2023-09-05

CTSimU Scenario Descriptions

A data format specification in JSON

Version 1.2

WIPANO CTSimU Project

Radiographic Computed Tomography Simulation for Measurement Uncertainty Evaluation

Summary: This is the specification for a JSON file format to handle the parameters of a full industrial CT scan scenario, intended for describing virtual CT scenarios for simulations, as well as for documenting real CT scan geometries and acquisition parameters and their measurement uncertainties.

Contents

About this work

This work was funded through the project CTSimU (WIPANO project 03TNH026A).

CTSimU

WIPANO projects are financed by the German Federal Ministry for Economic Affairs and Energy and managed by Project Management Jülich.

WIPANO

Gefördert durch:

Bundesministerium
für Wirtschaft
und Energie

aufgrund eines Beschlusses
des Deutschen Bundestages

1 Introduction

The WIPANO CTSimU project aims to define a qualification framework for radiographic simulation software. Such software is intended to be used for determining measurement uncertainties for geometrical properties of scanned samples, which arise from the overall procedure of a computed tomography (CT) scan.

On the following pages, a JSON structure (JavaScript Object Notation) is introduced. Its purpose is to contain all relevant parameters of an industrial CT scan, with the intention to

1. document the parameters of any real CT set-up with their respective uncertainties, and
2. completely describe CT scenarios for simulations within the project.

The JSON format was chosen because it is both *human-readable* and *human-writeable*, as well as *machine-readable*, which enables reproducible simulations based on the scenario description files.

A listing of a *full example* can be found at the end of this document. Additional example scenarios, along with simulated projection images, can be found on the Github repository of the file format specification in the `examples` folder:

https://github.com/BAMresearch/ctsimu-scenarios

The line numbers in the code snippets in the following description refer to the line numbers of the full example. It describes a tetrahedral pyramid made of a glass ceramic, placed 20 mm to the "left" of the rotation axis at a slight angle. The rotation stage is symmetrically surrounded by a fixed aluminium frame that does not follow the rotation of the stage (Fig. 1.1). The detector is tilted around its planar normal by 0.023 rad in clockwise direction (as seen from the source). The stage's rotation axis is tilted and performs a wobble motion at an angle of 4°.

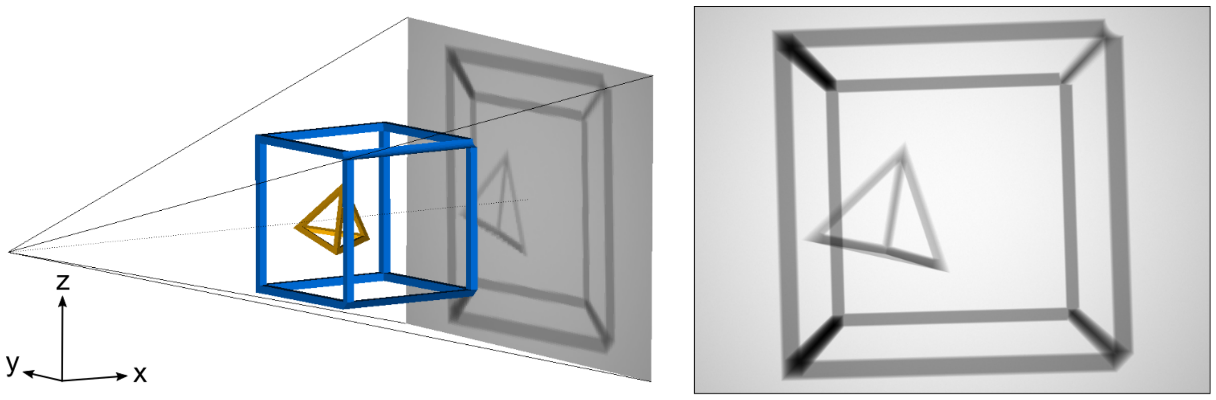

Fig. 1.1: Left: CT setup that is described by the example code snippets in the following sections.
Right: Resulting projection on the detector.

2 General structures

2.1 File structure

The JSON file should be encoded and decoded using the UTF-8 character set, even though all characters in this specification are from the ASCII set to ensure compatibility with other decoding interpretations.

The scenario description consists of the following main sections:

`"file":`
General file information: scenario name, file format version.

`"environment":`
Description of the environment: temperature, atmosphere.

`"geometry":`
Location and orientation of source, detector and sample stage (rotation axis).

`"detector":`
Parameters and characteristics of the detector.

`"source":`
Parameters and characteristics of the X-ray source.

`"samples":`
Definitions of all samples: surface models, location and orientation, size.

`"acquisition":`
Parameters for the CT acquisition: angular steps, flat fields, etc.

`"materials":`
Density and chemical composition of all materials relevant to the scenario.

`"simulation":`
Proprietary parameters specific to a simulation software.

2.2 Parameters

A parameter is given by its `"value"` and may have an associated physical `"unit"`. Additionally, a parameter may come with a measurement `"uncertainty"` and one or more `"drift"` components if its value changes during the CT scan. The following listing shows a full example of a parameter definition. Note that `"drifts"` is an array that may contain multiple drift components.

Depending on the situation, only the parameter properties of relevance must be specified. Irrelevant ones may be omitted or set to `null` at any level.

```
"parameter": {
  "value": 10.0,
  "unit": "mm",
  "uncertainty": {
    "value": 0.1,
    "unit": "mm"
  },
  "drifts": [
    {
      "value": [-100, 100],
      "file":  null,
      "unit":  "mm",
      "known_to_reconstruction": true
    },
    {
      "value": null,
      "file":  "vertical_motion_deviations.csv",
      "unit":  "mm",
      "known_to_reconstruction": false
    }
  ]
}
```

Values & Units

`"value"`:
gives the measured value or the value that should be used by the simulation software.

`"unit"`:
gives the physical unit of the value.

The following units are allowed for length, angle, time, voltage, current, density, temperature, angular velocity and the spatial frequency, and should be interpreted correctly by any parser.

```
"nm"  "deg"  "ms"   "MV"  "uA"  "g/cm^3"  "C"  "deg/s"    "lp/mm"  "px"  "relative"  null
"um"  "rad"  "s"    "kV"  "mA"  "kg/m^3"  "K"  "deg/min"  "lp/cm"
"mm"         "min"  "V"   "A"             "F"  "deg/h"    "lp/dm"
"cm"         "h"                               "rad/s"    "lp/m"
"dm"                                           "rad/min"
"m"                                            "rad/h"
```

The prefix `u` represents the SI prefix µ (10^{-6}). `"relative"` can be used for relative uncertainties or any values that express a fraction of a related measure. For properties without a unit, the keyword `null` is used.

Uncertainty

`"uncertainty": {"value": 0.1, "unit": "mm"}`
: gives a `"value"` for the standard measurement uncertainty and its physical `"unit"`. The intention is to use this to document (or model) a real, physical CT machine.

Drifts

`"drifts":`
: provides an array that may contain an arbitrary number of drift components. Typically, only one drift component is necessary, but in some cases it can be useful to provide more than one drift component for a parameter, especially if some drift contributions shall be unknown to the reconstruction software, whereas others shall be considered during the reconstruction. A typical example would be a helix scan: the vertical movement of the stage along the rotation axis can be modelled as a drift that must be known during the reconstruction. However, inhomogeneities in the vertical motion can be modelled as a second drift component unknown to the reconstruction software.

Drifts are applied for each frame individually once the stage has reached its intended position as described in the *acquisiton section* of the scenario file. Because any drift value describes an absolute deviation from the initial condition at frame 0, they are not accumulated over time. If multiple drift components are defined, they are applied in an additive, sequential manner in the given order.

Each drift component must provide a range of drift values (at least one). These drift values represent absolute deviations from the initial values at the start position (frame 0). They can be provided in the component's `"value"` array or through a single-column CSV `"file"` (i.e., a simple one-column list). If a drift component provides only a single `"value"` different from `null`, this deviation will stay constant throughout the scan.

For dynamic, non-constant drifts, more than one drift value can be provided in the `"value"` array or CSV `"file"`. Ideally, the number of rows in the CSV file would match the number of frames of the scan. If the number of provided drift values does not match the number of frames, the values are assumed to be spread in equidistant steps between start projection (first value) and last projection (last value), and a linear interpolation between neighbouring values is assumed to calculate each frame's deviation value for the parameter.

The physical `"unit"` of the drift values should be specified; otherwise, the main parameter unit is assumed. If the drift refers to a parameter that expects a string (e.g. file name of a spectrum file), the `"value"` array or CSV `"file"` should contain a string for each frame; otherwise, the same equidistant behaviour is assumed as for numerical parameters with the exception that no interpolation takes place. Instead, a string remains valid until the next key frame is reached.

The parameter `"known_to_reconstruction"` (either `true` or `false`) can be used to specify whether the drift should be considered during the reconstruction of the CT scan. This also applies to automatically generated configuration files or the calculation of projection matrices for the reconstruction software. If not specified, the standard value is `true`.

2.3 Formats of referred data files

A scenario description may refer to other data files.

One-dimensional data

For one-dimensional data such as response curves, characteristics files or spectra, the CSV (comma-separated values) or TSV format (tab-separated values) shall be used, with its columns in the order specified in the corresponding sections of this guide.

Two-dimensional data

For **two-dimensional** data such as intensity profiles or bad pixel maps, image files shall be used. The minimum set of supported image file formats should be TIFF and RAW. The image file is given by its `"file"` name which specifies the file's path relative to the JSON file, or provides an absolute path.

```
"intensity_map": {
  "file": {"value": "spot_profile.raw", "drifts": null},
  "type": "float32",
  "dim_x": 301,
  "dim_y": 301,
  "endian": "little",
  "headersize": 0
}
```

For the RAW format, the following additional parameters must be specified:

- The data `"type"`. Valid data types are: `"uint8"`, `"int8"`, `"uint16"`, `"int16"`, `"float32"`, `"float64"`,
- The image size in pixels, given by `"dim_x"` and `"dim_y"`.
- The `"endian"` can be `"little"` or `"big"`.
- As an optional parameter, the `"headersize"` gives the number of bytes to skip at the beginning of the file, or, in other words, the byte offset of the image data.

Data shall be written row-first. For an image with n_x columns and n_y rows, this results in an array with the following one-dimensional index representation, with coordinates (x, y) starting at $(0, 0)$:

$$\text{Index}(x, y) = (n_x \cdot y) + x. \tag{2.1}$$

Bytes are assumed to be written from most-significant to least-significant bit (MSB$_0$).

Three-dimensional data

For **three-dimensional** data, such as a 3D spot intensity profile, a RAW file shall be used. The notation is the same as for 2D data, with the addition of the third voxel dimension given by `"dim_z"`.

```
"intensity_map": {
  "file": {"value": "spot_profile.raw", "drifts": null},
  "type": "float32",
  "dim_x": 201,
  "dim_y": 201,
  "dim_z": 201,
  "endian": "little",
  "headersize": 0
}
```

Data shall be written as row-first, column-second slices. For an image with n_x columns, n_y rows and n_z slices, this results in an array with the following one-dimensional index representation, with coordinates (x, y, z) starting at $(0, 0, 0)$:

$$\mathrm{Index}(x, y, z) = (n_x \cdot n_y \cdot z) + (n_x \cdot y) + x. \tag{2.2}$$

Bytes are assumed to be written from most-significant to least-significant bit (MSB_0).

3 File

The `"file"` section leaves room for specifying a scenario `"name"` and a short `"description"`. Further meta data such as a `"contact"` person, dates of creation and modification, and a file `"version"` can be added here.

The `"file_type"` is a constant string to identify this as a CTSimU scenario and must not be changed. The `"file_format_version"` states which issue of the file format specification is used.

```
"file": {
  "name": "Example Scenario",
  "description": "Tetrahedron in a rigid frame.",

  "contact": "Jane Doe",
  "date_created": "2020-04-23",
  "date_changed": "2023-02-11",
  "version": {"major": 1, "minor": 8},

  "file_type": "CTSimU Scenario",
  "file_format_version": {"major": 1, "minor": 2}
}
```

4 Environment

In the `"environment"` section, the composition of the atmosphere can be described, as well as the environment temperature.

```
"environment": {
  "material_id": "Air",
  "temperature": {
    "value": 20, "unit": "C",
    "uncertainty": {"value": 0.5, "unit": "C"}
  }
}
```

5 Geometry

5.1 Placement of objects in the world coordinate system

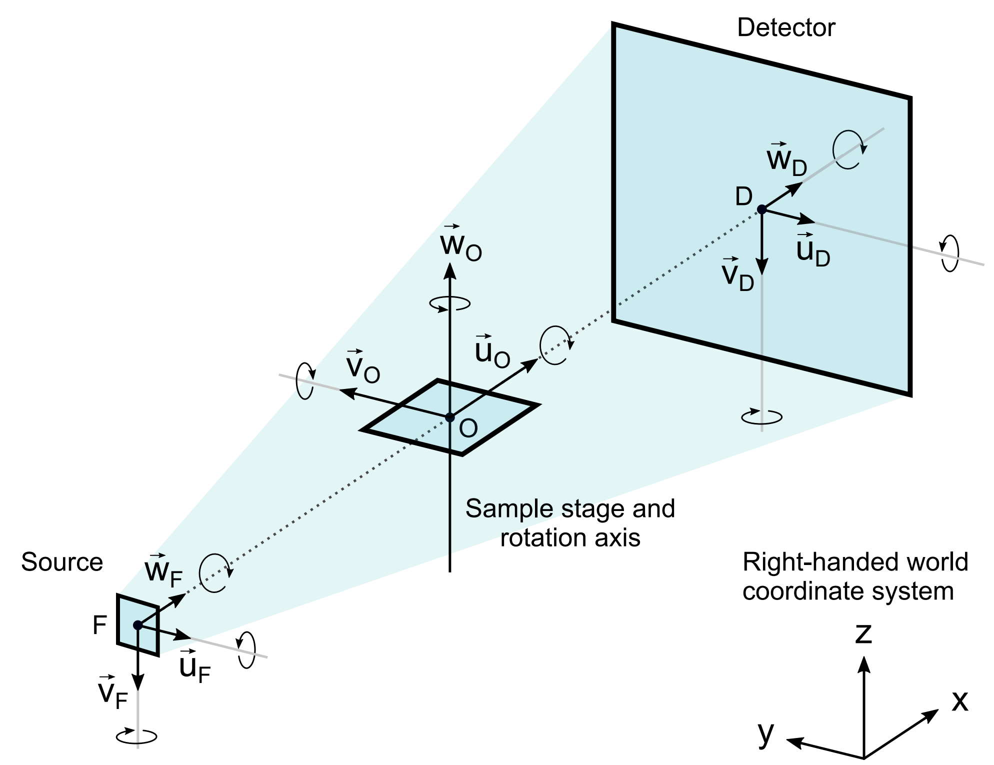

Fig. 5.1: World coordinate system {x, y, z} and local coordinate systems {u, v, w}.

This specification does not define a fixed coordinate system for the CT set-up. The only assumption is that a right-handed world coordinate system {x, y, z} is used, and the local coordinate systems {u, v, w} of source, detector and sample stage are to be placed in this world coordinate system. Fig. 5.1 illustrates the general placement that is used in the following examples.

The `"geometry"` section has three subsections to define the location and orientation of the principal CT scanner components: `"detector"`, `"source"` and `"stage"`. They all share a common set of JSON properties for positioning. A very similar description is used later on to place samples into the scene (see *sample positioning*). The placement description generally consists of the following parts:

"center":
specifies the object's center (x, y, z) in the world coordinate system, i.e., the geometric center of its bounding box. The bounding box is defined as the smallest cuboid (or rectangle, for 2D objects) that completely encloses the object, under the condition that the cuboid's edges are each strictly parallel to one of the coordinate axes of the object's own Cartesian coordinate system. The `"center"` is the origin of the local coordinate system {u, v, w} and also the pivot point for rotational deviations if no alternative pivot point is specified.

"vector_u": and "vector_w":
specify the object's orientation by defining the basis vectors $\vec{u}$ and $\vec{w}$ of the local coordinate system in terms of the world coordinate system {x, y, z}. They are not required to be unit vectors but they must be **orthogonal.**

$\vec{w}$ is usually meant to be a normal vector and $\vec{u}$ is one of the support vectors. $\vec{u}$ and $\vec{v}$ also serve a second meaning as row and column vector of the resulting projection (for the detector) or of a given spot intensity profile image (for the source). See the descriptions below for details.

5.2 Deviations

Deviations are small shifts (translations) or rotations that specify how much an object's position or orientation differs from its ideal placement for the given frame. Deviations can be part of an object's geometry definition. For example, a wobbling rotation axis of the stage could be described as a tilt of its local coordinate system:

```
"deviations": [
  {
    "type": "translation",
    "axis": "x",
    "amount": {"value": 0.5, "unit": "mm"},
    "known_to_reconstruction": false
  },
  {
    "type": "rotation",
    "axis": "w",
    "amount": {"value": 2.3e-2, "unit": "rad"},
    "known_to_reconstruction": true
  }
]
```

"deviations": an array that can be used to specify a sequence of small deviations from the ideal geometry, both rotational or translational. Examples would be tilts of the detector or the rotation axis, or small shifts that should or should not be considered during the reconstruction of the CT scan.

A simulation software should treat the array's deviation components as subsequent transformations of the frame's ideal local coordinate system.

A deviation component should provide the following properties:

- `"type"`: can be either `"translation"` or `"rotation"`.
- `"amount"`: the amount by which to deviate. For a translational deviation, this is the length by which the object should shift in the given direction. For a rotational deviation, it is the angle by which the object should rotate around the given axis (and optionally, the given pivot point). A `"value"` and `"unit"` must be specified for the amount.
- `"axis"`: specifies the direction of the deviation. For translations, it provides the direction of the shift. For rotations, it provides the rotation axis. The axis must not be a unit vector, but its length has no special significance.

  The axis can be given as a designation (name) or as an arbitrary vector:

  – `"x"`, `"y"` or `"z"` are the axes of the **world coordinate system.**

```
"axis": "x"
```

  – `"u"`, `"v"` or `"w"` are the axes of the **local coordinate system** or of the **stage coordinate system** (in case of samples).

```
"axis": "w"
```

  – `"r"`, `"s"` or `"t"` are the local axes of the **sample coordinate system** (see *samples* for details).

  – An **arbitrary axis** can be specified using the vector components from any of the three aforementioned sets (world, local or sample). For example, a vector defined in the **world coordinate system** would look like:

```
"axis": {
  "x": {"value":  2.0},
  "y": {"value": -3.0},
  "z": {"value":  5.3}
}
```

  whereas a vector that is fixed to the **local coordinate system** would look like in the following listing. If this were defined for the stage, the vector would follow the stage's rotation throughout the CT scan.

```
"axis": {
  "u": {"value": 1},
  "v": {"value": 1},
  "w": {"value": 0}
}
```

- `"pivot"`: For rotations, the pivot point defines where the rotation axis is attached. If not specified, the object's center point is assumed to be the rotation's pivot point. Similar to the `"axis"`, the pivot can be expressed in terms of the world coordinate system {x, y, z}, the local (or stage) coordinate system {u, v, w}, or the sample coordinate sytem {r, s, t}. It does not have to be the same coordinate system as for the `"axis"`.

```
"pivot": {
  "u": {"value":  1.0, "unit": "mm"},
  "v": {"value": -2.0, "unit": "mm"},
```

(continues on next page)

(continued from previous page)

```
  "w": {"value":  2.5, "unit": "mm"}
}
```

- `"known_to_reconstruction"`: Whether these deviations are known to the reconstruction software or not depends on the purpose of the scenario and can be specified by setting this property to either `true` or `false`. If not specified, the parameter's standard value is `true`.

```
"known_to_reconstruction": true
```

Note that deviations are applied on a per-frame basis. They do not propagate to the next frame, they do not accumulate. For a given frame, all deviations are assumed to be applied **after the rotating sample stage has arrived** at its intended angular position. This means that if the deviation's `"amount"` is drifting throughout the CT scan, the deviation is calculated anew for each frame, based on its current drift value, and assumed to be applied to the object's ideal position and orientation for the frame (without considering any previous deviations that have been applied in the frames before).

For example, a **static stage tilt** can be modelled by a rotation around on of the axes of the world coordinate system, or a vector that is given in terms of {x, y, z}:

```
"deviations": [
  {
    "comment": "static axis tilt around beam direction",
    "type": "rotation",
    "axis": "x",
    "amount": {"value": 4, "unit": "deg"},
    "known_to_reconstruction": false
  }
]
```

If a **dynamic stage tilt (wobble)** is modelled, it can be given as a rotation around one of its local axes, or any vector given in terms of the local coordinate system {u, v, w}. This way, the deviation's axis will rotate along with the CT stage. Since each deviation is applied on a per-frame basis from the frame's ideal geometry, the axis will precess around a cone.

```
"deviations": [
  {
    "comment": "axis wobble",
    "type": "rotation",
    "axis": {
      "u": {"value": 1},
      "v": {"value": 1},
      "w": {"value": 0}
    }
    "amount": {"value": 4, "unit": "deg"},
    "known_to_reconstruction": false
  }
]
```

5.3 Detector

For this specification, in general, vectors $\vec{u}$ and $\vec{v}$ designate an item's in-plane vectors, whereas $\vec{w}$ usually designates a normal vector. Following this convention, $\vec{u}_\mathrm{D}$ is the detector's **row vector,** pointing from left to right in the resulting projection image (as seen on a computer screen with a pixel coordinate system that has its origin in the upper left corner). $\vec{v}_\mathrm{D}$ is the detector's **column vector,** pointing from top to bottom in the resulting projection image. The orientation of these two vectors directly determines the orientation of the projection image.

Note that for any object only the normal vector $\vec{w}$ and the support vector $\vec{u}$ are given in the JSON file. The detector normal $\vec{w}_\mathrm{D}$ does not have any special meaning and should be arranged such that the detector's row and column vector point in the desired directions.

The size and further properties of the detector are defined later on in the *detector section* of the JSON file.

```
"detector": {
  "center": {
    "x": {"value": 400, "unit": "mm"},
    "y": {"value":   0, "unit": "mm"},
    "z": {"value":   0, "unit": "mm"}
  },

  "vector_u": {
    "x": {"value":  0},
    "y": {"value": -1},
    "z": {"value":  0}
  },
  "vector_w": {
    "x": {"value":  1},
    "y": {"value":  0},
    "z": {"value":  0}
  },

  "deviations": [
    {
      "type": "translation",
      "axis": "x",
      "amount": {"value": 0.5, "unit": "mm"},
      "known_to_reconstruction": false
    },
    {
      "type": "rotation",
      "axis": "w",
      "amount": {"value": 2.3e-2, "unit": "rad"},
      "known_to_reconstruction": true
    },
    {
      "type": "rotation",
      "axis": {
        "x": {"value":  2.0},
```

(continues on next page)

(continued from previous page)

```
        "y": {"value": -3.0},
        "z": {"value":  5.3}
      },
      "pivot": {
        "x": {"value":  2.0, "unit": "mm"},
        "y": {"value": -3.0, "unit": "mm"},
        "z": {"value":  5.3, "unit": "mm"}
      },
      "amount": {"value": 1.3, "unit": "deg"},
      "known_to_reconstruction": true
    }
  ]
}
```

5.4 Source

The source can be modelled either as a cone-beam geometry or a parallel beam geometry. This behaviour is set by the `"type":` property, which can be either `"cone"` or `"parallel"`.

```
"type": "cone",
"type": "parallel"
```

In the case of a parallel beam, its divergence can be specified along both planar axes of the source:

```
"beam_divergence": {
  "u": {"value": 0, "unit": "deg"},
  "v": {"value": 0, "unit": "deg"}
}
```

For a cone beam geometry, this property should be set to `null`:

```
"beam_divergence": null
```

Spatially extended source intensity profiles are modelled as a rectangle. For cone-beam geometries, this will be a very small rectangle on the size scale of the spot size.

For parallel beam geometries, the optical axis of the system is assumed to be the $\vec{w}_\mathrm{F}$ axis of the source, and all rays should be parallel to this axis (apart from beam divergence). This means that the source rectangle in the model should ideally be of a size such that the entire detector is covered by radiation.

If a spot intensity profile is given as an image (see *source* for details), $\vec{u}_\mathrm{F}$ is the image's row vector, pointing from left to right in this image, and $\vec{v}_\mathrm{F}$ is the image's column vector, pointing from top to bottom.

To allow the correct orientation of the intensity profile image, the vector $\vec{w}_\mathrm{F}$ does not necessarily point in the main direction of radiation, but could also point in the opposite direction (due to the restraint of a right-handed coordinate system).

Within the WIPANO CTSimU project, we agreed to the convention of placing the source at (0, 0, 0). It is not a requirement.

```
"source": {
  "type": "cone",
  "beam_divergence": {
    "u": {"value": 0, "unit": "deg"},
    "v": {"value": 0, "unit": "deg"}
  },

  "center": {
    "x": {"value": 0, "unit": "mm"},
    "y": {"value": 0, "unit": "mm"},
    "z": {"value": 0, "unit": "mm"}
  },

  "vector_u": {
    "x": {"value":  0},
    "y": {"value": -1},
    "z": {"value":  0}
  },
  "vector_w": {
    "x": {"value":  1},
    "y": {"value":  0},
    "z": {"value":  0}
  },

  "deviations": []
}
```

5.5 Stage

The normal vector $\vec{w}_O$ of the sample stage specifies the axis of rotation for the CT scan. By default, the sample stage coordinate system and all samples attached to it are meant to rotate around this axis, whereas the source and detector stay still (as for typical industrial CT scanners).

Samples that are placed in the stage coordinate system all take part in the rotation of the sample stage. Samples that are placed in the world coordinate system are fixed during the CT scan (i.e. fixed relative to source and detector, see *samples* for details).

```
"stage": {
  "center": {
    "x": {"value": 275, "unit": "mm"},
    "y": {"value":   0, "unit": "mm"},
    "z": {"value":   0, "unit": "mm"}
  },

  "vector_u": {
    "x": {"value":  1},
    "y": {"value":  0},
    "z": {"value":  0}
```

(continues on next page)

(continued from previous page)

```
  },
  "vector_w": {
    "x": {"value":  0},
    "y": {"value":  0},
    "z": {"value":  1}
  },

  "deviations": [
    {
      "comment": "axis wobble",
      "type": "rotation",
      "axis": "u",
      "amount": {"value": 4, "unit": "deg"},
      "known_to_reconstruction": false
    }
  ]
}
```

6 Detector

6.1 General properties

The following properties can be specified in the detector section of the scenario file.

The model name and the manufacturer, if an existing detector is modelled:

```
"model":        "DT-1",
"manufacturer": "Detector Company"
```

With the `"type"` property, it can be defined whether an **ideal** or a **real** detector is described or shall be simulated. An **ideal** detector is assumed to convert all incident radiation energy linearly into gray values. For an **ideal** detector, no scintillator should be defined in the scenario file. A **real** detector is assumed to take the absorption characteristics and quantum yield of the scintillator material into account.

```
"type": "real",
"type": "ideal"
```

The **number of detector pixels** in directions $\vec{u}_\text{D}$ and $\vec{v}_\text{D}$:

```
"columns": {"value": 200, "unit": "px"},
"rows":    {"value": 150, "unit": "px"},
```

The **pixel pitch** (distance between neighbouring pixels) in directions $\vec{u}_\text{D}$ and $\vec{v}_\text{D}$. The physical dimensions of the active detector area are calculated from the pixel pitch and the number of pixels per row and column.

```
"pixel_pitch": {
  "u": {"value": 1.3, "unit": "mm"},
```

(continues on next page)

(continued from previous page)

```
  "v": {"value": 1.3, "unit": "mm"}
}
```

The **bit depth** of the detector:

```
"bit_depth": {"value": 16}
```

The **integration time** is the exposure time to take one frame. The **dead time** is the time between exposure times, which includes the readout time of the detector.

```
"integration_time": {"value": 0.5, "unit": "s"},
"dead_time":        {"value": 50,  "unit": "ms"},
```

The **image lag** between subsequent frames can be specified, following loosely the definition of *GlobalLag1f* in ASTM E2597. [1]

$$\text{image lag} = \frac{\text{Gray value at first frame radiation fully off}}{\text{Gray value at radiation fully on}} \tag{6.1}$$

```
"image_lag": {"value": 0.05, "unit": null}
```

As opposed to *GlobalLag1f*, the image lag refers to the scenario described in this file, particularly the specified radiation intensity and integration time. Here, it is therefore not treated as a parameter intrinsic to the detector, but describes its lag characteristics only for the specified scenario.

6.2 Gray value characteristics

There are three ways provided to model the gray value characteristics of the detector.

```
"gray_value": {
  "imax":   {"value": 45000},
  "imin":   {"value":   180},

  "factor": {"value": 1.8e13, "unit": "1/J"},
  "offset": {"value":    180, "unit": null},

  "intensity_characteristics_file": {"value": "detector_gray_values.tsv"},
  "efficiency_characteristics_file": {"value": "detector_efficiency.tsv"}
}
```

Min/Max method

The first approach is to specify the average gray value at the maximum intensity of the free beam in the parameter `"imax"` and the average gray value at no incident radiation in the parameter `"imin"`. Between these two values, a linear interpolation based on the incident intensity should take place to calculate the gray value of a pixel. The given values refer to the first frame of the CT scan and are not adapted to different condition in other frames. For example, if the tube current rises during the scan, the maximum free beam gray value in the projection image should rise as well.

Linear response function

The second method allows to specify a linear response function. This function assigns a gray value to the collected energy E (in J) for each pixel.

$$\text{Gray Value} = (\text{factor} \cdot E) + \text{offset} \tag{6.2}$$

The file format allows to provide a `"factor"` and an `"offset"` for this linear function. This method allows the gray values to change with a change in pixel size, tube power, integration time or a change in intensity due to a different focus-detector distance.

This method has precedence over the first method if `"factor"` and `"offset"` are set and not `null`. From eq. (6.2) it becomes clear that the `"factor"` should have the inverse unit of the deposited energy: 1/J.

External characteristics file

For a more general approach, it is also possible to provide an arbitrary gray value characteristics from a CSV or TSV file specified by the parameter `"intensity_characteristics_file"`. The file should contain the columns listed in Tab. 6.1, separated by a comma or tab character.

A linear interpolation is assumed to take place between the given values. Therefore, if higher precision is required, a higher density of values should be provided in this file.

If a valid intensity characteristics CSV file is specified and the parameter is not set to `null`, this method has precedence over the first two methods.

Tab. 6.1: Table structure for intensity characteristics files

Column	Property
1	Energy E in J, collected by a pixel
2	Gray value
3	Gray value uncertainty *(optional)*

Quantum efficiency

The photon **conversion efficiency** (quantum efficiency) of the detector can be provided as a characteristics curve from a file, using the parameter `"efficiency_characteristics_file"`. This file should contain the columns listed in Tab. 6.2, separated by a comma or tab character.

A linear interpolation is assumed to take place between the given values. Therefore, if higher precision is required, a higher density of values should be provided in this file.

The quantum efficiency is assumed to already take the detector's *window* into account, but not additional, external *filters*.

Tab. 6.2: Table structure for efficiency characteristics files

Column	Property
1	Photon energy in keV
2	Quantum efficiency, as the ratio of incident photons to absorbed photons
3	Quantum efficiency uncertainty *(optional)*

6.3 Image quality

Noise

The noise in the projection image can be described by the signal-to-noise ratio (SNR) at the maximum free-beam intensity for the first frame of the CT scan, with no frame averaging applied. This value can be specified in the parameter `"snr_at_imax"`.

```
"noise": {
  "snr_at_imax": {"value": 205.3},
  "noise_characteristics_file": {"value": "detector_noise.tsv"}
}
```

It is also possible to provide an intensity-dependent noise characteristic using a CSV or TSV file. A valid characteristics file has precedence over the SNR at maximum intensity. The characteristics file should contain the columns listed in Tab. 6.3, separated by a comma or tab character.

Tab. 6.3: Table structure for noise characteristics files

Column	Property
1	Mean pixel gray value
2	Signal-to-noise ratio (SNR)
3	SNR uncertainty *(optional)*

Gain

A gain factor can be specified using the "gain" parameter. It does not have to be a number, but can also be the name of the gain mode used. This differs between CT or detector manufacturers.

```
"gain": {"value": 3},
```

This parameter is only for documentation purposes. A simulation software is not expected to take this value into account. Instead, the provided gray value and noise characteristics are assumed to match the gain mode that is recorded here.

Unsharpness

```
"unsharpness": {
  "basic_spatial_resolution": {"value": 0.1, "unit": "mm"},
  "mtf": {"value": "detector_mtf.tsv"}
}
```

There are two ways provided to specify the detector unsharpness:

1. The **basic spatial resolution,** as defined in ASTM E2597 [1], provided using the parameter "basic_spatial_resolution".
2. The **modulation transfer function** (MTF, [2]), provided through a CSV or TSV file. The file name can be given in the parameter "mtf". It should contain the columns listed in Tab. 6.4, separated by comma or tab characters. If a valid MTF is provided, this has precedence over the basic spatial resolution.

Tab. 6.4: Table structure for MTF files

Column	Property
1	Frequency in lp/mm
2	Modulation contrast, from the interval [0, 1]
3	Modulation contrast uncertainty *(optional)*

Bad pixel map

The bad pixel map provided here should be a 2D gray-scale image file with a signed data type (int8, int16, float32 or float64) with a number of columns and rows that matches the detector. A pixel gray value of -1 means that the pixel is working properly. A pixel gray value other than -1 means that the pixel does not function correctly and its value is instead replaced by the provided gray value from the bad pixel map.

```
"bad_pixel_map": {
  "file": {"value": "badpixels.raw", "drifts": null},
  "type": "int16",
  "endian": "little",
  "headersize": 0
}
```

6.4 Scintillator, window & filters

In this section, the scintillator and filter materials for the detector can be defined. The window and filter materials are split into `"front"` and `"rear"` components (e.g., to use as a back panel, mostly to consider backscattering). Any number of windows and filters can be defined. For the materials, only a material ID is given here. The actual material definition and declaration of its chemical composition is found in the *materials section* of the JSON file.

The `"window"` materials refer to the detector's built-in components, whereas `"filters"` are assumed to be additional components that are not originally part of the detector housing. This distinction is made because the detector's *quantum efficiency* (which can be defined in the `"efficiency_characteristics_file"`) already takes the window material into account, but none of the additional, external filters.

Note that front and rear of the detector are not explicitly identified by the detector's normal vector or any other property, but only given implicitly by the side of the detector facing the source.

```
"scintillator": {
  "material_id": "CsI",
  "thickness": {"value": 0.15, "unit": "mm"}
},
"window": {
  "front": [
    {
      "material_id": "Kapton",
      "thickness": {"value": 0.13, "unit": "mm"}
    }
  ],
  "rear": []
},
"filters": {
  "front": [
    {
      "material_id": "Al",
      "thickness": {"value": 0.2, "unit": "mm"}
    },
    {
      "material_id": "Cu",
      "thickness": {"value": 0.1, "unit": "mm"}
    }
  ],
  "rear": [
    {
      "material_id": "Al",
      "thickness": {"value": 2.0, "unit": "mm"}
    }
  ]
}
```

7 Source

7.1 General properties

The following properties can be specified in the source section of the scenario file.

The model name and the manufacturer, if an existing X-ray source is modelled:

```
"model":        "XS-1",
"manufacturer": "X-ray Tube Company",
```

The tube acceleration voltage and the target current:

```
"voltage": {"value": 130, "unit": "kV"},
"current": {"value": 143, "unit": "uA"}
```

7.2 Target

```
"target": {
  "material_id": "W",
  "type": "reflection",
  "thickness": null,
  "angle": {
    "incidence": {"value": 45, "unit": "deg"},
    "emission":  {"value": 45, "unit": "deg"}
  }
}
```

The target material is defined by providing a `"material_id"` that refers to a material declaration in the *materials section* of the file.

The `"type"` can be either a transmission target or a reflection target:

```
"type": "transmission",
"type": "reflection"
```

For a transmission target, its thickness must be specified:

```
"thickness": {"value": 3.0, "unit": "um"}
```

For a reflection target, this parameter should be set to `null`.

The angles of electron incidence and main X-ray emission can be defined and refer to the angles between the target surface and electron beam or main photon emission direction, respectively.

7.3 Spot intensity profile

```
"spot": {
  "size": {
    "u": {"value": 100.0, "unit": "um"},
    "v": {"value": 100.0, "unit": "um"},
    "w": {"value":   0.0, "unit": "um"}
  },
  "sigma": {
    "u": {"value":  50.0, "unit": "um"},
    "v": {"value":  50.0, "unit": "um"},
    "w": {"value":   0.0, "unit": "um"}
  },
  "intensity_map": {
    "file": {"value": "spot_profile.raw", "drifts": null},
    "type": "float32",
    "dim_x": 301,
    "dim_y": 301,
    "dim_z": null,
    "endian": "little",
    "headersize": 0
  }
}
```

The spot intensity profile is specified in the source's `"spot"` section. At first, the `"size"` of the virtual spot rectangle or volume is defined along the three axes of the source coordinate system. If only a two-dimensional spot profile is modelled, the size along the source's normal axis should be set to `0`.

If the spot size is set to `null`, the simulation software is free to choose a size that matches the required (Gaussian) shape.

```
"size": null
```

The shape of the spot can be defined in the following three ways.

Simple Gaussian profiles

A simple Gaussian profile can be modelled by specifying the spatial sigmas σ for each dimension:

$I(\vec{r}) = I_0 \cdot \exp(-|\vec{r}|^2/2\sigma^2)$

```
"sigma": {
  "u": {"value":  50.0, "unit": "um"},
  "v": {"value":  50.0, "unit": "um"},
  "w": {"value":   0.0, "unit": "um"}
},
```

2D images

For a more detailed approach, the intensity profile can also be provided from an external image file. In this case, the $\vec{u}$ vector of the source coordinate system points from left to right in the image, and the $\vec{v}$ vector points from top to bottom, in analogy to the *detector geometry*. The image shall be resized to match the given `"size"` of the spot rectangle, without necessarily retaining the original aspect ratio of the image. The picture is recommended to be a 32 bit float gray-scale image and the pixel values in the interval [0, 1], with 0 meaning no intensity, and 1 meaning full intensity. However, a simulation software must also support gray-scale integer file formats and be able to re-normalize the provided gray values accordingly.

If a valid spot intensity image is provided, this method takes precedence over the previously described specification of Gaussian sigmas.

Further details about referring to *two-dimensional data files* (RAW or TIFF) are given in the section about *General structures*.

3D volumes

To describe a three-dimensional spot profile, a RAW file can be provided. It is recommended to be a 32 bit float volume with values between `0` and `1`, and should otherwise be re-normalized by the simulation software. The lowest value means no intensity, the highest value means maximum intensity. If specified, this method takes precedence over the first two described methods.

The x direction of the given volume points along the $\vec{u}$ vector of the source coordinate system, y points in direction $\vec{v}$, and z in direction $\vec{w}$. The volume shall be resized to match the "`size`" of the spot volume, without necessarily retaining the original aspect ratio of the volume file.

Further details about referring to *three-dimensional data files* (RAW volumes) are given in the section about *General structures*.

7.4 Spectrum

If the spectrum is to be calculated by the simulation software, the following three parameters decide whether only a monochromatic energy scenario is described or a complete spectrum. If a valid spectrum file is provided, it takes precendence over the monochromatic mode. If no spectrum file is provided, a simulation software is free to generate its own spectrum from the given tube parameters.

```
"spectrum": {
  "monochromatic": false,
  "file": {"value": "tube_spectrum_130kV.tsv", "drifts": null}
}
```

The spectrum from the provided file is assumed to be already filtered by the tube's window material, but not yet by any of the filters in front of the tube (as specified under *Tube window and filters*). The CSV or TSV spectrum file should contain the columns listed in Tab. 7.1, separated by a comma or tab character.

Tab. 7.1: Table structure for X-ray spectrum files.

Column	Property
1	Photon energy in keV
2	Number of photons in 1 / (s · sr · mA)
3	Uncertainty in the number of photons *(optional)*

The energy values correspond to the centre of the histogram bins. The interpolation between the bin values shall not be specified here and is left to the simulation software. If a valid spectrum file is specified and the `"file"` parameter is not set to `null`, this has precedence over the spectrum calculated by the simulation software or the monochromatic mode.

7.5 Tube window and filters

The `"window"` material(s) and the additional `"filters"` in front of the tube are specified in two separate JSON arrays. For both, an arbitrary number of materials and thicknesses can be specified. If a *source spectrum* is provided in a file, the spectrum is assumed to be already filtered by all `"window"` materials but not yet by any `"filters"`.

The `"material_id"` refers to the *material definition* in the `"materials"` section of the file. Also, a window and filter `"thickness"` must be provided.

```
"window": [
  {
    "material_id": "Al",
    "thickness": {"value": 4.0, "unit": "mm"}
  }
],
"filters": [
  {
    "material_id": "Brass",
    "thickness": {"value": 0.2, "unit": "mm"}
  },
  {
    "material_id": "Cu",
    "thickness": {"value": 0.17, "unit": "mm"}
  }
]
```

8 Samples

8.1 General properties

Any number of samples can be defined in the `"samples"` array and placed in the scene, either *attached to the rotating sample stage*, or *fixed to the world coordinate system*. A sample object has the following properties.

The sample name:

```
"name": "Tetrahedron"
```

A reference to the model file (e.g. STL or CAD file) that describes the sample geometry:

```
"file": {"value": "tetra.stl", "drifts": null}
```

The unit of length that is used in the model file:

```
"unit": "mm"
```

A scaling factor for each axis of the sample coordinate system, if the model should be resized by a constant factor:

```
"scaling_factor": {
  "r": {"value": 0.75, "drifts": null},
  "s": {"value": 0.75, "drifts": null},
  "t": {"value": 0.75, "drifts": null}
}
```

The material of the sample, given by a `"material_id"` that references a material definition in the *materials section* of the file:

```
"material_id": "Glass Ceramic"
```

8.2 Sample positioning

Sample attached to the stage coordinate system

The sample coordinate system {r, s, t} is equivalent to the surface model's proper {x, y, z} coordinate system (except for the location of the origin). It is specific to each sample and can be placed in the stage coordinate system by providing its center coordinates in terms of $\{u, v, w\}_O$ (see Fig. 8.1 for an illustration). If done so, the sample is attached to the stage and will follow any rotations and translations performed by the stage during the CT scan.

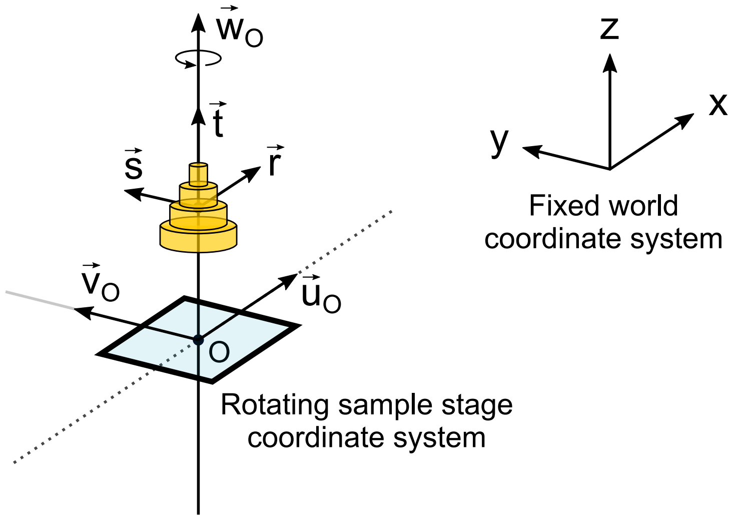
$\vec{w}_O$
$\vec{t}$
$\vec{s}$
$\vec{r}$
$\vec{v}_O$
$\vec{u}_O$
O
Rotating sample stage coordinate system
z
x
y
Fixed world coordinate system

Fig. 8.1: World coordinate system {x, y, z}, stage coordinate system $\{u, v, w\}_O$ and sample coordinate system {r, s, t}.

The description follows the convention that has been established for the *placement of objects in the world coordinate system*, with the following sub-elements of the sample's `"position"` property.

```
"position": {
  "center": {
    "u": {"value":  0, "unit": "mm"},
    "v": {"value": 20, "unit": "mm"},
    "w": {"value":  0, "unit": "mm"}
  },

  "vector_r": {
    "u": {"value":  1},
    "v": {"value":  0},
    "w": {"value":  0}
  },
  "vector_t": {
    "u": {"value":  0},
    "v": {"value": -0.2},
    "w": {"value":  1}
  },

  "deviations": []
}
```

The **center** is given in terms of the stage coordinate system $\{u, v, w\}_O$.

To define the sample's **orientation,** its $\vec{r}$ and $\vec{t}$ vector must also be expressed in terms of the stage coordinate system $\{u, v, w\}_O$:

In analogy to the **deviations** of the principal elements of the scene (see *deviations* in the geometry section), the sample's deviations can also take place along axes of the **sample coordinate system** {r, s, t}.

Fixed sample position in the world coordinate system

If the sample is placed in the fixed world coordinate system, it will not follow any motions performed by the sample stage. Instead, it will stay fixed in {x, y, z} if no custom drifts are specified.

The description is very similar to placing a sample in the stage coordinate system, as described in the *previous section*. The only difference is that the object's center and basis vectors $\vec{r}$ and $\vec{t}$ are now expressed in terms of the world coordinate system {x, y, z}, just like it is done for the source and detector. The following listing gives an example of an aluminium frame around the sample stage that is fixed to the world coordinate system.

```
{
  "name": "Attachment Frame",
  "file": {"value": "frame.stl", "drifts": null},
  "unit": "mm",
  "scaling_factor": {
    "r": {"value": 1.0, "drifts": null},
```

(continues on next page)

(continued from previous page)

```
    "s": {"value": 1.0, "drifts": null},
    "t": {"value": 1.0, "drifts": null}
  },
  "material_id": "Al",
  "position": {
    "center": {
      "x": {"value": 275, "unit": "mm"},
      "y": {"value":   0, "unit": "mm"},
      "z": {"value":   0, "unit": "mm"}
    },

    "vector_r": {
      "x": {"value":  1},
      "y": {"value":  0},
      "z": {"value":  0}
    },
    "vector_t": {
      "x": {"value":  0},
      "y": {"value":  1},
      "z": {"value":  0}
    },

    "deviations": []
  }
}
```

9 Acquisition

9.1 Sample stage rotation

The stage is assumed to perform a rotation around its $\vec{w}_O$ axis during the scan, resulting in a circular sample trajectory. Other trajectories can be modelled using *drifts*, especially drifts of the stage geometry parameters. For example, a helix scan can be modelled by starting with a standard circular trajectory of several rotations (as described in the following), and an additional drift of the stage center's z position. If the scan trajectory is completely described by drifts, the stage rotation described here should be deactivated by setting both `"start_angle"` and `"stop_angle"` to `0`.

For a circular CT scan, the start and stop angle of the sample stage are defined in this section. An angle of `0` refers to the orientation of the stage as defined in the *geometry section* (for frame 0). All other angles express a rotation around the $\vec{w}_O$ axis of the stage. The direction of rotation must be mathematically positive in the case of counter-clockwise acquisition direction (`"CCW"`), and mathematically negative in the case of clockwise acquisition direction (`"CW"`). The stage reaches the start and stop angle by rotating in the given `"direction"` around its normal axis (Fig. 9.1). This means that the `"direction"` parameter affects both the positions of start and stop angle, as well as the direction of rotation in which the CT scan is performed. It also means that **the start angle must always be less than (or equal to) the stop angle.** However, negative angular positions and positions greater than 360° are allowed, as well as an angular coverage of more than a full circle, e.g. to perform multiple rotations during one scan.

```
"start_angle": {"value":   0, "unit": "deg"},
"stop_angle":  {"value": 320, "unit": "deg"}
```

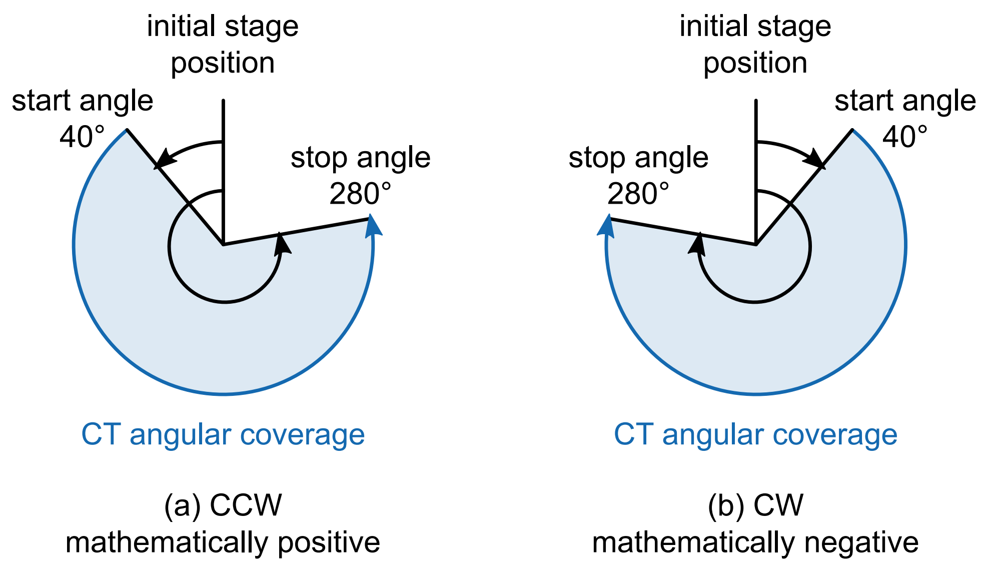

Fig. 9.1: The given start angle and stop angle refer to opposite angular positions in **(a)** counter-clockwise or **(b)** clockwise direction (as seen from "above" the stage). They describe the angular range covered by the CT scan.

The direction of the sample stage rotation can be counter-clockwise (`"CCW"`, mathematically positive) or clockwise (`"CW"`, mathematically negative) around the $\vec{w}_0$ axis:

```
"direction": "CW",
"direction": "CCW"
```

The parameter `"scan_mode"` defines if the rotation stops while a projection is taken, or if it runs continuously.

```
"scan_mode": "stop+go",
"scan_mode": "continuous"
```

The property for scan speed should only be used for continuous-motion scans. If undefined, it may be calculated from the detector's integration and dead time.

```
"scan_speed": {"value": 360, "unit": "deg/h"}
```

For stop&go scans, it should be set to `null`:

```
"scan_speed": null
```

9.2 Frames and projections

Number of projections

The `"number_of_projections"` is also given in the acquisition section.

```
"number_of_projections": 2001
```

Beginning from the start angle, the necessary number of angular steps is performed (in the case of a stop&go scan). It is assumed that a frame is taken before each step (starting with the first frame at the start angle). The parameter `"include_final_angle"` can be set to `true` if the last projection should be taken after the stop angle has been reached, or to `false` if it is meant to be taken at the last step before the stop angle is reached.

```
"include_final_angle": true
```

Frame averaging

The number of frames to be averaged for one projection image can be specified:

```
"frame_average": 3
```

Dark field and flat field acquisition and correction

If dark field and flat field images are acquired along with the projections, their numbers and frame averages can be specified. An `"ideal"` image means that the simulation of noise is to be omitted by the simulation software, possibly in contrast to the *noise* specification in the detector section. The parameter `"correction"` tells whether the projection images already come in a corrected form as a result of the scan (`true`) or if they are taken as uncorrected files (`false`).

```
"dark_field": {
  "number": 1,
  "frame_average": 1,
  "ideal": true,
  "correction": false
},
"flat_field": {
  "number": 3,
  "frame_average": 20,
  "ideal": false,
  "correction": false
}
```

Pixel binning

The number of pixels to bin in directions $\vec{u}$ and $\vec{v}$ of the detector:

```
"pixel_binning": {"u": 1, "v": 1}
```

The binning operation is not described here and left to the software.

9.3 Scattering

This parameter specifies if X-ray scattering should be simulated or not.

```
"scattering": false
```

10 Materials

In all previous declarations, materials are referred by their `"material_id"`. The last section of the JSON file, the materials array, finally contains the specifications for all materials used throughout the scenario description. Each element of the `"materials"` array must have the following properties:

`"id"`:
names the material ID that is used to refer to the material from the other sections of the file.

`"name"`:
is a trivial name that can be given to the material for better identification.

`"density"`:
provides the mass density of the material. The density of a material may drift during the CT scan.

`"composition"`:
gives the chemical composition. It is a JSON array that provides the empirical `"formula"` and `"mass_fraction"` for each component of the material. See examples below.

- The `"formula"` is a string of chemical symbols, each followed by their corresponding **number fraction** (integers or floating-point numbers) within the material. White space between symbols and numbers is allowed, but has no meaning. If the number fraction of an element is 1, the number can be omitted and the next chemical symbol can follow right away. All chemical symbols start with a capital letter, potentially followed by lower-case letters.
- A component's `"mass_fraction"` in a compound material can be specified by a number. All **mass fractions** of a material should add up to 1, but this is not required. A simulation software is assumed to re-normalize the sum of a material's mass ratios to 1 if this condition is not met.

The `"formula"` and `"mass_fraction"` may drift during a CT scan.

10.1 Single-component materials

The following gives an example of a single-component material: pure copper.

```
{
  "id":   "Cu",
  "name": "Copper",
  "density": {"value": 8.92, "unit": "g/cm^3"},
  "composition": [
    {
      "formula": {"value": "Cu"},
      "mass_fraction": {"value": 1}
    }
  ]
}
```

10.2 Multi-component materials

Simple multi-component materials with known number fractions can be modeled using only the `"formula"` parameter.

```
{
  "id":   "Brass",
  "name": "Brass",
  "density": {"value": 8860, "unit": "kg/m^3"},
  "composition": [
    {
      "formula": {"value": "CuZn5"},
      "mass_fraction": {"value": 1}
    }
  ]
}
```

Multi-component materials with different **mass fractions** can be modeled by adding further components with their proper mass ratios to the `"composition"` array. The following example defines a glass ceramic with a mass ratio of 40% Al_2O_3 and 60% SiO_2:

```
{
  "id":   "Glass Ceramic",
  "name": "Glass Ceramic",
  "density": {"value": 2.53, "unit": "g/cm^3", "drifts": null},
  "composition": [
    {
      "formula": {"value": "Al2O3", "drifts": null},
      "mass_fraction": {"value": 0.4, "drifts": null}
    },
    {
      "formula": {"value": "SiO2", "drifts": null},
      "mass_fraction": {"value": 0.6, "drifts": null}
    }
```

(continues on next page)

(continued from previous page)

```
  ]
}
```

Another example for a multi-component material would be the environment’s air:

```
{
  "id":   "Air",
  "name": "Air",
  "density": {"value": 1.293, "unit": "kg/m^3"},
  "composition": [
    {
      "formula": {"value": "N2"},
      "mass_fraction": {"value": 0.7552}
    },
    {
      "formula": {"value": "O2"},
      "mass_fraction": {"value": 0.2314}
    },
    {
      "formula": {"value": "Ar"},
      "mass_fraction": {"value": 0.0128}
    },
    {
      "formula": {"value": "CO2"},
      "mass_fraction": {"value": 0.0006}
    }
  ]
}
```

11 Software-specific properties

If a simulation software requires additional information or simulation parameters, those software-specific properties can be provided in the section called `"simulation"`. For each software, its own sub-section should be created.

```
"simulation": {
  "aRTist": {
    "multisampling_detector": {"value": "3x3"},
    "multisampling_spot": {"value": "30"},
    "spectral_resolution": {"value": 1, "unit": "keV"},
    "scattering_mcray_photons": {"value": 5e7},
    "scattering_image_interval": {"value": 5},
    "long_range_unsharpness": {
      "extension": {"value": 2, "unit":  "mm"},
      "ratio":     {"value": 10, "unit": "%"}
    },
    "primary_energies": false,
    "primary_intensities": false
```

(continues on next page)

(continued from previous page)

```
  }
}
```

12 Full Example

```
{
  "file": {
    "name": "Full example scenario",
    "description": "Tetrahedron in a rigid frame.",

    "contact": "Jane Doe",
    "date_created": "2020-04-23",
    "date_changed": "2023-02-12",
    "version": {"major": 1, "minor": 10},

    "file_type": "CTSimU Scenario",
    "file_format_version": {"major": 1, "minor": 2}
  },

  "environment": {
    "material_id": "Air",
    "temperature": {
      "value": 20, "unit": "C",
      "uncertainty": {"value": 0.5, "unit": "C"}
    }
  },

  "geometry": {
    "detector": {
      "center": {
        "x": {"value": 400, "unit": "mm"},
        "y": {"value":   0, "unit": "mm"},
        "z": {"value":   0, "unit": "mm"}
      },

      "vector_u": {
        "x": {"value":  0},
        "y": {"value": -1},
        "z": {"value":  0}
      },
      "vector_w": {
        "x": {"value":  1},
        "y": {"value":  0},
        "z": {"value":  0}
      },

      "deviations": [
```

(continues on next page)

(continued from previous page)

```
      {
        "type": "translation",
        "axis": "x",
        "amount": {"value": 0.5, "unit": "mm"},
        "known_to_reconstruction": false
      },
      {
        "type": "rotation",
        "axis": "w",
        "amount": {"value": 2.3e-2, "unit": "rad"},
        "known_to_reconstruction": true
      },
      {
        "type": "rotation",
        "axis": {
          "x": {"value":  2.0},
          "y": {"value": -3.0},
          "z": {"value":  5.3}
        },
        "pivot": {
          "u": {"value":  1.0, "unit": "mm"},
          "v": {"value": -2.0, "unit": "mm"},
          "w": {"value":  2.5, "unit": "mm"}
        },
        "amount": {"value": 1.3, "unit": "deg"},
        "known_to_reconstruction": true
      }
    ]
  },

  "source": {
    "type": "cone",
    "beam_divergence": {
      "u": {"value": 0, "unit": "deg"},
      "v": {"value": 0, "unit": "deg"}
    },

    "center": {
      "x": {"value": 0, "unit": "mm"},
      "y": {"value": 0, "unit": "mm"},
      "z": {"value": 0, "unit": "mm"}
    },

    "vector_u": {
      "x": {"value":  0},
      "y": {"value": -1},
      "z": {"value":  0}
    },
    "vector_w": {
```

(continues on next page)

(continued from previous page)

```
        "x": {"value":  1},
        "y": {"value":  0},
        "z": {"value":  0}
      },

      "deviations": []
    },

    "stage": {
      "center": {
        "x": {"value": 275, "unit": "mm"},
        "y": {"value":   0, "unit": "mm"},
        "z": {"value":   0, "unit": "mm"}
      },

      "vector_u": {
        "x": {"value":  1},
        "y": {"value":  0},
        "z": {"value":  0}
      },
      "vector_w": {
        "x": {"value":  0},
        "y": {"value":  0},
        "z": {"value":  1}
      },

      "deviations": [
        {
          "comment": "axis wobble",
          "type": "rotation",
          "axis": {
            "u": {"value": 1},
            "v": {"value": 1},
            "w": {"value": 0}
          },
          "amount": {"value": 4, "unit": "deg"},
          "known_to_reconstruction": false
        }
      ]
    }
  },

  "detector": {
    "model":        "DT-1",
    "manufacturer": "Detector Company",
    "type":         "real",
    "columns":      {"value": 200, "unit": "px"},
    "rows":         {"value": 150, "unit": "px"},
    "pixel_pitch": {
```

(continues on next page)

(continued from previous page)

```
      "u": {"value": 1.3, "unit": "mm"},
      "v": {"value": 1.3, "unit": "mm"}
    },
    "bit_depth":  {"value": 16},
    "integration_time": {"value": 0.5, "unit": "s"},
    "dead_time":  {"value": 50,   "unit": "ms"},
    "image_lag":  {"value": 0.05, "unit": null},
    "gray_value": {
      "imax":   {"value": 45000},
      "imin":   {"value":   180},

      "factor": {"value": 1.8e13, "unit": "1/J"},
      "offset": {"value":    180, "unit": null},

      "intensity_characteristics_file": {"value": "detector_gray_values.tsv"},
      "efficiency_characteristics_file": {"value": "detector_efficiency.tsv"}
    },
    "noise": {
      "snr_at_imax": {"value": 205.3},
      "noise_characteristics_file": {"value": "detector_noise.tsv"}
    },
    "gain": {"value": 3},
    "unsharpness": {
      "basic_spatial_resolution": {"value": 0.1, "unit": "mm"},
      "mtf": {"value": "detector_mtf.tsv"}
    },
    "bad_pixel_map": {
      "file": {"value": "badpixels.raw", "drifts": null},
      "type": "int16",
      "endian": "little",
      "headersize": 0
    },
    "scintillator": {
      "material_id": "CsI",
      "thickness": {"value": 0.15, "unit": "mm"}
    },
    "window": {
      "front": [
        {
          "material_id": "Kapton",
          "thickness": {"value": 0.13, "unit": "mm"}
        }
      ],
      "rear": []
    },
    "filters": {
      "front": [
        {
          "material_id": "Al",
```

(continues on next page)

(continued from previous page)

```
          "thickness": {"value": 0.2, "unit": "mm"}
        },
        {
          "material_id": "Cu",
          "thickness": {"value": 0.1, "unit": "mm"}
        }
      ],
      "rear": [
        {
          "material_id": "Al",
          "thickness": {"value": 2.0, "unit": "mm"}
        }
      ]
    }
  },

  "source": {
    "model":        "XS-1",
    "manufacturer": "X-ray Tube Company",
    "voltage": {"value": 130, "unit": "kV"},
    "current": {"value": 143, "unit": "uA"},
    "target": {
      "material_id": "W",
      "type": "reflection",
      "thickness": null,
      "angle": {
        "incidence": {"value": 45, "unit": "deg"},
        "emission":  {"value": 45, "unit": "deg"}
      }
    },
    "spot": {
      "size": {
        "u": {"value": 100.0, "unit": "um"},
        "v": {"value": 100.0, "unit": "um"},
        "w": {"value":   0.0, "unit": "um"}
      },
      "sigma": {
        "u": {"value":  50.0, "unit": "um"},
        "v": {"value":  50.0, "unit": "um"},
        "w": {"value":   0.0, "unit": "um"}
      },
      "intensity_map": {
        "file": {"value": "spot_profile.raw", "drifts": null},
        "type": "float32",
        "dim_x": 301,
        "dim_y": 301,
        "dim_z": null,
        "endian": "little",
        "headersize": 0
```

(continues on next page)

(continued from previous page)

```
      }
    },
    "spectrum": {
      "monochromatic": false,
      "file": {"value": "tube_spectrum_130kV.tsv", "drifts": null}
    },
    "window": [
      {
        "material_id": "Al",
        "thickness": {"value": 4.0, "unit": "mm"}
      }
    ],
    "filters": [
      {
        "material_id": "Brass",
        "thickness": {"value": 0.2, "unit": "mm"}
      },
      {
        "material_id": "Cu",
        "thickness": {"value": 0.17, "unit": "mm"}
      }
    ]
  },

  "samples": [
    {
      "name": "Tetrahedron",
      "file": {"value": "tetra.stl", "drifts": null},
      "unit": "mm",
      "scaling_factor": {
        "r": {"value": 0.75, "drifts": null},
        "s": {"value": 0.75, "drifts": null},
        "t": {"value": 0.75, "drifts": null}
      },
      "material_id": "Glass Ceramic",
      "position": {
        "center": {
          "u": {"value":  0, "unit": "mm"},
          "v": {"value": 20, "unit": "mm"},
          "w": {"value":  0, "unit": "mm"}
        },

        "vector_r": {
          "u": {"value":  1},
          "v": {"value":  0},
          "w": {"value":  0}
        },
        "vector_t": {
          "u": {"value":  0},
```

(continues on next page)

(continued from previous page)

```
          "v": {"value": -0.2},
          "w": {"value":  1}
        },

        "deviations": []
      }
    },
    {
      "name": "Attachment Frame",
      "file": {"value": "frame.stl", "drifts": null},
      "unit": "mm",
      "scaling_factor": {
        "r": {"value": 1.0, "drifts": null},
        "s": {"value": 1.0, "drifts": null},
        "t": {"value": 1.0, "drifts": null}
      },
      "material_id": "Al",
      "position": {
        "center": {
          "x": {"value": 275, "unit": "mm"},
          "y": {"value":   0, "unit": "mm"},
          "z": {"value":   0, "unit": "mm"}
        },

        "vector_r": {
          "x": {"value":  1},
          "y": {"value":  0},
          "z": {"value":  0}
        },
        "vector_t": {
          "x": {"value":  0},
          "y": {"value":  1},
          "z": {"value":  0}
        },

        "deviations": []
      }
    }
  ],

  "acquisition": {
    "start_angle": {"value":   0, "unit": "deg"},
    "stop_angle":  {"value": 320, "unit": "deg"},
    "direction": "CCW",
    "scan_mode": "stop+go",
    "scan_speed": null,
    "number_of_projections": 21,
    "include_final_angle": true,
    "frame_average": 3,
```

(continues on next page)

(continued from previous page)

```
    "dark_field": {
      "number": 1,
      "frame_average": 1,
      "ideal": true,
      "correction": false
    },
    "flat_field": {
      "number": 3,
      "frame_average": 20,
      "ideal": false,
      "correction": false
    },
    "pixel_binning": {"u": 1, "v": 1},
    "scattering": false
  },

  "materials": [
    {
      "id":   "Air",
      "name": "Air",
      "density": {"value": 1.293, "unit": "kg/m^3"},
      "composition": [
        {
          "formula": {"value": "N2"},
          "mass_fraction": {"value": 0.7552}
        },
        {
          "formula": {"value": "O2"},
          "mass_fraction": {"value": 0.2314}
        },
        {
          "formula": {"value": "Ar"},
          "mass_fraction": {"value": 0.0128}
        },
        {
          "formula": {"value": "CO2"},
          "mass_fraction": {"value": 0.0006}
        }
      ]
    },
    {
      "id":   "Al",
      "name": "Aluminium",
      "density": {"value": 2.6989, "unit": "g/cm^3"},
      "composition": [
        {
          "formula": {"value": "Al"},
          "mass_fraction": {"value": 1}
        }
```

(continues on next page)

(continued from previous page)

```
      ]
    },
    {
      "id":   "W",
      "name": "Tungsten",
      "density": {"value": 19.25, "unit": "g/cm^3"},
      "composition": [
        {
          "formula": {"value": "W"},
          "mass_fraction": {"value": 1}
        }
      ]
    },
    {
      "id":   "CsI",
      "name": "Caesium Iodide",
      "density": {"value": 4.51, "unit": "g/cm^3"},
      "composition": [
        {
          "formula": {"value": "CsI"},
          "mass_fraction": {"value": 1}
        }
      ]
    },
    {
      "id":   "Brass",
      "name": "Brass",
      "density": {"value": 8860, "unit": "kg/m^3"},
      "composition": [
        {
          "formula": {"value": "CuZn5"},
          "mass_fraction": {"value": 1}
        }
      ]
    },
    {
      "id":   "Cu",
      "name": "Copper",
      "density": {"value": 8.92, "unit": "g/cm^3"},
      "composition": [
        {
          "formula": {"value": "Cu"},
          "mass_fraction": {"value": 1}
        }
      ]
    },
    {
      "id":   "Kapton",
      "name": "Kapton 50HN (Polyimide film)",
```

(continues on next page)

(continued from previous page)

```
      "density": {"value": 1.42, "unit": "g/cm^3"},
      "composition": [
        {
          "formula": {"value": "C16H14N4O4"},
          "mass_fraction": {"value": 1}
        }
      ]
    },
    {
      "id":   "Glass Ceramic",
      "name": "Glass Ceramic",
      "density": {"value": 2.53, "unit": "g/cm^3", "drifts": null},
      "composition": [
        {
          "formula": {"value": "Al2O3", "drifts": null},
          "mass_fraction": {"value": 0.4, "drifts": null}
        },
        {
          "formula": {"value": "SiO2", "drifts": null},
          "mass_fraction": {"value": 0.6, "drifts": null}
        }
      ]
    }
  ],

  "simulation": {
    "aRTist": {
      "multisampling_detector": {"value": "3x3"},
      "multisampling_spot": {"value": "30"},
      "spectral_resolution": {"value": 1, "unit": "keV"},
      "scattering_mcray_photons": {"value": 5e7},
      "scattering_image_interval": {"value": 5},
      "long_range_unsharpness": {
        "extension": {"value": 2, "unit":  "mm"},
        "ratio":     {"value": 10, "unit": "%"}
      },
      "primary_energies": false,
      "primary_intensities": false
    }
  }
}
```

13 Metadata Files

Metadata files are used to document the result of a CT scan: the projection image data, as well as the reconstructed image data.

```
{
  "file":                 {
    "name":                 "01_full_example",
    "description":          "Tetrahedron in a rigid frame.",
    "contact":              "Jane Doe",
    "date_created":         "2023-09-05",
    "date_changed":         "2023-09-05",
    "file_type":            "CTSimU Metadata",
    "file_format_version": {"major": 1, "minor": 2}
  },
  "output":               {
    "system":        "aRTist v2.12.6-0-g30fb1d33, CTSimU Scenario Loader 1.2.3",
    "date_measured": "2023-09-05",
    "projections":   {
      "filename":      "../projections/corrected/01_full_example_%04d.tif",
      "number":        21,
      "frame_average": 3,
      "max_intensity": 45000.430564932205,
      "datatype":      "uint16",
      "byteorder":     "little",
      "headersize":    {"file":  0, "image": 0},
      "dimensions":    {
        "x": {"value": 200, "unit": "px"},
        "y": {"value": 150, "unit": "px"}
      },
      "pixelsize":     {
        "x": {"value": 1.3, "unit": "mm"},
        "y": {"value": 1.3, "unit": "mm"}
      },
      "dark_field":    {
        "number":        1,
        "frame_average": 1,
        "filename":      "../projections/01_full_example_dark.tif",
        "projections_corrected": true
      },
      "flat_field":    {
        "number":        3,
        "frame_average": 20,
        "filename":      "../projections/01_full_example_flat_%04d.tif",
        "projections_corrected": true
      },
      "bad_pixel_map": {
        "filename": null,
        "projections_corrected": false
      }
```

(continues on next page)

(continued from previous page)

```
    },
    "tomogram":      {
      "filename":   null,
      "datatype":   "float32",
      "byteorder":  "little",
      "headersize": {
        "file":  0,
        "image": 0
      },
      "dimensions": {
        "x": {"value": 200, "unit":  "px"},
        "y": {"value": 200, "unit":  "px"},
        "z": {"value": 150, "unit":  "px"}
      },
      "voxelsize":  {
        "x": {"value": 0.893752963236774, "unit":  "mm"},
        "y": {"value": 0.893752963236774, "unit":  "mm"},
        "z": {"value": 0.893752963236774, "unit":  "mm"}
      }
    }
  },
  "acquisition_geometry": {
    "path_to_CTSimU_JSON": "../01_full_example.json"
  },
  "reconstruction":       {
    "software": null,
    "settings": {}
  },
  "simulation":           {
    "full_simulation":     true,
    "compute_detector":    true,
    "compute_xray_source": true,
    "load_samples":        true,
    "set_multisampling":   true,
    "set_scattering":      true,
    "multisampling":       {
      "source":   "30",
      "detector": "3x3"
    },
    "scattering":          {
      "on":             false,
      "image_interval": null,
      "photons":        null
    },
    "ctsimu_scenario":     {}
  }
}
```

14 Parameter List

Tab. 14.1 gives an overview of all parameters in this file format specification. It lists the expected type and if the parameter supports drifts.

Tab. 14.1: Parameter overview and support for drifts.

Parameter	Type	Drifts
File		
`file name`	string	no
`file description`	string	no
`file contact`	string	no
`file date_created`	string	no
`file date_changed`	string	no
`file version major`	integer	no
`file version minor`	integer	no
`file file_type`	string	no
`file file_format_version major`	integer	no
`file file_format_version minor`	integer	no
Environment		
`environment material_id`	string	no
`environment temperature`	float	yes
Geometry		
`geometry detector center x/y/z`	float	yes
`geometry detector vector_u x/y/z`	float	yes
`geometry detector vector_w x/y/z`	float	yes
`geometry detector deviations`	array	see Tab. 14.2
`geometry source type`	string	no
`geometry source beam_divergence u/v`	float	yes
`geometry source center x/y/z`	float	yes
`geometry source vector_u x/y/z`	float	yes
`geometry source vector_w x/y/z`	float	yes
`geometry source deviations`	array	see Tab. 14.2
`geometry stage center x/y/z`	float	yes
`geometry stage vector_u x/y/z`	float	yes
`geometry stage vector_w x/y/z`	float	yes
`geometry stage deviations`	array	see Tab. 14.2
Detector		
`detector model`	string	no
`detector manufacturer`	string	no
`detector type`	string	no
`detector columns`	integer	yes
`detector rows`	integer	yes
`detector pixel_pitch u/v`	float	yes
`detector bit_depth`	integer	no
`detector integration_time`	float	yes
`detector dead_time`	float	yes
`detector image_lag`	float	yes
`detector gray_value imax`	float	yes
`detector gray_value imin`	float	yes

continues on next page

Tab. 14.1 – continued from previous page

Parameter	Type	Drifts
`detector gray_value factor`	float	yes
`detector gray_value offset`	float	yes
`detector gray_value intensity_characteristics_file`	string	yes
`detector gray_value efficiency_characteristics_file`	string	yes
`detector noise snr_at_imax`	float	yes
`detector noise noise_characteristics_file`	string	yes
`detector gain`	anything	no
`detector unsharpness basic_spatial_resolution`	float	yes
`detector unsharpness mtf`	string	yes
`detector bad_pixel_map file`	string	yes
`detector bad_pixel_map type`	string	no
`detector bad_pixel_map endian`	string	no
`detector bad_pixel_map headersize`	integer	no
`detector scintillator material_id`	string	no
`detector scintillator thickness`	float	yes
`detector window front`	array	–
`detector window front material_id`	string	no
`detector window front thickness`	float	yes
`detector window rear`	array	–
`detector window rear material_id`	string	no
`detector window rear thickness`	float	yes
`detector filters front`	array	–
`detector filters front material_id`	string	no
`detector filters front thickness`	float	yes
`detector filters rear`	array	–
`detector filters rear material_id`	string	no
`detector filters rear thickness`	float	yes
Source		
`source model`	string	no
`source manufacturer`	string	no
`source voltage`	float	yes
`source current`	float	yes
`source target material_id`	string	no
`source target type`	string	no
`source target thickness`	float	yes
`source target angle incidence`	float	yes
`source target angle emission`	float	yes
`source spot size u/v/w`	float	yes
`source spot sigma u/v/w`	float	yes
`source spot intensity_map file`	string	yes
`source spot intensity_map type`	string	no
`source spot intensity_map dim_x/y/z`	integer	no
`source spot intensity_map endian`	string	no
`source spot intensity_map headersize`	integer	no
`source spectrum monochromatic`	boolean	no
`source spectrum file`	string	yes
`source window`	array	–
`source window material_id`	string	no

continues on next page

Tab. 14.1 – continued from previous page

Parameter	Type	Drifts
source window thickness	float	yes
source filters	array	–
source filters material_id	string	no
source filters thickness	float	yes
Samples		
samples	array	–
samples name	string	no
samples file	string	yes
samples unit	string	no
samples scaling_factor r/s/t	float	yes
samples material_id	string	no
samples position center u/v/w/x/y/z	float	yes
samples position vector_r u/v/w/x/y/z	float	yes
samples position vector_t u/v/w/x/y/z	float	yes
samples position deviations	array	see Tab. 14.2
Acquisition		
acquisition start_angle	float	no
acquisition stop_angle	float	no
acquisition direction	string	no
acquisition scan_mode	string	no
acquisition scan_speed	float	yes
acquisition number_of_projections	integer	no
acquisition include_final_angle	boolean	no
acquisition frame_average	integer	no
acquisition dark_field number	integer	no
acquisition dark_field frame_average	integer	no
acquisition dark_field ideal	boolean	no
acquisition dark_field correction	boolean	no
acquisition flat_field number	integer	no
acquisition flat_field frame_average	integer	no
acquisition flat_field ideal	boolean	no
acquisition flat_field correction	boolean	no
acquisition pixel_binning u/v	integer	no
acquisition pixel_binning u/v	integer	no
acquisition scattering	boolean	no
Materials		
materials	array	–
materials id	string	no
materials name	string	no
materials density	float	yes
materials composition formula	string	yes
materials composition mass_fraction	float	yes

Objects in the scene may come with *geometrical deviations*. These are defined as an array of deviation objects. The structure of deviation objects is listed in Tab. 14.2.

Tab. 14.2: Deviation objects. The `"axis"` my be defined as a string or object.

Parameter	Type	Drifts
`type`	string	no
`axis`	string	no
`axis`	object	–
`axis x/y/z`	float	yes
`axis u/v/w`	float	yes
`axis r/s/t`	float	yes
`pivot x/y/z`	float	yes
`pivot u/v/w`	float	yes
`pivot r/s/t`	float	yes
`amount`	float	yes
`known_to_reconstruction`	boolean	no

15 Specification Changes

The following listing contains the changes to the file format in its prior versions.

15.1 Version 1.2

- Detector: new distinction between its `"window"` and additional `"filters"`. All `"filters"` defined in previous versions of the file format should rather be re-interpreted as windows now (sec. *Scintillator, window & filters*).

15.2 Version 1.1

- Material `"composition"` is now an array that can contain the components of a compound material, each with their specific `"mass_fraction"` (sec. *Materials*).
- Bad pixel maps are now defined in their new `"file"` parameter (which can come with its own drift). The `"value"` property has therefore been removed (sec. *Bad pixel map*).

15.3 Version 1.0

- Introduced a more general concept to specify geometric deviations (sec. *Deviations*). Instead of a strict order of rotations around the axes of the local coordinate system, an arbitrary order of translations and rotations around arbitrary axes and pivot points can be specified in a `"deviations"` array.
- Re-introduced American spelling to keep consistency with most other software and IT standards, also in regard to the world of Python packages. Therefore: `"centre"` → `"center"`, `"grey"` → `"gray"`.
- Added `"endian"` and `"headersize"` for RAW data files (sec. *Two-dimensional data* and *Three-dimensional data*). Also: more options for the data `"type"`.
- The detector's quantum efficiency can no longer be expressed by a single number because it would be meaningless (the gray value characteristics already takes it into account, and

it would currently have no effect on other parameters). For an energy-resolved quantum efficiency, an external characteristics file is still supported (sec. *Quantum efficiency*).

- The parameter `"mtf10_frequency"` for the detector unsharpness has been removed. Instead, a full MTF should be defined in an external file. For single-value unsharpness definitions, the `"basic_spatial_resolution"` should be used (sec. *Unsharpness*).
- Bad pixel maps: properly working pixels are now encoded with the value `-1` instead of `0`; the latter would now mean a defect, completely black pixel (sec. *Bad pixel map*).
- Detector `"gain"` is now purely reserved for the documentation of a real CT scan parameter, but has no meaning for a simulation. All detector parameters in the JSON file are assumed to refer to the gain setting specified in the `"gain"` parameter (sec. *Gain*). The gain property `"scale_signal_and_noise"` has been removed because it was not well-defined.
- X-ray source: the parameters `"bremsstrahlung"` and `"characteristic"` have been removed (sec. *Spectrum*).

15.4 Version 0.9

- Introduced a new concept for drifts (sec. *Drifts*).
- Uncertainties are now described in single `"uncertainty"` blocks that contain a `"value"` and `"unit"` element, instead of defining two separate elements for `"uncertainty"` and `"uncertainty_unit"` (sec. *Uncertainty*).
- Removed noise FWHM as an alternative measure to the SNR (sec. *Noise*) to avoid inconsistencies.
- Changed detector parameter name `"sharpness"` to `"unsharpness"` (sec. *Unsharpness*).

15.5 Version 0.8

- Added the simulation software-specific parameter section `"simulation"` (sec. *Software-specific properties*).
- Noise parameters (SNR, FWHM, sec. *Noise*) now specifically refer to one frame, not an averaged projection.

15.6 Version 0.7

- Added further meta data fields to the `"file"` section (sec. *File*) and changed the previous parameter designations `"type"` and `"version"` to `"file_format_type"` and `"file_format_version"` to make room for their meta data equivalents.
- The parameter `"frames_to_average"` has been renamed to `"frame_average"` (sec. *Frame averaging*).
- Introduced acquisition parameters for dark field and flat field images (sec. *Dark field and flat field acquisition and correction*).
- All uncertainties are assumed to be standard measurement uncertainties.

15.7 Version 0.6

- Added `"description"` to `"file"` section (sec. *File*).
- Split tube `"window"` and `"filters"` into two separate arrays (sec. *Tube window and filters*), and now demand that any tube spectrum provided through a file is only filtered by the window material, but none of the additional filters that can be placed in front of the tube (sec. *Spectrum*).
- Added the detector's noise FWHM as an equivalent measure next to the SNR (sec. *Noise*).
- Sample scaling factors now correctly refer to the sample coordinate system {r, s, t} instead of the stage coordinate system {u, v, w} (sec. *General properties*).

15.8 Version 0.5

- Added detector `"type"`, options are: `"real"` and `"ideal"` (sec. *General properties*).
- Spectra provided through CSV files are assumed to be filtered by all tube filters (sec. *Spectrum*).

15.9 Version 0.4

- Grey value characteristics of the detector is now based on total collected energy E (in J) for each pixel instead of incident energy density (sec. *Gray value characteristics*).
- Introduced `"environment"` section (sec. *Environment*), with the environment temperature and (atmospheric) composition.
- Introduced `"pixel_binning"` to `"acquisition"` parameters (sec. *Pixel binning*).
- The time index for drift trajectory files is now the frame number instead of the angular position to avoid floating point rounding discrepancies between description file and CT simulation (sec. *Drifts*).
- Under `"acquisition"`, the parameter `"angular_steps"` has been replaced by the parameter `"number_of_projections"` (sec. *Number of projections*).
- Added support for RAW files for any 2D or 3D data.
- 2D and 3D spot intensity profiles are now both handled by the parameter `"intensity_map"`, which has been extended to support RAW files in both cases (sec. *Spot intensity profile*).

15.10 Version 0.3

- New axis designations $\vec{u}$, $\vec{v}$, $\vec{w}$ for local coordinate systems, whereas $\vec{x}$, $\vec{y}$, $\vec{z}$ always designate the axes of the world coordinate system (sec. *Geometry*).
- Introduced sample coordinate system {r, s, t} (sec. *Sample positioning*).
- New geometry parameter `"deviations"` contains all deviations from the ideal (known) geometry. Deviation parameters can be unknown to the reconstruction. (Sec. *Geometry*)
- Introduced British spelling: `"center"` $\rightarrow$ `"centre"`, `"gray"` $\rightarrow$ `"grey"`.
- Removed FOD and FDD to prevent inconsistencies.

- Removed detector `"width"` and `"height"` to prevent inconsistencies.
- Removed option for gain characteristics file.
- Detector: filters and rear panel merged into `"front"` and `"rear"` filters (sec. *Scintillator, window & filters*).
- `"image_lag"` moved from `"acquisition"` to `"detector"` (sec. *General properties*).
- Moved `"integration_time"` from `"acquisition"` to `"detector"`, because other detector characteristics (such as image lag and possibly noise) are specific to it (sec. *General properties*).
- Source: removed number of photons from `"spectrum"` to prevent inconsistencies with tube current.
- Introduced uncertainties to CSV files.
- Removed option for purely random drifts because it is not reproducible and was so far only limited to Gaussian distributions. Drift trajectories from CSV files remain the most general and reproducible approach.

References

[1] ASTM Subcommittee E07.01. ASTM E2597 / E2597M-14, Standard Practice for Manufacturing Characterization of Digital Detector Arrays. *ASTM International, West Conshohocken, PA, 2014, www.astm.org*, 2014. doi:10.1520/E2597_E2597M-14.

[2] Kurt Rossmann. Point Spread-Function, Line Spread-Function, and Modulation Transfer Function: Tools for the Study of Imaging Systems. *Radiology*, 93(2):257–272, August 1969. doi:10.1148/93.2.257.