Direct Fourier Tomographic Reconstruction Image-to-Image Filter

Release 1.0

Dominique Zosso, Meritxell Bach Cuadra, Jean-Philippe Thiran

August 24, 2007

Ecole Polytechnique Fédérale de Lausanne (EPFL), Signal Processing Institute LTS5
Bâtiment ELD Station 11, CH-1015 Lausanne (Switzerland)
{dominique.zosso, meri.bach, jp.thiran}@epfl.ch

Abstract

We present an open-source ITK implementation of a direct Fourier method for tomographic reconstruction, applicable to parallel-beam x-ray images. Direct Fourier reconstruction makes use of the central-slice theorem to build a polar 2D Fourier space from the 1D transformed projections of the scanned object, that is resampled into a Cartesian grid. Inverse 2D Fourier transform eventually yields the reconstructed image. Additionally, we provide a complex wrapper to the `BSplineInterpolateImageFunction` to overcome ITK's current lack for image interpolators dealing with complex data types. A sample application is presented and extensively illustrated on the Shepp-Logan head phantom. We show that appropriate input zeropadding and 2D-DFT oversampling rates together with radial cubic b-spline interpolation improve 2D-DFT interpolation quality and are efficient remedies to reduce reconstruction artifacts.

Contents

1 Introduction

Direct Fourier reconstruction (DFR) is one of several reconstruction methods applied in parallel-beam computed tomography[6, 1]. An overview of different, transform-based – in contrast to algebraic or iterative – reconstruction methods can be found in [2].

1.1 Parallel-beam Computed tomography

In parallel-beam x-ray computed tomography (first generation CT), x-ray source and detector acquire parallel projections, then rotate circularly around the scanned object to the next frame. Most often the acquisition is performed slice by slice along the rotational axis (1D source and detector), several slices can however be scanned simultaneously by using a 2D source and detector.

The acquired data p are a discretely sampled cylindrical (in-slice) Radon transform of the scanned volume f, as illustrated in figure 1:

$$p(r,\theta,z) = \iint f(x,y,z)\delta(x\cos(\theta)+y\sin(\theta)-r)dxdy \tag{1}$$

and accordingly have the following 3 dimensions:

1. The angle of projection θ
2. The distance from the center r
3. The axial position z

From the Radon projections the image of the scanned volume can be reconstructed using different techniques.

1.2 Direct Fourier Reconstruction

Direct Fourier Reconstruction uses 3 major steps to reconstruct the image of the scanned object (per axial slice):

1. 1D-FFT of the projection to build a polar 2D Fourier space using the central-slice theorem.
2. Polar to Cartesian resampling.
3. Inverse 2D-FFT to obtain the reconstructed slice.

The central-slice theorem states that the Fourier transform in r of a Radon projection at a given angle is equal to the axial slice at the same angle of the Fourier transform of the original volume:

$$P_{\theta,z}(k_r) = F_z(k_r\cos(\theta), k_r\sin(\theta)) \tag{2}$$

where $P_{\theta,z}(k_r)$ is the 1D Fourier transform in r of the projection acquired at angle θ and axial position z, and where $F_z(u,v)$ is the in-plane 2D Fourier transform of the volume at axial position z. This relation is illustrated in figure 1.

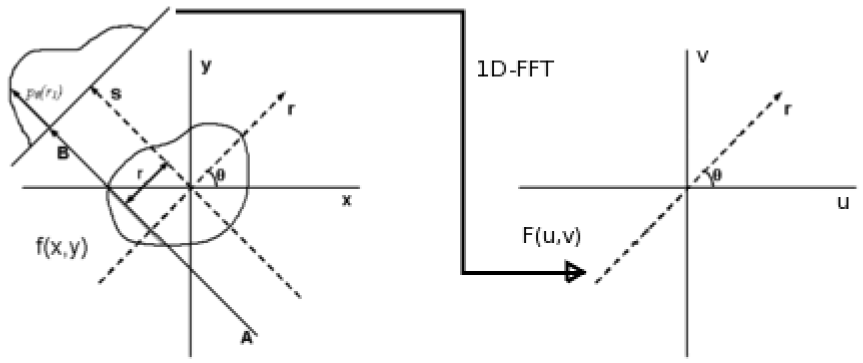

Figure 1: **Left:** *Parallel-beam x-ray tomography*. For each slice z, the projections $p(r,\theta,z)$ are made up by the line integrals from source to detector (A $\rightarrow$ B) through the object. **Right:** *Central-slice theorem*. The 1D Fourier transform $P_\theta(k_r)$ of a parallel projection $p(r,\theta)$ of an image $f(x,y)$ taken at an angle θ gives a slice of the 2D Fourier transform, $F(u,v)$, subtending the same angle θ with the u-axis.

The most striking advantage of DFR is its $O(N^2 \log N)$ complexity imposed by the inverse 2D-FFT in step 3, whereas the more widely used filtered backprojection (FBP) algorithm usually performs in $O(N^3)$[1]. The major drawback of DFR lies in step 2: inappropriate interpolation during the Cartesian resampling may cause important image artifacts[3]. However, the same source mentions some remedies to overcome these artifacts: Zeropadding of the input data and oversampling of the 2D Cartesian Fourier space.

The following sections report on the exact algorithm and its parameters, the contributed class implementations, and an example with some synthetic test results.

2 Algorithm

A formal description of our direct Fourier reconstruction method is given in algorithm 1. An illustration of the different symbols can be found in figure 2. In general the computations are index-based, rather than in physical space, with array indices beginning at 0. As first step, the number of effectively required input samples along the angular dimension is determined in line 1. The relative oversize Δ of the last angular segment (between the last angular sample and 180°) is determined (2). The dimensions of the zeropadded projection line and 1D-DFT, as well as the oversampled 2D-DFT are computed (3), and the radial lowpass cutoff frequency gives a maximal polar radius beyond which 2D Fourier samples are set to zero (4). For each input line, data is shifted to the line ends around the origin (8) and modulated to shift the DC to the center of the Fourier line (9). Once all projections of a slice have been transformed into Fourier space, the 2D-DFT is resampled. The oversampling rate n_g shortens the effective polar radius (19), while the angle is determined using `atan2`[2] (21). The angle is then converted into a floating point index (26), used to linearly interpolate between the two adjacent angular samples. Prior radial interpolation is obtained using b-splines of order n_b (28). Note that, if the upper angular index overflows, the modified interval size Δ applies, the angle is trimmed and the radius changes sign (30). After the inverse 2D-FFT, the requested region is collected from the edges of the image (36) and demodulated to undo the FFT-shift (37).

[1]Modified FBP versions exist, offering $O(N^2 \log N)$ as well

[2]`cmath` (`math.h`): `atan2(y,x)` returns the principal arc tangent of y/x, in the interval $[-\pi,+\pi]$ radians. To compute the value, the function uses the sign of both arguments to determine the quadrant.

Algorithm 1 Direct Fourier Reconstruction

Require: CT data as $p(r,\theta,z)$. a, b and c are the size of the input data (along z, θ and r, respectively). α is the angular range in degrees covered by the image series. The input projections are zero-padded n_z-times, and the 2D DFT is n_g-fold oversampled. $f_c \in [0,1]$ is the relative radial cutoff frequency. x_0, y_0 and m,n are the requested output offset and size. B-spline interpolation is of order n_b.

Ensure: Reconstructed image $f(x,y,z)$

```
 1: $b' \Leftarrow \lfloor 180 \cdot b/\alpha \rfloor$                              Cover just less than 180°
 2: $\Delta \Leftarrow 1 + 180 \cdot b/\alpha - b'$                                  Size of last angular interval
 3: $c' \Leftarrow c \cdot n_z, \quad c'' \Leftarrow c' \cdot n_g$                   Zeropad and oversample
 4: $r_{max} \Leftarrow f_c \cdot c'/2 - 1$                                          Radial cutoff frequency
 5: for $z = 0$ to $a-1$ do
 6:   for $\theta = 0$ to $b'-1$ do
 7:     for $r = 0$ to $c-1$ do
 8:       $r' \Leftarrow (r + c' - c/2) \mod c'$                                     Shift data to line ends (origin)
 9:       if $r' \mod 2 = 0$ then                                                   Modulate to shift DC to DFT center
10:         $p'(r') \Leftarrow p(r,\theta,z)$
11:       else
12:         $p'(r') \Leftarrow -p(r,\theta,z)$
13:       end if
14:     end for
15:     $P_\theta \Leftarrow FFT(p')$                                                Compute 1D-FFT
16:   end for
17:   for $u,v = 0$ to $c''-1$ do                                                   Re-sample Cartesian 2D-DFT
18:     $u' \Leftarrow u - c''/2, \quad v' \Leftarrow v - c''/2$                     Set origin to image center
19:     $r'' \Leftarrow \sqrt{u'^2 + v'^2}/n_g$                                      Oversample
20:     if $r'' < r_{max}$ then                                                     Lowpass filtering
21:       $\Theta \Leftarrow \tan_2^{-1}(v',u')$                                     Get angle $[-\pi,\pi]$
22:       if $\Theta < 0$ then                                                      Flip second half
23:         $\Theta \Leftarrow \Theta + \pi$
24:         $r'' \Leftarrow -r''$
25:       end if
26:       $\theta \Leftarrow b \cdot \Theta/\pi$                                     Convert angle to (float) index
27:       if $\lfloor\theta\rfloor + 1 < b'$ then                                   Radially b-spline and angular linear interpolation
28:         $F(u,v) \Leftarrow (1-(\theta-\lfloor\theta\rfloor))\beta^{n_b}[P_{\lfloor\theta\rfloor}(r''+c'/2)] + (\theta-\lfloor\theta\rfloor)\beta^{n_b}[P_{\lfloor\theta\rfloor+1}(r''+c'/2)]$
29:       else                                                                      $\theta$-overflow: enlarged angular skip ($\Delta$)
30:         $F(u,v) \Leftarrow (1-\Delta^{-1}(\theta-\lfloor\theta\rfloor))\beta^{n_b}[P_{\lfloor\theta\rfloor}(r''+c'/2)] + \Delta^{-1}(\theta-\lfloor\theta\rfloor)\beta^{n_b}[P_{\lfloor\theta\rfloor+1-b}(c'/2-r'')]$
31:       end if
32:     end if
33:   end for
34:   $f' \Leftarrow FFT^{-1}(F)$                                                    Inverse 2D-FFT
35:   for $x = x_0$ to $x_0 + (m-1)$, $y = y_0$ to $y_0 + (n-1)$ do                 Copy requested region
36:     $x' = x + c(n_z \cdot n_g - 1/2) \mod c'', \quad y' = y + c(n_z \cdot n_g - 1/2) \mod c''$      Get corners
37:     if $(x'+y') \mod 2 = 0$ then                                                Demodulate (reverse FFT-shift)
38:       $f(x,y,z) = f'(x',y')$
39:     else
40:       $f(x,y,z) = -f'(x',y')$
41:     end if
42:   end for
43: end for
```

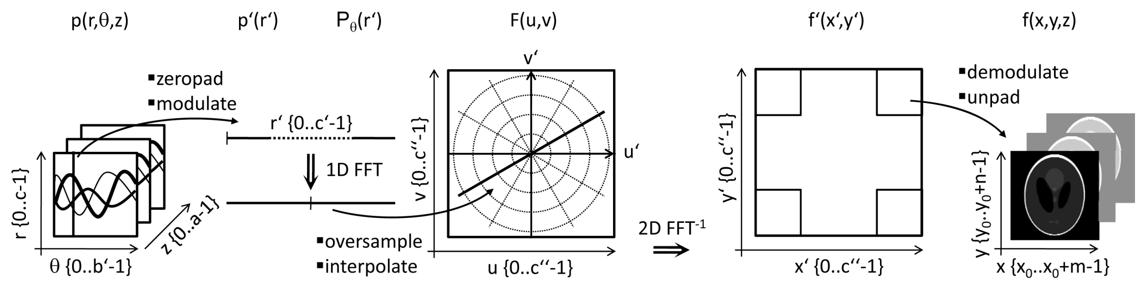

Figure 2: *Direct Fourier reconstruction algorithm overview.* For each slice, each radial line of the sinogram is zeropadded, shifted and modulated before transforming it in 1D Fourier domain. Modulation shifts the DC to the Nyquist frequency, i.e. the line center in Fourier domain. The Cartesian 2D Fourier space is resampled from the polar data. After inverse 2D Fourier transformation, the image is unpadded and demodulated, and the requested region is copied into the corresponding output slice.

3 Implementation

The presented algorithm has been implemented as the image-to-image filter class `itk::DirectFourierReconstructionImageFilter`. As currently no image interpolator exists in ITK supporting complex images, the wrapper class `itk::ComplexBSplineInterpolateImageFunction` is equally contributed thus providing b-spline interpolation of complex images. Both classes are briefly described in the following sections.

3.1 `itk::DirectFourierReconstructionImageFilter`

The proposed `itk::DirectFourierReconstructionImageFilter` class is derived from `itk::ImageToImageFilter` and is templated over the input and output pixel type. **The dimension of input and output images is fixed to 3**. The resulting input and output image types are provided as public traits `typedef Image< TInputPixelType, 3 > InputImageType` and `typedef Image< TOutputPixelType, 3 > OutputImageType`, respectively.

Class dependencies further include:

- `itk::VnlFFTRealToComplexConjugateImageFilter` and
- `itk::VnlFFTComplexConjugateToRealImageFilter`,
- `itk::ImageRegionIteratorWithIndex` and
- `itk::ImageSliceConstIteratorWithIndex`,
- `math.h`, as well as
- the provided `itk::ComplexBSplineInterpolateImageFunction`.

Several `Get`/`Set` methods allow configuring the reconstruction parameters:

- `RDirection`: image direction along *r* (projection lines)
- `ZDirection`: axial image direction (slices)

- `AlphaDirection`: image direction along angles[3]
- `AlphaRange`: angular range covered by the device in degrees (at least 180) α
- `ZeroPadding`: projection zeropadding rate n_z
- `Oversample`: 2D DFT oversampling rate n_g
- `Cutoff`: radial cutoff frequency f_c
- `RadialSplineOrder`: radial b-spline order n_b

The class is currently not multithread-enabled[4] and overrides the following 3 image-to-image filter methods:

- `void GenerateOutputInformation()`,
- `void GenerateInputRequestedRegion()` and
- `void GenerateData()`.

While the first two methods allow propagating region information along the pipeline (provided output regions and necessary input data, respectively), the `GenerateData()` contains the actual algorithm implementation.

The reader may refer to the source code and comments for further implementation details.

3.2 `itk::ComplexBSplineInterpolateImageFunction`

As can be seen in the examples below, an appropriate interpolation scheme in Fourier domain is crucial for good reconstruction quality. As currently the ITK interpolate image functions are unable to deal with complex images, we provide a complex wrapper around `itk::BSplineInterpolateImageFunction`. The `itk::ComplexBSplineInterpolateImageFunction` splits a complex input image into its real and imaginary parts and interpolates them using the existing scalar b-spline interpolator. It derives from `itk::InterpolateImageFunction` and is templated over `TImageType`, `TCoordRep` and `TCoefficientType`.

It instantiates a `itk::ComplexToRealImageFilter` and a `itk::ComplexToImaginaryImageFilter`, as well as two respective b-spline-interpolators as member objects. Upon construction, these member objects are initialized. `itk::BSplineInterpolateImageFunction` requires the spline order to be set prior to input image connection. In consequence the same restrictions apply to the wrapper as well. `SetSplineOrder` propagates the spline order to the underlying real-type interpolators, while `SetInputImage` connects the input image to the complex decomposition filters, and their output to the respective interpolators. Internally, the interpolators will at this point update their interpolation coefficients.

The overridden `EvaluateAtContinuousIndex` method becomes then particularly simple, as it only returns the merged results of two underlying calls to the scalar member interpolators:

[3] In the code, we consistently use `alpha` instead of `theta` to denote the angular index, while `theta` denotes the actual angle θ.

[4] Reconstruction of the different slices is embarrassingly parallel, and multi-threading would be easy to implement if regions were split along the z-axis only.

```
return typename ComplexBSplineInterpolateImageFunction<
    TImageType, TCoordRep, TCoefficientType >::OutputType(
      m_RealInterpolator->EvaluateAtContinuousIndex( x ),
      m_ImaginaryInterpolator->EvaluateAtContinuousIndex( x )
    );
```

4 Example

An example that shows how to use the `itk::DirectFourierReconstructionImageFilter` class is provided in `DirectFourierReconstruct.cxx`. The following section explains the incorporation of the filter in a small example application. We then illustrate its performance on a sample image.

4.1 Direct Fourier reconstruction example application

First, include some standard libraries used for image IO:

```
#include "itkImage.h"
#include "itkImageFileReader.h"
#include "itkImageFileWriter.h"
```

Then include the `DirectFourierReconstructionImageToImageFilter` headers:

```
#include "itkDirectFourierReconstructionImageToImageFilter.h"
```

We define the pixel type. For the sake of precision, it should be real rather than integer:

```
typedef double        PixelType;
```

As the image dimension is fixed to 3, the reconstruction filter is only templated over the input and output pixel types:

```
typedef itk::DirectFourierReconstructionImageToImageFilter<
                          PixelType, PixelType >     ReconstructionFilterType;
```

We then derive the corresponding input and output image types:

```
typedef ReconstructionFilterType::InputImageType     InputImageType;
typedef ReconstructionFilterType::OutputImageType    OutputImageType;
```

and define the IO types:

```
typedef itk::ImageFileReader< InputImageType >       ReaderType;
typedef itk::ImageFileWriter< OutputImageType >      WriterType;
```

The sinogram image is read:

```
ReaderType::Pointer reader = ReaderType::New();
reader->SetFileName( "sinogram.hdr" );
```

Now let's setup the reconstruction filter

```
ReconstructionFilterType::Pointer reconstruct = ReconstructionFilterType::New();
```

and connect the sinogram as its input

```
reconstruct->SetInput( reader->GetOutput() );
```

The image directions are set to the corresponding values:

```
reconstruct->SetRDirection( r_direction );          // r
reconstruct->SetZDirection( z_direction );          // slices
reconstruct->SetAlphaDirection( alpha_direction );  // angle
```

Zeropadding the input projections and oversampling the 2D DFT are two means of reducing aliasing artifacts in the reconstructed images (setting these factors, as well as the radial input size to powers of 2 allows for an efficient FFT calculation).

```
reconstruct->SetZeroPadding( 2 );
reconstruct->SetOverSampling( 2 );
```

Setting the radial cutoff frequency to values below 1.0 would allow low-pass filtering the reconstructed images (however, mind Gibb's phenomena):

```
reconstruct->SetCutoff( 1.0 );
```

While the angular interpolation is strictly linear, the degree of the radial b-splines interpolation can be set by the user. Degree 0 corresponds to a nearest neighbor interpolation, degree 1 equals linear interpolation. While values up to 5 are currently accepted, degree 3 (cubic b-spline) is most often an appropriate choice:

```
reconstruct->SetRadialSplineOrder( 3 );
```

Finally, the angular range covered by the projections has to be specified (at least 180 degree, depending on the imaging device):

```
reconstruct->SetAlphaRange( 180 );
```

After connecting the pipeline to the image writer, the pipeline update can finally be invoked:

```
WriterType::Pointer writer = WriterType::New();
writer->SetFileName( "reconstructed.hdr" );
writer->SetInput( reconstruct->GetOutput() );

try
{
    writer->Update();
}
catch ( itk::ExceptionObject err )
{
    std::cerr << "An error occurred somewhere:" << std::endl;
    std::cerr << err << std::endl;
    return 1;
}
```

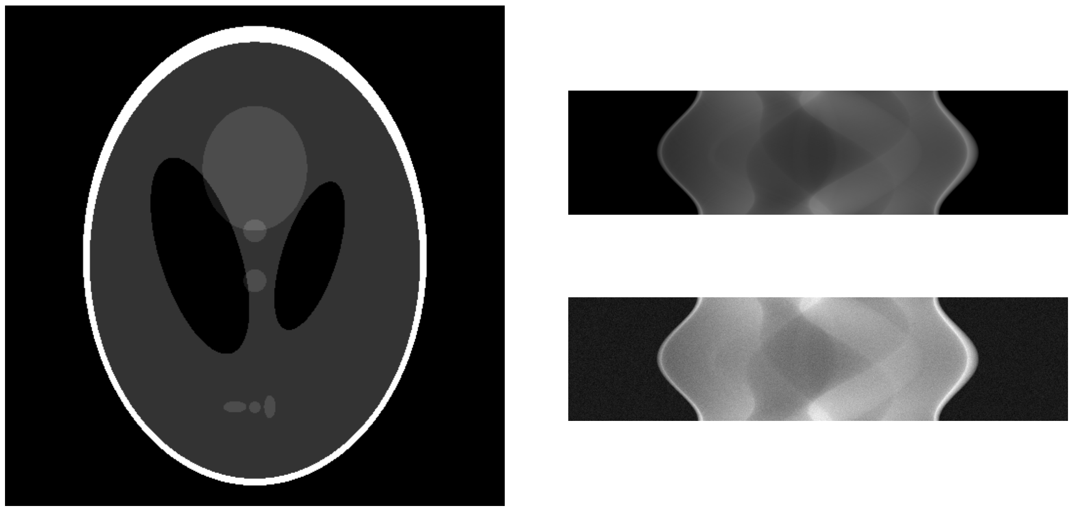

Figure 3: **Left:** *Shepp-Logan phantom.* Standard slice in 512×512 pixel resolution, provided by Matlab®. **Top right:** *Shepp-Logan sinogram.* 180 projections of the Shepp-Logan phantom, stepped at 1°. **Bottom right:** *Sinogram with additive Gaussian noise.*

4.2 Shepp-Logan Phantom

Input data. The most widely known reconstruction-from-projections test image is the Shepp-Logan phantom [4]. Introduced in 1974 it is still in common use today as a reference image for reconstruction algorithms. We obtained a 512×512 resolution image and calculated its projections using the `phantom` and `radon` functions provided by Matlab®. The radon projection (sinogram) generates 180 projections from 0 to 179°. After cropping to 512×180 pixels (the original sinogram is 729×180, respecting the size of the image diagonal), two instances, a noise-free and a version with additive Gaussian noise, have been stacked into a $512 \times 180 \times 2$ 3D volume to comply with the filter dimensionality requirements. The native Shepp-Logan phantom and the derived sinogram is shown in figure 3.

Zeropadding and oversampling. Figure 4 illustrates the reconstructed images obtained for a range of different zeropadding and DFT-oversampling configurations using radial nearest neighbor interpolation only. Images reconstructed with a low zeropadding and oversampling rate suffer from heavy interpolation artifacts, than can be mitigated at relatively high rates only.

Interpolation. A comparison of images reconstructed using different interpolation schemes in the radial direction are shown in figure 5. Although the ITK b-spline interpolate image function currently accepts spline orders up to 5, a good reconstruction quality can in general be obtained using cubic splines only, in combination with medium zeropadding and oversampling rates. A good interpolation scheme improves reconstruction quality and allows reducing the n_z and n_g rates.

Angular sampling rate. Figure 6 shows the reconstruction results for cubic b-spline interpolation and relatively high $n_z = n_g = 4$ rates. Although the overall reconstruction looks very convincing, narrowing of the optical window reveals high-frequency artifacts present over the whole image. They are equally reflected in the image intensity histogram: the sharp peaks of the original phantom are widened in the reconstructed image. These streak artifacts are due to the main weakness common to direct Fourier reconstruction methods: the limited high-frequency sample density in polar Fourier space degrades the interpolation due to angular undersampling with respect to bandwidth[5]. In figure 7 we show reconstructions from different numbers of views, illustrating the relation between streak artifacts and angular sampling rate.

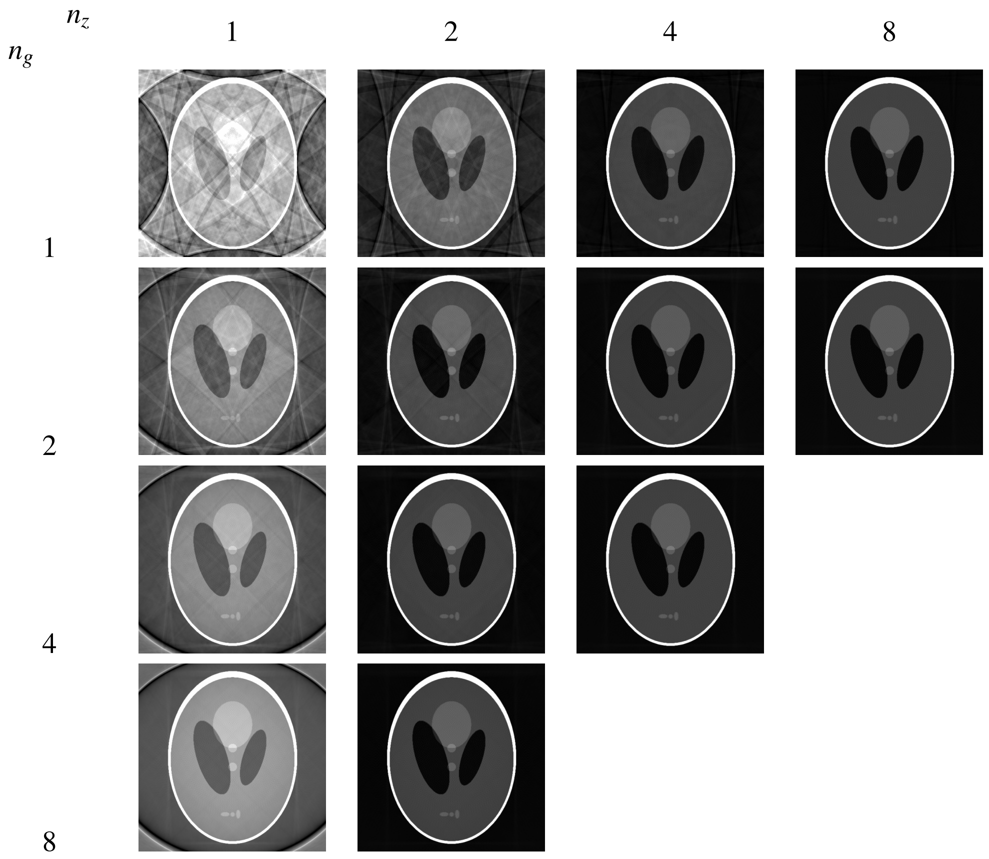

Figure 4: *Zeropadding and oversampling rates.* Zeropadding rates n_z increase from left to right, oversampling rates n_g increase with lines. Configurations using $n_z \cdot n_g > 16$ could not be run due to memory limitations. A significant improvement in image quality can be observed for the price of increasing resource requirements: while zeropadding removes time-domain overlap, DFT oversampling reduces interpolation noise.

Filtering. The angular bandwidth can be reduced by pre-filtering the projections with a low-pass smoothing filter. We used a Gaussian kernel along the radial (sic!) direction to smooth the sinogram before reconstruction. For the price of reduced contour sharpness, streak artifacts can be reduced significantly in the reconstructions, as illustrated in figure 8. In contrast, smoothing in the angular direction has a far more severe impact on contour sharpness, in particular further away from the image center, and does not result in the desired artifact reduction.

Noise. The important additive Gaussian noise is similarly affecting direct Fourier reconstruction, without any smoothing, and the commonly used filtered back projection (see figure 9). Prior radial smoothing with a Gaussian kernel cannot entirely remove the noise from the output images.

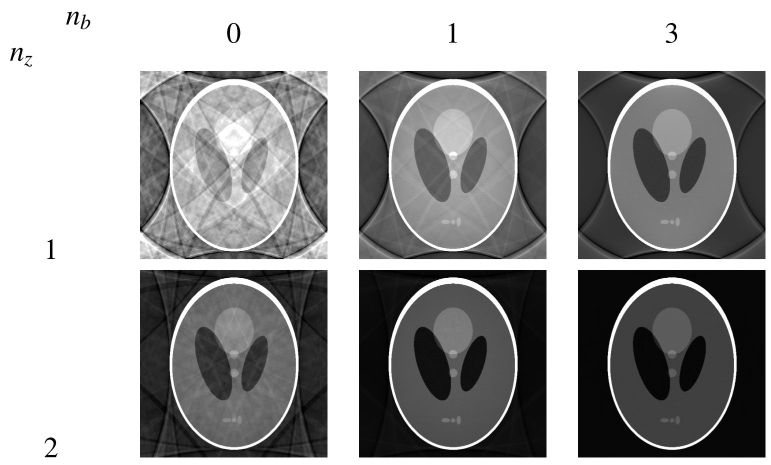

Figure 5: *DFT interpolation artifacts.* B-splines degree n_b increases from left to right, zeropadding rate increases with lines. Oversampling rate is set to $n_g = 1$. Increasing the degree of the interpolation scheme allows to achieve weak image artifacts even with low zeropadding and oversampling rates.

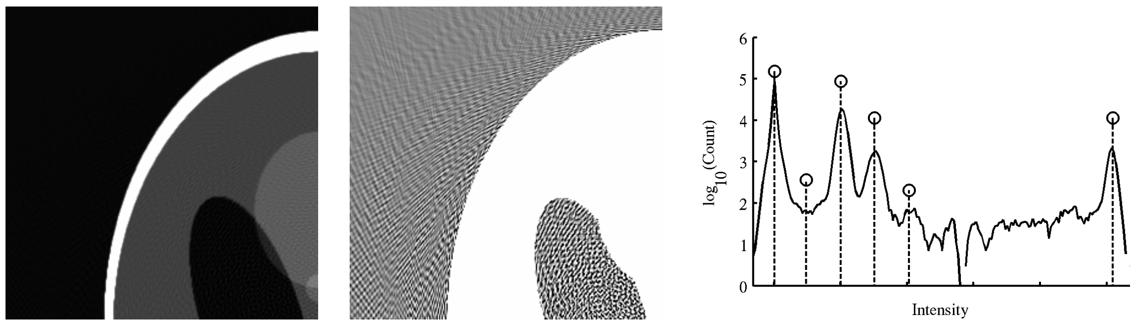

Figure 6: **Left:** *High-quality reconstruction.* Reconstruction with cubic b-spline interpolation and rates $n_z = n_g = 4$. **Center:** *Streak artifacts.* The optical window focuses on the background, revealing high-frequency streak artifacts. **Right:** *Image intensity histogram.* Image intensities of the reconstruction (solid line) spread around the phantom peaks (circles).

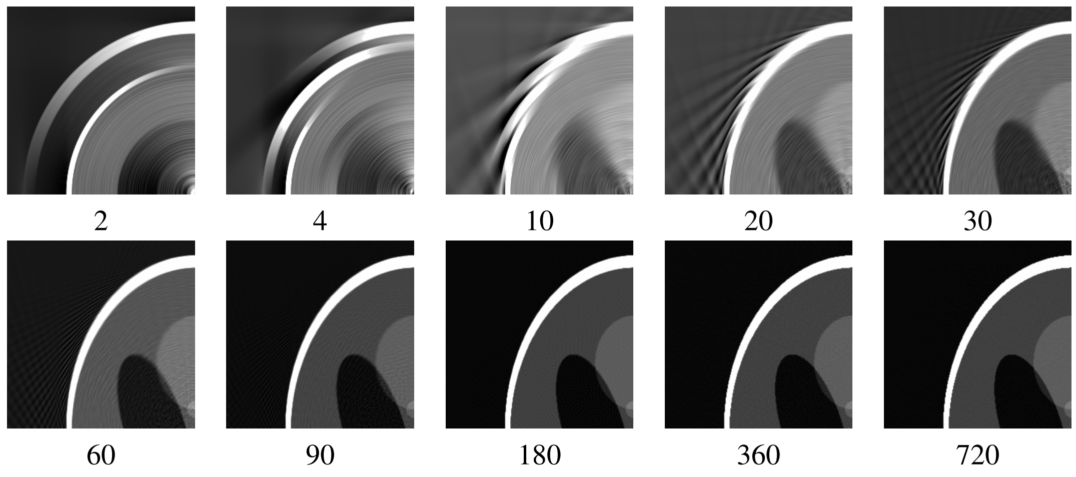

Figure 7: *Angular sampling and streak artifacts.* Reconstruction based on different numbers of views: Above 10 views, the reconstructed image exhibits the characteristic shape of the original phantom. With increasing number of views, the image quality improves and streak artifacts decrease.

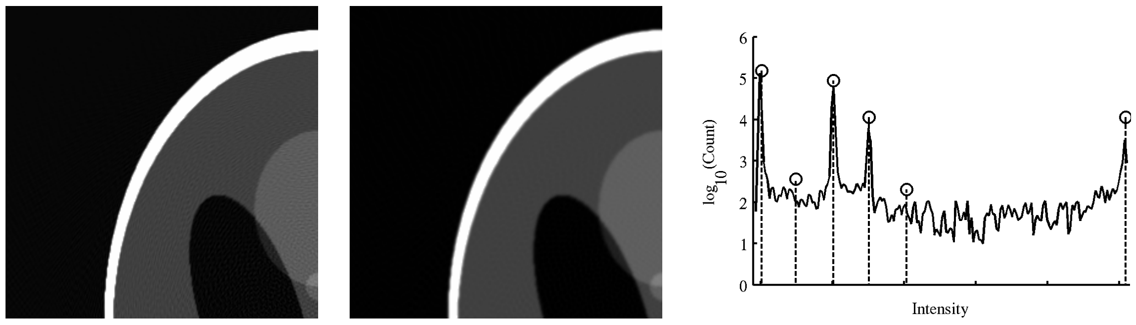

Figure 8: **Left:** *Reconstruction without band limiting pre-filtering.* Cubic b-spline interpolation and rates $n_z = n_g = 4$ **Center:** *Streak artifact reduction by Gaussian filtering.* A Gaussian kernel with $\sigma = 1$ is applied to the sinogram in radial direction to reduce the angular bandwidth. The reconstructed image shows less artifacts, but the contour sharpness has decreased. **Right:** *Improved image intensity histogram.* Filtering of input data drastically reduces the intensity spread in the reconstruction histogram.

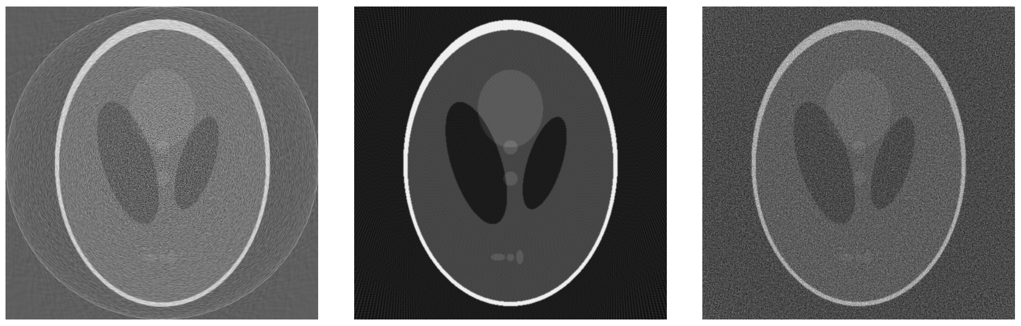

Figure 9: **Left:** *Direct Fourier reconstruction of noisy sinogram.* Configuration is $n_b = 3$, $n_z = 2$, $n_g = 2$, no smoothing. Important image degradation is apparent, the ring artifact results from the background-to-zero contrast. **Center:** *Filtered back projection of noiseless sinogram.* Obtained using Matlab® `iradon` function. Notice the streak artifacts. **Right:** *FBP of noisy sinogram.* The reconstructed image is severely affected by noise.

5 Conclusions

In this paper we reported on our ITK implementation of a direct Fourier tomographic reconstruction method for parallel-beam x-ray projections. We presented the overall structure of direct Fourier reconstruction and detailed the algorithm we developed. This algorithm was implemented as a `itk::ImageToImageFilter` class for 3D image processing. In addition, as currently no `itk::InterpolateImageFunction` is able to deal with complex datatypes, we provide a complex wrapper around the `itk::BSplineInterpolateImageFunction`.

The use of our `DirectFourierReconstructionImageFilter` class has been illustrated with a simple test application. We then provide extensive results based on the standard Shepp-Logan phantom image. We discuss the different reconstruction parameters and show their respective impact on the reconstruction outcome. We show how input zeropadding reduces signal domain replication (aliasing tails) and 2D-DFT oversampling reduces the dishing effect of Fourier domain interpolation (cyclic convolution). Chosing an appropriate radial interpolation spline order for Fourier domain resampling further improves the reconstruction quality. We finally compare the reconstruction results to a standard filtered back-projection method provided by Matlab®, confirming the high image quality of our DFR implementation.

References

[1] D. Gottlieb, B. Gustafsson, and P. Forssén. On the Direct Fourier Method for Computer Tomography. *IEEE Transactions on Medical Imaging*, 19(3):223–232, march 2000. 1

[2] R. M. Lewitt. Reconstruction Algorithms: Transform Methods. In *Proceedings of the IEEE*, volume 71, pages 390–408, march 1983. 1

[3] M. Magnusson, P.-E. Danielsson, and P. Edholm. Artefacts and Remedies in Direct Fourier Tomographic Reconstruction. *Nuclear Science Symposium and Medical Imaging Conference*, 2:1138–1140, october 1992. 1.2

[4] L. A. Shepp and B. F. Logan. The Fourier reconstruction of a head section. *IEEE Transactions on Nuclear Science*, 21:21–43, 1974. 4.2

[5] H. Stark. Sampling theorems in polar coordinates. *Journal of the Optical Society of America*, 69(11):1519–1525, nov 1979. 4.2

[6] H. Stark, J. Woods, I. Paul, and R. Hingorani. Direct Fourier reconstruction in computer tomography. *IEEE Transactions on Acoustics, Speech, and Signal Processing [see also IEEE Transactions on Signal Processing]*, 29(2):237–245, April 1981. 1