The *CTSimU* software toolbox for CT-related image processing and quality assessment

David Plotzki\(^1\), Bendix Hartlaub\(^1\), Fabricio Borges de Oliveira\(^2\), Tamara Reuter\(^3\), Florian Wohlgemuth\(^3, 4\), Carsten Bellon\(^3\), Tino Hausotte\(^3\)

\(^1\)Bundesanstalt für Materialforschung und -prüfung, Unter den Eichen 87, 12205 Berlin, Germany, e-mail: davidplotzki@gmail.com, bendix.hartlaub@bam.de, carsten.bellon@bam.de

\(^2\)Physikalisch-Technische Bundesanstalt, Bundesallee 100, 38116 Braunschweig, Germany, e-mail: fabricio.borges@ptb.de

\(^3\)Institute of Manufacturing Metrology (FMT), Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Nägelsbachstr. 25, 91052 Erlangen, Germany, e-mail: tamara.reuter@fmt.fau.de, tino.hausotte@fmt.fau.de

\(^4\)HEITEC PTS GmbH, Hauptstraße 49, 73329 Kuchen, Germany, e-mail: florian.wohlgemuth@heitec-pts.de

Abstract

A software toolbox is introduced that addresses several needs common to computed tomography (CT). Built for the WIPANO *CTSimU* project [1] to serve as the reference implementation for its image processing and evaluation tasks, it provides a Python 3 interface that is adaptable to many conceivable applications. Foremost, the toolbox features a pipeline architecture for sequential 2D image processing tasks, such as flat field corrections and image binning, and enables the user to create their own processing modules. Beyond that, it provides means to measure line profiles and image quality assessment algorithms to calculate modulation transfer functions (MTF) or to determine the interpolated basic spatial resolution (iSR) using a duplex wire image [2]. It can also be used to calculate projection matrices for the reconstruction of scans with arbitrary industrial CT geometries and trajectories. The *CTSimU* project defined a framework of projection- and volume-based test scenarios [3] for the qualification of radiographic simulation software towards its use in dimensional metrology. The toolbox implements the necessary evaluation routines and generates reports for all projection-based tests.

Keywords: image processing, image quality, WIPANO CTSimU, CT geometry, software, toolbox, Python

1 Motivation

Working with CT projection and volume data requires some basic tasks such as flat field corrections, image binning, noise reduction, or MTF measurements. In a production environment, those tasks are generally automated or easy to access in the manufacturer’s software, whereas a scientific environment requires more control and knowledge about the algorithmic steps. The *CTSimU* software toolbox presented here offers basic functions, objects and algorithms which facilitate these tasks. It is written in Python 3 with dependencies on the NumPy [4] and some SciPy packages [5, 6]. Its code and documentation can be found on Github [7]. The toolbox provides different levels of flexibility and automation. There are very basic features that offer read, write and editing routines for single images, and it also offers the ability to batch-process entire image stacks. Aspects of CT geometry are another application of the toolbox: it provides coordinate system objects (such as for source, stage and detector) that can be manipulated via function calls and be used to compute projection matrices for subsequent reconstruction in supporting software.

Its main features are summarized in four categories: image conversion, 2D image processing, image quality assessment, and CT geometry tools to model arbitrary trajectories and calculate projection matrices for subsequent CT reconstructions. While most of the functionality of the toolbox is already covered by other open-source software such as ImageJ [8, 9, 10], Drishti [11] or open_iA [12], these examples are primarily GUI-based applications, targeted towards end users, and offer the ability to extend them with plugins. On the other side, the *CTSimU* software toolbox provides a light-weight foundation framework meant to be incorporated into higher-level Python workflows concerning 2D image processing or quality assessment, CT geometry applications, or into scripts for the automation of CT-related tasks in scientific environments. It does not offer any GUI and is controlled either via documented API calls (application programming interface) or by using the classes and functions it provides on a lower level.

The *CTSimU* project (“Radiographic Computed Tomography Simulation for Measurement Uncertainty Evaluation”) developed a qualification framework for radiographic simulation software towards its ability to simulate complete CT scans for dimensional metrology and facilitate the creation of digital twins. The project designed a collection of test scenarios with dedicated reference standards [3] to assess the faithful simulation of the laws of physics and radiation as well as basic characteristic effects and functionalities relevant to CT measurements. Most test scenarios require the simulation of a few single projection images that will be evaluated using the toolbox (so-called “2D tests”). Each test specifies certain evaluation criteria and instructions to reach the test result. The software toolbox presented here serves as the project’s reference implementation for the evaluation instructions of all 2D tests to provide uniform evaluation procedures. For this purpose, it offers a generalized structure that can be employed by any user to create their own test evaluation routines.
2 Basic features

On the lowest level, the toolbox offers a Python class for image objects that allows the user to read, write and manipulate 2D image data. It can read and write TIFF files as well as RAW files with known header sizes. Image data is stored in memory as a 64-bit floating point Numpy array, which allows the manipulation of grey values without a loss in precision for all standard integer or floating-point image data types used in practice. Image objects can be rotated, flipped, cropped, binned, renormalized and used for arithmetic (adding, subtraction, multiplication, division, square root, etc.). Grey value statistics can be calculated for the complete image or a given region of interest (ROI). The grey values can be manipulated individually or using processing functions and filters, such as adding noise or applying a median filter.

On the next layer of abstraction, the toolbox provides a class for image stacks, i.e., a collection of 2D slices, which can be provided from a set of sequentially numbered 2D TIFF or RAW files, or a RAW volume file that contains slices. Such virtual stacks of image slices can be processed sequentially or used to access individual images within the stack.

3 Image processing pipelines

To facilitate the processing of entire image stacks, the CTsimU software toolbox provides modules that can be queued in a pipeline of arbitrary length (Figure 1). The pipeline reads the stack in sequential order and gradually passes them through its queue of processing modules.

A very simple application of such a pipeline is the conversion between image formats, considering that the input format may differ from the output format. Additional processing modules perform tasks such as flat/dark field correction with given reference images, binning, median and smoothing filters, simple transformations such as image rotations and mirroring, or the generation of artificial noise according to provided noise characteristics. Users can create their own pipeline modules and specify Python manipulation routines for the images. Each processing module may side-load additional images if needed (such as for a flat field correction) or save images or measurement data on its own, e.g., to store intermediate processing results.

4 Line profiles and image quality assessment

Line profiles across certain features in an image are an important analytical tool. They describe the course of grey value changes along a given line in the image. The toolbox offers a method to measure line profiles between two given points \( p_0 \) and \( p_1 \) in the pixel coordinate system (which may be given with subpixel precision) using a weighted grey value sampling technique (Figure 2). The line profile itself is a discretely binned entity, a chain of bins along the line, each with a size \( r \) (the resolution of the line profile). To achieve subpixel sampling, the resolution can be less than 1 px.

To compute the grey value for each bin, the pixels that have an overlap with the bin rectangle are identified. This is illustrated in Figure 2 for the 8th bin: the five greyly shaded pixels have an overlap with the highlighted bin rectangle. Each of these pixels contributes its grey value weighted with the area of their overlap (in Figure 2, the five overlap polygons are shaded in blue). To
calculate a pixel's overlap area, it is clipped using the bin rectangle as a clipping polygon. The Sutherland-Hodgman polygon clipping algorithm [13] is employed. Two features that can be calculated from line profiles are the modulation transfer function (MTF) and the interpolated basic spatial resolution ($iSR_b$) according to the ASTM E2002-15 standard [2]. The toolbox offers implementations for both.

The modulation transfer function (MTF) is a way of quantifying a system's response to input signals from a range of frequencies. For any given input frequency, it gives a value of how much the system dampens the signal's amplitude [14, 15] and is a representation of the point spread function in the frequency domain (assuming rotational symmetry of the point spread function). To measure the MTF of an X-ray projecting imaging system, the method is usually to measure a line profile perpendicular to an imaged ideal edge (Figure 3a), resulting in an edge spread function (ESF, Figure 3b). The derivative of the ESF is called the line spread function (LSF, Figure 3c). The absolute value of the LSF’s frequency transmission spectrum (i.e., its Fourier transform) is called the modulation transfer function (MTF, Figure 3d). The toolbox implements the method described by Samei et al. [16] to obtain the pre-sampled line spread function (LSF) and modulation transfer function (MTF).

Similarly, line profiles across duplex wire standards are used to quantify a detector’s response to increasingly higher frequencies (i.e., smaller wire distances) and to calculate a value for the detector’s unsharpness or basic spatial resolution [2], as illustrated in Figure 4. The toolbox provides a module for running this analysis on a given duplex wire image and allows the user to specify the wire distances for their own calibrated standard.

**Figure 3.** Illustration for measuring the MTF in a projection image of a simulated edge (a) using a line profile perpendicular to the edge. The toolbox was used to calculate the edge spread function (b), the line spread function (c) and the modulation transfer function (d).

**Figure 4.** Measuring a line profile across a duplex wire image (top). The line profile is used by the toolbox to determine the interpolated basic spatial resolution ($iSR_b$) as described in [2] and to generate a report diagram (seen on the bottom).

## 5 Tools for arbitrary cone-beam CT geometries and trajectories

A typical industrial cone-beam CT setup features an X-ray source, a rotating sample stage and a detector (Figure 5). It is essential to know their relative positions and orientations when reconstructing the volumetric data. For cone-beam CT, standard reconstruction software usually implements varieties of the FDK algorithm [17] for circular (source) trajectories or, in industrial CT, a rotating sample stage. However, other trajectories are conceivable, a well-established method being helical CT [18]. For the general description of arbitrary scan trajectories, projection matrices can be used [19, 20]. A projection matrix transforms a given coordinate from the 3D stage coordinate system into its projected 2D coordinate in the detector coordinate system. For
each projection image, a projection matrix can be calculated and provided to a supporting reconstruction software. Currently, the toolbox provides functions to generate configuration files for the openCT format used by VGSTUDIO (Volume Graphics, Heidelberg), and CERA (Siemens Healthineers, Erlangen) configuration files.

Figure 5. Industrial cone-beam CT setup with example coordinate systems. The world coordinate system is denoted by \{x, y, z\}, local coordinate systems by the letters \{u, v, w\} with corresponding indices.

The toolbox provides virtual representations for a cone-beam CT setup (like the one shown in Figure 5) and provides Python objects for source, stage and detector that can be manipulated using translations and rotations around any vector (including the axes of their local coordinate systems). For each specific constellation of source, stage and detector, a projection matrix can be calculated. The matrix representation depends on the detector coordinate system of the reconstruction software: its units and location of its origin. These parameters must be provided to the toolbox; openCT and CERA are supported out of the box. This abstraction makes it easy for programmers to simulate arbitrary trajectories. The resulting coordinate system vectors can also be used for further processing, e.g., using the ASTRA Toolbox [21]. Another use case for reconstruction methods based on projection matrices is the compensation of known geometrical drifts during the scan. Tracking systems for CT components are currently investigated [22] and would provide useful information to be incorporated into the reconstruction process.

For a comprehensive documentation of the parameters of industrial CT scans, a description format [23] using the JavaScript Object Notation (JSON) was developed within the CTSimU project. This format is used as an unambiguous set of simulation instructions for the test scenarios and can more generally be utilized to describe typical CT setup geometries and arbitrary scan trajectories, including drifts. The toolbox can read such JSON-based scenario files and directly provide reconstruction configuration files for VGSTUDIO/openCT and CERA.

### 6 Image-based 2D test evaluations

The test scenarios defined by the CTSimU project require the reproduction of physical effects and image quality parameters by the simulation software under test. For most of these tests, single projection images are sufficient (those are called “2D tests”). The reference implementations of the evaluation routines for these 2D tests are incorporated in the CTSimU toolbox. For example, a test scenario may require a certain basic spatial resolution, or the reproduction of a given MTF. Most 2D tests include specially designed test specimens [3] whose projection images can be used to evaluate the positioning of CT components and sample specimens, penetration lengths, grey value reproduction, properties of the X-ray spectrum, or the unsharpness contributions from detector and focal spot. Each of these tests describes a specific procedure for the projection image evaluation. The toolbox is used to calculate the numerical test results and to generate visual report diagrams. The general workflow of the CTSimU test framework is shown in Figure 6. It is currently being discussed for standardization within VDI/VDE technical committee 3.33 and cannot be described here in detail. However, in the following, an example test case will be illustrated using the inclined tungsten edge standard developed and described by Borges de Oliveira et al. [3] to measure the simulated focal spot size.

Figure 6. Workflow of the CTSimU test framework for all projection-based 2D tests.
The simplified scenario that we want to demonstrate here requests the radiographic simulator to reproduce a two-dimensional Gaussian spot profile for the X-ray source and specifies an expected standard deviation \( \sigma \) for the size of the spot intensity profile. The inclined tungsten edge is located halfway between source and detector to ensure that the projected point spread function is of the same size and shape as the spot profile. The detector is requested to create ideally sharp images (i.e., it should feature an ideal unsharpness value of zero). Any blurring of the edge should only result from the spatial extension of the focal spot and, to a minor extent, the pixel sampling. The simulator should now run the simulation of the required projection image according to the scenario description, using the provided CT geometry, specimen STL file and X-ray spectrum. A simulator is not required to perform a flat field correction; it is sufficient to provide a free beam image along with the simulated projection image, and it is even preferable, considering that there are usually several conceivable choices for the renormalization factor, whereas the test framework expects one specific way. A metadata file should be created for the simulation results [24], which is an additional JSON structure that comprises information about the projection and free beam image file names, their formats, sizes, etc. Such a metadata file can be passed to the toolbox along with the command to run an evaluation procedure for a specific test from the framework. If necessary, the toolbox will perform the flat field correction beforehand, in the manner that is expected by the test framework.

For this example, the software aRTist 2.10 (BAM, Berlin) was used to simulate the given scenario, using a plugin that automatically imports CTSimuU JSON scenario description files. The resulting projection image of the inclined tungsten edge is shown in Figure 7a. The aRTist software uses a ray tracing algorithm where discrete rays are simulated from the source to the individual detector pixels. A spatially extended spot profile is modelled with a certain number of rays originating from the spot profile’s surface, each ray weighted by the respective spot intensity at the point of origin. This simulation was done using a setting of 30 rays per detector pixel, distributed over the focal spot using an intensity-dependent Poisson disk algorithm. This results in a limited number of possible ray combinations that can hit a pixel, and therefore in a limited number of possible grey values. The pattern of the discretely sampled edge becomes visible in small discontinuities (steps) of the line profile across the edge (edge spread function, Figure 7b) and much more so in its derivative, the line spread function (Figure 7c), which shows a high-frequency pattern that matches the source spot sampling. However, a Gaussian fit to the LSF (blue-dashed line in Figure 7c) results in a shape that very closely matches the expected analytical LSF (black-dotted line). The analytical results are taken from a line profile measurement across an analytical edge image. The toolbox calculates this analytical image by assuming an isotropic source (as required by the test framework). In combination with the ideally energy-integrating detector,! The grey value of a pixel is purely proportional to the radiation flux through the exposed (uncovered) area, which in turn is proportional to the solid angle as seen from the source’s centre point. The toolbox calculates the solid angle for the exposed areas of all pixels and assigns grey values accordingly to generate an analytical edge image assuming an ideal point source. A convolution with the spot profile produces the expected analytical image for the given spatially extended focal spot, considering that the point spread function (PSF) in this specific scenario matches the spot profile. The modulation transfer function (MTF, Figure 7d) is calculated as well. In this example case, the lower frequencies match the expectation very well, but the discrete sampling leads to artifacts that become visible at higher frequencies in the MTF. Apart from the diagrams shown in Figure 7b-d, the toolbox generates the numerical test results shown in Figure 7e, which will become the basis for the software evaluation. In this example, the test results are the measured width \( \sigma \) from a Gaussian fit to the LSF, the frequency where the MTF drops to 20 % (MTF20 frequency), and their respective relative deviations from the expected (analytical) value. The actual software evaluation based on these test results is not part of the toolbox and a proposal for this independent procedure is currently discussed within the standardization committee.

From a technical perspective, test modules are special pipeline modules (see section 3) which can be incorporated into a test pipeline. This makes it easy to run the same test on several projection images or run a sub-test with different requirements for each individual projection image. For example, the test framework may require the simulation of a given set of several different

![Figure 7](image-url). (a) Example of a simulated projection image of the inclined tungsten edge standard located halfway between source and detector. The region and direction where the line profile is measured is shown as a grey overlay with the start point \( p_0 \) and the end point \( p_1 \). (b, c, d) Measurement results for the edge spread function (ESF), line spread function (LSF) and modulation transfer function (MTF). (e) Comparison of the test results with the analytical values.
spot profiles, and each resulting projection image would be evaluated by the toolbox considering the required individual parameters of the sub-test.

7 Summary and outlook

The CTSimU software toolbox primarily serves the need for reproducible image processing and test evaluations within the CTSimU project. Its general framework of documented API calls and Python structures makes it easy to incorporate into other CT-related workflows or evaluation scripts. It features a pipeline architecture for the batch-processing of 2D image stacks, and users can create their own pipeline modules for image processing and evaluation tasks. The toolbox currently provides a small set of image quality analysis tools, namely the calculation of the interpolated basic spatial resolution according to ASTM E2002-15 [2] as a quantifier for image unsharpness, and the calculation of the modulation transfer function (MTF) from an edge image, following a method described by Samei et al. [16]. It features geometry tools for the virtual representation and modification of a CT setup in Python with the ability to calculate projection matrices for arbitrary trajectory reconstruction algorithms. It provides reference implementations for all 2D test evaluations of the CTSimU test framework. An example for such a test has been illustrated for the focal spot size. In the upcoming follow-up project, CTSimU2, the toolbox will be extended with tests and evaluation routines that will focus on the calibration of real-world CT scanners with the intention of creating digital twins, and the subsequent testing of the digital twin reproduced by the simulation software.

Acknowledgements

This work was funded through the project CTSimU (WIPANO project 03TNH026A). WIPANO projects are financed by the German Federal Ministry for Economic Affairs and Energy and managed by Project Management Jülich. The authors thank all CTSimU project partners for the discussions and support, and in particular Markus Bartscher, Stefan Kasperl and Richard Schielein.

CRediT [25]

Tino Hausotte (TH) contributed to funding acquisition, project administration and resources. David Plotzki (DP) worked on the overall programming of the software and documentation of the toolbox and drafted this publication. Bendix Hartlaub (BH) contributed code for the image quality analysis modules, namely the iSRb and MTF module. Florian Wohlgemuth (FW) contributed a method for calculating the flux through a pixel from its solid angle as seen from the X-ray source and, together with DP, created the file format for CT metadata [24] used as one possible way to interface with the toolbox. Tamara Reuter (TR) and Fabrício Borges de Oliveira (FB) provided testing and contributed to the toolbox documentation. Carsten Bellon (CB) provided valuable supervision and advice. TR, FB, FW and CB contributed to the review, editing and finalization of the full paper.

References