

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

Daniil Kazantsev<sup>a</sup>, Valery Pickalov<sup>b</sup>, Srikanth Nagella<sup>c</sup>, Edoardo Pasca<sup>c</sup>, Philip J. Withers<sup>a</sup>

<sup>a</sup>*The Manchester X-Ray Imaging Facility, School of Materials, The University of Manchester, Manchester, M13 9PL, UK; The Research Complex at Harwell, Didcot, Oxfordshire, OX11 0FA, UK.*

<sup>b</sup>*Khristianovich Institute of Theoretical and Applied Mechanics SB RAS, Novosibirsk, 630090, Russia.*

<sup>c</sup>*Scientific Computing Department, Science & Technology Facilities Council - STFC, Rutherford Appleton Laboratory, Didcot, Oxfordshire, OX11 0QX, UK.*

---

## 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 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.

**Keywords:** phantoms, tomography, image reconstruction, iterative methods, open-source

---

## 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 method. 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 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].

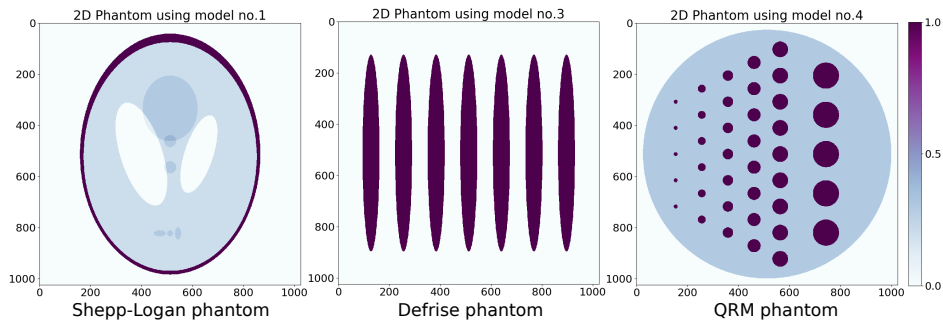


Figure 1: Some classical 2D phantoms plotted using TomoPhantom software: the SL phantoms consists of 10 ellipses, Defrise phantom consists of 7 elongated ellipses and the QRM phantom is made of 41 circular ellipses of different radii.

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** [11] has been released that provides an 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 while 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 [10] which delivers a large variety of data analysis and reconstruction tools for synchrotron X-ray imaging.

Another related software **syris** [12], provides an 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 a 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

67 data are compatible with widely-used open-source tomographic reconstruction software: the **ASTRA-toolbox** [18] and the **TomoPy** package [10].

69 **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.

75 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.

## 83 2. Software description

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

### 86 2.1. Software Architecture

In Fig. 2 we show a block-diagram of **TomoPhantom** v.1.1. All core mod-

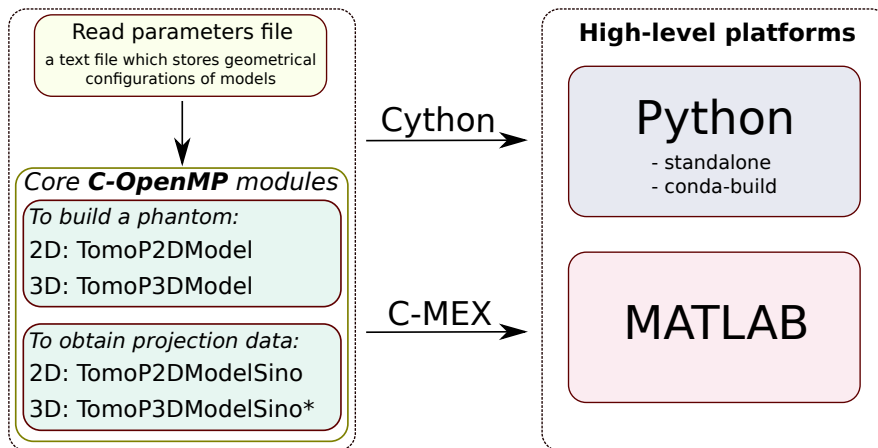


Figure 2: A block diagram of **TomoPhantom** v.1.1. The module **TomoP3dModelSino\*** - is under development.

87 ules 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] .

## 2.2. Software functionalities and sample code snippets

The main functionalities of **TomoPhantom** include building 2D-4D analytical phantoms and their exact projection data<sup>1</sup>.

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],  $C_0$ ,  $x_0$ ,  $y_0$ ,  $z_0$ ,  $a$ ,  $b$ ,  $c$ ,  $\alpha$ ,  $\beta$ ,  $\gamma$ ;. In Appendix A 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;
```

<sup>1</sup>In version 1.1., only 2D projection data capability is supported.

```

128 Object : paraboloid 1 0 0 0 0.2 0.4 0.6 30 0 0;
129 Endvar : 1 0.2 0.2 0 0.15 0.2 0.7 60 0 0;
130 Object : gaussian 1 -0.45 -0.45 -0.4 0.3 0.3 0.3 0 0 0;
131 Endvar : 1 0 -0.3 0.3 0.4 0.15 0.5 30 10 -40;
132

```

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

139 The current version of software comes with 40 pre-built models for 2D-  
140 4D cases (dynamic models are  $\geq 100$ ). In Fig. 3 and 4, we show some of  
141 2D and 3D models, respectively. Software users can create own models and  
142 through pull requests they can be added into the master branch on GitHub,  
143 see Table A.2. In future releases of **TomoPhantom**, only new models will be  
144 added and the existing ones will not be modified or deleted. Therefore, the  
145 unique model number can be used as a reference in publications.

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

```

148
149 model = 11 # the number of the selected model
150 N = 512 # phantom dimensions (squared grid)
151 P = 724 # detector dimension
152 T = 804 # the number of projection angles
153 pathf = '..Phantom2DLibrary.dat' # parameters file path
154

```

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

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

```

157 import numpy as np
158 from tomophantom import TomoP2D
159 # Generate [N,N] size phantom from model 11
160 phantom = TomoP2D.Model(model, N, pathf)
161 # Generate [P,T] size projection data (sinogram)
162 angles = np.linspace(0,180,T, dtype='float32')
163 sino = TomoP2D.ModelSino(model, N, P, angles, pathf, 1)
164 and similarly in MATLAB:

```

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

```

165 % Generate [N,N] size phantom from model 11

```

```

166 phantom = TomoP2DModel(model,N,pathf);
167 % Generate [P,T] size projection data (sinogram)
168 A = single(linspace(0,180,T)); % projection angles
169 sino = TomoP2DModelSino(model, N, P, A, pathf, 'radon');
170 The last parameter in ModelSino is responsible for the phantom centering.
171 It is different when one uses MATLAB's radon function or ASTRA-toolbox.
172 Similarly to the examples above, one can use TomoP3D.Model function to
173 generate 3D phantoms. Additionally, using TomoPhantom web-link in Table
174 A.2, one can find demos where phantoms created without the use of *.dat
175 files.

```

### 176 3. Illustrative Examples

177 In Fig. 1, we demonstrated some classical phantoms which exist in  
178 TomoPhantom. In Fig. 3, we show novel 2D models and in Fig. 4, 3D models,  
179 respectively. Along with the given model number in the library, we also pro-  
180 vide its composition described by objects constituting a model. By showing  
181 these models we emphasize the capabilities of TomoPhantom software in cre-  
182 ating quite complex scenes made of piecewise-constant and piecewise-smooth  
183 objects.

184 In order to demonstrate the computational agility of TomoPhantom, in  
185 Table 1, we provide computation times and memory usage for the example of  
186 building 3D model no. 9 (see Fig. 4). Since models are built recursively from  
187 the set of objects, the computational complexity of a model increases with the  
188 number of components. The model no. 9 consists of nine different objects. In  
189 addition to the number of objects present in the model, the computation time  
190 depends on the number of CPU threads available. In general, it is quick to  
191 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.

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

192

### 193 4. Impact

194 Rigorous testing and benchmarking of reconstruction algorithms is fre-  
195 quently performed using over-simplistic low-resolution phantoms. The sim-  
196 ulated projection data which generated using the same grid and the same

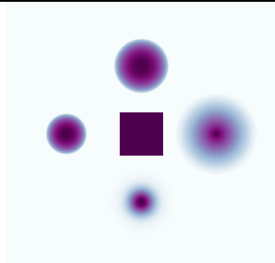
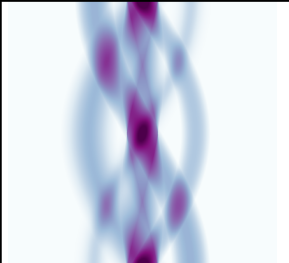
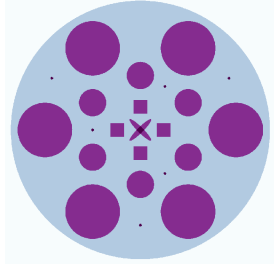
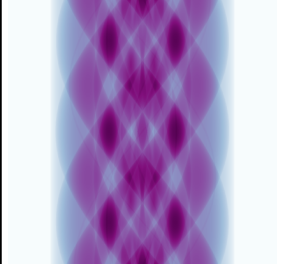
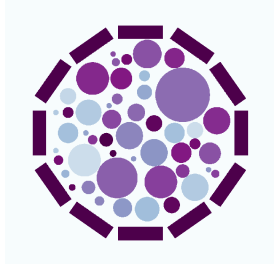
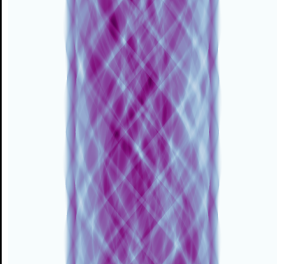
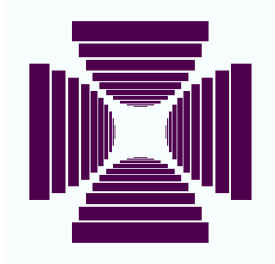
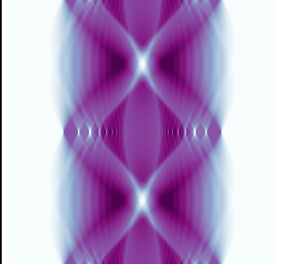
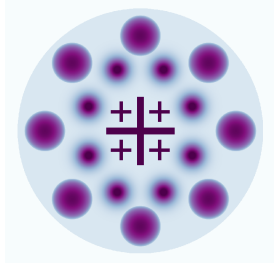
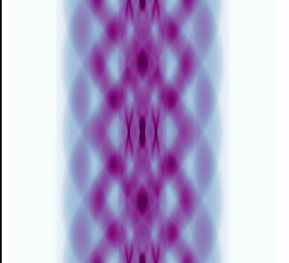
Model no.	Composites	Phantom	Analytical sinogram
10	1 rectangle 2 parabolas 1 Gaussian 1 cone		
11	21 ellipses 4 rectangles		
12	50 ellipses 12 rectangles		
13	40 rectangles		
14	1 ellipse 8 parabolas 8 Gaussians 18 rectangles		

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



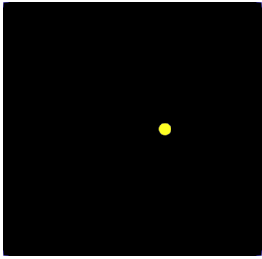
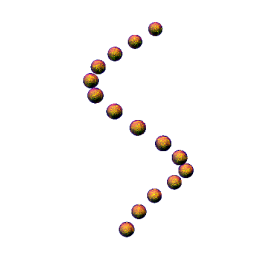
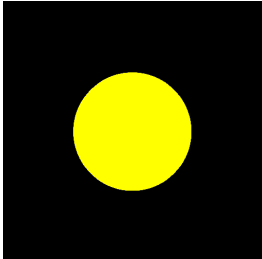
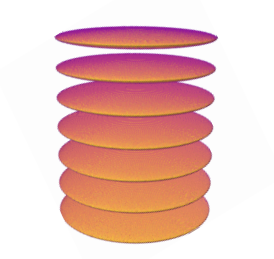
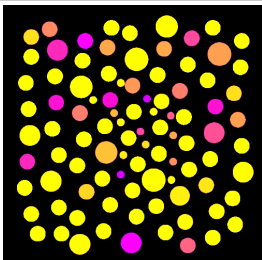
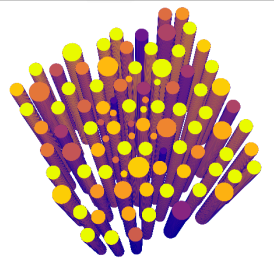
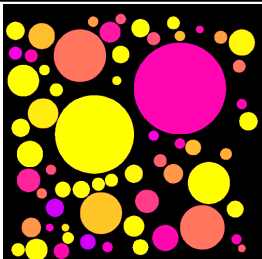
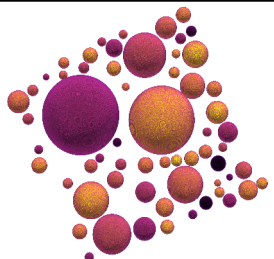
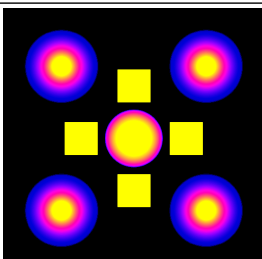
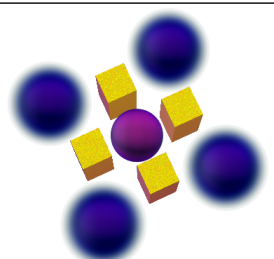
Model no.	Composites	Phantom, ax. slice	3D Phantom
1 'Snake'	15 ellipsoids		
2 'Defrise'	7 ellipsoids		
7	94 elliptical cylinders		
8	62 ellipsoids		
9	1 paraboloid 4 Gaussians 4 cuboids		

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

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 [11, 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.

## 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.

## Acknowledgements

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 A. 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.

231 *Appendix A.1. Elliptical Gaussian*

232 The function is given as:

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

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

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

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

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

237 *Appendix A.2. Parabola of  $\lambda$ -order with an elliptical support*

238 The function is given as:

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

239 The Radon transform of this function is:

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

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

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

243 **References**

- 244 [1] Kak AC, Slaney M. Principles of computerized tomographic imaging.  
 245 N.Y.: IEEE Press 1988.
- 246 [2] Shepp LA, Logan BF. The Fourier reconstruction of a head sec-  
 247 tion. IEEE Transactions on Nuclear Science 1974; 21(3): 21–43. DOI:  
 248 10.1109/TNS.1974.6499235
- 249 [3] Wernick MN, Aarsvold JN. Emission tomography: the fundamentals of  
 250 PET and SPECT: Elsevier 2004.

- [4] Balandin AL, Likhachev AV, Panferov NV, Pikalov VV, Rupasov AA, Shikanov AS. Tomographic diagnostics of radiating plasma objects. *Journal of Soviet Laser Research* 1992, 13(6):472–498. DOI: 10.1007/BF01120652
- [5] Arridge SR. Optical tomography in medical imaging. *Inverse Problems* 1999; 15(2): p.R41. DOI: 10.1088/0266-5611/15/2/022
- [6] Correia T, Aguirre J, Sisniega A, Chamorro-Servent J, Abascal J, Vaquero JJ, Desco M, Kolehmainen V, Arridge, S. Split operator method for fluorescence diffuse optical tomography using anisotropic diffusion regularisation with prior anatomical information. *Biomedical Optics Express* 2011; 2(9): 2632–2648. DOI: 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. *Measurement Science and Technology* 2017; 28(9): p. 094004. DOI: 10.1088/1361-6501
- [8] De Carlo F, Gürsoy D, Ching DJ, Batenburg KJ, Ludwig W, Mancini L, Marone F, Mokso R, Pelt DM, Sijbers J, Rivers M. TomoBank: A tomographic data repository for computational X-ray science. *Measurement Science and Technology* 2018; 29(3): p.034004. DOI: 10.1088/1361-6501/aa9c19
- [9] Jørgensen JS, Coban SB, Lionheart WR, McDonald SA, Withers PJ. SparseBeads data: benchmarking sparsity-regularized computed tomography. *Measurement Science and Technology* 2017; 28(12): p.124005. DOI: 10.1088/1361-6501/aa8c29
- [10] Gürsoy D, De Carlo F, Xiao X, Jacobsen C. TomoPy: a framework for the analysis of synchrotron tomographic data. *Journal of Synchrotron Radiation* 2014; 21(5): 1188–1193. DOI: /10.1107/S1600577514013939
- [11] Ching DJ, Gürsoy D. XDesign: an open-source software package for designing X-ray imaging phantoms and experiments. *Journal of Synchrotron Radiation* 2017; 24(2):537–544. DOI: 10.1107/S1600577517001928
- [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. *Journal of Synchrotron Radiation* 2017; 24(6): 1283–1295. DOI: 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. *Medical Physics* 1985; 12(2): 252–255. DOI: 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. DOI: 10.1016/j.cam.2005.09.027
- [17] Beister M, Kolditz D, Kalender WA. Iterative reconstruction methods in X-ray CT. *Physica Medica: European Journal of Medical Physics.* 2012; 28(2): 94–108. DOI: 10.1016/j.ejmp.2012.01.003
- [18] van Aarle W, Palenstijn WJ, Cant J, Janssens E, Bleichrodt F, Dabrovolski A, De Beenhouwer J, Batenburg KJ, Sijbers J. Fast and flexible X-ray tomography using the ASTRA toolbox. *Optics Express* 2016; 24(22): 25129–25147. DOI: 10.1364/OE.24.025129
- [19] Pikalov VV, Preobrazhenskii NG. Computer-aided tomography and physical experiment. *Soviet Physics-Uspekhi* 1983; 26(11):974–990. DOI: 10.1070/PU1983v026n11ABEH004538
- [20] Likhachov AV, Pikalov VV. Subpixel resolution in 3D cone - beam microtomography. *Nucl. Instrum. Meth. Phys. Res. (A)* 1995; 359(1-2):370–375. DOI: 10.1016/0168-9002(94)01677-1
- [21] Likhachov AV, Pikalov VV. Three-dimensional tomography with finite aperture beams. *Nucl. Instrum. Meth. Phys. Res. (A)* 1998; 405(2-3):506–510. DOI: 10.1016/S0168-9002(97)00178-2
- [22] Derevtsov EY, Pikalov VV. Reconstruction of vector fields and their singularities from ray transforms. *Numerical Analysis and Applications* 2011; 4(1):21–35. DOI: 10.1134/S1995423911010034
- [23] Kazantsev D, Pikalov V. New iterative reconstruction methods for fan-beam tomography. *Inverse Problems in Science and Engineering* 2018; 26(6): 773–791. DOI: 10.1080/17415977.2017.1340946

318 [24] Jiang Y, Padgett E, Hanwell MD, Quammen C, Harris C, Waldon  
319 S., Muller DA, Hovden R. tomviz: Providing Advanced Electron To-  
320 mography by Streamlining Alignment, Reconstruction, and 3D Visu-  
321 alization. Microscopy and Microanalysis 2017; 23(S1): 222–223. DOI:  
322 10.1017/S1431927617001799

### 323 **Current code version**

Nr.	Code metadata description	Please fill in this column
C1	Current code version	Version 1.1
C2	Permanent link to code/repository used for this code version	<a href="https://github.com/dkazanc/TomoPhantom">https://github.com/dkazanc/TomoPhantom</a>
C3	Legal Code License	Apache License v.2.0
C4	Code versioning system used	git
C5	Software code languages, tools, and services used	C, Python, Matlab
C6	Compilation requirements, operating environments	C compilers (GCC/MinGW), Cython; Linux, Windows, Mac OS
C7	If available Link to developer documentation/manual	For example: <a href="https://github.com/dkazanc/TomoPhantom/tree/master/docs">https://github.com/dkazanc/TomoPhantom/tree/master/docs</a>
C8	Support email for questions	dkazanc@hotmail.com

Table A.2: TomoPhantom v.1.1.1. package metadata