Luxendo Image

In the following, the "Luxendo Image" (`.lux.h5`) format is defined and explained. Its purpose is to provide an open, transparent image format for 3d SPIM (*Single Plane Illumination Microscopy*) images acquired by Luxendo microscopes, as well as for output images from further processing / analysis. We designed this format to contain all the relevant acquisition, viewing, and processing metadata in a clear and flexible form. Luxendo Image is based on the widely used, open HDF5 (`.h5`) format, to achieve broad compatibility.

Table of contents

Goals

The main goals for the Luxendo Image format are:

- Stability and long-term compatibility
- Avoid data duplication and provide flexibility by linking to the raw data from a "main" file.
- Enable everyone to load and create Luxendo Image files in their own software and custom processing / analysis pipelines.
- Metadata:
  - Clearly distinguish between "acquisition" and "derived" metadata to transparently keep track of metadata, e.g. when multiple images are fused.
  - Use the same form of metadata for both newly acquired images as well as processed images so that already processed images can be fed back into pipelines for further processing.

Structure

The experiment folder

An experiment has a folder structure similar to the example below.

- 📁YYYY-MM-DD_hhmmss
  - 📁 processed
  - 📁 raw
    - 📁 stack_0-x00-y00_channel_0_obj_left
    - 📁 stack_0-x00-y00_channel_0_obj_right
    - 📁 stack_0-x00-y01_channel_0_obj_left
      - Cam_left_00000.json
      - Cam_left_00000.lux.h5
      - Cam_left_00001.json
      - Cam_left_00001.lux.h5
    - 📁 stack_0-x00-y01_channel_0_obj_right
      - Cam_right_00000.json
      - Cam_right_00000.lux.h5
      - Cam_right_00001.json
      - Cam_right_00001.lux.h5
  - 📄 main.lux.h5

The root folder is automatically generated, and it's named with a unique timestamp.

The `processed` subfolder contains the output of the Image Processor. This folder is only present is there was at least one Image Processor task started for the experiment.

The `raw` subfolder contains the raw image data as well as the metadata, grouped by the stack and channel IDs. There are two distinct file types present:

- image data files: `.lux.h5` extension
- metadata file: `.json` extension

In addition, the root folder contains "wrapper" or "main" files, which link to the image data in the `raw` folder. The wrapper files can provide a compatibility interface to multiple image analysis software.

To move or copy the images to a different location, the experiment folder should be moved/copied as a whole, to not break the links in the main files pointing to the image data.

The individual image files (`.lux.h5`) should also not be renamed manually or moved to different subfolders inside the experiment folder, since this would break the links as well.

🔝 back to top

Image Data

A Luxendo Image (`.lux.h5`) file is a regular `.h5` (HDF5) file that has a defined internal structure and metadata.

The image data is stored in a h5 dataset called `"Data"` , as 3d array with 16-bit unsigned-integer (`uint16`) elements.

If there are multiple datasets in one `.lux.h5` file, then they correspond to different resolution layers of a resolution pyramid:

- `"Data"` : dataset containing the highest-resolution image data.
- `"Data_2_2_2"` : dataset containing image data that is downsampled in each dimension by a factor 2.
- `"Data_3_3_3"` : image data downsampled by factor 3 in each dimension.

and so on.

The order of the downsampling factors is: `Data_<width>_<height>_<depth>` , referring to the width, height, and depth of the image. And they are always *integer* factors.

The highest-resolution image data is typically chunked with chunk size `64*64*64` , whereas lower-resolution data may be chunked with chunk size `32*32*32` .

back to top

Metadata

A `.lux.h5` file has a h5 dataset called `metadata` that contains JSON data `{"processingInformation": ...}` . The entry `"processingInformation"` has all the relevant metadata (in JSON form). This metadata is divided into "acquisition" metadata and "derived" metadata.

Its high-level structure is:

```
"processingInformation": {
  // ... all "derived" metadata (e.g. transforms) ...
  "acquisition": [
    // ... all "acquisition" metadata (i.e. microscope settings, geometry) ...
  ]
}
```

In this structure:

- All the acquisition settings are inside the `"acquisition"` field.

- Metadata outside `"acquisition"` is derived / calculated from the `"acquisition"` metadata or stems from processing (e.g. transforms from image registration).
- The `"acquisition"` field (list) can have *multiple* items corresponding to different acquired images, e.g. when the given Luxendo Image contains image data that results from a fusion of multiple images.

Note that the same form of `"processingInformation"` is used for both newly acquired images as well as processed / fused images. Thus, already processed images can be fed back into a processing pipeline for further processing.

The concrete form of the `"processingInformation"` metadata is the following (values are just examples):

```
"processingInformation": {
    "version": "1.0.0",
    "image_id": "2023-03-22T13:21:21.569Z-f2ce69b8-50bd-4569-81e2-9374a72bdb1c",
    "sources": ["Luxendo MuVi-SPIM, Embedded v3.0.0, serial-nr: 10007"],
    "contains_beads": false,
    "time_point": "00003",
    "channel": "1",
    "channel_description": "Green-22",
    "pm": "5",
    "stack": "0",
    "stack_description": "top-left",
    "objective": "left",
    "camera": "left",
    "repetition": 0,
    "voxel_size_um": {
        "width": 0.40625,
        "height": 0.40625,
        "depth": 1
    },
    "image_size_vx": {
        "width": 2048,
        "height": 2048,
        "depth": 441
    },
    "affine_to_sample": [
        {
            "matrix": [
                [1, 0, 0],
                [0, 1, 0],
                [0, 0, 1]
            ],
            "translation": [-1023.5, -1023.5, -220],
            "type": "same-stack:center"
        },
        {
            "matrix": [
                [-0.40625, 0, 0],
                [0, 0.40625, 0],
                [0, 0, 1]
            ],
            "translation": [0, 0, 0],
            "type": "same-stack:geometry"
        },
        {
            "matrix": [
                [1, 0, 0],
                [0, 1, 0],
                [0, 0, 1]
            ],
            "translation": [150, 3200, 380],
            "type": "inter-stack:translation"
        },
        {
            "matrix": [
                [1, 0, 0],
                [0, 1, 0],
                [0, 0, 1]
            ],
            "translation": [-36, 0, -297],
            "type": "inter-stack:rotation-offset"
        },
        {
            "matrix": [
                [-0.8660254037844387l,-0, -0.49999999999999994],
                [0, 1, 0],
                [0.49999999999999994, 0, -0.8660254037844387l]
            ],
            "translation": [246, 0, 488],
            "type": "inter-stack:rotation"
        }
    ],
    "detection_directions": [
        [0, 0, 1]
    ],
    "acquisition": [
        {
            "image_id": "2023-03-22T13:21:21.569Z-f2ce69b8-50bd-4569-81e2-9374a72bdb1c",
            "microscope_type": "MuVi-SPIM",
            "serial_number": "10047",
            "embedded_version": "3.0.0",
            "edits": [],
            "contains_beads": false,
            "time_point": "00003",
            "channel": "1",
            "channel_description": "Green-22",
            "pm": "5",
            "stack": "0",
            "stack_description": "top-left",
            "objective": "left",
            "camera": "left",
            "repetition": 0,
            "number_planes": 50,
            "stage_positions": [
                {
                    "name": "x",
                "start_um": 3200,
                "end_um": 3200,
                "movement": {
                    "direction": [0, -1, 0],
                    "offset_um": [0, 0, 0]
                },
                "type": "linear"
            },
            {
                "name": "y",
                "start_um": -150,
                "end_um": -150,
                "movement": {
                    "direction": [1, 0, 0],
                    "offset_um": [0, 0, 0]
                },
                "type": "linear"
            },
            {
                "name": "z",
                "start_um": 160,
                "end_um": 600,
                "movement": {
                    "direction": [0, 0, -1],
                    "offset_um": [0, 0, 0]
                },
                "type": "linear"
            },
            {
                "name": "r",
                "start_deg": 150,
                "end_deg": 150,
                "movement": {
                    "direction": [0, 1, 0],
                    "offset_um": [-246, 0, -488]
                },
                "type": "rotation"
            }
        ],
        "image_plane_vectors": {
            "cam_left_to_right": [-1, 0, 0],
            "cam_top_to_bottom": [0, 1, 0]
        },
        "refractive_index": 0,
        "detection": {
            "wavelength_nm": 0,
            "numerical_aperture": 0.8000000000000004,
            "magnification": 16,
            "cam_pixel_size_um": {
                "width": 6.5,
                "height": 6.5
            },
            "sensor_size_px": {
                "width": 2048,
                "height": 2048
```

```
                    },
                    "crop_offset_px": {
                        "left": 0,
                        "top": 0
                    },
                    "crop_size_px": {
                        "width": 2016,
                        "height": 1050
                    },
                    "direction": [0.0, 0.0, 1.0]
                },
                "illuminations": [
                  {
                    "objective": "front",
                    "direction": [0.0, 1.0, 0.0]
                  },
                  {
                    "objective": "back",
                    "direction": [0.0, -1.0, 0.0]
                  }
                ],
                "time_stamps": ["2021-04-19T14:45:38.073876Z", "2021-04-19T14:45:38.075876Z"
            }
        ]
    }
```

In the following, the different fields are explained:

"Derived" metadata

This metadata is derived (calculated) from the `acquisition` metadata.

- `version` : Version number of the Luxendo Image format.
- `image_id` : [Optional] Unique identifier for the image (typically composed of time stamp of first image plane and system UUID).
- `sources` : List of sources (software) that generated / contributed to the given Luxendo Image file.
- `contains_beads` : Whether the image contains beads (relevant e.g. for image registration).
- `time_point` : Name of the time point corresponding to the image. A valid name for a time point is an (unsigned) integer (in the form of a string), including zero, that may also have leading zeros (e.g. `"00003"`). When multiple images are merged into one image (e.g. by image fusion), the `time_point` of the first image is used as `time_point` for the merged image. But in the `"acquisition"` field, in each acquisition-metadata entry for the respective images that were merged, their original `time_point` will be kept.
- `channel` : Name of the "channel". A channel refers to a combination of different settings such as illumination and detection wavelengths and other settings that do *not* decide the positioning of the sample.
- `channel_description` : [Optional] User-provided, short description of the channel.

- `pm` : [Optional: missing if no PM set up] Name (ID) of a photo-manipulation (PM) object that was applied during acquisition of this image stack. This PM-object ID uniquely refers to the parameter settings used for the PM, such as geometrical shape of the applied PM inside the sample, laser intensity, timings, etc.
- `stack` : Name of the "stack". A stack refers to a specific positioning of the sample during acquisition.
- `stack_description` : [Optional] User-provided, short description of the stack (e.g. what it contains or where it is located).
- `objective` : Name of the detection objective through which the given image was acquired.
- `camera` : Name of the camera that acquired the image data.
- `repetition` : [Optional: assumed to be 0 when missing] Maximum of the `repetition` numbers in the `acquisition` metadata (see explanation of the `repetition` field in `acquisition` metadata below).
- `voxel_size_um` : Physical size of the voxels in sample space, in micrometers. `"depth"` is the shortest spacing between the image planes (perpendicular to planes). Important things to note:
    - This voxel size is *also* contained in the (first applied) scaling part of the transform given by `affine_to_sample` (see below). This is because, when opening the image with a viewer/software that is unable to apply the `affine_to_sample` transform, it should still be able to show the data with the correct voxel size, read from `voxel_size_um` .
    - On the other hand, any viewer or processing software that *can* apply the `affine_to_sample` transform should *ignore* `voxel_size_um` and use `(1,1,1)` as voxel size instead, and then automatically get the correct voxel size from applying `affine_to_sample` , which contains the scaling.
    - Consequently, software that outputs Luxendo Image files should not only write the voxel size to "voxel_size_um", but *also* include the corresponding scaling transform as first transform in `affine_to_sample` : a diagonal matrix with the voxel sizes in `voxel_size_um` on the diagonal.
- `image_size_vx` : The image size in each dimension, in number of voxels.
- `affine_to_sample` : The affine (scaling, rotation, shear, translation) transform that brings the image data to the "sample space", the space permanently connected with the sample stage. Note that for a rotation stage, the sample space may be rotated with respect to the "microscope space" (reference frame of the microscope, which can be thought of as axes along the edges of the microscope "box"), e.g. for the Luxendo MuVi SPIM system. For zero sample-stage rotation and zero sample-stage translation, the "sample space" and the "microscope space" are aligned. `affine_to_sample` can be a list of multiple affine transforms that are to be combined, where the ordering is such that the first transform in the list is the first one applied, and the other transforms follow in the given order. Each affine transform has a `matrix` field containing the *rows* of the affine-transform matrix, and a `translation` field with the 3d translation vector. An *optional* field `type` can be used to label the transform -- e.g. `"rotation"` if the transform corresponds to a rotation (note that the `type` field is not required and can not be relied on -- it may or may not be present and is only for annotation). The `translation` vectors are also used to position the image

stack at the correct physical location in sample space. `affine_to_sample` may also contain only a single or multiple transforms that are already a combination of multiple other transforms.

- `detection_directions` : The direction of detection (axis of detection objective) as vector in 3d in "sample space": it is the original detection-direction vector, but with the stage-rotation transform applied so that the vector is brought from the microscope-coordinate system to the sample-coordinate system. There can be multiple detection directions, e.g. in case of a fused image consisting of multiple original images with given original detection directions.

Acquisition metadata

Inside the `acquisition` field, we have the following metadata fields:

- `image_id` : [Optional] Unique identifier for the image (typically composed of time stamp of first image plane and system UUID).
- `microscope_type` : The type of microscope from which the acquired image data stems (e.g. "Luxendo MuVi-SPIM").
- `serial_number` : The serial number of the microscope.
- `embedded_version` : Version number of the embedded system running on the microscope.
- `edits` : In case there were edits to the `acquisition` metadata, this field holds a list of origins and times of the edits.
- `contains_beads` , `time_point` , `channel` , `channel_description` , `stack` , `stack_description` , `objective` , `camera` : These have the same meaning as in the "derived" metadata. They appear here again since the image could be a fused image consisting of multiple original images, in which case in the "derived" metadata section we would e.g. have `"camera": "Fused-left-right"` , while in the `acquisition` section we would have one set of metadata containing `"camera": "left"` and another set containing `"camera": "right"` .
- `repetition` : [Optional: assumed to be 0 when missing] If the acquisition of the image stack was repeated, this counter indicates the n-th repetition, with n > 0. When n = 0, the acquisition of the stack was NOT repeated, i.e., the stack was only acquired once.
- `number_planes` : Number of image planes in the image stack.
- `stage_positions` : Start and end positions (in micrometers, or degrees in case of rotation) for each stage-movement direction, together with the name of the respective direction. Each movement direction and offset are specified as vectors in 3d "microscope space" (reference frame of the microscope).
- `image_plane_vectors` : The two vectors in 3d microscope space that "span" the imaging plane (plane of the light sheet) along the image "edges" of the camera. These two vectors describe the orientation of the imaging plane with respect to the microscope reference frame, and thus with respect to the stage-movement directions. This flexibly defines the geometrical arrangement of the microscope.
    - `cam_left_to_right` : Vector along the upper edge of the camera image, from left to right.

    - `cam_top_to_bottom` : Vector along the left edge of the camera image, from top to bottom.
- `refractive_index` : The refractive index of the medium that encloses the sample.
- `detection` : Detection-related parameters, such as:
    - `wavelength_nm` : Detection wavelength (in nanometers).
    - `numerical_aperture` : Numerical aperture.
    - `magnification` : Effective magnification.
    - `cam_pixel_size_um` : Size of a pixel on the camera sensor (in micrometers).
    - `sensor_size_px` : Size of the camera sensor in pixels.
    - `crop_offset_px` : Offset in case the image is cropped (in pixels).
    - `crop_size_px` : Size (width and height) of the region of interest when cropping (in pixels).
    - `direction` : [Optional] The direction of detection (axis of detection objective) as vector in 3d in "microscope space". If it's not provided (e.g. because it's not specified in the config file of the microscope), then it is assumed that the detection direction is the cross product of the left-to-right and top-to-bottom image-plane vectors (see above).
- `illuminations` : [Optional entries: can be empty list]. Illumination-related parameters. Not fully defined yet. Subject to change.
    - `objective` : Name of the respective illumination objective.
    - `direction` : The direction of illumination (axis of illumination objective) as vector in 3d in "microscope space".
- `time_stamps` : Acquisition date and time for each image plane, including microseconds. The format is `2021-04-19T14:45:38.073876Z` in UTC (ISO-8601: https://en.wikipedia.org/wiki/ISO_8601).

🔝 back to top

Example structure

This is an example for the structure inside a `.lux.h5` file:

```
Data
Data_2_2_2
Data_3_3_3
metadata
```

There are three h5 datasets `Data` , `Data_2_2_2` , and `Data_3_3_3` containing three different resolutions of image data. The content of the dataset `metadata` has the form described above. Note that each of the items can also be a *link* pointing to a corresponding dataset in a *different* h5 file.

🔝 back to top

Optional nested groups in a `.lux.h5` file

Optionally, a `.lux.h5` can also have a nested structure: on the highest level there are groups `timepoint_<name>` for the different time points, and these contain groups `channel_<name>` for different channels. The channel groups then contain groups for different "views", which can have arbitrary names. And inside the views, there are the same datasets for image data and metadata as in a flat (not nested) `.lux.h5` file described above. So for instance:

```
timepoint_First
  channel_First
    someView
      Data
      Data_2_2_2
      metadata
    someOtherView
      Data
      Data_2_2_2
      Data_3_3_3
      metadata

timepoint_Second
  channel_First
    someView
      Data
      Data_2_2_2
      metadata
    someOtherView
      Data
      Data_2_2_2
      Data_3_3_3
      metadata
```

Also here (as for a flat `.lux.h5` file), instead of a dataset, there can also be a *link* pointing to a dataset in a *different* h5 file.

To check whether a `.lux.h5` file is flat or nested, it can just be checked if at the highest level it contains a dataset `Data` or not.

🔝 back to top

The "main" file

A "main" file links to all the datasets in the other Luxendo Image files (other `.lux.h5` files), including the `metadata` dataset (containing `processingInformation`). It has a nested structure as described above. And it contains separate links to all the different resolution levels of the image data. It can link to multiple channels, time points, cameras, stacks, etc. The "main" file that links to the data in the `raw` folder is called `main_raw.lux.h5` . A "main" file that links to both `raw` and `processed` data is called `main_processed.lux.h5` .

A "main" file has a structure (example):

```
timepoint_First
  channel_First
    raw_First
      Data
      Data_2_2_2
      metadata
    proc_First
      Data
      Data_2_2_2
      Data_3_3_3
      metadata

timepoint_Second
  channel_First
    raw_First
      Data
      Data_2_2_2
      metadata
    proc_First
      Data
      Data_2_2_2
      Data_3_3_3
      metadata
```

Within a time point `timepoint_<name>` , there can be multiple channels `channel_<name>` , which in turn can contain multiple views `raw_<name>` or `proc_<name>` , depending whether the view links to `raw` or `processed` data. At the deepest nesting level, there are the links to the datasets for the different resolutions (`Data` , `Data_2_2_2` , etc.) and to the metadata (`metadata`).

Authors

This document was written by

- Jan Roden, Senior Software Architect at Luxendo
- Balint Balazs, Senior Software Architect at Luxendo

Feedback can be sent to software.luxendo@bruker.com

License