PT4 (PhanTom-4d) Software Overview

Stephen Catsamas[1], Glenn Myers[1], Andrew Kingston[1*]

[1]Department of Materials Physics, Research School of Physics, The Australian National University, Canberra ACT 2601, Australia
* Correspondence: andrew.kingston@anu.edu.au

Version 1.0.0

October 6, 2023

Contents

PT4 (PhanTom-4d) is an open-source software tool for generating phantoms. It is specialised for time-evolving (4D) phantoms. PT4 and its source code is available for download at https://github.com/MaterialsPhysicsANU/pt4. It was created to generate ground-truth datasets to assess the performance of dynamic-CT models and reconstruction methods. For static-CT, reconstruction methods are often compared against benchmark phantoms such as the Shepp-Logan [1] or Defrise phantom [2], however benchmark phantoms for 4D-CT are scarce and do not cover all types of dynamics. We therefore present a range of 4D phantoms: both to serve as these benchmarks, and also demonstrate the capabilities of PT4.

1 Phantom description

1.1 Overview

In PT4 phantoms are built by composing primitives (currently ellipsoids, cylinders, or cubeoids). Primitives are described by their location and attenuation whose parameters are stored in the `loc` and `atn` data structures. These parameters are piecewise defined over a set of contiguous time domains. On each time domain each parameter of `loc` and `atn` can be an arbitrary function of global time, `t`, and the time since the beginning of the current piecewise domain `dt`. This architecture allows concise descriptions of arbitrary continuous and discontinuous dynamics. Furthermore, it does not restrict the continuous dynamics to linear dynamics.

When composing primitives the attenuation of the new primitive is combined with the existing value in the phantom via:

$$\texttt{blend}\,(a, b, \texttt{blend\_mode}) = \begin{cases} a + b, & \texttt{blend\_mode} = \texttt{ADD} \\ a \times b, & \texttt{blend\_mode} = \texttt{MULTIPLY} \\ b, & \texttt{blend\_mode} = \texttt{REPLACE} \\ b \textbf{ if } b > 0 \textbf{ else } a, & \texttt{blend\_mode} = \texttt{MASK} \end{cases}. \quad (1)$$

Here a is the existing value, and b is the value from the new primitive. The blending modes (`ADD`, `MULTIPLY`, `REPLACE`, `MASK`) allow primitives to be combined in rich ways to create more detailed phantoms. For instance, `MASK` blending is used in fig. 8 to overlay a pore structure on fluid flow. As the time domains for the dynamics of each primitive may be different, if the current time is outside any of the domains of a primitive it is ignored.

Unlike existing phantom description software, in PT4 the attenuation need not be constant throughout the primitive. Instead, it may be an arbitrary function `atnf(x, y, z, s, t, dt)` where `x,y,z` are the coordinates in *texture space*, $V_T \cong \mathbb{R}^3$, and `s` is known as the fill variable, which can be set to $\texttt{s} = 1$ via `FILL_SOLID` or $\texttt{s} \sim \mathcal{U}_{[-1,1]}$, the continuous uniform distribution over each unit voxel of texture space via `FILL_NOISE`.

Thus, `atnf` allows for the creation of complex attenuation 'texture' without needing hundreds or thousands of constant-attenuation primitives. This texture emulates the spatial variation of attenuation seen in real world objects which are rarely homogeneous and uniform. It can be used to model, for instance, rock grains by using `FILL_NOISE` to give a random attenuation to each voxel (ie: grain) this is done in fig. 6. Adding object texture also becomes useful when analysing the effectiveness of dynamics models and dynamics reconstruction as it provides highly detailed structure which both the model and reconstructions should preserve.

Texture space, V_T, is transformed with the primitive. For example scaling the primitive along the x-axis by factor of 2 will also scale the texture space x-axis by 2. This emulates how texture transforms in a physical material.

We have defined coordinates relative to the centre of each unit primitive as *primative space*, notated $V_P \cong \mathbb{R}^3$. Primative space is then again transformed and embedded into *screen space*, $V_S \cong \mathbb{R}^3$, to get the position of the primitive in the final phantom volume. This process is shown in fig. 1. The transformation which takes points from V_S to V_P

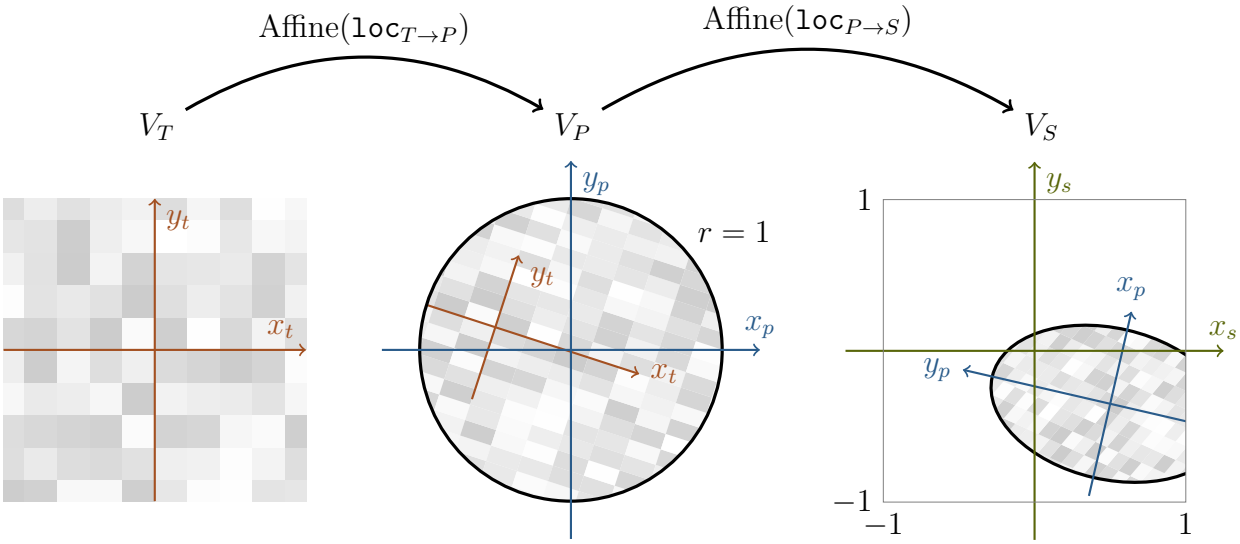


Figure 1: Transformation between texture space V_T, primative space V_P, and screen space V_S in PT4. The action of Affine is specified in eq. (2). The z dimension has been omitted for illustration.

and V_P to V_S are encoded in `loc` data structures. Each `loc` data structure has a position $\texttt{pos} = (\texttt{p}_x, \texttt{p}_y, \texttt{p}_z)$, scaling $\texttt{sma} = (\texttt{s}_x, \texttt{s}_y, \texttt{s}_z)$, a single axis-angle rotation in `axis`, `angle`, and acts as

$$\begin{bmatrix}\texttt{x}\\ \texttt{y}\\ \texttt{z}\end{bmatrix} \xmapsto{\text{Affine}(\texttt{loc})} \left(\begin{bmatrix} \\ R_{\texttt{axis,angle}} \\ \\ \end{bmatrix} \begin{bmatrix}\texttt{s}_x & 0 & 0\\ 0 & \texttt{s}_y & 0\\ 0 & 0 & \texttt{s}_z\end{bmatrix} \begin{bmatrix}\texttt{x}\\ \texttt{y}\\ \texttt{z}\end{bmatrix} \right) - \begin{bmatrix}\texttt{p}_x\\ \texttt{p}_y\\ \texttt{p}_z\end{bmatrix}, \tag{2}$$

where $R_{\texttt{axis,angle}}$ is the rotation matrix about `axis` by `angle`. This is not a general affine transformation - shears cannot be represented for instance - which is a current limitation of this approach. The region $[-1, 1]^3 \subset V_S$ of screen space forms the bounds of the rendered phantom. ie: $[-1, -1, -1]$ is the bottom-left-back edge of the phantom and $[1, 1, 1]$ is the top-right-front. Each primitive is has unit radius and centred in its primative space.

1.1.1 Metadata

Along with specifying the primitives, metadata can also be specified. These determine properties such as the render resolution, noise model, projection scheme, and projection integral. A full list of properties is found in table 1. The `projection_integrand` property of the metadata specifies whether projection should be done in `ATTENUATION` space, eq. (3), or `INTENSITY` space, eq. (4).

$$A = \int \mathrm{d}\boldsymbol{\ell}\ \mu(\boldsymbol{\ell}) \tag{3}$$

$$I = I_0 \mathrm{e}^{-A} = I_0 \mathrm{e}^{-\int \mathrm{d}\boldsymbol{\ell}\mu(\boldsymbol{\ell})} \tag{4}$$

Here $I_0 = \texttt{photon\_flux}$, the initial photon flux, I is the photon flux at the detector, and A is the integral value, μ the phantom's attenuation, and $\boldsymbol{\ell}$ is the X-ray path. A physical

property	value	description
`version`	⟨major⟩.⟨minor⟩.⟨patch⟩	`.pt4` file syntax version
`size_<x,y,z>`	natural number	phantom resolution in dimension `x,y,z`
`projection_integrand`	`ATTENUATION` (eq. (3)) `INTENSITY` (eq. (4))	integrand for projection integral
`noise_seed`	natural number	noise model random engine seed
`photon_flux`	floating point	photon count per projection per unit area of screen space
`noise_quantisation`	boolean	quantise photon count
`noise_possion`	boolean	apply Poisson noise to projection data
`noise_gaussian`	floating point	standard deviation in pixel photon count for projection data
`unit_time_per_volume`	floating point	time step between phantom volumes renders
`revolutions_per_unit_time`	floating point	phantom stage revolution frequency
`projections_per_revolution`	floating point	projections per revolution
`projection_supersampling_ratio`	natural number	resolution multiplier for projection rendering

Table 1: Description of `.pt4` file metadata values. For boolean values `0` is `false` and `1` is `true`.

CT scanner produces all projection data in intensity space, however attenuation space can be useful for debugging or for an intermediate output.

1.2 Implementation

1.2.1 Syntax

Users describe their phantoms in `.pt4` files. We have written a syntax for this phantom description and used GNU Bison [3] to generate the parser code for this syntax. An example `.pt4` file is given in fig. 2. This describes a cylindrical sample which is deformed under tensile strain and then fractures. The snippet shows cylinder `top`, the top fragment of the sample with domains `wait` where it is deforming, and `split` where it fractures and moves apart. This example omits the bottom fragment for brevity. The value of each numeric parameter described in section 1 is taken as an expression which is evaluated using the exprtk library [4]. The parameters of `loc` and `atn` can be expressions written in terms of the `t` and `dt` variables. The `atnf` parameter can additionally be written in terms of `x`, `y`, `z`, and `s`. All other parameters and metadata should be self-contained expressions. The expressions evaluated with exprtk are written with a C-like syntax and are Turing-

complete. This therefore allows parameters to be arbitrary functions. Parameters which are not specified in a domain are set equal to their value in the previous domain. All characters on a line after `//` are considered comments. `.pt4` files are first run through the simplecpp [5] C-preprocessor. This allows users to define macros, which are evaluated before the `.pt4` file is parsed. Macros can include simple text replacement functions via `#define`, conditional statements with `#if`, and inclusion of other `.pt4` files via the `#include` directive. A helpful idiom is to define the primitives in a separate `.pt4` file and `#include` these into multiple files containing the metadata. This reduces repetition and synchronises changes of the primitives between all files. Currently, primitive and domain names are used solely as user labels however in the future they could be used for higher order functions. For instance, inheriting the properties of one primitive onto another if only a few changes are needed.

1.2.2 Volume rendering

Volumes of the phantom are rendered beginning at `t` $= 0$ and at increments of `unit_time_per_volume` until the final time the phantom is defined. To render the volume at time $t =$ `t`

$$\mu_{v,t} = \sum_{\delta\mathbf{r}\in\mathcal{M}} \frac{\mu(\mathbf{r}+\delta\mathbf{r},t)}{|\mathcal{M}|} \tag{5}$$

is evaluated for each voxel, v, of the volume. Here μ is defined by algorithm 1, and $\mathcal{M}$ is a set of 8 uniform-psudorandomly chosen offsets $\delta\mathbf{r} = (\delta x, \delta y, \delta z)$ such that $\mathbf{r}+\delta\mathbf{r}$ still lies within the initial voxel. This averaging over multiple samples per voxel is known as super-sample anti-aliasing (SSAA) [6]. It reduces artefacts when a sudden change in μ occurs over the voxel. Each volume is computed with single-precision floating-point and saved as a netCDF (network common data form) [7] file - a format which allows saving three-dimentional (3D) arrays of values.

Algorithm 1 Definition of phantom attenuation μ. Note: for each primitive `s`, `dt` are implicit functions of $\mathbf{r}, t$. Hence μ is only a function of $\mathbf{r}, t$.

function $\mu(\mathbf{r}, t)$
  $\mu_c \leftarrow 0$
  for all primitive $\in$ Primatives **do**
    $\mu_p \leftarrow$ `atnf`$(\mathbf{r}_x, \mathbf{r}_y, \mathbf{r}_z,$ `s`$, t,$ `dt`$)$
    $\mu_c \leftarrow$ `blend`$(\mu_c, \mu_p,$ `blend_mode`$)$ ▷ eq. (1)
  end for
  return μ_c
end function

```
version  = 0.5.4;

#define SIZE 256

size_x = SIZE;
size_y = SIZE;
size_z = SIZE;

projection_integrand = INTENSITY;

noise_seed        = 0;              //
photon_flux       = 4096E6;         // photons/([0,1]^2)
noise_quanisation = 0;              //
noise_poisson     = 1;              //
noise_gaussian    = 25;             //counts per pixel

unit_time_per_volume           = 0.1;
revolutions_per_unit_time      = 0.5;
projections_per_revolution     = 200;
projection_supersampling_ratio = 3;

cylinder top = {
  domain wait = {
    .len = 1,
    .val = {
      .loc = {
        .pos = {0.0,0.0,0.4+0.02*t,},
        .sma = {0.6-0.01*t,0.6-0.01*t,0.4+0.02*t,},
        .axis = {0,0,1,},
        .angle = 0,
      },
      .atn = {
        .blend = ADD,
        .fill = FILL_NOISE,
        .atnf = 1+0.2*s,
        .loc = {
          .pos = {0,0,-5.0,},
          .sma = {0.1,0.1,0.1,},
          .axis = {0.1,-0.4,1.3,},
          .angle = 3.1,
        },
      },
    },
  },
  domain split = {
    .len = 1,
    .val = {
      .loc = {
        .pos = {0.0,0.0,0.42+0.1*dt*dt,},
        .sma = {0.59,0.59,0.42,},
        .axis = {0,0,1,},
        .angle = 0,
      },
    },
  },
};
```

Figure 2: A typical `.pt4` file which specifies (one half) of a cylinder deforming under tension and then fracturing (fig. 6)

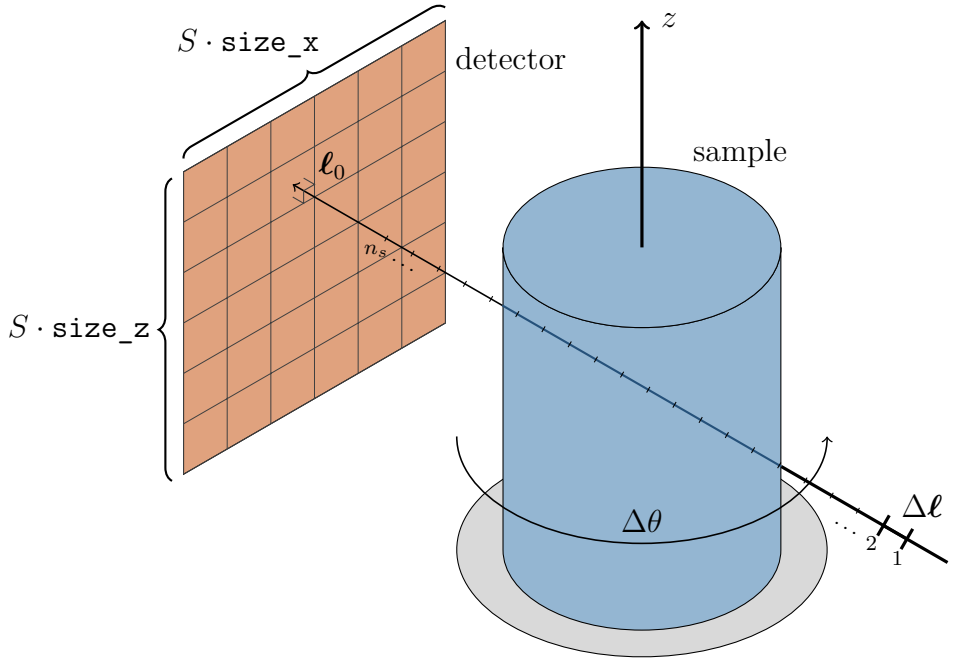


Figure 3: Projection geometry in PT4. The relative spacing of the detector, sample, and integration sampling ticks and the enumeration of the integration ticks have been modified for illustration.

1.2.3 Projection rendering

PT4 evaluates the projection integrals eqs. (3) and (4) on each pixel via a Riemann sum with fixed partition size:

$$\int \mathrm{d}\boldsymbol{\ell}\ \mu(\boldsymbol{\ell}, t) \to \sum_{n=1}^{n=n_s} \mu(\boldsymbol{\ell}_0 + n\Delta\boldsymbol{\ell})|\Delta\boldsymbol{\ell}|. \tag{6}$$

Here, n_s is the number of samples. The integration line $\boldsymbol{\ell}$ has $\boldsymbol{\ell}_0$ at the centre of the detector pixel and $\Delta\ell$ perpendicular to the detector. This achieves a parallel beam projection, however it would be straight forward to modify the integration line to achieve cone beam geometry. The projection is simulated at resolution of $(S\cdot\texttt{size\_x})\times(S\cdot\texttt{size\_z})$ and with $n_s = S \cdot \texttt{size\_y}$, for $S = \texttt{projection\_supersampling\_ratio}$. After rendering the projection is downsampled to a resolution of $\texttt{size\_x} \times \texttt{size\_y}$. Projections are taken with a fixed time step, Δt, and fixed stage angle increment, $\Delta\theta$, from $t = 0$ until the final time t_f the phantom is defined. Letting $f = \texttt{revolutions\_per\_unit\_time}$ and $n_p = \texttt{projections\_per\_revolution}$, then the time step and angle increment are given by:

$$\Delta t = \frac{1}{n_p t_f}, \tag{7}$$

$$\Delta\theta = \frac{2\pi}{n_p}. \tag{8}$$

This projection setup is summarised in fig. 3.

Projections are considered instantaneous and do not encompass mechanical errors such as slack of the stage and or settling time of the state after rotation.

The noise model (eq. (9)) is implemented as a simplified version of the noise model given in section 5 of [8]. The noise model can account for the Poisson shot noise of photon counts, Gaussian noise due to the combine effect of dark currents, readout noise, and electronic noise, and quantisation noise of detector readout. Importantly, we need to be aware that the noise model does not account for polychromatic X-ray effects (such as beam-hardening), physical X-ray effects (such as scattering, diffraction and refraction), detector efficiency, or source intensity variations. The noise model is applied to each pixel of the projection data and transforms the photon flux I to the detector output N under the equations:

$$\begin{aligned} N_n &\sim \mathcal{N}(0, \sigma^2) + \begin{cases} \text{Poisson}(I) & \text{if } p \\ I & \text{else} \end{cases} \\ N &= \begin{cases} \lfloor N_n \rfloor & \text{if } q \\ N_n & \text{else} \end{cases} \end{aligned} \tag{9}$$

Here $\sigma =$ `noise_gaussian`, $p =$ `noise_poisson`, and $q =$ `noise_quantisation`. The noise model is deterministic for an input `.pt4` file (as with all stochastic behaviour in PT4). Projections are also computed with single-precision floating points. They are saved as TIFF (tag image file format)[9] images which allows for storage of the exact floating point data representation.

1.2.4 Performance

One of the challenges when working in computed tomography is the massive datasets and computational requirements brought on by its 3D nature. This is exacerbated by the added time dimension in 4D-CT. We have therefore kept PT4's performance in mind lest it become an infeasible tool to use. To this end we have written PT4 in C++ and most routines are parallelised (run on multiple central processing unit (CPU) cores and multiple nodes simultaneously) using OpenMP [10]. Rendering and projection both have theoretical time complexity of $\mathcal{O}(N^3 \times M \times K)$, where $N^3 =$ `size_x` $\times$ `size_y` $\times$ `size_z`, K is the number of primitives, and M is (`projection_supersampling_ratio`)^3 for projection and the number of SSAA samples for volume generation. Due to the arbitrary nature of the primitive attenuation, we could not use faster analytic schemes for computing projection data. In table 2, we present volume rendering times at various resolutions. This data agrees closely with the theoretical N^3 behaviour (observed $N^{2.95}$, $R^2 > 0.999$) and demonstrates the feasibility of PT4 up to the thousands-of-voxels volume resolution.

All rendering in PT4 is done on the CPU. Due to the highly parallelisable nature of rendering (all pixel/voxel are independent), a large speedup could be obtained via moving the computation to the graphics processing unit. However this has not been implemented for the initial release of PT4.

2 Example phantoms

To demonstrate the capabilities of PT4 we will now give a few examples of phantoms which can be created. We will describe the types of dynamics that they exhibit and link

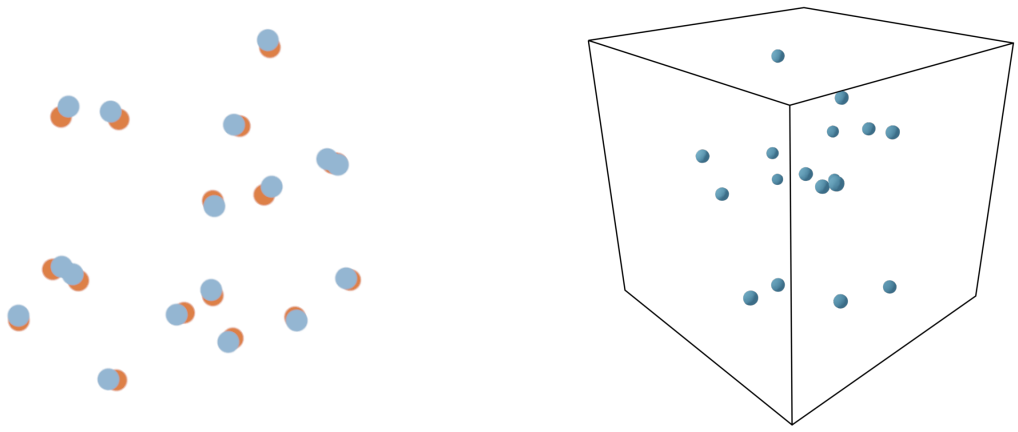
Resolution	128^3	256^3	512^3	1024^3
Time (s)	0.3	2.2	16.7	143

Table 2: Rending times for phantom composed of 10 primitives using an Intel(R) Core(TM) i9-13900HX

them to existing CT imaging research problems.

2.1 Spheres translating

The simplest example we present of a dynamic phantom is 'spheres translating' (fig. 4). This phantom consists of 16 spheres of equal and constant attenuation each moving independently. This phantom is designed to test dynamic models on smooth and continuous deformation. It can test whether motion is captured at a high enough resolution to distinguish the motion of each of the spheres independently. While this is a technical test, the reconstruction artefacts of moving spheres has been studied [11], and these spheres are also analogous to the gold fiducial markers which are used to align projection data and trace object motion [12–14].



(a) (b)

Figure 4: Visualisations of the spheres translating phantom. (a) z-axis projection of spheres phantom at $t = 0$ (light blue) superimposed on $t = 1$ (orange). (b) Perspective 3D rendering of spheres phantom. This subfigure was created in-part using Drishti [15].

2.2 Bread baking

The bread baking phantom (fig. 5) takes inspiration from a loaf of bread rising during baking. Throughout its evolution it expands, while voids appear that also expand. In its final state the phantom has seven voids. To emulate conservation of mass, the phantom reduces in density as it expands. This manifests as phantom's main ellipsoid decreasing in attenuation proportionally to its volume ($\mu(t) = \mu(0)\frac{V(t)}{V(0)}$, where $\mu(t)$ and $V(t)$ are the attenuation and volume at time t). This phantom tests not only the continuous deformation from the expanding spheres and voids, but also spontaneous dynamics when voids appear. Research into problems such as the aeration of gluten-free bread [16] and fabrication of metal foam materials [17] image similar systems.

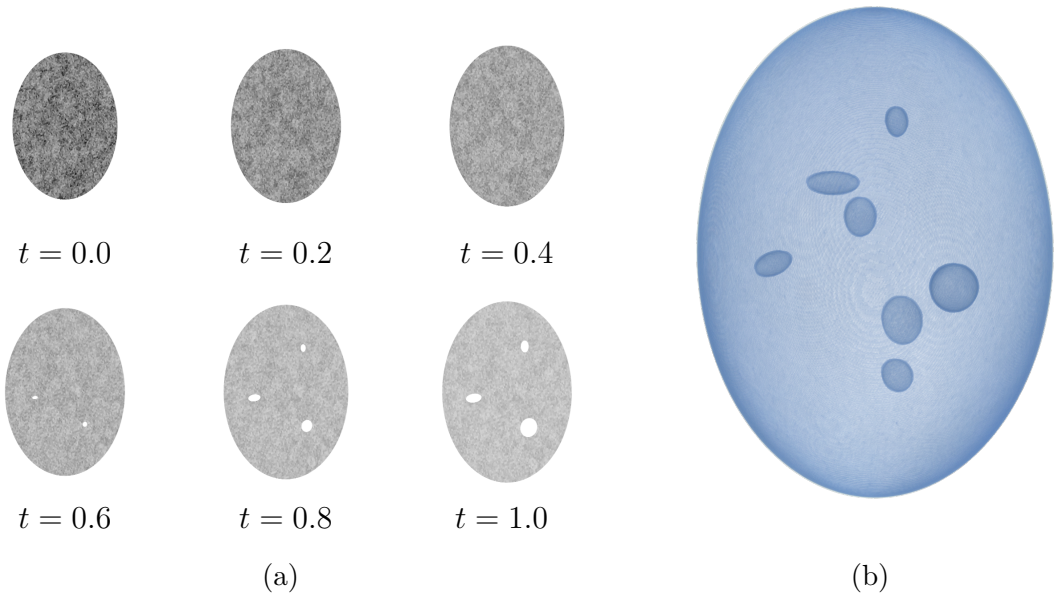


Figure 5: Visualisations of the bread baking phantom. (a) Time series of central zx-plane. (b) 3D rendering with internal voids visible. This subfigure was made in Drishti [15].

2.3 Tensile failure

The tensile failure phantom (fig. 6) emulates a sample undergoing a tensile test to failure. This phantom exhibit both motion and fracturing dynamics. In the first half of the test, the sample stretches vertically and contracts in radius such that its volume remains constant. At $t = 0.5$, the sample fractures and the two halves begin to move away from each other. While the fracture is a clear horizontal plane which may not be the case for real materials, any dynamics model which captures this fracture will need to identify the two rigid body components and the appearance of a void between them. This phantom has obvious relevance as CT is commonly applied to image samples, before during and after tensile tests [18–21].

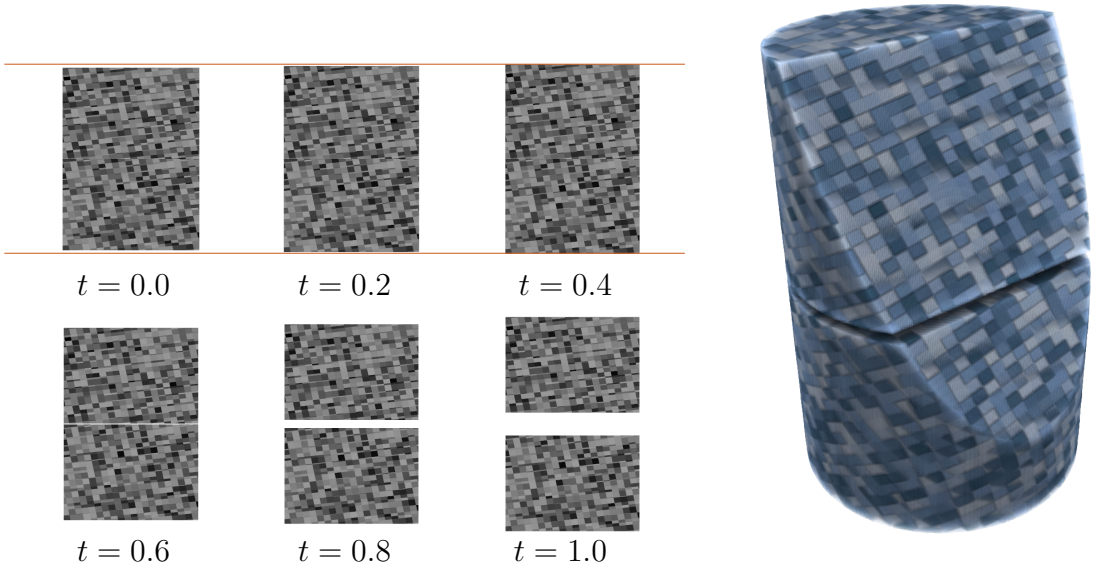


Figure 6: Visualisations of the tensile failure phantom. (a) Timeseries of the central zx-plane. The orange line emphases the slight vertical expansion of the sample before fracture. (b) 3D rendering of phantom with cutaway. This subfigure was made in Drishti [15].)

2.4 Brazil crush

The Brazil crush phantom (fig. 7) represents a sample undergoing the Brazilian test. In this test a cylindrical sample is placed between two jaws which apply compressive load to the sample. This results in a tensile load on the sample that increases until fracture [22]. In the phantom, multiple fractures appear and close up. For simplicity, the jaws have been modelled nonphysically as they form a contiguous piece of material, however the contact points between the jaws and the sample still close with time. Like tensile testing, CT is commonly used for imaging Brazilian crush test, for instance in the study of crack formation [23].

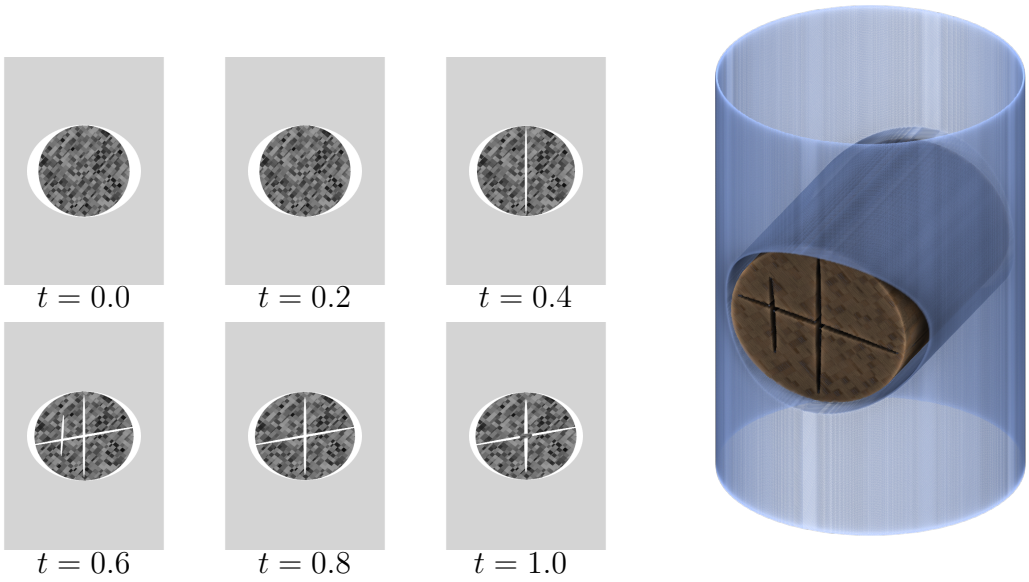
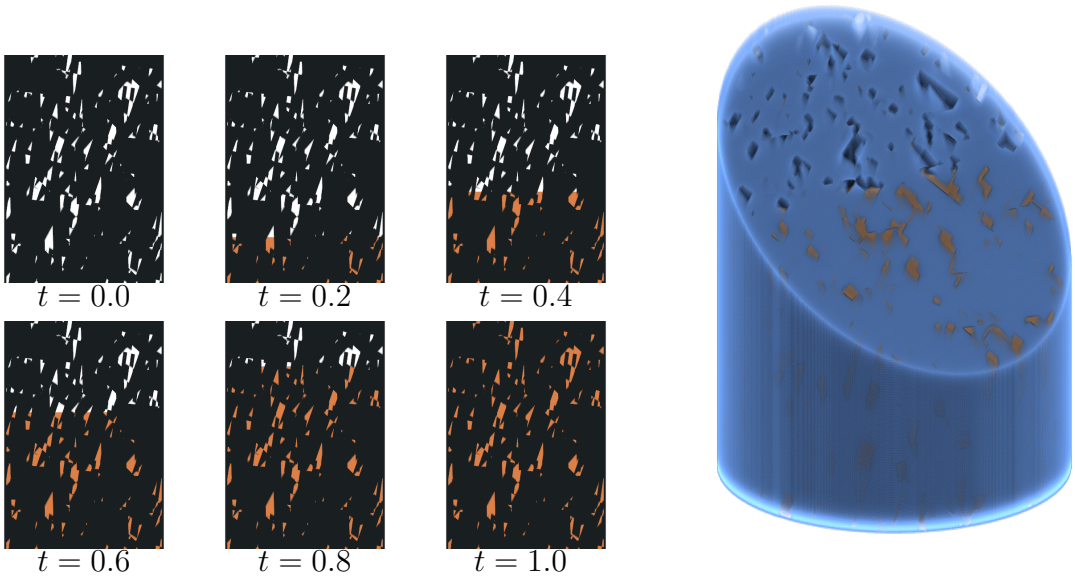


Figure 7: Visualisation of the Brazil crush phantom. (a) Timeseries of the central zx-plane. (b) 3D render of phantom with jaws shown in blue and sample in orange. This subfigure was made in Drishti [15].

2.5 Fluid flow

The fluid flow phantom (fig. 8) gives an example of fluid flowing through a porous medium. This has been implemented as the fluid level rising uniformly in time inside the porous medium. Again, while this is nonphysical, any dynamic imaging method which works on fluids should capture arbitrary appearance and removal of fluid anywhere in the tomogram. Additionally, the sudden appearance of fluid in empty voids is somewhat representative of the Haines jumps phenomena [24]. Imaging of such fluid flow is of interest to applications such as: understanding solute transport in agricultural soils [25], carbon dioxide sequestration and, oil and gas exploration [26, 27], or the recovery of water stressed plants [28].



(a) (b)

Figure 8: Visualisation of the fluid flow phantom. (a) Timeseries of the central zx-plane, the fluid has been highlighted in orange for emphasis. (b) 3D render of phantom with porous medium shown in blue and fluid in orange. This subfigure was made in Drishti [15].

3 Similar projects

Below are a list of similar projects for phantom generation :

- syris (https://github.com/ufo-kit/syris/tree/master) [29].
- TomoPhantom (https://github.com/dkazanc/TomoPhantom) [30].
- XDesign (https://github.com/AdvancedPhotonSource/xdesign) [31].

References

[1] L. A. Shepp and B. F. Logan. "The Fourier Reconstruction of a Head Section". *IEEE Transactions on Nuclear Science* 21.3 (1974), pp. 21–43. DOI: 10.1109/TNS.1974.6499235. (Visited on 09/24/2023).

[2] Hiroyuki Kudo, Frédéric Noo, and Michel Defrise. "Cone-Beam Filtered-Backprojection Algorithm for Truncated Helical Data". *Physics in Medicine and Biology* 43.10 (1998), pp. 2885–2909. DOI: 10.1088/0031-9155/43/10/016. (Visited on 09/24/2023).

[3] Robert Corbett and Richard Stallman. *Bison*. GNU. 2021. URL: https://www.gnu.org/software/bison/ (visited on 09/20/2023).

[4] Arash Partow. *C++ Mathematical Expression Library (ExprTk)*. 2023. URL: http://www.partow.net/programming/exprtk/ (visited on 09/20/2023).

[5] Daniel Marjamäki. *Simple C/C++ Preprocessor*. 2023. URL: https://github.com/danmar/simplecpp (visited on 09/20/2023).

[6] A. Mammen. "Transparency and Antialiasing Algorithms Implemented with the Virtual Pixel Maps Technique". *IEEE Computer Graphics and Applications* 9.4 (1989), pp. 43–55. DOI: 10.1109/38.31463. (Visited on 09/24/2023).

[7] Russ Rew, Glenn Davis, Steve Emmerson, Cathy Cormack, John Caron, Robert Pincus, Ed Hartnett, Dennis Heimbigner, et al. *Unidata NetCDF*. 1989. DOI: 10.5065/D6H70CW6.

[8] Johan Nuyts, Bruno De Man, Jeffrey A Fessler, Wojciech Zbijewski, and Freek J Beekman. "Modelling the Physics in the Iterative Reconstruction for Transmission Computed Tomography". *Physics in Medicine and Biology* 58.12 (2013), R63–R96. DOI: 10.1088/0031-9155/58/12/R63. (Visited on 03/05/2023).

[9] Adobe Developers Association. *TIFF (Tagged Image File Format), Revision 6.0*. 1992. URL: https://developer.adobe.com/content/dam/udp/en/open/standards/tiff/TIFF6.pdf (visited on 09/29/2023).

[10] OpenMP Architecture Review Board. *OpenMP Application Programming Interface Specification Version 5.0*. Ed. by Michael Klemm and Bronis R. Supinski. Independently Publishing: Independently published, 2019. URL: https://www.openmp.org/wp-content/uploads/OpenMP-API-Specification-5.0.pdf.

[11] George T. Y Chen, Jong H Kung, and Kevin P Beaudette. "Artifacts in Computed Tomography Scanning of Moving Objects". *Seminars in Radiation Oncology*. High-Precision Radiation Therapy of Moving Targets 14.1 (2004), pp. 19–26. DOI: 10.1053/j.semradonc.2003.10.004. (Visited on 09/22/2023).

[12] Yongbin Zhang, Lifei Zhang, X. Ronald Zhu, Andrew K. Lee, Mark Chambers, and Lei Dong. "Reducing Metal Artifacts in Cone-Beam CT Images by Preprocessing Projection Data". *International Journal of Radiation Oncology*Biology*Physics* 67.3 (2007), pp. 924–932. DOI: 10.1016/j.ijrobp.2006.09.045. (Visited on 09/24/2023).

[13] Jennifer C. Fung, Weiping Liu, W. J. de Ruijter, Hans Chen, Craig K. Abbey, John W. Sedat, and David A. Agard. "Toward Fully Automated High-Resolution Electron Tomography". *Journal of Structural Biology* 116.1 (1996), pp. 181–189. DOI: 10.1006/jsbi.1996.0029. (Visited on 09/24/2023).

[14] Renmin Han, Liansan Wang, Zhiyong Liu, Fei Sun, and Fa Zhang. "A Novel Fully Automatic Scheme for Fiducial Marker-Based Alignment in Electron Tomography". *Journal of Structural Biology* 192.3 (2015), pp. 403–417. DOI: 10.1016/j.jsb.2015.09.022. (Visited on 09/24/2023).

[15] Yuzhi Hu, Ajay Limaye, and Jing Lu. "Three-Dimensional Segmentation of Computed Tomography Data Using Drishti Paint: New Tools and Developments". *Royal Society Open Science* 7.12 (2020), p. 201033. DOI: 10.1098/rsos.201033. (Visited on 09/22/2023).

[16] Dana Elgeti, Mario Jekle, and Thomas Becker. "Strategies for the Aeration of Gluten-Free Bread A Review". *Trends in Food Science & Technology* 46.1 (2015), pp. 75–84. DOI: 10.1016/j.tifs.2015.07.010. (Visited on 09/24/2023).

[17] L. Helfen, T. Baumbach, H. Stanzick, J. Banhart, A. Elmoutaouakkil, and P. Cloetens. "Viewing the Early Stage of Metal Foam Formation by Computed Tomography Using Synchrotron Radiation". *Advanced Engineering Materials* 4.10 (2002), pp. 808–813. DOI: 10.1002/1527-2648(20021014)4:10<808::AID-ADEM808>3.0.CO;2-U. (Visited on 09/24/2023).

[18] Eric Maire, Christophe Le Bourlot, Jérôme Adrien, Andreas Mortensen, and Rajmund Mokso. "20 Hz X-ray Tomography during an in Situ Tensile Test". *International Journal of Fracture* 200.1-2 (2016), pp. 3–12. DOI: 10.1007/s10704-016-0077-y. (Visited on 09/24/2023).

[19] Eric Maire, Vincent Carmona, Joel Courbon, and Wolfgang Ludwig. "Fast X-ray Tomography and Acoustic Emission Study of Damage in Metals during Continuous Tensile Tests". *Acta Materialia* 55.20 (2007), pp. 6806–6815. DOI: 10.1016/j.actamat.2007.08.043. (Visited on 09/24/2023).

[20] Le Zhang, Faning Dang, Weihua Ding, and Lin Zhu. "Comparative Study on Damage Process of Concrete Subjected to Uniaxial Tensile and Compression Loads Based on CT Test and Improved Differential Box Counting Method". *Construction and Building Materials* 285 (2021), p. 122693. DOI: 10.1016/j.conbuildmat.2021.122693. (Visited on 09/24/2023).

[21] Clément Jailin, Ante Buljac, Amine Bouterf, François Hild, and Stéphane Roux. "Fast Four-Dimensional Tensile Test Monitored via X-ray Computed Tomography: Elastoplastic Identification from Radiographs". *The Journal of Strain Analysis for Engineering Design* 54.1 (2019), pp. 44–53. DOI: 10.1177/0309324718810593. (Visited on 09/24/2023).

[22] F.G. Bell. "ENGINEERING GEOLOGY | Rock Properties and Their Assessment". In: *Encyclopedia of Geology*. Elsevier, 2005, pp. 566–580. DOI: 10.1016/B0-12-369396-9/00211-2. (Visited on 09/24/2023).

[23] Yongtao Gao, Yang Peng, Yu Zhou, and Yu Wang. "Meso-Mechanism Research on Mechanical Properties of Rock-like with Two Fissures in Brazilian Test". *Indian Geotechnical Journal* (2023). DOI: 10.1007/s40098-023-00760-1. (Visited on 09/24/2023).

[24] William B. Haines. "Studies in the Physical Properties of Soil. V. The Hysteresis Effect in Capillary Properties, and the Modes of Moisture Distribution Associated Therewith". *The Journal of Agricultural Science* 20.1 (1930), pp. 97–116. DOI: 10.1017/S002185960008864X. (Visited on 09/24/2023).

[25] Sheela Katuwal, Trine Norgaard, Per Moldrup, Mathieu Lamandé, Dorthe Wildenschild, and Lis W. de Jonge. "Linking Air and Water Transport in Intact Soils to Macropore Characteristics Inferred from X-ray Computed Tomography". *Geoderma* 237–238 (2015), pp. 9–20. DOI: 10.1016/j.geoderma.2014.08.006. (Visited on 09/24/2023).

[26] Gidon Han, Weon Shik Han, Kue-Young Kim, Johyun Baek, Minji Kim, Chan Yeong Kim, and Jae-Hong Lim. "Characterizing Locality- and Scale-Dependent Heterogeneity in Conglomerate Core and Associated Fluid Flow Using X-ray CT Imaging". *Journal of Hydrology* 602 (2021), p. 126736. DOI: 10.1016/j.jhydrol.2021.126736. (Visited on 09/24/2023).

[27] Yang Su, Ming Zha, Lin Jiang, Xiujian Ding, Jiangxiu Qu, Jiehua Jin, and Stefan Iglauer. "Pore Structure and Fluid Distribution of Tight Sandstone by the Combined Use of SEM, MICP and X-ray Micro-CT". *Journal of Petroleum Science and Engineering* 208 (2022), p. 109241. DOI: 10.1016/j.petrol.2021.109241. (Visited on 09/24/2023).

[28] Tomás I. Fuenzalida, Matthew J. Blacker, Michael Turner, Adrian Sheppard, and Marilyn C. Ball. "Foliar Water Uptake Enables Embolism Removal in Excised Twigs of Avicennia Marina". *New Phytologist* 237.4 (2023), pp. 1136–1145. DOI: 10.1111/nph.18613. (Visited on 04/16/2023).

[29] T. Faragó, P. Mikulík, A. Ershov, M. Vogelgesang, D. Hänschke, and T. Baumbach. "Syris: A Flexible and Efficient Framework for X-ray Imaging Experiments Simulation". *Journal of Synchrotron Radiation* 24.6 (2017), pp. 1283–1295. DOI: 10.1107/S1600577517012255. URL: http://scripts.iucr.org/cgi-bin/paper?ay5502 (visited on 09/10/2023).

[30] Daniil Kazantsev, Valery Pickalov, Srikanth Nagella, Edoardo Pasca, and Philip J. Withers. "TomoPhantom, a Software Package to Generate 2D4D Analytical Phantoms for CT Image Reconstruction Algorithm Benchmarks". *SoftwareX* 7 (2018), pp. 150–155. DOI: 10.1016/j.softx.2018.05.003. (Visited on 08/19/2023).

[31] D. J. Ching and D. Gürsoy. "XDesign: An Open-Source Software Package for Designing X-ray Imaging Phantoms and Experiments". *Journal of Synchrotron Radiation* 24.2 (2017), pp. 537–544. DOI: 10.1107/S1600577517001928. URL: http://scripts.iucr.org/cgi-bin/paper?pp5098 (visited on 09/10/2023).