes through the unit disc (the support of $f$) and zero otherwise. In each step, a Radon transform $R f^k$ and a back-projection $R^\# \left( \frac{g}{R^k f} \right)$ is computed.

It is common in GPU computing that the host-device memory transfers constitute an essential part of the total computational cost. For iterative methods (such as the one described above) it is possible to keep all the necessary data in the GPU memory, and thereby limiting the data copy between host and device memory to an initial guess $f^0$; the measured data $g$; and the final result. As most methods, the proposed method require some initialization steps (e.g., geometry parameters and convolution kernels $\zeta, \zeta^\#$).

Figure 9a) shows the Radon data with Poisson noise, and Figure 9b) shows the corresponding filtered back-projection. Note that the noise level is rather higher in this reconstruction. In contrast, the de-noised recovered data after 50 iterations of the EM-algorithm is presented in Figure 9c). We present performance results for the EM-algorithm for the proposed method and the methods implemented on GPU from ASTRA and NiftyRec tomography toolboxes. The computational times for conducting 100 iterations of the EM-algorithm for different resolution parameters are presented in Table 4. The Radon data was sampled with parameters $N_\theta = \frac{3}{2}N$, $N_s = N$, and $M = 3$ partial back-projection was used.

10. Conclusions. We have described how to implement the Radon transform and the back-projection as convolution operators in log-polar coordinates efficiently on GPUs. We present sampling conditions; provide formulas and numerical guidelines for how to compute the kernels associated with the Radon transform and the back-projection operator; and discuss how the convolutions can be rapidly evaluated by using FFT. The procedure involves several steps of
Fig. 9: Reconstruction of Radon data with noise. Radon data with Poisson noise is shown in the left panel, the result of applying filtered back-projection is shown in the middle panel, the right panel shows the result after 50 iterations of the EM-algorithm.

Table 4: Time (in seconds) for 100 iterations of the EM-algorithm for reconstruction 3d data of sizes \((N \times N \times N)\).

<table>
<thead>
<tr>
<th>Package</th>
<th>Method</th>
<th>Package</th>
<th>Method</th>
<th>Package</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LP</td>
<td>log-polar GPU</td>
<td>128</td>
<td>2.6e+01</td>
<td>256</td>
<td>2.1e+02</td>
</tr>
<tr>
<td>ASTRA</td>
<td>FBP GPU</td>
<td>2.2e+01</td>
<td>6.8e+01</td>
<td>4.2e+02</td>
<td>5.3e+03</td>
</tr>
<tr>
<td>NiftyRec</td>
<td>FBP GPU</td>
<td>8.5e+01</td>
<td>1.3e+03</td>
<td>2.1e+04</td>
<td>3.0e+05</td>
</tr>
</tbody>
</table>

interpolation between data in the Radon domain; the spatial domain; and the log-polar domain. It is comparatively favorable to conduct interpolation in these domains compared to conducting interpolation in the Fourier domain, as the functions will tend to be less oscillatory there. We use cubic spline interpolation which can be efficiently implemented on GPUs, and optimized routines for FFT on the GPU. We conduct numerical tests and conclude that reasonable accuracy can be obtained at favorable computational cost.

REFERENCES


