Simulation tools for two-dimensional experiments in x-ray computed tomography using the FORBILD head phantom

Phys. Med. Biol. **57** (2012) N237–N252 doi:10.1088/0031-9155/57/13/N237

NOTE

Simulation tools for two-dimensional experiments in x-ray computed tomography using the FORBILD head phantom

Zhicong Yu[1,2], Frédéric Noo[1], Frank Dennerlein[3], Adam Wunderlich[1], Günter Lauritsch[3] and Joachim Hornegger[2]

[1] Department of Radiology, University of Utah, Salt Lake City, UT, USA
[2] The Chair of Pattern Recognition, University of Erlangen-Nuremberg, Erlangen, Germany
[3] Siemens AG, Healthcare Sector, Forchheim, Germany

E-mail: zyu@ucair.med.utah.edu

Received 28 February 2012, in final form 17 May 2012
Published 20 June 2012
Online at stacks.iop.org/PMB/57/N237

Abstract
Mathematical phantoms are essential for the development and early stage evaluation of image reconstruction algorithms in x-ray computed tomography (CT). This note offers tools for computer simulations using a two-dimensional (2D) phantom that models the central axial slice through the FORBILD head phantom. Introduced in 1999, in response to a need for a more robust test, the FORBILD head phantom is now seen by many as the gold standard. However, the simple Shepp–Logan phantom is still heavily used by researchers working on 2D image reconstruction. Universal acceptance of the FORBILD head phantom may have been prevented by its significantly higher complexity: software that allows computer simulations with the Shepp–Logan phantom is not readily applicable to the FORBILD head phantom. The tools offered here address this problem. They are designed for use with Matlab®, as well as open-source variants, such as FreeMat and Octave, which are all widely used in both academia and industry. To get started, the interested user can simply copy and paste the codes from this PDF document into Matlab® M-files.

(Some figures may appear in colour only in the online journal)

1. Introduction

Mathematical phantoms are essential for the development and early stage evaluation of image reconstruction algorithms in x-ray computed tomography (CT). An important phantom was developed for this purpose in the initial years of CT. This phantom, called the Shepp–Logan phantom (Shepp and Logan 1974), has been heavily used, as well as a three-dimensional (3D) version of it that can be found in Kak and Slaney (2001). However, by the mid-1990s, it was recognized that this phantom and its 3D version were too simple for the evaluation of advanced reconstruction theories; more challenging phantoms were needed to improve the robustness of evaluations. In 1999, a solution to this problem was proposed by a group of CT researchers at the Institute of Medical Physics (Erlangen, Germany). Working together with scientists from Siemens Healthcare, this group developed the FORBILD phantoms[4], which model different parts of the anatomy. Among these phantoms, the models of the head and the thorax were quickly adopted by CT researchers, particularly for early stage evaluations of new image reconstruction algorithms; for example, see Sidky *et al* (2005), Bontus *et al* (2007), Tang and Hsieh (2007), Yan *et al* (2010), Yu and Wang (2010), Dennerlein and Noo (2011), Maass *et al* (2011), Noo *et al* (2004), Manzke *et al* (2005), King *et al* (2006), Li *et al* (2006), Dennerlein *et al* (2007), Ye *et al* (2007), Maass *et al* (2009), Schmidt (2009).

Unfortunately, the Shepp–Logan phantom is still largely used in papers that focus on two-dimensional (2D) image reconstruction (Horbelt *et al* 2002, Rieder and Faridani 2004, Zhang and Froment 2005, Chen *et al* 2006, You and Zeng 2007, Baek and Pelc 2009, Bilgot *et al* 2011, Supanich *et al* 2009, Louis 2011, Averbuch *et al* 2012). We attribute this situation to the fact that software for data simulation with the Shepp–Logan phantom is not readily applicable to the FORBILD head phantom. Two difficulties must be overcome, and handling these typically requires a significant investment in software development. The first difficulty stems from the fact that the primitive objects forming the phantom are not merely ellipsoids; elliptical cylinders, cones and truncated ellipsoids are also needed. Second, these objects are not combined together with the common addition formula: a precedency rule is used instead. That is, the order of the objects matters, because each object in the list is assigned the entire space it occupies, so that it can hide portions of objects that are listed prior to it.

In this paper, we present a 2D phantom that models the central axial slice through the FORBILD head phantom, and we offer simulation tools for 2D experiments with this phantom. This slice exhibits much of the complexity of the FORBILD head phantom and has been extensively employed in the development of 2D reconstruction algorithms (De Man and Basu 2002, Kachelriess *et al* 2003, Zou *et al* 2005, Yu and Wang 2007, Wunderlich and Noo 2008, Riviere and Vargas 2008). In particular, it includes more sophisticated features than the Shepp–Logan phantom, allowing the evaluation of low-contrast object detectability in the presence of artifact-inducing, high-contrast objects. The tools are designed for use with Matlab®, as well as any open-source versions of Matlab®, such as Octave and FreeMat.

2. Phantom definition

The FORBILD head phantom consists of 17 objects plus two insets, which we refer to as the left and right ears. Each of the 17 objects is either an ellipsoid, a cylinder, or a cone, or a portion thereof. The left ear is a resolution pattern built from ellipsoids of various sizes. The right ear is a model of the temporal bone, defined as a set of spherical air cavities within a high-density (bone-like) material. A depiction of how the 17 objects and the two insets appear within the central axial plane through the phantom is given in figure 1. Note that only 16, not 17 objects are seen in this figure, because the two cones that model the petrous bone cannot be distinguished from each other in this plane; they are reduced to the object labeled as 16.

The goal of this paper is to provide simulation tools for a 2D phantom that models the image displayed in figure 1. A proper name for this phantom may be ‘The 2D FORBILD head phantom’. The tools are offered with the option of including or leaving out either of the two ears. Our description is based on Cartesian coordinates, *x* and *y*, that are measured along the

[4] http://www.imp.uni-erlangen.de/phantoms.

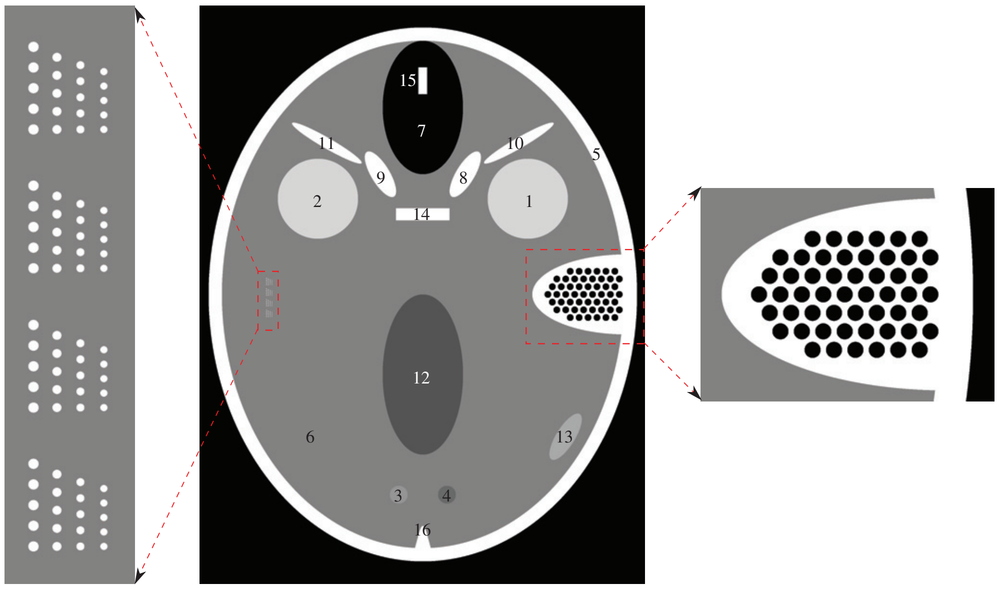

Figure 1. Central axial slice through the FORBILD head phantom. In this slice, only 16 of the 17 objects forming the phantom can be seen. These objects are each labeled with a number between 1 and 16. The phantom also includes two insets: the left and right ears, a magnified version of which is shown. Display window: [35, 65]HU.

minor and major axes of the outer ellipse in figure 1. The origin $(x, y) = (0, 0)$ is at the center of this ellipse.

3. Modeling technique

We present here our mathematical model for the 2D FORBILD head phantom. We start by discussing the overall design, then provide details on the most important aspects of the model. Note that lengths are always expressed in centimeters.

3.1. Design

The 2D FORBILD head phantom is built as a linear combination of functions that are each equal to 1 within an ellipse or the portion of an ellipse and 0 outside. Whenever the portion of an ellipse is involved, this portion is identified through the use of a clipping line, as discussed later. This construction has the advantage of yielding a simple description and enabling fast data simulation, particularly in comparison with using object superposition with a precedency rule (which is the method employed in the definition of the FORBILD phantom). However, these advantages come with a cost: our model is not an exact representation of the central axial slice through the FORBILD head phantom—it is a slight approximation. The misrepresentation issue is linked to the petrous bone (object 16), which is created from a cone in such a way that its boundary is a parabola that cannot be exactly described by an ellipse. In our model, the petrous bone is approximately represented, whereas all other features of the FORBILD phantom are accurately reproduced.

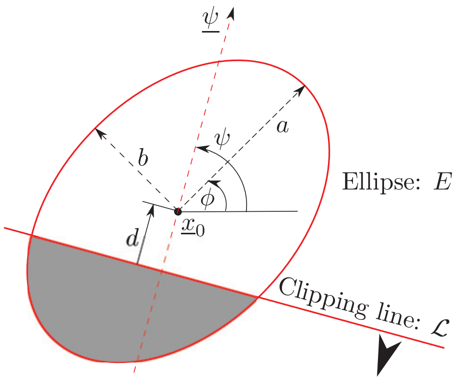

Figure 2. Illustration of an ellipse together with a clipping line. The portion retained after clipping is indicated by the arrow and the shaded area. Note that d is a signed distance that is negative in this drawing.

3.2. Ellipses and clipping lines

Let E be the ellipse of half-axes a and b that is centered at $\underline{x}_0$ and makes an angle ϕ with the x-axis, as shown in figure 2. By definition, any point $\underline{x}$ within E satisfies the equation

$$(\underline{x}-\underline{x}_0)^T\mathbf{Q}^T\mathbf{D}^2\mathbf{Q}(\underline{x}-\underline{x}_0) \leqslant 1, \tag{1}$$

where

$$\mathbf{D}=\begin{bmatrix}1/a & 0\\ 0 & 1/b\end{bmatrix}, \qquad \mathbf{Q}=\begin{bmatrix}\cos\phi & \sin\phi\\ -\sin\phi & \cos\phi\end{bmatrix}. \tag{2}$$

Next, let $\mathcal{L}$ be the line that is at signed distance d from the center of the ellipse in the direction of a given vector $\underline{\psi} = (\cos\psi, \sin\psi)$, as illustrated in figure 2. When the magnitude of d is small enough relative to the size of the ellipse, line $\mathcal{L}$ intersects the ellipse and can be used to single out a portion of it. When performing this role, we say that $\mathcal{L}$ is a clipping line. The portion of the ellipse that is identified as the object of interest is the set of points $\underline{x}$ such that

$$(\underline{x}-\underline{x}_0)^T\underline{\psi} < d. \tag{3}$$

Given an ellipse E and a number K of clipping lines, described with d_i and $\underline{\psi}_i = (\cos\psi_i, \sin\psi_i)$, where $i = 1,\dots,K$, we introduce a characteristic function:

$$f(\underline{x}) = \begin{cases}1 & \text{if } \underline{x}\in E \text{ and } (\underline{x}-\underline{x}_0)^T\underline{\psi}_i < d_i,\ \ i=1,\dots,K,\\ 0 & \text{otherwise.}\end{cases} \tag{4}$$

Our description of the 2D FORBILD head phantom without ears involves 17 objects that are each defined as one ellipse together with a number of clipping lines. The nth object is associated with a relative density value, μ_n, as well as with a function $f_n(\underline{x})$ that is defined in the same way as $f(\underline{x})$ in (4), so that the mathematical expression for the phantom with no ears is

$$\mu(\underline{x}) = \sum_{n=1}^{17}\mu_n f_n(\underline{x}). \tag{5}$$

When desired, the right ear (see figure 1) is included using the following transformations. First, object 6 is truncated, and a new clipped ellipse is introduced, as drawn in figure 3, to create the main body of this ear. The relative density assigned to this new ellipse is such

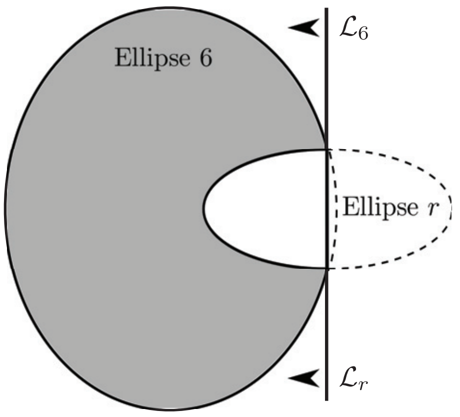

Figure 3. Construction of the right ear body. Object 6, which represents the brain matter, is clipped by a vertical line denoted as $\mathcal{L}_6$, and a new ellipse, called ellipse r, is introduced. This new ellipse is clipped by line $\mathcal{L}_r$, which is geometrically identical to $\mathcal{L}_6$, and is added with a relative density such that the overall density of the right ear body is equal to that of object 5, which depicts the skull.

that the value of $\mu(\underline{x})$ within the ear is equal to that of the skull (object 5) before adding the air cavities. Next, the air cavities are inserted within the ear as 53 small circular disks. The inclusion of the left ear is much simpler, as it amounts to just adding 80 circular disks.

Note that the full description of all objects involved in the phantom is embedded within the first Matlab® function that we supply. Note also that the values taken by $\mu(\underline{x})$ are to be interpreted as attenuation factors relative to that of water, so that $1000\,(\mu(\underline{x}) - 1)$ is in Hounsfield units. To perform simulations with noise or a polychromatic x-ray beam, some corrections must be applied, which will be discussed later.

3.3. Modeling of the petrous bone

As discussed earlier, our model involves an approximation because the petrous bone (object 16 in figure 1) could not be exactly represented using ellipses. We have represented this object using a combination of two clipped ellipses, denoted as objects 16a and 16b in figure 4, where ellipse a is the same ellipse as that describing object 6. Note that figure 4 is not drawn to scale: objects 16a and 16b are magnified for visibility purpose. Object 16a is the portion of ellipse a that remains after it is clipped by lines $\mathcal{L}_1^a$, $\mathcal{L}_2$ and $\mathcal{L}_3$. Object 16b is the portion of another ellipse, called ellipse b, that is clipped by line $\mathcal{L}_1^b$. The major axis of ellipse a is colinear with that of ellipse b, and lines $\mathcal{L}_2$ and $\mathcal{L}_3$ are mirror images of each other relative to this major axis. Moreover, these two lines are tangent to ellipse b, and lines $\mathcal{L}_1^a$ and $\mathcal{L}_1^b$, which are on top of each other, pass through the two tangency points, so that the boundary of the union of objects 16a and 16b is both continuous and differentiable.

The level of approximation involved in our representation of the petrous bone can be appreciated through inspection of images given in figure 5. A very good approximation is achieved near the points where $\mathcal{L}_2$ and $\mathcal{L}_3$ meet ellipse b, as well as near the tip of the bone. Away from these regions, the size of our petrous bone slightly differs from that of the original bone: see the plot in figure 5, which shows the width difference in x as a function of y.

Since the approximation involved in our representation of the petrous bone is fairly smooth and small in size, it is not expected to affect image quality evaluations in x-ray CT experiments.

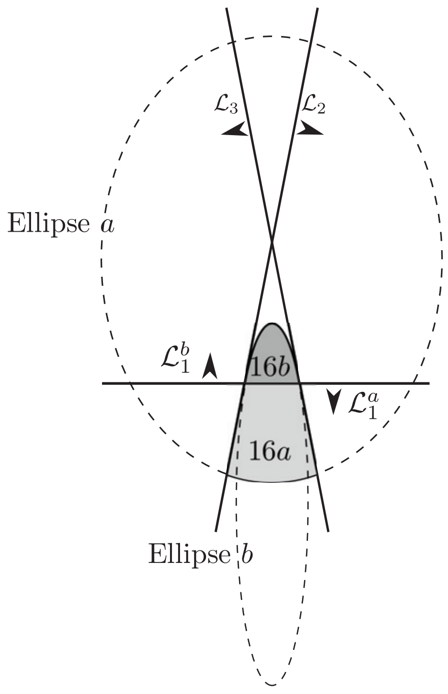

Figure 4. Modeling of the petrous bone (object 16 in figure 1). Our model is based on two ellipse portions: 16*a* from ellipse *a* and 16*b* from ellipse *b*.

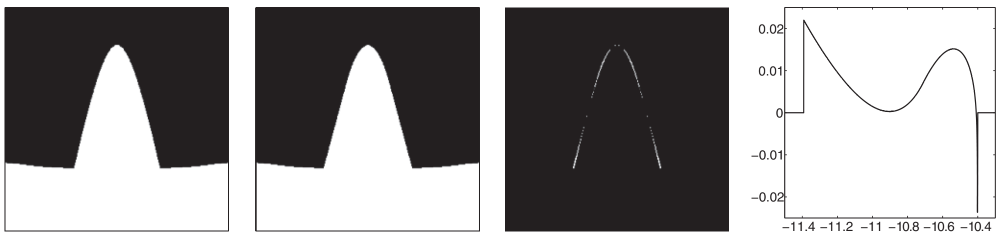

Figure 5. Approximation in our representation of the petrous bone. From left to right: the original bone; our approximation; mask highlighting the locations where the representation differs from the original; difference in width (in cm) as a function of *y*, from baseline to tip, between the representation and the original.

4. Simulation tools

We provide the reader with four Matlab® functions. The first function creates a matrix-based analytical representation of the phantom, and it must be executed before running the other functions. When executing this first function, the user gets the opportunity to decide which, if any, of the two ears should be included. The second function changes the attenuation value within the description of the phantom; it is only needed for experiments involving a polychromatic x-ray beam. The last two functions allow the user to obtain a discretized version of the phantom and to compute exact line integrals through the phantom, both with freely selectable sampling conditions.

This section is divided into three subsections that provide guidance on how to use each of the four Matlab® functions, as well as specific details on how these functions are built. Note that all four functions can be copied and pasted directly from the PDF version of this paper into Matlab® M-files, naming each file with the name of the function it contains.

4.1. Analytical representation

The function to generate the analytical representation of the 2D FORBILD head phantom is

$$[\mathbf{phantom}] = \mathbf{analytical\_phantom}(\mathbf{oL}, \mathbf{oR}),$$

where [oL] and [oR] are two input parameters acting on the inclusion or not of the left and right inner ears. By setting [oL = 1] (resp. [oR = 1]), the left inner ear (resp. the right inner ear) is added to the phantom, whereas any other value chosen for [oL] (resp. [oR]) results in the corresponding ear being left out.

The output of this function, called **phantom** above, is a structure containing two elements, **phantom.E** and **phantom.C**, which are both matrices containing the parameters for the ellipses and clipping lines forming the phantom, respectively. Each row of **phantom.E** defines one ellipse and the number of lines clipping it; the elements of any row successively correspond to the (x, y) coordinates of the ellipse center, the half-lengths, the orientation angle, the relative attenuation coefficient and the number of clipping lines. (Note that the correspondence between the object labels in figure 1 and the rows of E is highlighted in the Matlab® function using a comment (number) at the end of each row of E.)

The columns of **phantom.C** each define one clipping line. The first element of each column specifies the signed distance d, whereas the second element gives the orientation angle, ψ, in degrees. The order of the columns is critical and defined in agreement with the rows in **phantom.E**. Since there are no clipping lines associated with the first 12 rows in **phantom.E**, the first four columns of **phantom.C** define the clipping lines for the ellipse described in the 13th row of **phantom.E** (object 14), whereas the next four columns are the clipping lines for the ellipse in the 14th row, and so on.

As discussed earlier, the attenuation values within the phantom are expressed relative to that of water. There are two options to perform experiments with physical attenuation values. The first option is straightforward and generally sufficient for experiments assuming a monochromatic beam: simply multiply any line integral computed from **phantom** with the linear attenuation coefficient of water at some energy of interest. For example, a multiplication factor of 0.183 can be used when the energy of interest is 80 keV, which is close to the average energy of the beam for a diagnostic CT scan performed at 120 kVp. In defining this factor, recall that all lengths are expressed in centimeters; the attenuation coefficient for water at 80 keV is 0.183 cm^{-1}.

This first option discussed above to create physical line integrals is a minimal requirement to perform experiments with noise, which can be performed as explained, for example, in Wunderlich and Noo (2008). To perform experiments that involve a polychromatic x-ray beam, it is better to adopt another option, which accounts for the fact that the linear attenuation coefficients for water and bone change with the photon energy, denoted as E_p. This second option corresponds to using the function

$$[\mathbf{physphantom}] = \mathbf{physical\_value}(\mathbf{phantom}, \mathbf{muW}, \mathbf{muB}),$$

which transforms all attenuation coefficients within the phantom into physical values based on two materials, water and bone, so that any line integral computed from the output of this function, **physphantom**, has a physical meaning. The second and the third arguments are the linear attenuation coefficients for water and bone, respectively, at a given value for E_p.

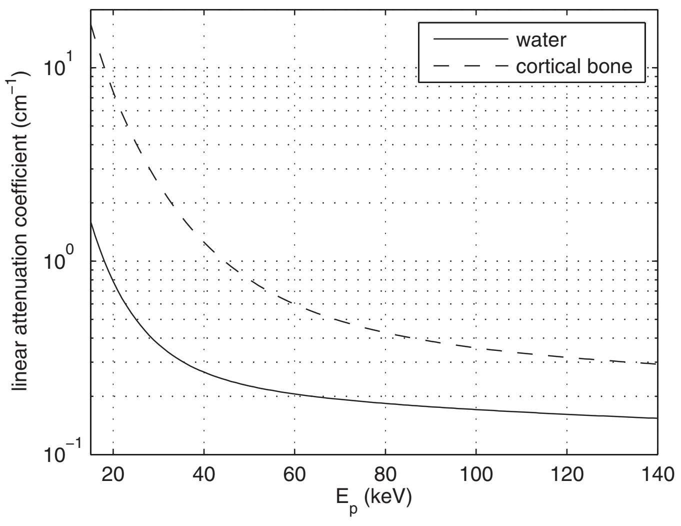

Figure 6. Linear attenuation coefficients for water and cortical bone as a function of the photon energy, E_p.

Table 1. Coefficients for the polynomial-based approximation of curves in figure 6; see equation (6).

	p_1	p_2	p_3	p_4	p_5
Water	−0.014 027	−0.045 959	2.366 105	−13.683 202	21.867 818
Cortical bone	−0.179 564	2.851 439	−16.055 087	35.924 159	−23.704 935

Conceptually, **physical_value** changes all attenuation values of 1.8 to muB, and replaces all other attenuation values by their product with muW (e.g., a value of 1.05 becomes 1.05 muW). Thus, all attenuation values that are equal to 1.8 are assumed to represent bone; and all attenuation values that differ from 1.8 are assumed to represent materials that behave like water, except for a relative change in mass density. A typical procedure for an experiment with a polychromatic x-ray beam is to compute the line integrals of interest for a number of energy values and combine those afterward using energy spectrum information (Whiting *et al* 2006).

Figure 6 shows how the linear attenuation coefficients for water and cortical bone vary with E_p. These two curves were obtained using the cross-sectional databases from NIST[5]. They are well approximated using the following polynomial-based expression:

$$\mu \simeq \exp\left\{p_1\,\varepsilon^4 + p_2\,\varepsilon^3 + p_3\,\varepsilon^2 + p_4\,\varepsilon + p_5\right\}, \tag{6}$$

where $\varepsilon = \ln E_p$, i.e. ε is the natural logarithm of E_p. The coefficients that appear in this expression are given in table 1. Note that μ and E_p are expressed in cm^{-1} and keV, respectively. Note also that the fitting is only accurate for E_p between 15 and 140 keV. The maximum relative error within this range is 2.5%. Finally, observe that the ratio between the attenuation coefficients of cortical bone and water at 80 keV is not 1.8, as could be expected; it is 2.32. Technically speaking, the human skull is not made of cortical bone only: the inner portion of the skull is made of less attenuating material, but the FORBILD head phantom does not allow for this level of detail. If a lower attenuation effect is desired, the third input of **physical_value** may be chosen as 0.77 times the linear attenuation coefficient of cortical bone. Either way, the

[5] http://www.nist.gov/pml/data/xray_gammaray.cfm.

most important issue is to clearly report in any research paper the expression that was used for muB.

4.2. Phantom discretization

A discrete version of the 2D FORBILD head phantom can be obtained using the following Matlab® function:

$$[\textbf{image}] = \textbf{discrete\_phantom}(\textbf{xcoord}, \textbf{ycoord}, \textbf{phantom}).$$

In this function, the third input argument is the output of **analytical_phantom**, or the output of **physical_value**, whereas the first and the second inputs are two matrices of the same size that, respectively, specify the x and the y coordinates corresponding to all desired samples. By construction, the output result, **image**, is a matrix that has the same size as the first two input arguments, with element (k, l) of this matrix being the value taken by the phantom at location (x, y) with x and y given by element (k, l) of the first and the second input arguments, respectively.

For example, suppose that we wish to create a 400×400 image of the phantom that is centered on the origin with a sampling step of 0.075 cm in both x and y. This task can be performed using the following Matlab® command lines to generate the first two input arguments:

```
x      = ((0:399)-199.5)*0.075;
y      = ((0:399)-199.5)*0.075;;
xcoord = ones(400,1)*x;
ycoord = transpose(y)*ones(1,400);
```

From these inputs, the output, **image**, will be a 400×400 matrix that contains pixel values for a fixed y in each row, with the first row corresponding to the most negative value of y. Within each row, x increases from left to right.

4.3. Line integrals

The second Matlab® function is for computation of line integrals through the phantom. We describe lines using the Radon transform notation, i.e. any line L through the object is specified using a scalar, s, and an angle, θ. By definition, line $L(\theta, s)$ is orthogonal to vector $\underline{\theta} = (\cos\theta, \sin\theta)$ at signed distance s from the origin, with s measured positively in the direction of $\underline{\theta}$, as shown on the left side of figure 7.

Any sets of line integrals through the 2D FORBILD head phantom can be obtained using the following function:

$$[\textbf{sino}] = \textbf{line\_integrals}(\textbf{scoord}, \textbf{theta}, \textbf{phantom}).$$

In this function, the third input is the output of **analytical_phantom**, or the output of **physical_value**, whereas the first and the second inputs are two matrices of the same size that, respectively, specify the parameters s and θ corresponding to all desired lines. By construction, the output result, **sino**, is a matrix that has the same size as the first two input arguments, with element (k, l) of this matrix being the integral of the phantom along the line $L(\theta, s)$ that results from using the element (k, l) of the first and the second input arguments to define s and θ, respectively.

For example, suppose that we wish to obtain a set of 1160 non-truncated parallel-beam projections, with a sampling step of 0.075 cm in s, and with the first projection in the direction

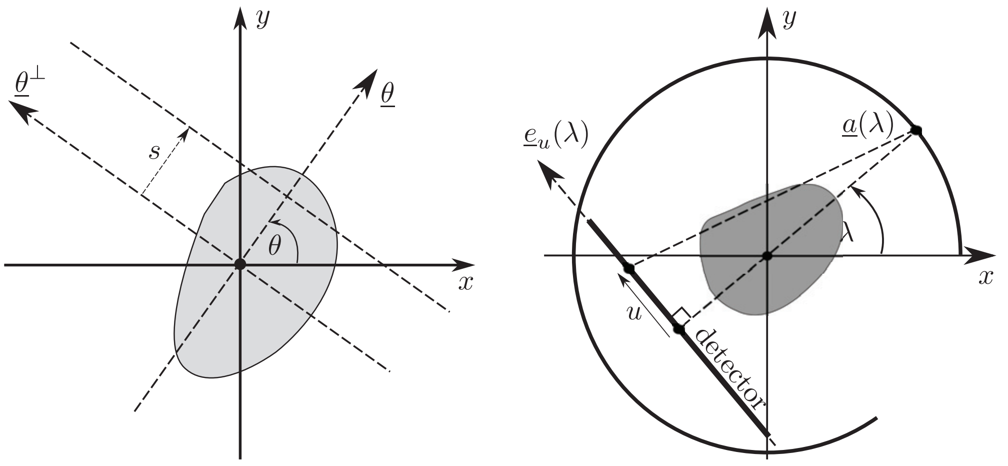

Figure 7. Circular scan geometries using a flat detector. Left: the parallel-beam geometry; right: the fan-beam geometry with radius of R and source-to-detector distance of D.

of y. Given that the phantom fits within a disk of radius 12 cm centered on the origin, a minimum of 320 rays per view is needed. Suppose that we set out for 351 rays evenly distributed across the origin $s = 0$. Then, the following command lines in Matlab® yield the inputs for **lines_integrals**:

```
s      = ((0:350)-175)*0.075;
theta  = (0:1159)*pi/1160-pi/2;
scoord = ones(1160,1)*s;
theta  = transpose(theta)*ones(1,351);
```

From these inputs, the output, **sino**, will be a matrix of size 1160×351, with each row yielding one of the desired parallel-beam projections.

Our Matlab® function can also be used to create fan-beam data. For illustration, consider a fan-beam data acquisition geometry with a linear detector array, as shown on the right side of figure 7. The distance from the source to the rotation center is R, and angle λ describes the rotation, so that the source position is $\underline{a}(\lambda) = (R\cos\lambda, R\sin\lambda)$. Also, the detector is parallel to $\underline{e}_u(\lambda) = (-\sin\lambda, \cos\lambda)$ at distance D from the source, with each point on the detector identified using a coordinate u that is measured positively in the direction of $\underline{e}_u(\lambda)$, and with $u = 0$ coinciding with the orthogonal projection of the source onto the detector. In this geometry, any line that connects the source to a point on the detector is given by

$$\begin{cases} \theta = \lambda + \pi/2 - \arctan(u/D), \\ s = uR/\sqrt{D^2 + u^2}. \end{cases} \tag{7}$$

Therefore, the following Matlab® command lines will create the two inputs required to obtain M fan-beam projections over 360° with N rays per view sampled with a stepsize W in u:

```
u        = ((0:N-1)-(N-1)/2)*W;
lambda   = (0:M-1)*2*pi/M; %lambda
uc       = ones(M,1)*u;
lambdac  = transpose(lambda)*ones(1,N);
scoord   = R*uc./sqrt(D^2+uc.^2);
theta    = lambdac+pi/2-atan(uc/D);
```

For the interested reader, appendix A provides details on the method and the associated notation used to compute the line integrals. There is however one important numerical aspect that we wish to highlight here. This aspect regards our handling of cases where $L(\theta, s)$ is parallel to one of the clipping lines. For any such line, we decided to add a tiny increment to the value of θ (specifically, 10^{-10}) to circumvent any mathematical singularity that could appear at some values of s. While having a negligible effect on the accuracy of simulations, this increment avoids errors that could occur when $L(\theta, s)$ is identical to one of the clipping lines used to define the petrous bone or the right ear. Moreover, it regularizes the output in cases where the line integral through an object is undefined, which occurs when the boundary of an object includes a segment of line and the line integral is to be taken along this line segment, as with object 14.

5. Conclusions

The primary purpose of this note was to offer simulation tools for 2D experiments in x-ray CT using a phantom that models the central slice through the FORBILD head phantom. These tools consist of four different Matlab® functions, the first two of which are dedicated to the phantom definition, whereas the other two are for phantom discretization and line integral computation. The reader might have noted that the last two functions are not restricted to the 2D FORBILD phantom. They apply to any 2D phantom that is built by addition of clipped ellipses. The tools offered here should enable more widespread use of the FORBILD head phantom, thereby facilitating the development of modern 2D image reconstruction theories.

Acknowledgments

This work was partially supported by a grant of Siemens AG, Healthcare Sector and by the US National Institutes of Health (NIH) under grant R21 EB009168. The concepts presented in this paper are based on research and are not commercially available. Its contents are solely the responsibility of the authors and do not necessarily represent the official views of the NIH.

Appendix A. Computation of line integrals

Here, we describe the technique used to compute a line integral through the object described by equation (4). Any line integral through the 2D FORBILD head phantom is a weighted sum of such line integrals, obtained using equation (5).

By definition, any point $\underline{x}$ on $L(\theta, s)$ can be written in the form

$$\underline{x} = s\,\underline{\theta} + t\,\underline{\theta}^{\perp} \tag{A.1}$$

for some $t \in \mathbb{R}$, with $\underline{\theta} = (\cos\theta, \sin\theta)$ and $\underline{\theta}^{\perp} = (-\sin\theta, \cos\theta)$. To obtain the integral of the function f in (4) on $L(\theta, s)$, we determine if $L(\theta, s)$ intersects the support of f or not, and at what values of t. If there are no intersections, the integral is zero. Otherwise, since the support of f is convex, there is only one entrance point and one exit point, and these two points fully determine the value for the desired integral, given that f is constant within its support. To determine the locations where $L(\theta, s)$ possibly meets the support of f, we first examine the intersections between $L(\theta, s)$ and the ellipse E forming (4), then we refine the results by examining the impact of each clipping line, as explained hereafter.

Assuming that $\underline{x}$ is on the boundary of the ellipse, which is given by (1), we obtain a quadratic equation for t:

$$(\underline{s}_0 + t\underline{\theta}^{\perp})^T \mathbf{Q}^T \mathbf{D}^2 \mathbf{Q} (\underline{s}_0 + t\underline{\theta}^{\perp}) = 1, \tag{A.2}$$

where $\underline{s}_0 = s\underline{\theta} - \underline{x}_0$. For the Matlab® function, this equation is rewritten in the form:

$$At^2 + Bt + C = 0, \qquad \text{with} \begin{cases} A & = \| \mathbf{DQ}\underline{\theta}^\perp \|^2 \\ B & = 2(\mathbf{DQ}\underline{\theta}^\perp) \cdot (\mathbf{DQ}\underline{s}_0) \\ C & = \| \mathbf{DQ}\underline{s}_0 \|^2 - 1. \end{cases}$$

When $B^2 > 4AC$, there exist two solutions t_P and t_Q, which indicates two intersection points between E and $L(\theta, s)$, denoted as P and Q, with $t_Q > t_P$. When $B^2 \leqslant 4AC$, the line integral is zero.

Once t_P and t_Q are known, the impact of each clipping line is determined by evaluating whether t_P and t_Q correspond to points within the retained portion of the ellipse or not. For clipping line $\mathcal{L}(\psi_i, d_i)$, this evaluation is performed using the following inequality test:

$$t\,\underline{\theta}^\perp \cdot \underline{\psi_i} < d_i - \underline{s}_0 \cdot \underline{\psi_i}\,. \tag{A.3}$$

If t_Q fails to satisfy this inequality, then t_Q is replaced by

$$t_Z = (d_i - \underline{s}_0 \cdot \underline{\psi_i})/(\underline{\theta}^\perp \cdot \underline{\psi_i}), \tag{A.4}$$

and the same operation is applied to t_P. Geometrically, t_Z corresponds to the intersection point between $L(\theta, s)$ and $\mathcal{L}(\psi_i, d_i)$. Once all clipping lines have been considered, the line integral is attributed the value of $t_Q - t_P$, which may be zero if both t_Q and t_P satisfy none of the clipping-line inequalities.

Naturally, the expression for t_Z is only well defined when the denominator in (A.4) is non-zero, which is always the case when $L(\theta, s)$ and $\mathcal{L}(\psi_i, d_i)$ are non-parallel. In our Matlab® function, this geometrical arrangement is enforced through the addition of a tiny increment to θ whenever $L(\theta, s)$ is parallel to one of the clipping lines (recall the discussion at the end of section 4.3).

Appendix B. Matlab® functions

```
function [ phantom ] = analytical_phantom(oL, oR)
if (oL~=1) oL=0; end; if (oR~=1) oR=0; end;
 sha    = 0.2*sqrt(3);            y016b = -14.294530834372887;
a16b    = 0.443194085308632;        b16b =   3.892760834372886;
E=[-4.7    4.3      1.79989    1.79989     0        0.010    0; %1
    4.7    4.3      1.79989    1.79989     0        0.010    0; %2
   -1.08  -9        0.4        0.4         0        0.0025   0; %3
    1.08  -9        0.4        0.4         0       -0.0025   0; %4
    0      0        9.6        12          0        1.800    0; %5
    0      8.4      1.8        3.0         0       -1.050    0; %7
    1.9    5.4      0.41633    1.17425   -31.07698 0.750    0; %8
   -1.9    5.4      0.41633    1.17425    31.07698 0.750    0; %9
   -4.3    6.8      1.8        0.24       -30       0.750    0; %10
    4.3    6.8      1.8        0.24        30       0.750    0; %11
    0     -3.6      1.8        3.6         0       -0.005    0; %12
  6.39395 -6.39395  1.2        0.42        58.1     0.005    0; %13
    0      3.6      2          2           0        0.750    4; %14
    0      9.6      1.8        3.0         0        1.800    4; %15
    0      0        9.0        11.4        0        0.750    3; %16a
    0     y016b     a16b       b16b        0        0.750    1; %16b
    0      0        9.0        11.4        0       -0.750   oR; %6
    9.1    0        4.2        1.8         0        0.750    1];%R_ear
```

```
%generate the air cavities in the right ear
cavity1     = transpose(8.8:-0.4:5.6);    cavity2 = zeros(9,1);
cavity3_7   = ones(53,1)*[0.15  0.15  0  -1.800  0];
for j = 1:3    kj = 8-2*floor(j/3);    dj = 0.2*mod(j,2);
cavity1 = [cavity1; cavity1(1:kj)-dj;  cavity1(1:kj)-dj];
cavity2 = [cavity2; j*sha*ones(kj,1); -j*sha*ones(kj,1)]; end
E_cavity    = [cavity1 cavity2 cavity3_7];
%generate the left ear (resolution pattern)
x0    = -7.0;y0 = -1.0;d0_xy = 0.04;
d_xy  = [0.0357, 0.0312, 0.0278, 0.0250];
x00   = zeros(0,0);      y00 = zeros(0,0);
ab  = 0.5*ones(5,1)*d_xy; ab = ab(:)*ones(1,4);
leftear4_7 = [ab(:) ab(:) ones(80,1)*[0  0.750  0]];
for i = 1:4 y00 = [y00; transpose(y0+(0:4)*2*d_xy(i))];
x00 = [x00; (x0+2*(i-1)*d0_xy)*ones(5,1)]; end
x00 = x00*ones(1,4);
y00 = [y00;y00+12*d0_xy; y00+24*d0_xy; y00+36*d0_xy];
leftear = [x00(:) y00 leftear4_7];
C=[ 1.2     1.2     0.27884  0.27884         0.60687  0.60687  0.2 ...
    0.2    -2.605  -2.605  -10.71177  y016b+10.71177  8.88740 -0.21260;
    0       180     90       270             90       270      0   ...
  180       15      165      90              270      0        0     ];
if    (oL==0&oR==0) phantom.E=E(1:17,:);           phantom.C=C(:,1:12);
elseif(oL==0&oR==1) phantom.E=[E;E_cavity];        phantom.C=C;
elseif(oL==1&oR==0) phantom.E=[leftear;E(1:17,:)]; phantom.C=C(:,1:12);
else                phantom.E=[leftear;E;E_cavity];phantom.C=C;    end
end


function [ physphantom ] =  physical_value(phantom,muW,muB)
physphantom.E=phantom.E;
physphantom.C=phantom.C;
nrows=size(phantom.E,1);
shift=0;if (nrows > 71) shift=80;end
if (nrows >= 97) physphantom.E(1:80,6)=muB-1.05*muW;end
if (nrows == 71 || nrows == 151)
    physphantom.E(18+shift,6)=muB-1.05*muW;
    physphantom.E((19:71)+shift,6)=-muB;
end
physphantom.E(5+shift,6)=muB;
physphantom.E(17+shift,6)=1.05*muW-muB;
physphantom.E(6+shift,6)=-1.05*muW;
physphantom.E(14+shift,6)=muB;
physphantom.E([7 8 9 10 13 15 16]+shift,6)=muB-1.05*muW;
j=[1 2 3 4 11 12];
physphantom.E(j+shift,6)=muW*phantom.E(j+shift,6);
end
```

```
function [ image ] = discrete_phantom(xcoord,ycoord,phantom)
image   = zeros(size(xcoord));
nclipinfo = 0;
for k   = 1:length(phantom.E(:,1))
    Vx0 = [transpose(xcoord(:))-phantom.E(k,1);
           transpose(ycoord(:))-phantom.E(k,2)];
    D   = [1/phantom.E(k,3) 0;0 1/phantom.E(k,4)];
    phi =  phantom.E(k,5)*pi/180;
    Q   = [cos(phi) sin(phi); -sin(phi) cos(phi)];
    f   = phantom.E(k,6);
    nclip     = phantom.E(k,7);
    equation1 = sum((D*Q*Vx0).^2);
    i   = find(equation1<=1.0);
    if (nclip > 0)
        for j = 1:nclip
            nclipinfo = nclipinfo+1;
            d         = phantom.C(1,nclipinfo);
            psi       = phantom.C(2,nclipinfo)*pi/180;
            equation2 = ([cos(psi) sin(psi)]*Vx0);
            i         = i(find(equation2(i)<d));
        end
    end
    image(i) = image(i)+f;
end


function [ sino ] = line_integrals(scoord,theta,phantom)
sinth = sin(transpose(theta(:)));costh = cos(transpose(theta(:)));
meps  = 1e-10;  nclipinfo = 0;    mask  = zeros(size(theta));
for k=1:size(phantom.C,2)
    psi = phantom.C(2,k)*pi/180;  tmp = -sinth*cos(psi)+costh*sin(psi);
    kk  = find(abs(tmp)<meps); mask(kk) = meps;
end
theta= theta+mask;              sino  = zeros(1,length(scoord(:)));
sinth = sin(transpose(theta(:)));costh = cos(transpose(theta(:)));
sx    = transpose(scoord(:)).*costh; sy = transpose(scoord(:)).*sinth;
for k=1:length(phantom.E(:,1))
    x0    = phantom.E(k,1);         y0    = phantom.E(k,2);
    a     = phantom.E(k,3);         b     = phantom.E(k,4);
    phi   = phantom.E(k,5)*pi/180;  f     = phantom.E(k,6);
    nclip = phantom.E(k,7);         s0    = [sx-x0;sy-y0];
    DQ = [cos(phi)/a sin(phi)/a; -sin(phi)/b cos(phi)/b];
    DQthp = DQ*[-sinth;costh];      DQs0  = DQ*s0;
    A  = sum(DQthp.^2);                B  = 2*sum(DQthp.*DQs0);
    C  = sum(DQs0.^2)-1;         equation = B.^2-4*A.*C;
    i = find(equation>0);
    tp = 0.5*(-B(i)+sqrt(equation(i)))./A(i);
    tq = 0.5*(-B(i)-sqrt(equation(i)))./A(i);
    if (nclip>0)
       for j = 1:nclip
         nclipinfo = nclipinfo+1;
         d  = phantom.C(1,nclipinfo);
         psi= phantom.C(2,nclipinfo)*pi/180;
         xp = sx(i)-tp.*sinth(i);     yp = sy(i)+tp.*costh(i);
         xq = sx(i)-tq.*sinth(i);     yq = sy(i)+tq.*costh(i);
         tz = d-cos(psi)*s0(1,i)-sin(psi)*s0(2,i);
         tz = tz./(-sinth(i)*cos(psi)+costh(i)*sin(psi));
         equation2 = ((xp-x0)*cos(psi)+(yp-y0)*sin(psi));
         equation3 = ((xq-x0)*cos(psi)+(yq-y0)*sin(psi));
         m1 = find(equation3>=d);  tq(m1) = tz(m1);
         m2 = find(equation2>=d);  tp(m2) = tz(m2);
       end
    end
    sinok = f*abs(tp-tq); sino(i) = sino(i)+sinok;
end
sino = reshape(sino,size(theta));
end
```

References

Averbuch A, Sedelnikov I and Shkolnisky Y 2012 CT reconstruction from parallel and fan-beam projections by a 2-D discrete Radon transform *IEEE Trans. Image Process.* **21** 733–41

Baek J and Pelc N 2009 Direct two-dimensional reconstruction algorithm for an inverse-geometry CT system *Med. Phys.* **36** 394–401

Bilgot A, Desbat L and Perrier V 2011 Filtered backprojection method and the interior problem in 2D tomography http://hal.archives-ouvertes.fr, hal-00591849, version 1

Bontus C, Koken P, Köhler T and Proksa R 2007 Circular CT in combination with a helical segment *Phys. Med. Biol.* **52** 107–20

Chen G, Tokalkanahalli R, Zhuang T, Nett B and Hsieh J 2006 Development and evaluation of an exact fan-beam reconstruction algorithm using an equal weighting scheme via locally compensated filtered backprojection (LCFBP) *Med. Phys.* **33** 475–81

De Man B and Basu S 2002 Distance-driven projection and backprojection *IEEE Nuclear Science Symp. Conf. Record* vol 3 pp 1477–80

Dennerlein F and Noo F 2011 Cone-beam artifact evaluation of the factorization method *Med. Phys.* **38** S18–24

Dennerlein F, Noo F, Hornegger J and Lauritsch G 2007 Fan-beam filtered-backprojection reconstruction without backprojection weight *Phys. Med. Biol.* **52** 3227–40

Horbelt S, Liebling M and Unser M 2002 Discretization of the Radon transform and of its inverse by spline convolutions *IEEE Trans. Med. Imaging* **21** 363–76

Kachelriess M, Berkus T and Kalender W 2003 Quality of statistical reconstruction in medical CT *IEEE Nuclear Science Symp. Conf. Record* vol 4 pp 2748–52

Kak A and Slaney M 2001 *Principles of Computerized Tomographic Imaging* (Philadelphia: SIAM)

King M, Pan X, Yu L and Giger M 2006 Region-of-interest reconstruction of motion-contaminated data using a weighted backprojection filtration algorithm *Med. Phys.* **33** 1222–38

Li L, Chen Z, Xing Y, Zhang L, Kang K and Wang G 2006 A general exact method for synthesizing parallel-beam projections from cone-beam projections via filtered backprojection *Phys. Med. Biol.* **51** 5643–54

Louis A 2011 Feature reconstruction in inverse problems *Inverse Problems* **27** 065010

Maass C, Baer M and Kachelriess M 2009 Image-based dual energy CT using optimized precorrection functions: a practical new approach of material decomposition in image domain *Med. Phys.* **36** 3818–29

Maass C, Meyer E and Kachelriess M 2011 Exact dual energy material decomposition from inconsistent rays (MDIR) *Med. Phys.* **38** 691–700

Manzke R, Koken P, Hawkes D and Grass M 2005 Helical cardiac cone beam CT reconstruction with large area detectors: a simulation study *Phys. Med. Biol.* **50** 1547–68

Noo F, Defrise M and Kudo H 2004 General reconstruction theory for multislice x-ray computed tomography with a gantry tilt *IEEE Trans. Med. Imaging* **23** 1109–16

Rieder A and Faridani A 2004 The semidiscrete filtered backprojection algorithm is optimal for tomographic inversion *SIAM J. Numer. Anal.* **41** 869–92

Riviere P La and Vargas P 2008 Correction for resolution nonuniformities caused by anode angulation in computed tomography *IEEE Trans. Med. Imaging* **27** 1333–41

Schmidt T 2009 Optimal image-based weighting for energy-resolved CT *Med. Phys.* **36** 3018–27

Shepp L and Logan B 1974 The Fourier reconstruction of a head section *IEEE Trans. Nucl. Sci.* **21** 21–43

Sidky E, Zou Y and Pan X 2005 Minimum data image reconstruction algorithms with shift-invariant filtering for helical, cone-beam CT *Phys. Med. Biol.* **50** 1643–57

Supanich M, Tao Y, Nett B, Pulfer K, Hsieh J, Turski P, Mistretta C, Rowley H and Chen G 2009 Radiation dose reduction in time-resolved CT angiography using highly constrained back projection reconstruction *Phys. Med. Biol.* **54** 4575–93

Tang X and Hsieh J 2007 Handling data redundancy in helical cone beam reconstruction with a cone-angle-based window function and its asymptotic approximation *Med. Phys.* **34** 1989–98

Whiting B, Massoumzadeh P, Earl O, O'Sullivan J, Snyder D and Williamson J 2006 Properties of preprocessed sinogram data in x-ray computed tomography *Med. Phys.* **33** 3290–303

Wunderlich A and Noo F 2008 Image covariance and lesion detectability in direct fan-beam x-ray computed tomography *Phys. Med. Biol.* **53** 2471–93

Yan H, Mou X, Tang S, Xu Q and Zankl M 2010 Projection correlation based view interpolation for cone beam CT: primary fluence restoration in scatter measurement with a moving beam stop array *Phys. Med. Biol.* **55** 6353–75

Ye Y, Yu H, Wei Y and Wang G 2007 A general local reconstruction approach based on a truncated Hilbert transform *Int. J. Biomed. Imaging* **2007** 63634

You J and Zeng G 2007 Hilbert transform based FBP algorithm for fan-beam CT full and partial scans *IEEE Trans. Med. Imaging* **26** 190–9

Yu H and Wang G 2007 Data consistency based rigid motion artifact reduction in fan-beam CT *IEEE Trans. Med. Imaging* **26** 249–60

Yu H and Wang G 2010 A soft-threshold filtering approach for reconstruction from a limited number of projections *Phys. Med. Biol.* **55** 3905–16

Zhang X and Froment J 2005 Total variation based Fourier reconstruction and regularization for computer tomography *IEEE Nuclear Science Symposium Conf. Record* vol 4 pp 2332–6

Zou Y, Pan X and Sidky E 2005 Image reconstruction in regions-of-interest from truncated projections in a reduced fan-beam scan *Phys. Med. Biol.* **50** 13–27