SoftwareX 7 (2018) 150–155

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

TomoPhantom, a software package to generate 2D–4D analytical phantoms for CT image reconstruction algorithm benchmarks

Daniil Kazantsev [a,b], Valery Pickalov [c], Srikanth Nagella [d], Edoardo Pasca [d,*], Philip J. Withers [a,b]

[a] The Manchester X-ray Imaging Facility, School of Materials, The University of Manchester, Manchester, M13 9PL, UK
[b] The Research Complex at Harwell, Didcot, Oxfordshire, OX110 FA, UK
[c] Khristianovich Institute of Theoretical and Applied Mechanics SB RAS, Novosibirsk, 630090, Russia
[d] Scientific Computing Department, Science & Technology Facilities Council - STFC, Rutherford Appleton Laboratory, Didcot, Oxfordshire, OX110 QX, UK

ARTICLE INFO

Article history:
Received 13 March 2018
Received in revised form 17 May 2018
Accepted 17 May 2018

Keywords:
Phantoms
Tomography
Image reconstruction
Iterative methods
Open-source

ABSTRACT

In the field of computerized tomographic imaging, many novel reconstruction techniques are routinely tested using simplistic numerical phantoms, e.g. the well-known Shepp–Logan phantom. These phantoms cannot sufficiently cover the broad spectrum of applications in CT imaging where, for instance, smooth or piecewise-smooth 3D objects are common. TomoPhantom provides quick access to an external library of modular analytical 2D/3D phantoms with temporal extensions. In TomoPhantom, quite complex phantoms can be built using additive combinations of geometrical objects, such as, Gaussians, parabolas, cones, ellipses, rectangles and volumetric extensions of them. Newly designed phantoms are better suited for benchmarking and testing of different image processing techniques. Specifically, tomographic reconstruction algorithms which employ 2D and 3D scanning geometries, can be rigorously analyzed using the software. TomoPhantom also provides a capability of obtaining analytical tomographic projections which further extends the applicability of software towards more realistic, free from the "inverse crime" testing. All core modules of the package are written in the C-OpenMP language and wrappers for Python and MATLAB are provided to enable easy access. Due to C-based multi-threaded implementation, volumetric phantoms of high spatial resolution can be obtained with computational efficiency.

`TomoPhantom` v.1.1. package metadata.

Current code version	Version 1.1
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX_2018_19
Legal Code License	Apache License v.2.0
Code versioning system used	git
Software code languages, tools, and services used	C, Python, Matlab
Compilation requirements, operating environments	C compilers (GCC/MinGW), Cython; Linux, Windows, Mac OS
If available Link to developer documentation/manual	For example: https://github.com/ElsevierSoftwareX/SOFTX_2018_19/tree/master/docs
Support email for questions	dkazanc@hotmail.com

1. Motivation and significance

The variety of image processing techniques, such as, reconstruction, denoising, deblurring, inpainting, and segmentation require simulated ground truth data in order to numerically verify, benchmark and evaluate the proposed methods. The ground truth data can have various specific to acquisition hardware characteristics, which also can be the focus of image processing methods. In many situations, however, over-simplistic or unsuitable simulated data are used for testing.

In computerized tomographic (CT) imaging [1], the well-known Shepp–Logan (SL) phantom [2] has been routinely used for reconstruction methods benchmarking for more than 40 years. Consisting of 10 ellipses of constant intensity (see Fig. 1), the SL phantom does not suit the diversity of objects for different imaging tasks (CT imaging supports many applications in materials science and medical imaging). For instance, in emission tomography [3], the investigated objects are intrinsically piecewise-smooth, so the use of the SL phantom would be an inaccurate approach. Similarly, in plasma tomography [4] or optical tomography [5,6], the object of

* Corresponding author.
E-mail addresses: daniil.kazantsev@manchester.ac.uk (D. Kazantsev), edoardo.pasca@stfc.ac.uk (E. Pasca).

https://doi.org/10.1016/j.softx.2018.05.003

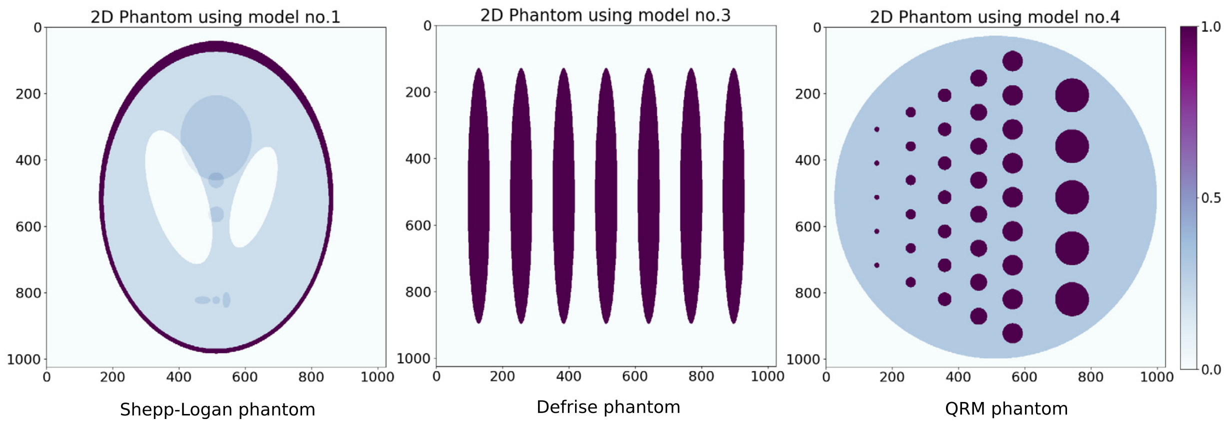


Fig. 1. Some classical 2D phantoms plotted using TomoPhantom software: the SL phantoms consists of 10 ellipses, Defrise phantom consists of 7 elongated ellipses and the QRM phantom is made of 41 circular ellipses of different radii.

interest can be represented by smooth, continuously differentiable Gaussian functions. In material science, it is common for a sample to comprise both piecewise-constant and piecewise-smooth objects [7].

One approach to test various CT reconstruction methods is to use phantoms which are generated directly from the experimental data [8,9]. Unfortunately, the spatial resolution of such datasets is normally fixed, there is a limited selection of scanned objects is available and large data storage is frequently required for 3D high-resolution data. Alternatively, one can build a parameterized phantom by using analytical geometrical objects in accordance to some mathematical formulae. This enables a customizable approach to select a desired discretization for a model with task-specific objects. Importantly, this approach does not require large storage since reproducible models can be built "on-the-fly" from the library of object descriptors (parameters).

Recently, an open-source software package XDesign [10] has been released that provides access to analytical phantoms mainly based on application to X-ray CT (XCT). XDesign is written in the Python language and supports 2D objects, including circles, triangles and triangular meshes. It can generate analytical projection data (sinograms) from custom designed phantoms and has integrated image quality metrics, which can be useful when assessing various reconstruction algorithms. However, XDesign lacks the capability of creating 3D phantoms, which is essential for testing cone-beam reconstruction methods [9]. Furthermore, XDesign delivers only piecewise-constant models which are ill-suited for many applications as explained above. XDesign has been developed in conjunction with TomoPy software [11] which delivers a large variety of data analysis and reconstruction tools for synchrotron X-ray imaging.

Another related software syris [12], provides access to basic 3D objects with no control of customization. syris delivers a significant contribution towards realistic physical modeling behind the image formation. Additionally, syris offers simple temporal modeling, i.e. the generation of 3D+time data. Syris has a Python interface and some core parts are written in OpenCL with GPU acceleration.

Here we present the TomoPhantom package which provides a number of complementary features to the existing capabilities of both XDesign and syris. The core modules are written in the C-OpenMP language, and wrappers for Python and MATLAB interfaces are provided. The C-based implementation enables computationally efficient generation of 2D/3D high resolution phantoms with temporal capability and also corresponding projection data. Notably, projection data are calculated following analytical Radon transform derivations applied to elementary functions [13,14]. This approach is different to conventional numerical calculation of projections for the forward X-ray acquisition model using ray-tracing algorithms [15]. Analytical projections are exact and discretized over a different grid which normally used for projection and backprojection steps in reconstruction methods. TomoPhantom enables a convenient testing environment for novel reconstruction methods, therefore avoiding the "inverse crime" issue [16] (data simulation using the same grid). Overall, the use of exact projections and known ground truth is especially beneficial for benchmarking of iterative reconstruction algorithms [17]. In TomoPhantom, the generated projection data are compatible with widely-used open-source tomographic reconstruction software: the ASTRA-toolbox [18] and the TomoPy package [11].

TomoPhantom is distributed with a library of pre-built enumerated models (a list of unique parameters) that one can refer to during benchmarking or testing of numerical algorithms. Some models also include classical phantoms (see Fig. 1) in 2D and 3D representations. With a plethora of reconstruction methods, there is a need of a robust benchmark system and TomoPhantom contributes to the issue.

Another important capability of TomoPhantom is the customizable temporal extension to available models. It is possible to generate 2D+time and 3D+time phantoms with desirable spatial and temporal resolution. This feature is especially useful when one needs to test reconstruction methods employing temporal information. The temporal capability in TomoPhantom is introduced through a special syntax in the library file. This is different from syris implementation, where temporal capability is given as a demonstration.

2. Software description

Here we provide information on the architecture of TomoPhantom software and discuss its main characteristics.

2.1. Software architecture

In Fig. 2 we show a block-diagram of TomoPhantom v.1.1. All core modules are developed in the C-OpenMP language, while the wrappers enable easy access to software from MATLAB and Python environments. We use Cython for Python and the C-MEX interface for MATLAB in order to wrap the C code. To compile the C code one also needs OS-specific compilers (e.g. GNU GCC, MinGW, Microsoft Visual Studio). TomoPhantom is partially based on the TOPAS-Micro software written using the Fortran language [19–22] and also the MATLAB version used in [23].

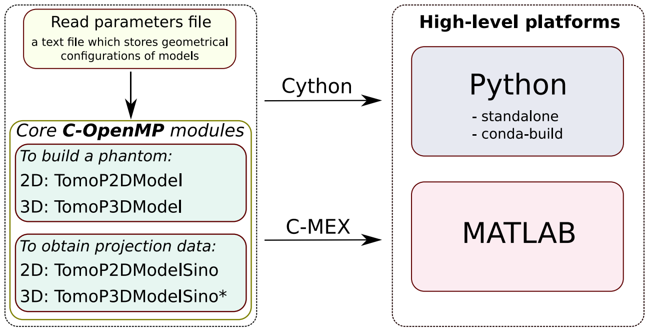


Fig. 2. A block diagram of `TomoPhantom` v.1.1. The module `TomoP3dModelSino*` - is under development.

2.2. Software functionalities and sample code snippets

The main functionalities of `TomoPhantom` include building 2D–4D analytical phantoms and their exact projection data.[1]

The pre-built models are given in files *Phantom2DLibrary.dat* and *Phantom3DLibrary.dat* for 2D and 3D cases, respectively. New models can be built by editing existing models in provided **.dat* files. There is a special syntax to follow, e.g. for a 3D object one specifies: Object: [gaussian, paraboloid, ellipsoid, cone, cuboid, elliptical cylinder], C_0, x_0, y_0, z_0, a, b, c, α, β, γ;. In Appendix we provide mathematical formulae which correspond to some of the objects used in `TomoPhantom`. Here $C_0 \in \mathbb{R}$ is an amplitude of an object, $(x_0, y_0, z_0) \in [-1, 1]$ are the center coordinates of the object, $(a, b, c) \in [-1, 1]$ the half-widths, and $(\alpha, \beta, \gamma) \in \mathbb{R}$ the Euler angles (in degrees). By adding geometrical objects recursively into a **.dat* file, one can build a modular phantom consisting of different objects. For instance, the volumetric model no. 10 in *Phantom3DLibrary.dat* is defined as follows:

```
# Static model: 1 volumetric Gaussian + 1 cuboid
Model : 10;
Components : 2;
TimeSteps : 1;
Object : gaussian 1 −0.25 −0.15 0 0.3 0.2 0.3 35 0 0;
Object : cuboid 1 0.1 0.2 0 0.15 0.35 0.6 −60 0 0;
```

This model has two geometrical objects (components). Note that the number of components must be specified in advance. The line *TimeSteps: 1* means that the given model is stationary. If one needs to specify a dynamic model, then the syntax is the following:

```
# Temporal (3D +time) model, 1 volumetric Gaussian +
1 paraboloid
Model : 100;
Components : 02;
TimeSteps : 5;
Object : paraboloid 1 0 0 0 0.2 0.4 0.6 30 0 0;
Endvar : 1 0.2 0.2 0 0.15 0.2 0.7 60 0 0;
Object : gaussian 1 −0.45 −0.45 −0.4 0.3 0.3 0.3 0 0 0;
Endvar : 1 0 −0.3 0.3 0.4 0.15 0.5 30 10 −40;
```

Note that *TimeSteps: 5* results in 5 volumetric time-frames, i.e. 4D phantom is produced. The line *Endvar:* pinpoints the final position of the *Object* above and its parameters. Therefore in this model, 5 equidistant steps are calculated between the original parameters of the object and the ones in *Endvar*. In `TomoPhantom` v.1.1. we provide only linear motion pattern, however, in future more complex nonlinear schemes can be introduced.

[1] In version 1.1., only 2D projection data capability is supported.

The current version of software comes with 40 pre-built models for 2D–4D cases (dynamic models are ≥ 100). In Figs. 3 and 4, we show some of 2D and 3D models, respectively. Software users can create own models and through pull requests they can be added into the master branch on GitHub, see metadata table. In future releases of `TomoPhantom`, only new models will be added and the existing ones will not be modified or deleted. Therefore, the unique model number can be used as a reference in publications.

Once `TomoPhantom` is compiled and installed, one can use it from Python or MATLAB. Let us first set parameters for the 2D case:

```
model = 11 # the number of the selected model
N = 512 # phantom dimensions (squared grid)
P = 724 # detector dimension
T = 804 # the number of projection angles
pathf = '..Phantom2DLibrary.dat' # parameters
        # file path
```

Then the 2D phantom and the corresponding parallel beam sinogram can be generated in Python as:

Listing 1: Python example using `TomoPhantom` v.1.1.

```
import numpy as np
from tomophantom import TomoP2D
# Generate [N,N] size phantom from model 11
phantom = TomoP2D.Model(model, N, pathf)
# Generate [P,T] size projection data (sinogram)
angles = np.linspace(0,180,T,dtype='float32')
sino = TomoP2D.ModelSino(model, N, P, angles, pathf, 1)
```

and similarly in MATLAB:

Listing 2: Matlab example using `TomoPhantom` v.1.1.

```
phantom = TomoP2DModel(model,N,pathf);
A = single(linspace(0,180,T));
sino = TomoP2DModelSino(model, N, P, A, pathf, 'radon');
```

The last parameter in ModelSino is responsible for the phantom centering. It is different when one uses MATLAB's *radon* function or `ASTRA-toolbox`. Similarly to the examples above, one can use TomoP3D.Model function to generate 3D phantoms. Additionally, using `TomoPhantom` web-link in metadata table, one can find demos where phantoms created without the use of **.dat* files.

3. Illustrative examples

In Fig. 1, we demonstrated some classical phantoms which exist in `TomoPhantom`. In Fig. 3, we show novel 2D models and in Fig. 4, 3D models, respectively. Along with the given model number in the library, we also provide its composition described by objects constituting a model. By showing these models we emphasize the capabilities of `TomoPhantom` software in creating quite complex scenes made of piecewise-constant and piecewise-smooth objects.

In order to demonstrate the computational agility of `TomoPhantom`, in Table 1, we provide computation times and memory usage for the example of building 3D model no. 9 (see Fig. 4). Since models are built recursively from the set of objects, the computational complexity of a model increases with the number of components. The model no. 9 consists of nine different objects. In addition to the number of objects present in the model, the computation time depends on the number of CPU threads available. In general, it is quick to build phantoms up to 1024^3 voxels in size, which would require approximately 4 GB of RAM or disk storage in single-precision floating-point format.

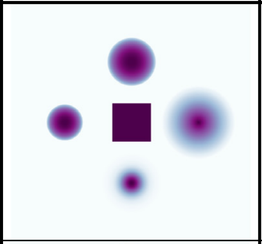
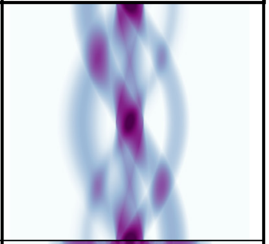
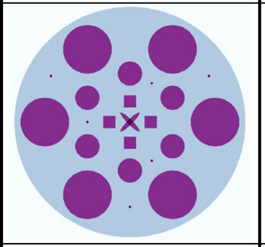
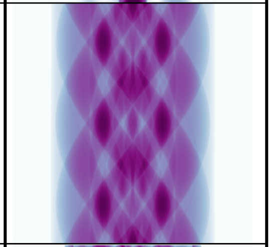
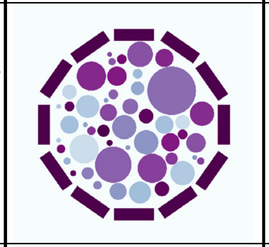
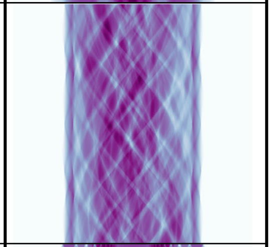
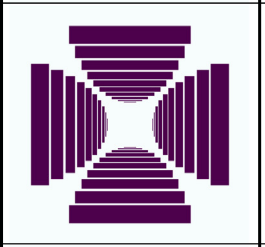
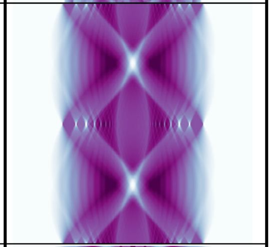
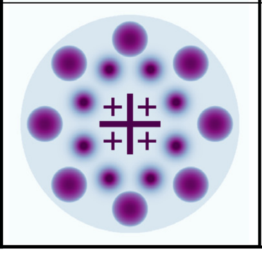
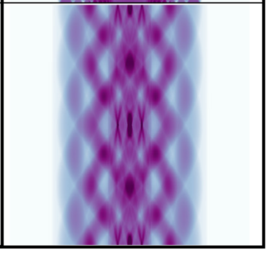
Model no.	Composites	Phantom	Analytical sinogram
10	1 rectangle 2 parabolas 1 Gaussian 1 cone		
11	21 ellipses 4 rectangles		
12	50 ellipses 12 rectangles		
13	40 rectangles		
14	1 ellipse 8 parabolas 8 Gaussians 18 rectangles		

Fig. 3. Selection of 2D models available in the library (see *Phantom2DLibrary.dat* file).

Table 1
Computation times to build volumetric model No. 9 (see Fig. 4) using Intel(R) Xeon(R) CPU E5-2630 with 24 CPU threads.

Dimension (voxels)	256^3	512^3	1024^3	2048^3
Time (seconds)	0.7	4.9	37.5	1295
Space required (GB)	0.06	0.5	4.0	32

4. Impact

Rigorous testing and benchmarking of reconstruction algorithms is frequently performed using over-simplistic low-resolution phantoms. The simulated projection data which are generated using the same grid and the same ray-tracing model leads to idealistic "inverse crime" reconstructions [16]. In order to ensure better testing practices for numerical algorithms in tomography, one needs flexible and efficient simulation software [10,12]. This is especially crucial with the growing number of novel iterative reconstruction methods [17].

TomoPhantom software aims to provide a set of tools for objective testing of reconstruction algorithms with a capability of referencing to the library of novel and classical models (phantoms). By using combined models with discontinuous and smooth objects, various properties of a numerical method can be investigated and highlighted. Provided analytical projections are not correlated with ray-tracing algorithms used in tomographic reconstruction software. Therefore, TomoPhantom can deliver more realistic simulated data for XCT methods testing. In addition to the simulation of exact projections, TomoPhantom offers routines for adding noise and various acquisition artifacts.

Apart from using TomoPhantom just for tomographic reconstruction purposes, the designed phantoms can be also used for a large variety of image processing tasks, e.g. denoising, deblurring, and segmentation. Therefore, the potential user community for TomoPhantom is large.

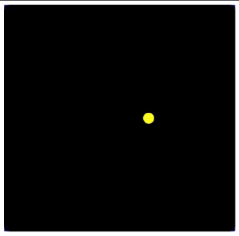
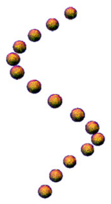
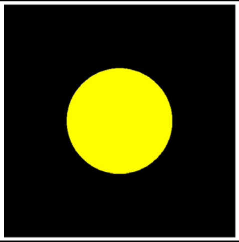
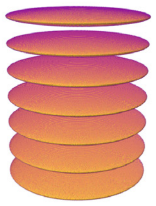
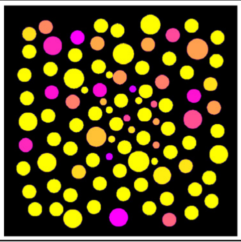
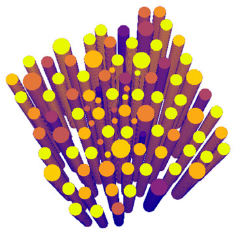
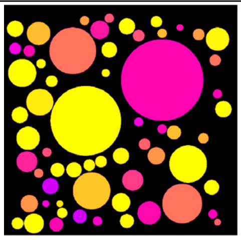
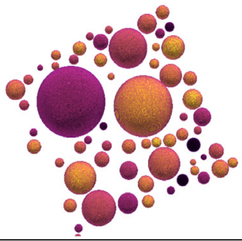
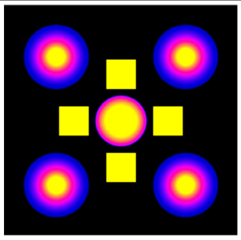
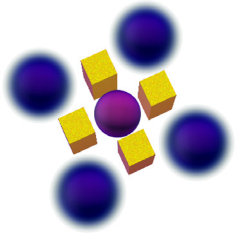
Model no.	Composites	Phantom, ax. slice	3D Phantom
1 'Snake'	15 ellipsoids		
2 'Defrise'	7 ellipsoids		
7	94 elliptical cylinders		
8	62 ellipsoids		
9	1 paraboloid 4 Gaussians 4 cuboids		

Fig. 4. Selection of 3D models available in the library (see *Phantom3DLibrary.dat* file). Models are visualized using Tomviz software [24].

5. Conclusions

In this paper we present open-source software `TomoPhantom` which can be used for testing and benchmark studies. A library of enumerated 2D–4D phantoms is provided for future referencing in scientific papers. The proposed software enables a quick and easy access to analytical tomographic projections which can be used to rigorously test image reconstruction algorithms. The core is written in the C-OpenMP language, and the wrappers for Python and MATLAB environments are provided.

Acknowledgments

This work has been funded by the EPSRC grant EP/P02226X/1: A 'Reconstruction Toolkit for Multichannel CT' and the CCPi initiative (EP/M022498/1). The authors acknowledge facilities and the support provided by the Research Complex at Harwell.

Appendix. Some functions and their Radon Transforms

In this section we provide some formulae which correspond to 2D elementary objects and also their analytical Radon Transforms.

A.1. Elliptical Gaussian

The function is given as:

$$g(x,y) = C\exp[-(4\ln 2)t^2], \tag{A.1}$$

$$t^2 = \frac{((x-x_0)\cos\varphi + (y-y_0)\sin\varphi)^2}{\tilde{a}^2} + \frac{(-(x-x_0)\sin\varphi + (y-y_0)\cos\varphi)^2}{\tilde{b}^2},$$

where (x_0, y_0) describe the center coordinates of an ellipse, $(\tilde{a}, \tilde{b})$ are the major/minor axes of an ellipse, respectively. φ is the rotation angle of the axis $\tilde{a}$ with respect to the OX axis. The Radon transform of the function (A.1) from the angle ξ to the axis OX is given as:

$$f(\xi, p) = \frac{\tilde{a}\tilde{b}C\sqrt{\pi}}{|\varsigma|\sqrt{4\ln 2}} \exp\left(-4\ln 2\frac{(p-p_0)^2}{\varsigma^2}\right), \tag{A.2}$$

$$p_0 = -x_0\sin\xi + y_0\cos\xi,\ \ \varsigma^2 = \tilde{a}^2\sin^2(\xi-\varphi) + \tilde{b}^2\cos^2(\xi-\varphi).$$

A.2. Parabola of λ-order with an elliptical support

The function is given as:

$$g(x, y) = \begin{cases} C(1-t^2)^\lambda, & t < 1, \\ 0, & t \geq 1. \end{cases} \tag{A.3}$$

The Radon transform of this function is:

$$f(\xi, p) = \frac{\sqrt{\pi}abC\,\Gamma(\lambda+1)}{|\varsigma|\,\Gamma(\lambda+3/2)}\left(1 - \frac{(p-p_0)^2}{\varsigma^2}\right)^{\lambda+1/2}, \tag{A.4}$$

where $\Gamma(\lambda)$ is the gamma function and p_0 and ς^2 are given as above.

Other elementary functions, such as, rectangular, circles, triangles etc. can be found in the references [13,14].

References

[1] Kak AC, Slaney M. Principles of computerized tomographic imaging. N.Y.: IEEE Press; 1988.
[2] Shepp LA, Logan BF. The Fourier reconstruction of a head section. IEEE Trans Nucl Sci 1974;21(3):21–43. http://dx.doi.org/10.1109/TNS.1974.6499235.
[3] Wernick MN, Aarsvold JN. Emission tomography: The fundamentals of PET and SPECT. Elsevier; 2004.
[4] Balandin AL, Likhachev AV, Panferov NV, Pikalov VV, Rupasov AA, Shikanov AS. Tomographic diagnostics of radiating plasma objects. J Sov Laser Res 1992;13(6):472–98. http://dx.doi.org/10.1007/BF01120652.
[5] Arridge SR. Optical tomography in medical imaging. Inverse Problems 1999;15 (2):R41. http://dx.doi.org/10.1088/0266-5611/15/2/022.
[6] Correia T, Aguirre J, Sisniega A, Chamorro-Servent J, Abascal J, Vaquero JJ, et al. Split operator method for fluorescence diffuse optical tomography using anisotropic diffusion regularisation with prior anatomical information. Biomed Opt Express 2011;2(9):2632–48. http://dx.doi.org/10.1364/BOE.2.002632.
[7] Kazantsev D, Guo E, Phillion AB, Withers PJ, Lee PD. Model-based iterative reconstruction using higher-order regularization of dynamic synchrotron data. Meas Sci Technol 2017;28(9):094004. http://dx.doi.org/10.1088/1361-6501.
[8] De Carlo F, Gürsoy D, Ching DJ, Batenburg KJ, Ludwig W, Mancini L, et al. TomoBank: A tomographic data repository for computational X-ray science. Meas Sci Technol 2018;29(3):034004. http://dx.doi.org/10.1088/1361-6501/aa9c19.
[9] Jørgensen JS, Coban SB, Lionheart WR, McDonald SA, Withers PJ. Sparsebeads data: Benchmarking sparsity-regularized computed tomography. Meas Sci Technol 2017;28(12):124005. http://dx.doi.org/10.1088/1361-6501/aa8c29.
[10] Ching DJ, Gürsoy D. Xdesign: An open-source software package for designing X-ray imaging phantoms and experiments. J Synchrotron Radiat 2017;24(2):537–44. http://dx.doi.org/10.1107/S1600577517001928.
[11] Gürsoy D, De Carlo F, Xiao X, Jacobsen C. TomoPy: A framework for the analysis of synchrotron tomographic data. J Synchrotron Radiat 2014;21(5):1188–93. http://dx.doi.org/10.1107/S1600577514013939.
[12] Faragó T, Mikulik P, Ershov A, Vogelgesang M, Haenschke D, Baumbach T. Syris: A flexible and efficient framework for X-ray imaging experiments simulation. J Synchrotron Radiat 2017;24(6):1283–95. http://dx.doi.org/10.1107/S1600577517012255.
[13] Deans SR. The Radon transform and some of its applications. N.Y: John Wiley; 1983.
[14] Toft P. The Radon transform Theory and implementation [Ph.D. thesis], DTU; 1996.
[15] Siddon RL. Fast calculation of the exact radiological path for a three-dimensional CT array. Med Phys 1985;12(2):252–5. http://dx.doi.org/10.1118/1.595715.
[16] Kaipio J, Somersalo E. Statistical inverse problems: discretization, model reduction and inverse crimes. J Comput Appl Math 2007;198(2):493–504. http://dx.doi.org/10.1016/j.cam.2005.09.027.
[17] Beister M, Kolditz D, Kalender WA. Iterative reconstruction methods in X-ray CT. Phys Med: Eur J Med Phys 2012;28(2):94–108. http://dx.doi.org/10.1016/j.ejmp.2012.01.003.
[18] van Aarle W, Palenstijn WJ, Cant J, Janssens E, Bleichrodt F, Dabravolski A, et al. Fast and flexible X-ray tomography using the ASTRA toolbox. Opt Express 2016;24(22):25129–47. http://dx.doi.org/10.1364/OE.24.025129.
[19] Pikalov VV, Preobrazhenskii NG. Computer-aided tomography and physical experiment. Sov Physics-Uspekhi 1983;26(11):974–90. http://dx.doi.org/10.1070/PU1983v026n11ABEH004538.
[20] Likhachov AV, Pickalov VV. Subpixel resolution in 3D cone - beam microtomography. Nucl Instrum Methods Phys Res A 1995;359(1–2):370–5. http://dx.doi.org/10.1016/0168-9002(94)01677-1.
[21] Likhachov AV, Pickalov VV. Three-dimensional tomography with finite aperture beams. Nucl Instrum Methods Phys Res A 1998;405(2–3):506–10. http://dx.doi.org/10.1016/S0168-9002(97)00178-2.
[22] Derevtsov EY, Pickalov VV. Reconstruction of vector fields and their singularities from ray transforms. Numer Anal Appl 2011;4(1):21–35. http://dx.doi.org/10.1134/S1995423911010034.
[23] Kazantsev D, Pickalov V. New iterative reconstruction methods for fan-beam tomography. Inverse Probl. Sci. Eng. 2018;26(6):773–91. http://dx.doi.org/10.1080/17415977.2017.1340946.
[24] Jiang Y, Padgett E, Hanwell MD, Quammen C, Harris C, Waldon S, et al. tomviz: providing advanced electron tomography by streamlining alignment, reconstruction, and 3D visualization. Microsc Microanal 2017;23(S1):222–3. http://dx.doi.org/10.1017/S1431927617001799.