Research Article Vol. 32, No. 3 / 29 Jan 2024 / Optics Express 3425
Optics EXPRESS

CAD-ASTRA: a versatile and efficient mesh projector for X-ray tomography with the ASTRA-toolbox

P. PARAMONOV,* N. FRANCKEN, J. RENDERS, D. IUSO, T. ELBERFELD, J. DE BEENHOUWER, AND J. SIJBERS

Imec-Vision Lab, Department of Physics, University of Antwerp, Universiteitsplein 1, Building N, Antwerp, B- 2610, Belgium
*pavel.paramonov@uantwerpen.be

Abstract: Accurate and fast simulation of X-ray projection data from mesh models has many applications in academia and industry, ranging from 3D X-ray computed tomography (XCT) reconstruction algorithms to radiograph-based object inspection and quality control. While software tools for the simulation of X-ray projection data from mesh models are available, they lack either performance, public availability, flexibility to implement non-standard scanning geometries, or easy integration with existing 3D XCT software. In this paper, we propose CAD-ASTRA, a highly versatile toolbox for fast simulation of X-ray projection data from mesh models. While fully functional as standalone software, it is also compatible with the ASTRA toolbox, an open-source toolbox for flexible tomographic reconstruction. CAD-ASTRA provides three specialized GPU projectors based on state-of-the-art algorithms for 3D rendering, implemented using the NVIDIA CUDA Toolkit and the OptiX engine. First, it enables X-ray phase contrast simulations by modeling refraction through ray tracing. Second, it allows the back-propagation of projective errors to mesh vertices, enabling immediate application in mesh reconstruction, deep learning, and other optimization routines. Finally, CAD-ASTRA allows simulation of polychromatic X-ray projections from heterogeneous objects with a source of finite focal spot size. Use cases on a CAD-based inspection task, a phase contrast experiment, a combined mesh-volumetric data projection, and a mesh reconstruction demonstrate the wide applicability of CAD-ASTRA.

© 2024 Optica Publishing Group under the terms of the Optica Open Access Publishing Agreement

1. Introduction

X-ray computed tomography (XCT) is an imaging technique for scanning objects from multiple angles with X-rays, enabling the reconstruction of cross-sectional images for diverse applications in industrial inspection, materials science, and academic research [1,2]. In XCT, objects are typically represented by 3D images composed of uniformly-sized voxels arranged regularly. While voxel volumes allow for efficient modeling of density variation, they often require a large amount of memory for storage, especially for large grids or grids with high resolution. Alternatively, an object's geometry can be defined by a surface mesh, which accurately represents the boundaries between material types. While a surface mesh representation assumes a uniform density between surfaces, it can be more memory-efficient. Moreover, it suffers less from partial volume effects, which is inherent to voxel grids.

The use of surface meshes in XCT is, for example, highly relevant in the manufacturing industry, where XCT is an essential tool for non-destructive testing (NDT) [3–5]. The conventional approach to X-ray-based NDT is to extract the surface mesh from a reconstructed voxel-based XCT image of the scanned object and compare it with a reference surface mesh from a computer-aided design (CAD) model. However, this is a computationally expensive and time-consuming procedure. Moreover, as the surface mesh is extracted from a reconstructed image, it is vulnerable

#498194 https://doi.org/10.1364/OE.498194
Journal © 2024 Received 16 Jun 2023; revised 29 Nov 2023; accepted 6 Dec 2023; published 18 Jan 2024

Optics EXPRESS

to image reconstruction artifacts. Alternatively, X-ray-based NDT can be performed by directly comparing simulated radiographs of the CAD model's surface mesh with measured radiographs of the real object [6,7]. However, such a comparison requires fast mesh projection simulators.

Simulating projections directly from a surface mesh offers a number of additional advantages over simulating these from a voxel-based sample, including a simplified sample representation, faster ray tracing, simpler parameterization, and a reduced solution space in optimization problems. Additionally, such a simulation can be implemented as a differentiable program, which enables the use of gradient-based optimization and back-propagation calculations in deep learning contexts. XCT extensions, such as phase contrast XCT or 4DCT, may also benefit from a mesh projection simulation tool. For instance, X-ray phase contrast imaging (XPCI) simulations need to include the X-ray beam phase shift, which is often modeled as X-ray refraction [8]. Refraction occurs at material interfaces; therefore a sample can be accurately and efficiently described by a collection of surface meshes, where the mesh faces represent the material interfaces. Another example is 4DCT setups [9], where the sample undergoes substantial changes over the scanning time, allowing for a time-dependent reconstruction to provide insights into dynamic processes. In 4DCT, a surface mesh sample representation may be preferred over a voxel representation, as mesh deformation is a much simpler process than voxel grid deformation.

An efficient tool for simulating forward projections and performing tomographic reconstruction is the ASTRA-toolbox, an open source XCT software package [10]. Its flexible approach to defining scanning geometries allows it to be applied to conventional circular XCT as well as non-standard scanning schemes, such as helical XCT, laminography, or cycloidal XCT. However, its application for tasks involving a surface mesh is currently limited, because ASTRA only supports voxel data. While this limitation could be overcome by discretizing the mesh surface (as proposed, e.g., in [6]), such a work-around would introduce significant overhead. Hence, dedicated and efficient mesh projectors that can be easily integrated into a reconstruction toolbox would be valuable.

Several open source tools to simulate radiographs from surface mesh models have been described in the literature. McXtrace allows for simulating X-ray imaging through ray tracing, using a Monte Carlo method [11]. Starting from version 3.0, it supports GPU-acceleration through the OpenACC programming model. The Syris framework, described in [12], is implemented using the OpenCL framework, and allows for realistic simulations of X-ray radiographs by modeling the wavefield propagation. While these software packages provide high performance tools for simulating X-ray radiographs from 3D meshes, they lack generality and flexibility in the definition of a scanning setup. Furthermore, they only simulate forward projections and do not integrate easily with existing XCT reconstruction software. These drawbacks hinder their efficient application in areas such as X-ray-based NDT.

In this paper, we propose a toolbox for the simulation of projections of surface meshes that is fully compatible with the ASTRA toolbox, facilitating simultaneous use of forward mesh model projections with tomographic reconstruction in NDT tasks. Our work builds on the projector presented in [13], which provided a basic implementation of an ASTRA-compatible mesh projector. Unlike its predecessor, our toolbox provides various projection algorithms implemented with CUDA [14] and OptiX [15], and supports the same generic definition of the scanning geometry as the ASTRA toolbox. Additionally, it includes features such as ray refraction, mesh transformations, and the vector-Jacobian product calculation. The vector-Jacobian product calculation is particularly useful for optimization techniques, as in deep learning frameworks. While the proposed toolbox is complementary to ASTRA, it can also be used as a standalone package through its Python interface. An open-source version of CAD-ASTRA can be found at [16].

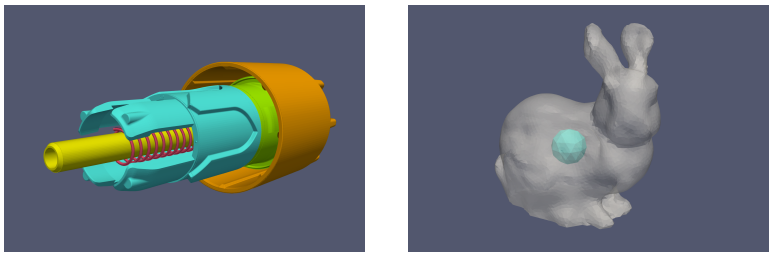
Optics EXPRESS

2. Methods

This section provides an overview of the imaging model used in CAD-ASTRA, which is based on the principle of ray optics. Firstly, general considerations on object modeling and definitions of a forward model and its differential are given. Next, two forward projection operators are described, one of which is implemented as a differentiable program. Then, options for defining realistic X-ray scanning geometry are presented. Finally, the implementation details of the projection algorithms are given, along with code snippets that demonstrate a typical simulation pipeline.

2.1. Object modeling with meshes

In CAD-ASTRA, watertight triangular surface meshes represent sets of interfaces that enclose volumes of homogeneous materials. Two primary configurations exist for modeling real-world objects: the assembly configuration where each mesh represents a separate part, and the composite configuration where each mesh delineates a homogeneous region within a single part (cfr. Fig. 1). The optical properties of the represented volume are linked with each mesh through two key parameters: the linear attenuation coefficient, which describes the intensity decay of light as it traverses the material, and the refractive index, which indicates the degree to which incident light undergoes deflection upon entering the material.



(a) (b)

Fig. 1. (a) A CAD model illustrating an assembly of distinct components. (b) A composite configuration that represents a Stanford bunny with a spherical cavity.

2.2. Mesh projector as a differentiable program

The non-linear projection operator Proj, which maps the vertex coordinates to the projection pixels, generates k projections from a set of meshes with a total number of vertices v to a virtual detector with $m \times n$ pixels [17]:

$$\text{Proj}_d : \mathbb{R}^{v\times 3} \rightarrow \mathbb{R}^{m\times n\times k}, \tag{1}$$

with d the metadata that includes vertices connectivity and material optical properties. The utilization of Proj_d in gradient-based optimization requires the knowledge of its differential, or Jacobian, which is a linear operator with the same domain and codomain [17]:

$$\text{Jac}_d : \mathbb{R}^{v\times 3} \rightarrow \mathbb{R}^{m\times n\times k}. \tag{2}$$

The adjoint of Jac_d is part of the gradient of an objective function of the form $f(\mathbf{x}) = g\left(\text{Proj}(\mathbf{x})\right)$:

$$\nabla f(\mathbf{x}) = \text{Jac}_d^T(\mathbf{x}) \nabla g\left(\text{Proj}_d(\mathbf{x})\right), \tag{3}$$

with $\mathbf{x} \in \mathbb{R}^{v\times 3}$ the mesh vertices, and $g\left(\text{Proj}_d(\mathbf{x})\right)$ a user-defined function of the mesh projection data, e.g., two-norm. As the matrix representation of $\text{Jac}_d(\mathbf{x})$ has $v \times 3$ columns and $m \times n \times k$

Optics EXPRESS

rows, it is often too large to be stored in memory. In practice, however, direct access to the Jacobian matrix is not required, since Eq. (3) can be directly used in the optimization routines. The implementation of Proj_d along with Eq. (3) for a specific objective function allows the use of such a mesh projector as a differentiable program, enabling a wide range of applications which rely on parameter tuning and automatic differentiation, such as deep learning.

2.3. Forward projection models

The Beer-Lambert law postulates that the intensity of an X-ray beam decreases exponentially as it travels through a homogeneous medium:

$$I = I_0 \exp(-\mu l), \tag{4}$$

where I is the intensity after passing through the material, I_0 is the initial intensity, μ is the linear attenuation coefficient, and l is the thickness of the material. Based on Eq. (4), the absorbance A of the X-ray beam is defined as follows:

$$A = -\log\left(\frac{I}{I_0}\right) = \mu l. \tag{5}$$

This linear relationship between absorbance, material thickness, and the attenuation coefficient, allows simulation of the X-ray projections by calculating the distance l that each ray $R \subset \mathbb{R}^3$ traverses through the mesh, starting at the source $\mathbf{s} \in \mathbb{R}^3$ [13,18].

CAD-ASTRA includes the implementation of two projection operators that exploit the reduction of the problem to distance calculations. The operator Proj_d^A creates projections by simulating the propagation of rays along straight lines and assigning absorbance values to the corresponding detector pixels. To illustrate this for a single detector pixel, consider a ray that passes through the center of the corresponding detector pixel with position $\mathbf{d}$, such that $R = \{\mathbf{s} + t\mathbf{d} | t{>}0\}$, propagating through a single mesh, which it enters through the face i and leaves through the face k. The projection value p is then calculated as the ray path difference between the faces i and k multiplied by μ:

$$p = \mu \sum_{r \in \{i,k\}} \text{sign}(\mathbf{n}_r^{\mathrm{T}} \mathbf{d}) t_r, \tag{6}$$

with $\mathbf{n}_r$ the outward normal of the face r, and t_r the distance from the source to the r-th face. Note that, since $\mathbf{n}_i^{\mathrm{T}}\mathbf{d}$ and $\mathbf{n}_k^{\mathrm{T}}\mathbf{d}$ have opposite signs, the sum in Eq. (6) will result in the l in Eq. (5). The extension of Eq. (6) to the case when a ray passes the mesh multiple times is straightforward, as it merely requires the inclusion of all intersected faces in the set $\{i, k\}$.

Although the non-linear operator Proj_d^A is not differentiable everywhere, it is possible to compute its piecewise derivative [17,19]. As explained in Section 2.2, Proj_d^A can be used as a differentiable program when the computation of the gradient of the some objective function is implemented. CAD-ASTRA includes such an implementation for the following objective function:

$$f(\mathbf{x}) = \frac{1}{2} \| \text{Proj}_d^A(\mathbf{x}) - \mathbf{b} \|_2^2, \tag{7}$$

with piecewise gradient [17]:

$$\nabla f(\mathbf{x}) = \text{Jac}_d^T(\mathbf{x}) \left(\text{Proj}_d^A(\mathbf{x}) - \mathbf{b} \right), \tag{8}$$

with $\mathbf{x} \in \mathbb{R}^{v \times 3}$ the vertices and $\mathbf{b} \in \mathbb{R}^{m \times n \times k}$ the reference projection images. The application of Proj_d^A as a differentiable program for mesh reconstruction from a tomography scan will be demonstrated in Section 3.2.4.

Optics EXPRESS

Alternatively, the projection operator Proj_d^I generates projection images that approximate the beam intensity after passing through the sample using Eq. (4). Unlike Proj_d^A, here users have the flexibility to define a finer volumetric sampling of the space by customizing the number of traced rays. These rays are pointing to equidistantly distributed endpoints laying on the detector plane. When multiple rays are generated per pixel, their intensities are mapped to the pixel values. To that end, the simulated intensity of each ray is distributed amongst the four neighboring pixels, with the pixel having the center closest to the hit point receiving the majority of the intensity. Additionally, Proj_d^I allows simulation of the beam refraction by executing ray tracing in a recursive manner. This approach allows for the tracing of a ray's path through multiple transitions between the materials by initiating new traces recursively. However, the incorporation of refraction in the context of transmissive optical rays presents practical challenges for efficient implementation as a differentiable program. Although theoretically feasible, it is particularly complex due to the interdependence between mesh faces during gradient accumulation. This complexity arises from the fact that alterations in vertex positions directly influence the ray path, thereby complicating the calculations. For this reason, we opted not to implement Proj_d^I as a differentiable program at this time, though this may be reconsidered in the future.

In addition to straightforward projection simulations with the aforementioned operators, more sophisticated image generation models can be developed, leveraging either Proj_d^A or Proj_d^I. Firstly, since the majority of X-ray detectors are energy-integrating and conventional X-ray sources have a broad energy spectrum, a polychromatic adaptation of the Beer-Lambert law is required to model the attenuation of X-rays arriving on the detector. For a polychromatic X-ray source discretized over E energy levels, the intensity of a single ray I, measured after its propagation through a sample consisting of K materials, is given by

$$I = \sum_{e=1}^{E} s(\epsilon_e) \exp\left(-\sum_{k=1}^{K} \mu_k(\epsilon_e) l_k\right) \Delta\epsilon_e, \tag{9}$$

with $s(\epsilon_e)$ and $\Delta\epsilon_e$ the weight and width of the e-th energy bin, respectively, and l_k the distance that the ray travels through material k with the energy-dependent attenuation coefficient $\mu_k(\epsilon_e)$.

Next, although an X-ray source is often assumed to be point-like in projection simulation, in practice it has a finite focal spot size. This can be accounted for by computing a weighted sum of point-source projections where the weights are determined by the focal spot shape, which yields the following computation of the projection pixels $\hat{\mathbf{p}}$ [20]:

$$\hat{\mathbf{p}} = \frac{\sum_i G_i \mathbf{p}_i}{\sum_i G_i}, \tag{10}$$

with $\mathbf{p}_i \in \mathbb{R}^{m \times n \times k}$ the projection image generated with the i-th point source, and G_i the position-dependent weight corresponding to the i-th point-source projection.

2.4. Scanning geometry

The description of an XCT imaging setup is based on the concepts of projection views and a projection geometry. A single projection view consists of the specification of the source and detector positions, the size of the detector, and its orientation. A set of projection views determines the trajectory of the X-ray source and detector that yields an XCT scan. Alternatively, the source and detector can remain stationary, with the XCT scan resulting from transformations of the sample. The projection geometry contains a description of either of the two. CAD-ASTRA provides three possibilities to specify a projection geometry, of which the first two replicate the cone-beam projection geometries available in the ASTRA toolbox [10], while the third allows a scan description through the movement of the mesh models (see Fig. 2):

Optics EXPRESS

- In the circular geometry, a rotation of the source and detector about the $+z$ axis with the mesh model located at the origin is assumed. Each projection view is defined as a triplet $\{SOD, ODD, \alpha_j\}$, with SOD the source-origin distance, ODD the origin-detector distance, and α_j the rotation angle for the j-th view.
- In the vector geometry, the $j-$th projection view is defined by a 12-element vector $[\mathbf{S}_j, \mathbf{D}_j, \mathbf{u}_j, \mathbf{v}_j]$, where $\mathbf{S}_j = [S_{jx}, S_{jy}, S_{jz}]$ is the source position, $\mathbf{D}_j = [D_{jx}, D_{jy}, D_{jz}]$ is the detector position, and $\mathbf{u}_j = [u_{jx}, u_{jy}, u_{jz}]$ and $\mathbf{v}_j = [v_{jx}, v_{jy}, v_{jz}]$ are the detector plane basis vectors. Note that $|\mathbf{u}_j|$ and $|\mathbf{v}_j|$ specify the pixel size along the detector columns and rows, respectively. The vector geometry offers a high degree of flexibility in specifying the source and detector trajectories, which allows easy implementation of non-conventional scanning techniques.
- In the mesh motion geometry, the projector, described by a single 12-element view vector, remains stationary, while the meshes that populate the scene are transformed. A mesh transformation is defined by a set of motion keys $\mathcal{T} = \{T(t)|t_s \leq t \leq t_e\}$, with t_s the start time and t_e the end time. The motion keys $T(t)$ are evenly distributed between the start and end time, with each key describing either a translation, rotation, scaling, or a combination thereof. To obtain a correct motion geometry, at least two motion keys must be specified for each transformed mesh and the points in time at which projections are to be generated. Note that these projection times are independent of the time points $\{t|t_s \leq t \leq t_e\}$ that correspond to the motions keys. If the projection time does not coincide with the motion key time, the transformation data is linearly interpolated between the two nearest motion keys. The mesh motion geometry allows to model applications containing complex sample transformations with different independent pieces, such as 4DCT.

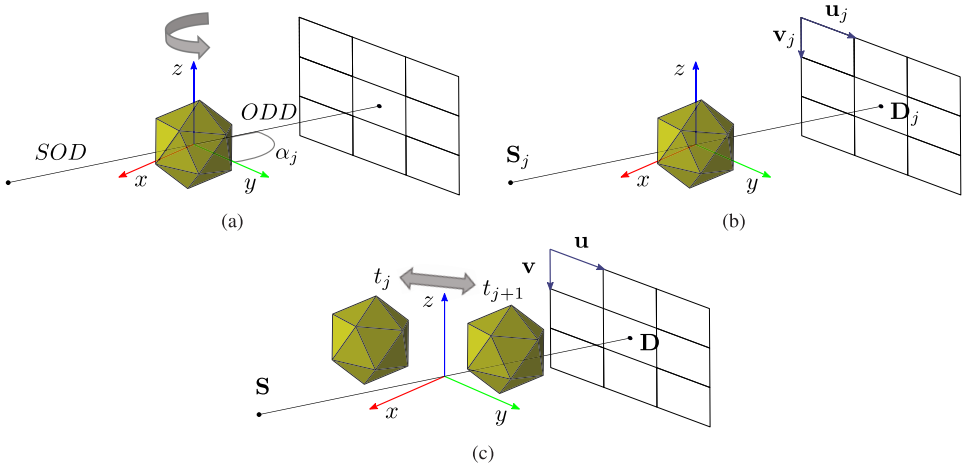


Fig. 2. Types of cone beam projection geometries available in CAD-ASTRA: (a) circular geometry, (b) vector geometry, (c) mesh motion geometry, with t_j and t_{j+1} denoting the projection times.

2.5. Projection algorithms

CAD-ASTRA's projectors are high-performance GPU implementations of projection algorithms which operationally describe the process of generating projection images from input meshes for the corresponding forward models. In Table 1, a brief overview is given of the projectors

Optics EXPRESS

available in the toolbox. Two of them implement the Proj_d^A operator. The first, Differentiable Rasterization (Raster-Diff), is a CUDA-implemented program based on rasterization [21]. The second projector is Differentiable Ray tracing (Ray-Diff), while providing similar functionality, leverages OptiX for ray tracing. The third algorithm, Ray Tracing with Refraction (Ray-Ref), corresponds to the Proj_d^I operator. Like Ray-Diff, it is implemented with the OptiX ray tracing engine, with added support for beam refraction. Since OptiX supports time-dependent ray tracing with transformations applied to the mesh(es), both the Ray-Diff and Ray-Ref are compatible with the motion projection geometry.

Table 1. Overview of the CAD-ASTRA projection algorithms.

Projector	Raster-Diff	Ray-Diff	Ray-Ref
Forward model	Proj_d^A	Proj_d^A	Proj_d^I
Gradient	**Yes**	**Yes**	No
GPU implementation	CUDA	OptiX	OptiX
Projection of multiple meshes	Multi-Pass	Multi-Pass	One-Pass
Mesh motion geometry	No	**Yes**	**Yes**
Refraction	No	No	**Yes**

Before running the projection algorithms, data must be transferred to the GPU memory from the host memory. The projectors Raster-Diff and Ray-Diff require multiple passes to project a collection of meshes due to the unordered way ray-mesh intersections are processed. Each mesh from an iterated list is projected separately and the projected image is aggregated to the final projection. In contrast, Ray-Ref adopts an ordered processing approach for ray-mesh intersections, enabling it to track the material the ray passes through. This allows multiple meshes to be projected in a single pass. Once the final projections and/or gradient values are stored in the GPU memory, the data can be transferred back to the host memory. The selected projection algorithm can be configured through a Python interface, allowing users to choose and fine-tune the algorithm that is best suited to their specific needs or applications.

2.6. Simulation pipeline

The main design concepts of CAD-ASTRA are the projection geometries, mesh data structures, CUDA arrays, and projector objects. Application of these concepts is presented by means of an example simulation pipeline, using the Python programming interface:

1. **Import of the CAD-ASTRA module, creation of the CUDA arrays, and transfer of the mesh data to the GPU memory.** The mesh is defined by a list of vertices and faces, or with an ordered list of three vertices representing a face. Providing normal vectors is purely optional since the projection algorithms can calculate them on-the-fly. In the example below, the volume inside the mesh has a linear attenuation coefficient μ of 0.5 mm^{-1} and a refractive index η of 1. Note that μ depends on the length unit of the world coordinate system, which in this case was chosen to be millimeters.

```
import cad_astra
mu = 0.5 # [1/mm]
eta = 1.0
# vert and norms are the numpy arrays with dtype=np.float32
# faces in the numpy array with dtype=np.int32
vert_id = cad_astra.create_cuda_array("vertices", nd_array=vert)
face_id = cad_astra.create_cuda_array("faces",    nd_array=faces)
norm_id = cad_astra.create_cuda_array("normals",  nd_array=norms)
mesh    = cad_astra.create_mesh( mu, eta, vert_id, face_id, norm_id )
```

Optics EXPRESS

2. **Definition of the projection geometry.** In this code snippet, a circular cone beam geometry is defined, using the parameters depicted in Fig. 2(a). The projection geometry consists of 360 projection images in a $[0, 2\pi)$ angular range, with a 512×512 detector of $1\,\text{mm} \times 1\,\text{mm}$ sized pixels. The source-object distance is 50 mm, and the object-detector distance is 500 mm. Note that, as with the mesh data above, the angles are defined using 32-bit floating point numbers.

```
import numpy as np
angles = np.linspace(0, 2*np.pi, 360, endpoint=False, dtype=np.float32)
proj_geom = cad_astra.create_projection_geometry('cone', 1, 1,
                                                 512, 512, angles,
                                                 50, 500)
```

3. **Configuring and running the projector.** A sinogram array is prepared in the GPU memory by creating a dedicated CUDA array, with its shape matching the above defined projection geometry. A projector object is created and configured, after which the selected projector is started using its identifier. After the forward projection is ready, the sinogram is transferred to the host memory.

```
sino_shape = (angles.shape[0], det_rows, det_cols)
sino_id = cad_astra.create_cuda_array('sino', shape=sino_shape)

cfg = {}
cfg['mesh'    ] = mesh
cfg['geometry'] = proj_geom
cfg['sino'    ] = sino_id
proj_id = cad_astra.create_projector('optix', cfg)
cad_astra.run(proj_id)

sino = cad_astra.get_cuda_array(sino_id)
```

4. **Free GPU memory.** Delete all CUDA arrays and the CAD-ASTRA projector object.

```
cad_astra.delete_all_cuda_arrays()
cad_astra.delete_projector(proj_id)
```

3. Experiments and results

3.1. Performance evaluation

To study the execution time of the different CAD-ASTRA projectors, simulation experiments for varying detector and mesh resolutions were performed. For each simulation, 180 cone-beam projections were computed, equiangularly distributed over $[0, 2\pi)$. Processed versions of the classic Stanford bunny mesh were used as samples. A total of seven meshes with $3.7 \cdot 10^4$ to $4.5 \cdot 10^6$ faces were prepared in MeshLab [22] by applying uniform mesh resampling and isotropic explicit re-meshing filters to the original Stanford bunny model (see Fig. 3). Every mesh was projected to four detectors containing 256^2, 512^2, 1024^2, and 2048^2 pixels, with the number of rays coinciding with the number of detector pixels. The simulation timings were recorded as the average over 20 repeated runs. All numerical experiments were run on a workstation equipped with an Intel Core i7-9700K CPU working at 3.60GHz, 32GB of RAM, an NVIDIA GeForce RTX 2080 graphics card, and running Ubuntu 22.04. The GPU was operated by display driver version 510.108.03, with NVIDIA CUDA and NVIDIA OptiX versions 10.2 and 7.4, respectively. To run OptiX JIT-compilation of the PTX code and optimize caching, all projectors were pre-launched before starting the benchmark.

The results of the performance study are summarized in Fig. 4. Figures 4(a) and 4(b) show the mean execution time as a function of the mesh size for detectors of size 256×256 and 2048×2048,

Optics EXPRESS

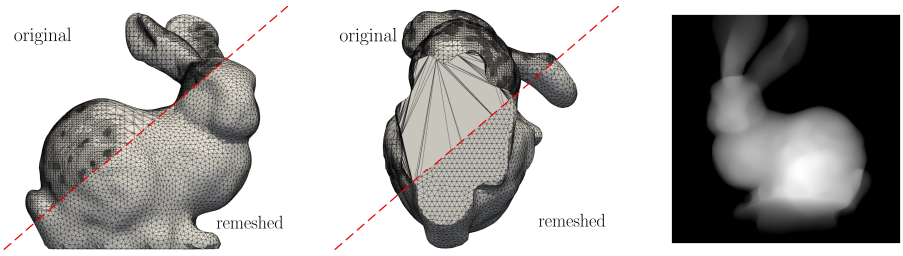


Fig. 3. Stanford bunny model from the performance study. Only remeshed versions of the model (a),(b) were projected. (c) Example of a projection image.

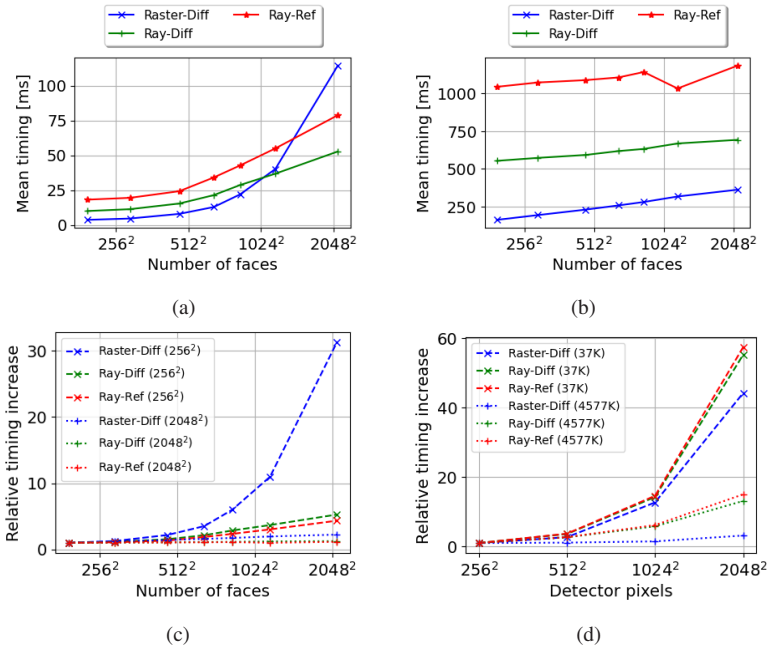


Fig. 4. Top row: computation time for Stanford bunny projections for the detector containing (a) 256×256 pixels and (b) 2048×2048 pixels. Bottom row: computation time increase relative to the minimum value for (c) combinations of projector and detector size (given in brackets), (d) combinations of projector and number of mesh faces (given in brackets).

respectively. The figures show that, for smaller detector sizes, all projectors are influenced predominantly by the mesh size. With increasing number of detector pixels, the performance of the mesh projector becomes less dependent on the mesh size. Fig. 4(c) shows how the timing for each combination of projector and detector size (indicated in brackets) increases, relative to the minimum value recorded for this combination. The relative timing increase for combinations of projector and mesh faces (also given in brackets), presented in Fig. 4(d), showcases each projector's scalability to the detector resolution.

Of the three implementations, the Raster-Diff projector exhibits the best scalability to detector size for all mesh resolutions (see Fig. 4(d)). As the Ray-Ref and Ray-Diff projectors show better scalability to mesh size than the Raster-Diff projector, they were faster at an unbalanced case of 256×256 detector and mesh size of $>1024^2$ faces. However, the Raster-Diff projector still

Optics EXPRESS

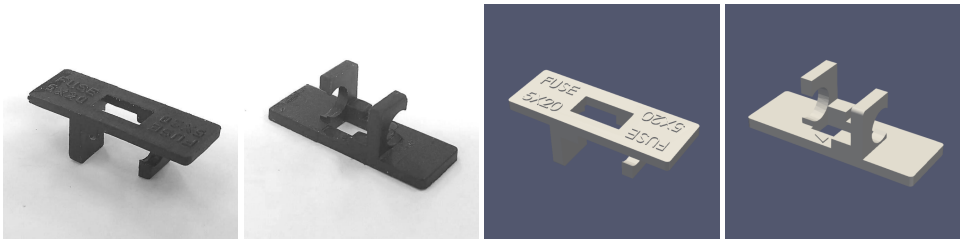
demonstrated the fastest execution times in most testing scenarios, especially for higher-resolution detectors.

3.2. Use cases

3.2.1. CAD-based inspection

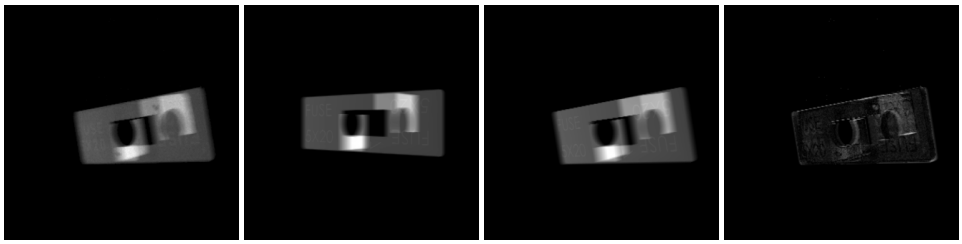
In many NDT X-ray based inspection tasks, the XCT image of an object is compared to the surface mesh of the CAD model from which this object was manufactured. In order to do this, the mesh has to be geometrically aligned with the XCT image, after which the deviations between the object and its CAD model are visualized. To demonstrate this use case, a fuse cover, shown in Fig. 5(a,b) was scanned with the FleXCT scanner [23] at 50 kVp, using a circular cone beam trajectory with radiographs equally distributed over a full angular range $[0, 2\pi)$. To facilitate direct comparison between the FleXCT images and mesh projections, the radiographs were corrected for beam hardening by the linearization method [24]. The CAD model was aligned by minimizing the difference with its forward projection and the measured projection as a function of the rigid transformation parameters of the CAD model. More details on the alignment algorithm can be found in Appendix 5.

The simulated projections from the CAD model surface mesh were computed with the Raster-Diff projector which was configured with the vector cone beam projection geometry extracted from the scan acquisition setting. For accurate simulation of the projection blur, a 2D-Gaussian focal spot of the source was assumed. In Fig. 6, the result of the alignment process (a-c), as well as the residuals image (d), are shown for a single projection angle. By reusing the projection geometry employed in the simulation of CAD model projections, a 3D difference image was reconstructed with the ASTRA-toolbox directly from the projection differences (Fig. 7). The reconstructed image reveals internal defects such as sample porosity, indicated by coloured



(a) (b) (c) (d)

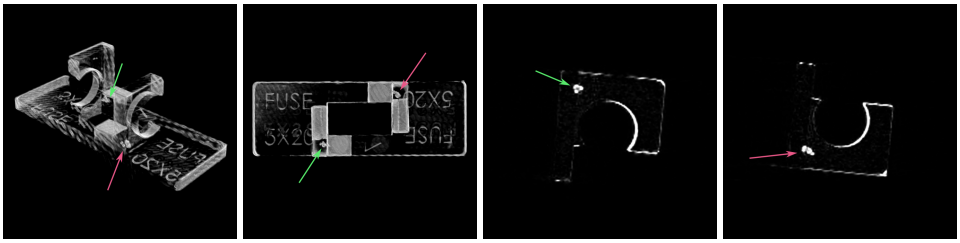
Fig. 5. Fuse cover sample used in the experiments: (a),(b) the photographs, (c),(d) CAD model surface mesh renderings.



(a) (b) (c) (d)

Fig. 6. Attenuation contrast images at a single angle: (a) FleXCT projection after beam hardening correction, (b) mesh projection before the alignment, (c) projection of the aligned CAD model surface mesh, (d) projection absolute difference image after the alignment.

Optics EXPRESS



(a) (b) (c) (d)

Fig. 7. A 3D image reconstructed with ASTRA that represents the absolute difference between the inspected sample and its aligned CAD model surface mesh: (a),(b) rendered in Mayavi [25] as a 3D volume with the color transfer function highlighting both surface and internal deviations, (c),(d) slices that correspond to the marked internal deviations.

arrows, as well as surface deviations from the CAD model (e.g., misaligned letters and thick edges).

3.2.2. Propagation-based phase contrast imaging

X-ray beam refraction at sample material interfaces may result in a phase contrast signal visible as dark and bright fringes at sample edges. The signal strength is dictated by the difference in refractive indices of the sample materials and by the distance between the sample and the detector. Phase contrast imaging techniques include edge illumination [26], speckle-based imaging [27], and free space propagation [28]. The availability of software to efficiently simulate phase contrast X-ray projections is crucial for the further development of these techniques and their implementation in laboratory and industry setups. In this subsection, the simulation of a free space propagation setup is described, leveraging the Ray-Ref algorithm. To mimic a typical synchrotron setup, a point source was placed at 10 m from the sample shown in Fig. 8, generating a nearly parallel X-ray beam propagating towards a detector containing 512×512 pixels, with the size of each pixel 12 µm. The sample mesh represented a collection of 100 water spheres with random radii of up to 800 µm placed at a distance of 250 mm from the detector. The CT scan consisted of 600 projection images over an angular range of $[0, \pi]$. To compare, images were generated of an X-ray scan with the object-to-detector distance moved down to 10 mm. The simulation used monochromatic X-rays at an energy of 30 keV, where the attenuation and refractive index of the water sample were set based on [29] and [30], respectively.

After generating the forward projections, the same projection geometry was used to reconstruct a volumetric image with the ASTRA toolbox. Results are shown in the top row of Fig. 9, where Poisson distributed projections were created with a flatfield photon count of 10^4 photons per pixel. It can be observed that the larger object-detector-distance (ODD) leads to stronger fringes

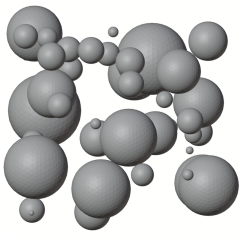
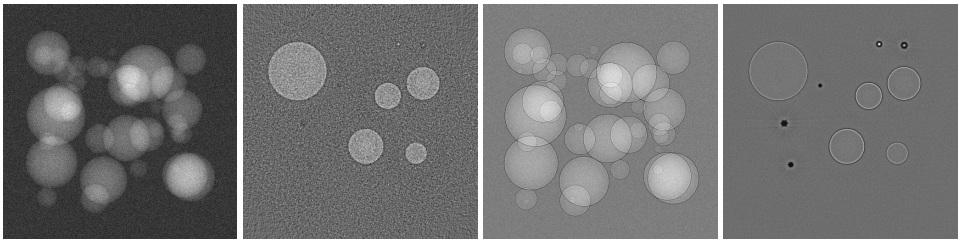
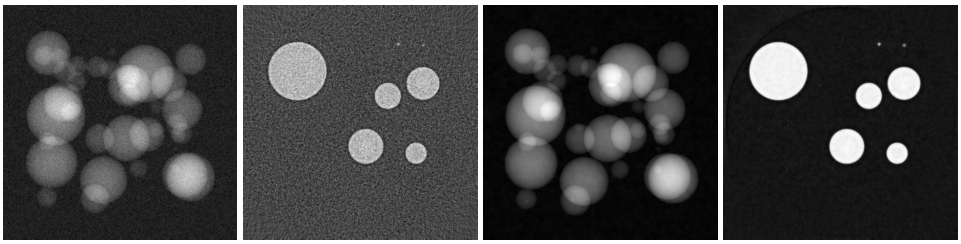


Fig. 8. A collection of surface meshes representing water spheres, used for simulating free space propagation phase contrast images.

Optics EXPRESS



(a) 10 mm projection, unfiltered (b) 10 mm reconstruction (c) 250 mm projection (d) 250 mm reconstruction



(e) 10 mm projection (f) 10 mm reconstruction (g) 250 mm projection (h) 250 mm reconstruction

Fig. 9. Projections and reconstructions (central slice) with varying ODD, where (a)-(d) show the unfiltered results, while (e)-(h) show the Paganin filtered results. For the ASTRA reconstruction, the same projection geometry was used as that of the forward projection.

at the sphere edges as a result of refraction. These fringes are not (or only faintly) present in the image with shorter ODD. A conventional method to convert these so-called phase fringes into extra contrast relies on the Paganin filter [31]. The corresponding Paganin filtered results are shown in the bottom row of Fig. 9. It can be seen that, after applying the filter, the projections with a longer propagation distance have a much higher signal-to-noise ratio, as was expected. These results demonstrate the capability of the Ray-Ref algorithm to model X-ray refraction.

3.2.3. Combined volume and mesh projection

In 4DCT (3D + time), objects that vary in shape, structure, and/or consistency during the acquisition time are studied. A particular field of interest is fluid flow, in which fluids and solid matter are typically represented as a voxel volume, yielding an XCT image containing both [32,33]. It has been shown that stationary regions can be excluded from the reconstruction, thereby reducing reconstruction artifacts and improving overall image quality [32]. One way to do that is to model the stationary region by a surface mesh, allowing it to be efficiently projected separately from the voxelized dynamic region.

To demonstrate the simulation of such combined volume-mesh projection, a fluid dynamics scene was simulated using mantaflow through Blender [34,35] (Figs. 10(a) and 10(b)). The scene consists of a solid dome, represented by a mesh, that interacts with a rising fluid, represented by a voxel volume. The dome is punctured with a single hole, such that part of the fluid seeps through. The resulting projection was computed by directly superposing the images simulated with ASTRA and CAD-ASTRA for the same projection geometry.

3.2.4. Mesh reconstruction

In numerous XCT applications, object modeling with surface meshes delivers distinct advantages over a voxel-based representation. With their capacity for high geometric precision and efficient storage, surface meshes provide a significantly smaller solution space, by simultaneously limiting the influence of limited or noisy projection data [17,19]. This use case demonstrates the

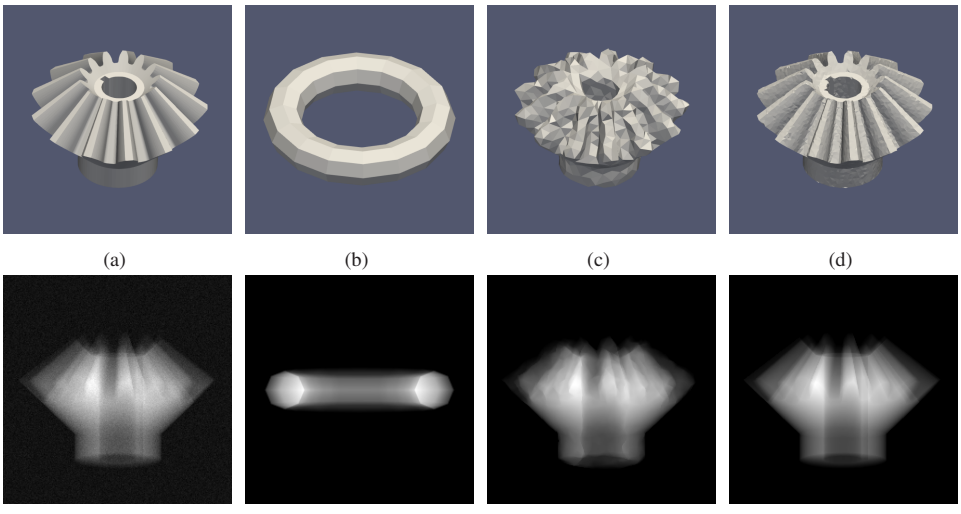
Optics EXPRESS

(a) (b) (c)

Fig. 10. A fluid simulation containing a dome represented by a mesh (green), and a fluid represented by a voxel volume: (a), (b) scene rendered in Blender, (c) combined projection simulated with ASTRA and CAD-ASTRA.

reconstruction of a 3D shape directly from projection images by minimizing the objective function described by Eq. (7). To simulate an XCT scan, a reference mesh (see Fig. 11(a)) was projected with the Raster-Diff algorithm, resulting in 180 projection images evenly spread over a full angular range $[0, 2\pi)$ to which normally distributed noise was added (see projection example in Fig. 11(e)).

The same algorithm was applied to project the current surface fit and to calculate the gradient of the objective function. The initial mesh was selected to be a torus (see Fig. 11(b)). To prevent the mesh from becoming degenerate, remeshing, as described in [36], was applied after every optimization round. The final reconstruction after 16 rounds of optimization and remeshing is shown in Fig. 11(d). The minor imperfections observed are attributable to factors such as the addition of noise to the simulated projections, the limited amount of the projection data simulated, and the use of single-precision floating-point computations.



(a) (b) (c) (d)

(e) (f) (g) (h)

Fig. 11. Mesh reconstruction example. In the top row, the renderings are shown for (a) the original mesh, (b) the initial mesh recontruction, (c) the reconstructed mesh after 43 iterations and remeshing (d) the final reconstructed mesh. In the bottom row, the projections for the corresponding meshes are shown for a selected angle.

Optics EXPRESS

4. Conclusion

In this paper, we presented CAD-ASTRA, a versatile toolbox for simulating X-ray projections of objects represented as surface meshes, which has an open-source version. CAD-ASTRA provides high-performance differentiable and non-differentiable GPU implementations of projection algorithms based on ray optics. Selecting an appropriate projector allows the user to conduct numerical experiments in X-ray phase contrast imaging, reconstruct meshes directly from X-ray projections, or embed the forward projection operators into deep learning models. Furthermore, a more sophisticated forward model can be implemented on top of the projectors, incorporating source polychromaticity and a finite-sized focal spot. CAD-ASTRA's flexible approach to defining a scanning geometry allows the implementation of a wide range of non-standard scanning techniques. By maintaining full compatibility with the ASTRA toolbox, CAD-ASTRA facilitates the integration of forward mesh model projections with conventional voxel-based tomographic reconstruction. Various use cases of CAD-ASTRA demonstrated its potential in both academic and industrial application domains. The open-source version of CAD-ASTRA is available at [16].

Appendix A. Mesh alignment

The position of a surface mesh can be aligned by optimizing the translations $\mathbf{T} = (t^x, t^y, t^z)^T$ of the mesh vertices along the x, y and z-axes, respectively, while its orientation can be aligned by optimizing the rotation matrix $\mathbf{R} = \mathbf{R}^x(\alpha)\mathbf{R}^y(\beta)\mathbf{R}^z(\gamma)$, with α, β, γ the rotation angles about the x, y and z-axes, respectively. Let $\mathbf{M} \in \mathbb{R}^{3\times v}$ be a matrix representing the v mesh vertices. By minimizing the difference between the vector $\tilde{\mathbf{P}}$ containing the sample radiograph pixel values and a vector containing simulated projection pixel values, simulated with the CAD-ASTRA projection operator Proj_d^A, the transformed mesh coordinates $\mathbf{RM} + \mathbf{T}$ corresponding to the aligned surface mesh can be found. The alignment procedure is summarized in Algorithm 1.

Algorithm 1. Mesh alignment algorithm.

$\mathbf{T}_0 = \mathbf{0}$;
$\mathbf{R}_0 = \mathbf{R}^x(0)\mathbf{R}^y(0)\mathbf{R}^z(0)$;
for $i = 1$ **to** N **do**
  $\hat{\mathbf{T}} = \arg\min_{\mathbf{T}} \|\text{Proj}_d^A\left(\mathbf{R}_{i-1}\mathbf{M} \quad \mathbf{T}_{i-1} \quad \mathbf{T}\right) - \tilde{\mathbf{P}}\|^2$;
  $\hat{\mathbf{R}} = \arg\min_{\mathbf{R}} \|\text{Proj}_d^A\left(\mathbf{R}\left(\mathbf{R}_{i-1}\mathbf{M} \quad \mathbf{T}_{i-1} \quad \hat{\mathbf{T}}\right)\right) - \tilde{\mathbf{P}}\|^2$;
  $\mathbf{T}_i = \hat{\mathbf{R}}(\mathbf{T}_{i-1} \quad \hat{\mathbf{T}})$;
  $\mathbf{R}_i = \hat{\mathbf{R}}\mathbf{R}_{i-1}$;
end

Funding. Fonds Wetenschappelijk Onderzoek (1SA2922N, G090020N, G094320N, FoodPhase S003421N); Agentschap Innoveren en Ondernemen (Multiplicity HBC.2022.0094).

Disclosures. The authors declare no conflicts of interest.

Data availability. Data underlying the results presented in this paper are not publicly available at this time but may be obtained from the authors upon reasonable request.

References

1. C. Fragnaud, C. Remacha, J. Betancur, *et al.*, "CAD-based X-ray CT calibration and error compensation," Meas. Sci. Technol. **33**(6), 065024 (2022).
2. P. Withers, C. Bouman, S. Carmignato, *et al.*, "X-ray computed tomography," Nat. Rev. Methods Primers **1**(1), 18 (2021).
3. J. Kruth, M. Bartscher, S. Carmignato, *et al.*, "Computed tomography for dimensional metrology," CIRP Ann. **60**(2), 821–842 (2011).

Optics EXPRESS

4. L. De Chiffre, S. Carmignato, J.-P. Kruth, *et al.*, "Industrial applications of computed tomography," CIRP Ann. **63**(2), 655–677 (2014).
5. A. Thompson, I. Maskery, and R. Leach, "X-ray computed tomography for additive manufacturing: A review," Meas. Sci. Technol. **27**(7), 072001 (2016).
6. M. van Dael, P. Verboven, J. Dhaene, *et al.*, "Multisensor X-ray inspection of internal defects in horticultural products," Postharvest Biol. Technol. **128**, 33–43 (2017).
7. A. Presenti, J. Sijbers, and J. De Beenhouwer, "Dynamic few-view X-ray imaging for inspection of CAD-based objects," Expert. Syst. with Appl. **180**, 115012 (2021).
8. M. Endrizzi, "X-ray phase-contrast imaging," Nucl. Instruments Methods Phys. Res. Sect. A: Accel. Spectrometers, Detect. Assoc. Equip. **878**, 88–98 (2018).
9. E. Zwanenburg, M. Williams, and J. Warnett, "Review of high-speed imaging with lab-based X-ray computed tomography," Meas. Sci. Technol. **33**, 1 (2021).
10. W. van Aarle, W. J. Palenstijn, J. Cant, *et al.*, "Fast and flexible X-ray tomography using the ASTRA toolbox," Opt. Express **24**(22), 25129–25147 (2016).
11. E. Bergbäck Knudsen, A. Prodi, J. Baltser, *et al.*, "McXtrace: A Monte Carlo software package for simulating X-ray optics, beamlines and experiments," J. Appl. Crystallogr. **46**(3), 679–696 (2013).
12. T. Faragó, P. Mikulík, A. Ershov, *et al.*, "Syris: A flexible and efficient framework for X-ray imaging experiments simulation," J. Synchrotron Radiat. **24**(6), 1283–1295 (2017).
13. Á. Marinovszki, J. De Beenhouwer, and J. Sijbers, "An efficient CAD projector for X-ray projection based 3D inspection with the ASTRA toolbox," in *8th Conference on Industrial Computed Tomography* (2018).
14. J. Nickolls, I. Buck, M. Garland, *et al.*, "Scalable parallel programming with CUDA," Queue **6**(2), 40–53 (2008).
15. S. G. Parker, J. Bigler, A. Dietrich, *et al.*, "OptiX: A general purpose ray tracing engine," ACM Trans. Graph. **29**(4), 1–13 (2010).
16. CAD-ASTRA GitHub repository, [retrieved 16 June 2023], https://github.com/cad-astra/cad-astra.
17. J. Renders, J. De Beenhouwer, and J. Sijbers, "Mesh-based reconstruction of dynamic foam images using X-ray CT," in *International Conference on 3D Vision* (IEEE, 2021), pp. 1312–1320.
18. F. Vidal, M. Garnier, N. Freud, *et al.*, "Simulation of X-ray attenuation on the GPU," in *Theory and Practice of Computer Graphics* (2009), pp. 25–32.
19. J. Koo, A. B. Dahl, J. A. Baerentzen, *et al.*, "Shape from projections via differentiable forward projector for computed tomography," Ultramicroscopy **224**, 113239 (2021).
20. A. Presenti, Z. Liang, L. F. A. Pereira, *et al.*, "Fast and accurate pose estimation of additive manufactured objects from few X-ray projections," Expert. Syst. with Appl. **213**, 118866 (2023).
21. S. G. Parker, H. Friedrich, D. Luebke, *et al.*, "GPU ray tracing," Commun. ACM **56**(5), 93–101 (2013).
22. P. Cignoni, M. Callieri, M. Corsini, *et al.*, "MeshLab: an open-source mesh processing tool," in *Eurographics Italian Chapter Conference*, V. Scarano, R. D. Chiara, and U. Erra, eds. (The Eurographics Association, 2008).
23. B. De Samber, J. Renders, T. Elberfeld, *et al.*, "FleXCT: a flexible X-ray CT scanner with 10 degrees of freedom," Opt. Express **29**(3), 3438–3457 (2021).
24. G. Herman, "Correction for beam hardening in computed tomography," Phys. Med. Biol. **24**(1), 81–106 (1979).
25. P. Ramachandran and G. Varoquaux, "Mayavi: 3D visualization of scientific data," Comput. Sci. Eng. **13**(2), 40–51 (2011).
26. A. Olivo, F. Arfelli, G. Cantatore, *et al.*, "An innovative digital imaging set-up allowing a low-dose approach to phase contrast applications in the medical field," Med. Phys. **28**(8), 1610–1619 (2001).
27. L. Quénot, E. Brun, J. M. Létang, *et al.*, "Evaluation of simulators for X-ray speckle-based phase contrast imaging," Phys. Med. Biol. **66**(17), 175027 (2021).
28. S. Wilkins, T. Gureyev, D. Gao, *et al.*, "Phase-contrast imaging using polychromatic hard X-rays," Nature **384**(6607), 335–338 (1996).
29. C. T. Chantler, "Theoretical form factor, attenuation, and scattering tabulation for z=1-92 from e=1-10 ev to e=0.4-1.0 mev," J. Phys. Chem. Ref. Data **24**(1), 71–643 (1995).
30. B. Henke, E. Gullikson, and J. Davis, "X-ray interactions: Photoabsorption, scattering, transmission, and reflection at e = 50-30,000 ev, z = 1-92," At. Data Nucl. Data Tables **54**(2), 181–342 (1993).
31. D. Paganin, S. C. Mayo, T. E. Gureyev, *et al.*, "Simultaneous phase and amplitude extraction from a single defocused image of a homogeneous object," J. Microsc. **206**(1), 33–40 (2002).
32. G. Van Eyndhoven, K. J. Batenburg, D. Kazantsev, *et al.*, "An iterative CT reconstruction algorithm for fast fluid flow imaging," IEEE Trans. on Image Process. **24**(11), 4446–4458 (2015).
33. P. W. Rasmussen, H. O. Sørensen, S. Bruns, *et al.*, "Improved dynamic imaging of multiphase flow by constrained tomographic reconstruction," Sci. Rep. **11**(1), 12501 (2021).
34. Blender Online Community, "Blender - a 3D modelling and rendering package," [retrieved 16 June 2023], http://www.blender.org.
35. N. Thuerey and T. Pfaff, *MantaFlow* http://mantaflow.com (2022).
36. M. Botsch and L. Kobbelt, "A remeshing approach to multiresolution modeling," in *Eurographics/ACM SIGGRAPH Symposium on Geometry Processing* (2004), pp. 185–192.