User Guide: Image Processor

PDF version

Overview

Images given to the Image Processor are in the Luxendo-Image format (`.lux.h5`) and contain important metadata, including transforms that bring the image data from the "pixel space" of the cameras to the "sample space", the 3d-coordinate system in which the sample is represented.

The Image Processor performs the following steps:

1. (Optional) `Registration` (content or landmark-based): align the different objective views, tiles, and rotated views against the selected `Reference` view, to get finer "correction" transforms to sample space (in addition to the transforms from the image metadata, e.g. from stage positions). The resulting transforms are saved to `.tf.json` files in `reg_...` (registration) subfolders in the output folder.

2. `Fusion`: the different views are transformed to sample space and fused. This produces a separate output `.lux.h5` file for each time point and channel.

3. Generate main files that link to the output-data files for all time points, channels, etc (e.g. a `.ims` main file to open the output images in Imaris, and `.xml`, `.h5` files for Fiji BigDataViewer).

The settings for each step can be found in the corresponding sections in the GUI. For specific information on the individual settings, please also see the tool tips in the GUI (they are shown when pointing at the label of the respective setting).

When `Fusion` is deactivated, no output images will be produced. This is useful if only the transforms from the content-based `Registration` are needed.

Tasks

The Image Processor allows to set up multiple separate processing `Tasks`, with different settings and/or different input images. Then many `Tasks` can be enqueued and run one after the other, without need of supervision or interaction from the user, and the results of all completed tasks can be evaluated at a later time. Each `Task` is run as a separate process on the computer. A task's output is saved to a separate output folder (named with `Task Name`, time stamp, and user-provided `Task description`) within the `processed` folder, inside the "experiment" folder that contains the used `raw` input images from the acquisition.

A `Task` can have different statuses:

- `New`: a new task.
- `Queuing`: the task is enqueued and waiting to be run, when the tasks that are ahead of it in the queue are completed or have been stopped by the user.
- `Running`: the task is currently running, and output is being written to its output folder.
- `Stopped`: the task was manually stopped by the user. For instance when the user realized that the settings needed to be changed or another task behind it in the queue was more important and should be run first. `_S` (Stopped) is appended to the task's output-folder name. Only a `Stopped` task can be *resumed* by starting it again; it will then continue writing outputs to *the same* output directory. That's why the settings of a `Stopped` task can't be changed.
- `Failed`: the task has failed with an error. This error is shown in a red error notification. `_F` (Failed) is appended to the task's output-folder name. A `Failed` task can be started again, e.g. after adjusting its settings, or to try it again with the same settings; it will then start from zero, with a fresh output directory.

- `Completed`: the task is completed. It can not be run again. `_C` (Completed) is appended to the task's output-folder name.

All present tasks are shown in the task-list panel. When clicking on a task in the list, its settings (configuration) are shown in the configuration panel, and can be changed (unless the task is currently `Running`, `Stopped`, or is already `Completed`). For each task, all its settings are saved in a `config_task.json` file in its output folder, when the task starts to run.

A task can be

- Newly added by clicking the `Add` button. A new task will show up in the task list with a new name (number), for which the user can then provide a description to easily identify the task later, and adjust the settings of the task.
- Copied by clicking the `Copy` button. All settings of the task will be copied to a new task created in the list. This is particularly useful when e.g. another task should be run that has mostly the same settings but only a few specific settings need to be changed.
- Exported (saved) as a `.json` file that contains all the settings of the task, by clicking the `Export` button.
- Imported, by clicking the `Import` button and selecting the `.json` file of a task that is e.g. in the output folder of a previously run task, or that was exported before.
- Saved as new default settings: by clicking the `Save as default` button, the settings of the currently selected task will be saved as new defaults, so that whenever a new task is added afterwards, it will at first have these settings. This function can be very useful, when the user often needs the same settings, but these settings are different from the pre-configured defaults.
- Deleted: the task will be only deleted from the task list, but its output folder will *not* be deleted.
- Run: the task will be added at the end of the queue of tasks (enqueued) and will be waiting until it is its turn to run.
- Stopped: the task can be manually stopped by the user, e.g., to let other more important tasks that are behind it in the queue run first.
- Resumed: a `Stopped` task can be resumed by starting it again: it will continue producing output data in the same output directory.

Usage level

With the `Usage level` slider, the power/difficulty level can be chosen, and the corresponding controls for the settings of a task will be accessible in the user interface. The settings that are not shown at a chosen level are set to their default values. The levels are:

- Basic (0): only the most commonly used controls are accessible.
- Intermediate (1): settings are accessible that provide more fine-grained control, e.g., if a sample requires specific `Registration` settings.
- Advanced (2): these are settings which advanced users can use to fine-tune the processing results.

Should a setting that is mentioned in this guide not be visible in the user interface, please try if it shows up in a higher `Usage level`.

Task configuration: automatic defaults based on input images

Input images for a task can be loaded

- in the `Task configuration` panel, in the `Input` tab, via the `Browse` button.
- Or by opening images in the Image Viewer and then clicking `Add task in Image Processor`.

Once the images are loaded, for some processing settings in the `Task configuration` panel, defaults are set automatically based on the properties (metadata) of the input images. More precisely, these defaults are set based on the metadata of the images of *only the chosen reference time point*, which is sufficient, because normally the basic properties in the image metadata don't change across time points.

To keep things simple for the user, this automatic setting of defaults is done only once directly after loading an input directory -- *not* every time the selection of a subset of images is changed. As a consequence, the defaults are set based on *all* images found in the loaded directory, *not* based on only the selected images.

Of course, the user can still choose to change these settings after the automatic defaults were set.

The default settings that were changed automatically, compared to the original defaults, are also listed in an info popup.

What defaults are set, and based on what input-image properties?

Below, the automatic default settings are listed.

Common settings for registration and fusion:

- If there are multiple detection objectives *and* we have only one camera per detection objective (e.g. MuVi-SPIM):
  - `Fusion` > `Fuse different cams` > ON
- If `Fusion` > `Fuse different cams` > ON:
  - `Registration` > `Objective, camera registration` > ON
  - `Register also different cameras` > ON
  - `Register also different objectives` > ON
- else:
  - `Register also different cameras` > OFF

Output sampling:

- If all detection directions of all images are either parallel or antiparallel (opposing views):
  - `Output` > `Down/up-sample factors` > `Depth (non-isotropic)` > ON (keep anisotropic sampling of input images)

Single-sided illumination:

- If we have only a single illumination objective per image *and* have overall multiple illumination objectives *and* have only one illumination objective per channel:
  - `Fusion` > `Merge all channels` > ON
- If `Fusion` > `Merge all channels` > ON:
  - `Registration` > `Tiles registration` > `Register channels` > `RegisterDifferentChannels`

Typical workflow

A typical workflow has usually two phases:

1. Run multiple fast low-resolution "test" processing tasks, to test different settings quickly and find the settings that lead to the desired result. We recommend setting the parameter `Output` > `Down/up-sample factor` to 2 or larger (e.g. factor 4), to speed up the processing and keep the output small. This factor has a very large effect on run time and output size, since it is applied in all three dimensions of the output images.
2. Once good settings have been found in one test task, copy the task to a new task and run this task with full-resolution output (factor 1), to produce the final result.

In Phase 1, often the following tasks with low-resolution output are useful (**Tip**: copy the first task and only change specific settings):

1. A task *without* content-based registration (`Registration` tab switched off), to see if the alignment of views / tiles from only the transforms in the input-image metadata is roughly correct. **Tip**: To also get the original views in sample space as output, in addition to the fusion result, choose the option `Output` > `Add views separately`. To be able to view the original, aligned data separately can often be helpful.
2. A task with the desired content-based registrations, e.g., `Objective, camera registration`, `Tiles registration`, to see the effect of the content registration.
3. When satisfied with the content alignment, copy this task to a few tasks in which different settings for the `Fusion` are tried out. For these tasks, you can use `Registration` > `Reuse registrations from other task` to select the previous task with the good alignment, to reuse its registrations. This can shorten the processing time significantly.

In Phase 2:

- When good `Registration` and `Fusion` settings are found in a task in Phase 1, copy this task and set the output resolution to full resolution (if desired), i.e., `Output` > `Down/up-sample factor` to 1. Also here, to shorten processing time, you can use `Registration` > `Reuse registrations from other task` to select a previous task which obtained satisfying content alignment. (If the selected previous task had a lower output resolution, this does not affect the accuracy of its registrations -- its registrations can be reused by a task with higher output resolution.) Then run this final task. Due to the high resolution, this task will take significantly longer and create much larger output-image files. **Tips**:
    - Note that choosing full resolution (especially isotropic) can lead to very large output-image data, and may not always be necessary, depending on further use of the data.
    - `Add views separately` usually only makes sense for low-resolution "test" tasks (Phase 1). It should be switched off in high-resolution tasks, since there it leads to much longer processing times and much larger output data (each view added will have the same size as the fusion output).

Registration (content-based)

The `Registration` phase of the Image Processor tries to improve the rough to-sample-space transforms from the input-image metadata by aligning (registering) the content of different views / tiles. By default, only content from the same channel will be registered, because the content in different channels can of course be quite different and does not necessarily match.

The content-based registration is *not* guaranteed to succeed. Depending on the sample / content, it can also fail. It can, for instance, be thrown off by "noise" or other content not of interest to the user, such as edges (borders) of images, or small features that are only present in one of the views to be registered. Sometimes, it may seem surprising that a seemingly easy alignment task does not give a satisfactory result. Please keep in mind, however, that the alignment algorithm does not necessarily know what content is of interest to the user and what content is irrelevant. The fewer *different* features / content there are in the views to be registered, the more likely it is that the registration will be successful. We try to make our registration algorithms more and more robust. So it may be that some registration task that did not work, will work in future versions of the software.

Tip: If the alignment after content registration looks wrong, then run the task again *without* content registration and check that the views are roughly correctly aligned based on the transforms from the input-image metadata.

MuVi-SPIM registration

Objective, camera registration:

- For the MuVi, when input images from *different* objectives/cameras are present, the option `Registration` > `Objective, camera registration` > `Register also different cameras` needs to be switched on, and `Register also different objectives` needs to be switched on as well.

Rotation registration: The content-based registration of rotated views (`Tiles registration` > `Rotation`) is only performed on rotated views from the "reference" objective / camera (selected in the `Input` tab). And the resulting transform is then also applied to the images from the other objective / camera.

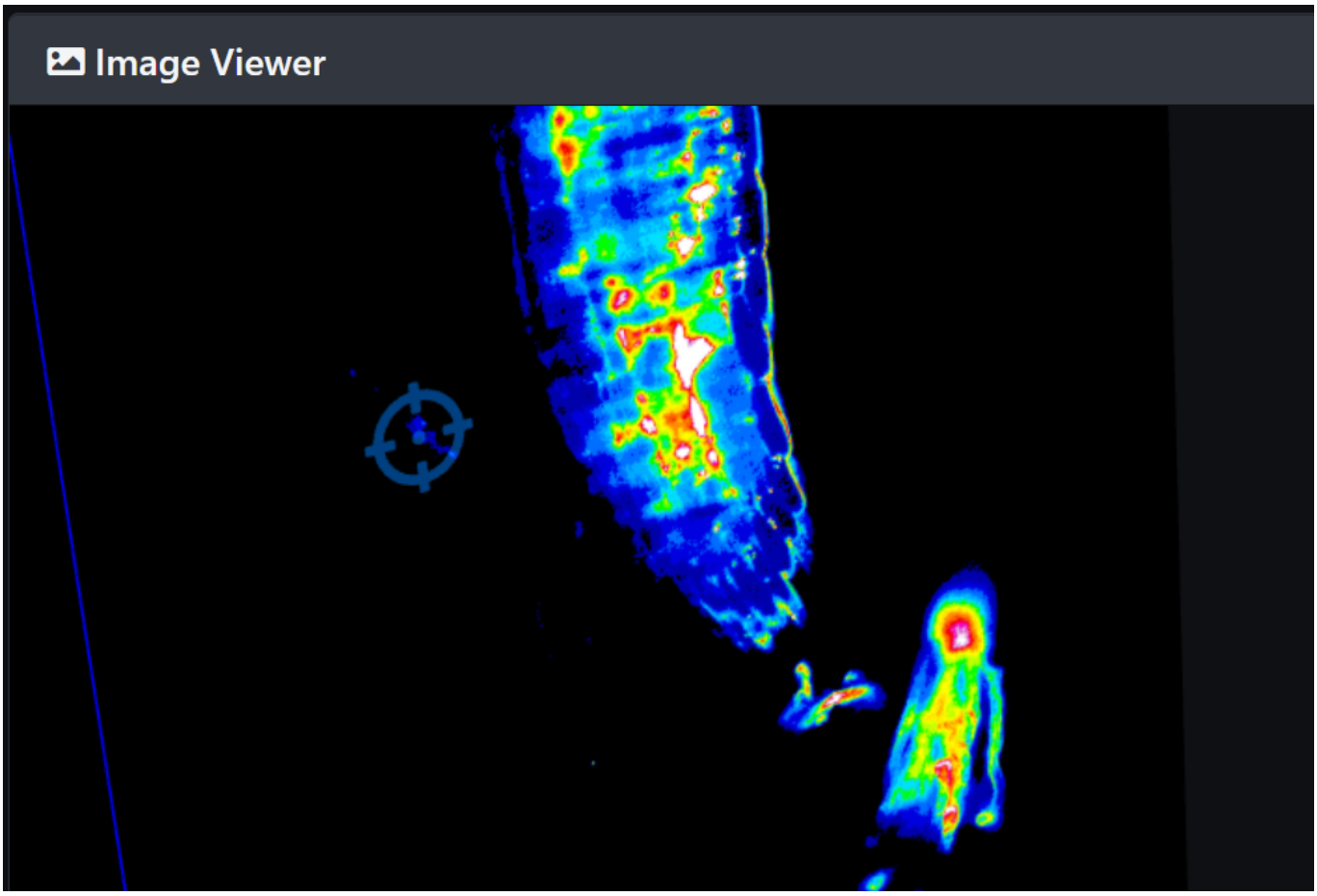
- It is recommended to calibrate the stage-rotation axis *before* acquisition, to facilitate the rotation registration of the MuVi images.
- If the different rotated views have significant content/features *outside* their intersection volume, then the option `Tiles registration` > `Rotation` > `Views are pre-aligned` should be selected, to use only the intersection volume of the views for registration instead of registering in the full volume. Because the content outside the intersection volume can throw off the content-based registration. However, this option requires that the stage-rotation axis had been calibrated (at least roughly) *before* acquisition.

Tips:

- If the result of `Objective, camera registration` (i.e. multiple objective views present) and `Rotation` registration and `Fusion` looks misaligned, then, in a separate task, first only do registration and fusion of the opposing objective views (no rotated views), and check that the result is aligned. If not, then the problem already is in the registration of the opposing objective views.
- If the rotation registration fails, try
    1. Select a different "reference" objective / camera in the `Input` tab.
    2. Calibrate the stage-rotation axis before the acquisition (check that fusion without content registration aligns the rotated views roughly), and then select `Views are pre-aligned`.
    3. Switch `Do fine alignment` off.

4. Subtract a bit more or less background with the `Subtract constant background value` parameter.

Landmarks registration



With the option `Registration` > `Landmarks registration`, corresponding groups of landmarks (with same label) set in the `Image Viewer`, can be used to align the images. If the Image Processor finds corresponding landmarks for input images, it will use these to align the images instead of doing content-based registration.

Reasons to use landmarks registration:

- Improve alignment in cases in which content-based registration gave insufficient results, e.g., because of too different content in the images.
- Chromatic alignment (correction) between images of different color (different channels / cameras), which is not possible with content registration.
- Landmarks registration allows alignment by 3d *affine* transform, which may give better alignment than the *rigid* transform obtained from content registration.

In order to keep the number of landmarks that need to be set in the `Image Viewer` to a minimum, when `Landmarks registration` is switched on, the Image Processor will automatically (wherever it makes sense) apply found landmark-based alignments to other images that don't have landmarks. We call this "batch alignment". This relies on landmarks of the `Reference` image selected in the `Input` tab.

If an image pair doesn't have corresponding landmarks and can also not be aligned by "batch alignment" from other images that have landmarks (see examples below), then the Image Processor will automatically use

content registration (if switched on) to align this image pair.

If three or fewer corresponding-landmark pairs are set in the images to be aligned, then the landmarks are aligned by 3d translation; if four or more landmark pairs are set, by 3d affine transform. If more landmark pairs are set than needed for the alignment, a best-fit transform is calculated that minimizes the distances between the corresponding landmarks. This way, setting more landmarks can give better alignment.

Note: In the case of affine landmarks registration (four or more landmark pairs), the landmarks should *not* all be in the same plane, to enable the algorithm to find the 3d affine transform aligning the landmarks.

Note: If additional transforms are applied to the input images through `Input` > `Create in input-image folders` > `New metadata for input images` > `Use multi-transform (MTF) file` (see this section for details), landmarks registration can still be done on top of this, because these additional transforms are automatically applied to the landmarks as well, before the landmarks registration is performed. That is, the landmarks always stay at the correct position relative to the content of the image.

Example use cases

Here are three example use cases for `Landmarks registration`:

Example 1: Stitching of tiles using only content registration gave insufficient alignment between one or more pairs of neighboring tiles. Goal: use `Landmarks registration` to improve the alignment of the insufficiently aligned tiles. Steps:

1. In the misaligned neighboring tiles, set corresponding landmarks in the `Image Viewer`, giving the landmarks in a pair of corresponding landmarks the same `label`. The landmarks should all be set for tiles with the same `Objective`, `Channel`, and `Camera`. **Note**: setting four or more corresponding-landmark pairs in neighboring tiles will give an *affine* transform between this pair of tiles (instead of only a translation, in case of three or fewer landmark pairs). Consequently, the global optimization of the arrangement of the tiles will automatically try to find an optimal *affine* transform for each tile. If however, there is no pair of neighboring tiles with more than three corresponding-landmark pairs, then only the translations of the tiles are optimized in the global optimization, which can be more robust than finding affine transforms for the tiles (though provides fewer degrees of freedom).
2. In the Image Processor, copy the previous `Task` to a new `Task` and switch on `Landmarks registration`.
3. In the `Input` tab, for the `Reference` image, select the same `Objective`, `Channel`, `Camera`, and `Time point` where the landmarks were set.
4. Run the task.

In the resulting stitched output image, the alignment between the tiles should have improved.

Example 2: For stitching of tiles with different `Channel`s, correct for chromatic shifts between the channels, and align different `Objective`s. Use `Landmarks registration` to do this alignment of the channels and objectives. Steps:

1. In `Image Viewer`, pick one tile (e.g. `stack1-x03-y04`) and in this tile, set corresponding landmarks between all possible different combinations of `Objective`, `Channel`, and `Camera`.
2. In the Image Processor, create a new `Task` and switch on `Landmarks registration`.
3. In the `Input` tab, for the `Reference` image, select the `Stack`, `Tile`, and `Time point` for which the landmarks were set. For the `Reference` `Objective`, `Channel`, and `Camera`, one of the combinations

for which landmarks were set should be selected. (If `Tiles registration` is switched on, alignments between different tiles with the `Reference Objective`, `Channel`, and `Camera` will be done by content registration, since there are no corresponding landmarks set between different tiles. Remember: in Step 1 the landmarks were all set in the same tile.)
4. Run the task.

In the resulting stitched output image, the landmark-based, chromatic alignment (correction) and alignment of objectives should now have been applied to all tiles ("batch alignment").

Example 3: Use `Landmarks registration` for accurate alignment of rotated views of the MuVi (e.g., to remove "doublings" after content registration: artificially duplicated structures due to insufficiently aligned content from different views). Steps:

1. In `Image Viewer`, pick one `Objective`, `Channel`, `Camera`, `Time point`, and for this, set corresponding landmarks between the different rotated `Stack`s.
2. In the Image Processor, create a new `Task` and switch on `Landmarks registration`.
3. In the `Input` tab, for the `Reference` image, select the `Objective`, `Channel`, `Camera`, `Time point` for which the landmarks were set. For the `Reference Stack`, one of the ones for which landmarks were set should be selected. (If `Objective, camera registration` is switched on, alignment between different objectives will be done by content registration, since there are no corresponding landmarks set between different objectives. Remember: in Step 1 no landmarks have been set in a different "objective view" other than the picked one.)
4. Run the task.

In the resulting fused output image, the landmark-based alignment between the rotated stacks should now have been applied, across all `Objective`s, `Channel`s, and `Camera`s ("batch alignment").

The steps in Examples 2 and 3 can of course also be combined for cases in which landmark-based alignment between different colors / objectives *and* different stacks is desired.

Important notes

- For the batch alignment to work, it is important to make sure that the `Reference` image selected in the `Input` tab has corresponding landmarks with the needed other images with same / different `Stack`, `Objective`, `Channel`, `Camera` (as described in the examples above).
- The alignment transforms for `Objective, camera registration` are automatically extended to all `Time points`.
- If `Use registration of reference time point for all time points` is switched on, then the other landmark registrations are extended to all `Time points` as well.
- If landmarks were set in a specific image (specific `Stack`, `Objective`, `Channel`, `Camera`, and `Time point`), then the alignment of these landmarks with the corresponding landmarks in the selected `Reference` image takes priority over any (indirect) "batch alignment" that would have applied to this image. That is, setting landmarks in a specific image overwrites the transform that would have been otherwise applied to this image via batch alignment.

Alignment of different time points

By setting landmarks in different specific time points, it is also possible to align these time points against the selected `Reference` time point. Please note the following points:

- Make sure that the selected `Reference` image has the desired corresponding landmarks set.
- This time-points alignment also works when `Use registration of reference time point for all time points` is switched on, because the registration of landmarks set in specific time points will take priority over such batch alignment.
- If you're using `Registration` > `Time-points registration` ON, to perform *content-based* registration of time points: if content-based registration of certain time points wasn't sufficient (e.g. not accurate enough or failed), you can then set corresponding landmarks only in these time points and the respective previous time point (and keep content-based registration between the other time points where it worked). Then run this in a new task.

Reuse static registration transforms across different samples

Some transforms found by (landmarks or content) registration can be reused across different samples (different experiments), as long as the physical microscope configuration is not changed. For instance, registrations between different objectives, cameras, or channels (colors) are such "static" registrations that can be reused, if they are for the *same time point and stack*, i.e., the sample couldn't move between the registered images.

This has the advantage that if for instance the user performed a landmarks registration between different combinations of objective, camera, and channel, this registration doesn't have to be done again when processing images for a new sample (new experiment).

For this purpose, the Image Processor writes such reusable registration transforms to a so-called "multi-transform" (MTF) file `reuse_static_regis.mtf.json` in the output folder of the task that performed the registrations (please see this section for an in-depth description of these MTF files).

If a reuse of these transforms is desired in a new task, you can just switch on `Input` > `Create in input-image folders` > `New metadata for input images` > `Use multi-transform (MTF) file`, and with `Non-default MTF file path` select the `reuse_static_regis.mtf.json` file in the output folder of the desired previous task.

To automatically reuse the same set of registration transforms for this microscope for every newly created processing task, just add a new task, select the desired `reuse_static_regis.mtf.json` file in `Non-default MTF file path`, and use `Task Configuration` > `Save as default` to have this MTF file selected by default in every new task. This is for instance helpful when a shift between two color channels, or a slight rotation of a camera, should always be corrected by the same transform, without having to manually select a MTF file every time.

When reusing the registration transforms this way, it is currently *very important* though that channels with same properties also *need to have the same name* (i.e. properties affecting the alignment transform, e.g. wavelength). This is because in the MTF file, the channels are matched *by their name* to the corresponding transform. That is, if e.g. Channel 2 had a wavelength of 405 nm in one experiment where (landmarks) registration between channels was done, and the registration transform should be reused to align the channels in another experiment, then it's important that the Channel with the same wavelength is called Channel 2 also in the other experiment. For objectives and cameras, however, it is easier, because their names anyway stay the same across experiments.

Fusion

In the `Fusion` phase, all selected input views, i.e., objective views, tiles, and rotated views, are fused at once. This happens in a chunk-wise manner, to allow

- fusion of very large images, without running out of RAM
- (optionally) calculating the image quality in each chunk for each view and in the fusion weighting each view's chunk based on its image quality
- efficient parallel processing of chunks

The fusion `Image-quality weighting` enables weighting of different views based on their local image qualities with the granularity of the chunk size -- typically 64 pixels (for details, please see the descriptions in the tool tips). Such weighting can be very useful, because it allows "stitching" fusion. This provides, for instance, a way to suppress doublings from inaccurate alignments in the output image, which can occur in many situations.

Tip: To ensure that the fused output images can be viewed in Fiji BigDataViewer, the option `Fusion` > `Half intensity` should be switched on. This will divide the intensity of the output images by two.

MuVi-SPIM fusion

- For the MuVi, when input images from *different* objectives/cameras are present, the option `Fusion` > `Fuse different cams` needs to be switched on.

QuVi-SPIM fusion

- For the QuVi, the option `Fusion` > `Load only full chunks` should be switched on, to avoid empty spots in the fused output images, at the boundary between views (though some content at the edges of the input images may be lost when this option is used).

Stitching

When doing stitching of multiple tiles (fields of view), the following points are important:

- If multiple *rotated* views with multiple *horizontal* tiles are to be stitched, it is crucial to (at least roughly) calibrate the position of the stage rotation axis in LuxControl, *before* acquisition. In the Image Processor, then check `Registration` > `Tiles registration` > `Rotation` > `Views are pre-aligned`. Otherwise, the registration algorithm doesn't know which rotated tiles intersect and can be content registered.

- For all tiles, the "reference" objective selected in the `Input` tab needs to be present in the list of input images.

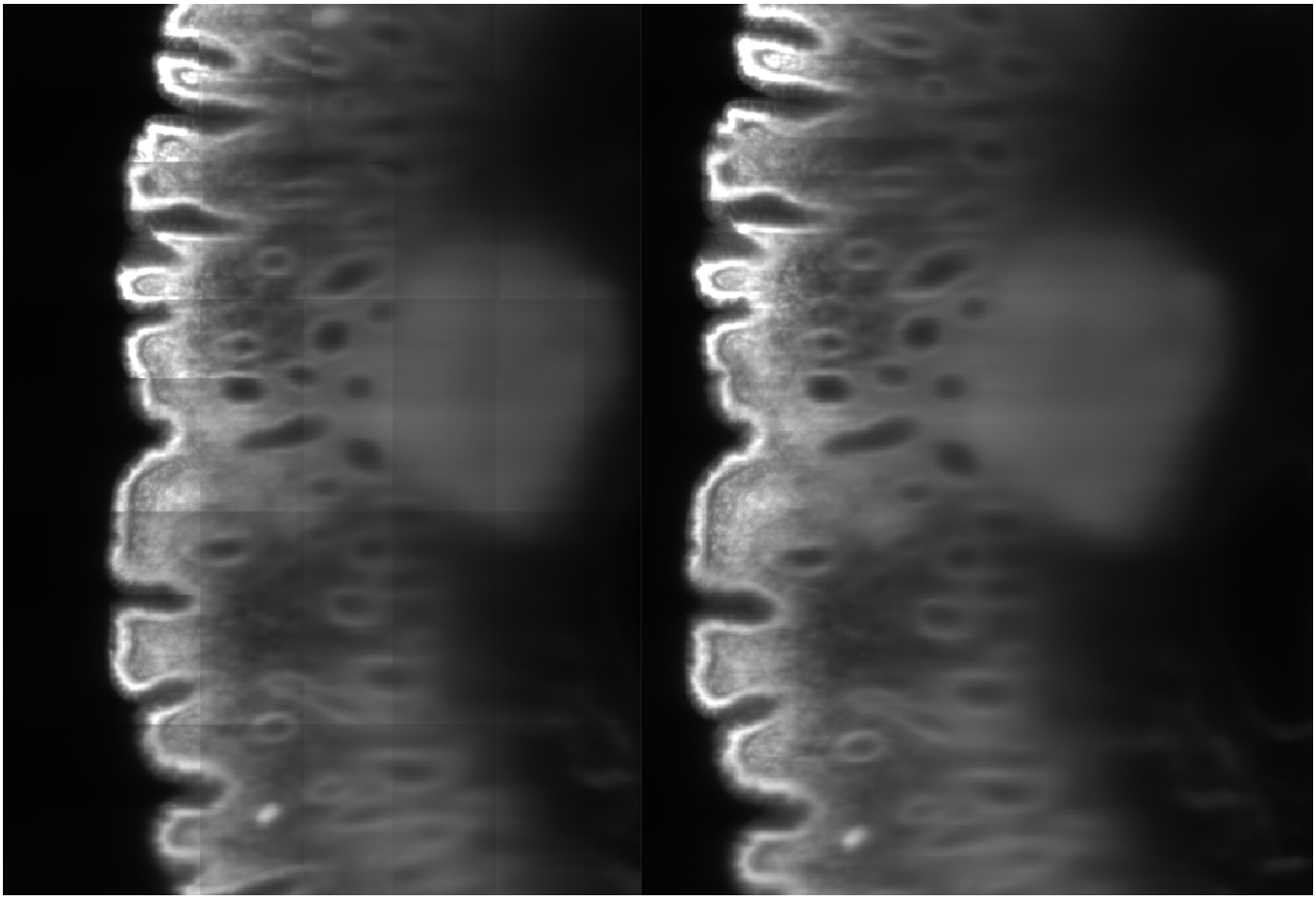
- Content-based alignment of the tiles can be switched on and configured with `Registration` > `Tiles registration`. If this is switched on, all overlapping tile pairs are registered based on the content in their overlap regions. Afterwards, the resulting tile-pair registrations are used in a global-optimization procedure, to find the best overall arrangement of the tiles. If any of the registered tile pairs contribute an unusually high "cost" during this global optimization, a warning will be shown, since this likely means that the corresponding tile-pair registration is wrong or inaccurate and thus disagrees with the other tile-pair registrations. In this case, it is recommended to copy the task and run it again, but with `Tiles registration` > `Remove outlier pairs` > `Automatic with cost threshold` with threshold 10 enabled, to automatically remove these "outlier" tile-pair registrations in

the global optimization. (Note that often there are anyway more tile-pair registrations than needed for a good overall arrangement of tiles, so removing some inaccurate ones will usually lead to a better result.) For this new task, the tile-pair registrations from the previous task can be reused, with `Registration` > `Reuse registrations from other task`, to avoid doing all the tile-pair registrations again, to save time.

- For images acquired with relatively small magnification (e.g. < 10x), depending on the use case, the alignment of the tiles may already be accurate enough *without* content-based registration (`Registration` > `Tiles registration` switched *off*), i.e., using only the stage-position-based alignment from the input-image metadata. If this is not sufficient, the `Registration` > `Tiles registration` can be switched on, to add content-based alignment.

- For stitching of tiles, under `Output` > `Bounding boxes`, either `UnionAll` or `UnionsOverlapping` should be selected (regarding the differences between these modes, please see the tool tip for `Bounding boxes` in the GUI).

- If there are strong intensity borders visible between the stitched tiles, you can smoothen the transitions between tiles by stitching again with `Fusion` > `Blending` switched on.



(Large intestine, imaged with Luxendo MuVi-SPIM at ClearLight Biotechnologies)

Typical workflow for stitching

Run different "test" tasks with low-resolution output (e.g. `Output` > `Down/up-sample factor` set to 4 or larger), and finally one task with high-resolution output:

- Task 1 (low-resolution output):

    - `Registration` off, to see arrangement of tiles based on only stage positions (no content alignment).
    - `Fusion` > `Half intensity` on, to be able to view result in Fiji BigDataViewer.
    - `Output` > `Add views separately` can be used to enable seeing tiles separately in the viewer later on, for easier evaluation of alignment and overlap of tiles. However, this will add a dataset of the size of the *full stitched image*, *for each tile* to the output, i.e., *multiply the output size by the number of tiles.* Thus, this is only recommended for either smaller numbers of tiles, or strongly downsampled output (large `Output` > `Down/up-sample factor`).
    - Open result in Fiji BigDataViewer, and see how much the sample moved between acquisition of different tiles, and whether the chosen overlap size between the tiles was large enough.
    - If the overlap of tiles is too small compared to the movement of the sample (i.e. corresponding features are not in the overlap region), then the sample should be re-imaged with
        - better mounting
        - less stage acceleration
        - more overlap
    - If the overlap was sufficiently large, proceed with Task 2.
- Task 2 (low-resolution output):
    - `Registration` on.
    - Are there warnings about tile-pair registrations with high cost?
        - If so, proceed with Task 3, otherwise go directly to final Task 4.
- Task 3 (low-resolution output):
    - To shorten processing time, you can select `Registration` > `Reuse registrations from other task` > Task 2.
    - Switch on `Tiles registration` > `Remove outlier pairs` > `Automatic with cost threshold` with threshold 10.
    - If in the result the alignment of the tiles is satisfying, proceed with final Task 4. Otherwise, try to improve the alignment further by changing other parameters in `Tiles registration`, e.g. limit the translation by setting `Maximum translation`, if you know that the alignment shift should not have to be larger than a certain distance in micrometers. However, please keep in mind that perfect alignment of all features often will not be possible, due to a limited number of degrees of freedom of the registration.
- Task 4 (high-resolution output):
    - To shorten processing time, you can select `Registration` > `Reuse registrations from other task` > Task 2.
    - Set the `Output` > `Down/up-sample factor` to the desired final resolution.
    - For smoother transitions between tiles, we recommend to switch on `Fusion` > `Blending`.

Stitching single-sided-illumination channels

For acquisitions with single-sided illumination, the acquired images for the different illumination directions normally go into different Channels: one Channel then typically images one half of the sample and the other Channel the other half. This for instance results in:

- Channels 0, 1: the two halves of the sample with color A
- Channels 2, 3: the two halves of the sample with color B

As an example, let's consider the following goal: we want to stitch the two halves of the sample together and end up with a stitched image for color A and one for color B. Also, let's say we know that color A has better

signal / more content, so we want to use color A for content-based registration between the two halves of the sample (and between tiles within each half), and use the same relative alignment shifts found within color A also for color B. That is, we don't want to do a new content registration for color B. Additionally, we may want to globally shift color B relative to color A, for chromatic correction. This can be achieved with the following steps:

1. Run a Task 1 with:
    - `Input`: Channels 0, 1 (two halves of sample, color A)
    - `Registration` (ON) > `Tiles registration` (ON) > `Register channels` > `RegisterDifferentChannels` (so that the two halves are registered with each other)
    - `Fusion` (ON) > `Merge all channels` > ON (so that the two halves are fused)
    - `Output`: `Bounding boxes` > `UnionAll`
2. After Task 1 is completed, run a Task 2 with:
    - `Input`:
        - Channels 2, 3 (two halves of sample, color B)
        - `Create in input-image folders` > `New metadata for input images` (ON) > `Use multi-transform (MTF) file` (ON), for global chromatic correction between color A and B (the needed MTF file can e.g. be generated by setting a landmark pair in one tile of color A and the corresponding tile of color B)
    - `Registration` (ON) > `Tiles registration` (ON) > `Register channels` > `RegisterDifferentChannels` (so that the two halves are registered with each other)
    - `Registration` > `Reuse registrations from other task` (ON):
        - `Task folder` > Task 1
        - `Match channel` > OFF (so that transforms from Task 1 can be reused, which are for different channels)
    - `Fusion` (ON) > `Merge all channels` > ON (so that the two halves are fused)
    - `Output`: `Bounding boxes` > `UnionAll`

After the tasks are completed, the stitched image for color A will be in the output folder of Task 1 and that for color B in the output folder of Task 2.

To also create main files (for Imaris and Fiji BigDataViewer) that have (link to) both colors A and B as different channels (useful e.g. to measure distances between features in the different channels, etc), you can do the following steps:

1. Copy (or move) the image-data file `uni_...lux.h5` from the output folder of Task 2 to that of Task 1 (note that moving it instead of copying can avoid data duplication, but then the main files for Imaris and Fiji BigDataViewer in the output folder of Task 2 will not work anymore, because the image-data file they linked to has been moved to a different path).
2. Run a Task 3 with:
    - `Input`:
        - select the output folder of Task 1 as input, then choose the two Channels that are now in the folder of Task 1 (colors A and B).
        - `Only create main files for input images` > ON

After Task 3 is completed, the Imaris and BigDataViewer main files should be in the folder of Task 1 and are named `main_ADDED...`.

Output

MuVi and QuVi SPIM

The option `Output` > `Down/up-sample factors` > `Depth` can be switched *off* to produce isotropically sampled output images. For the QuVi, the output should normally be sampled isotropically. For the MuVi, isotropic output makes usually only sense when rotated views are fused (angular fusion). In the case of only one stage-rotation angle, for one objective or two opposing objectives, it is recommended to use the original z (depth) sampling, i.e., the same number of planes as the input images: `Depth` (z) sampling factor of 1.

Bounding boxes

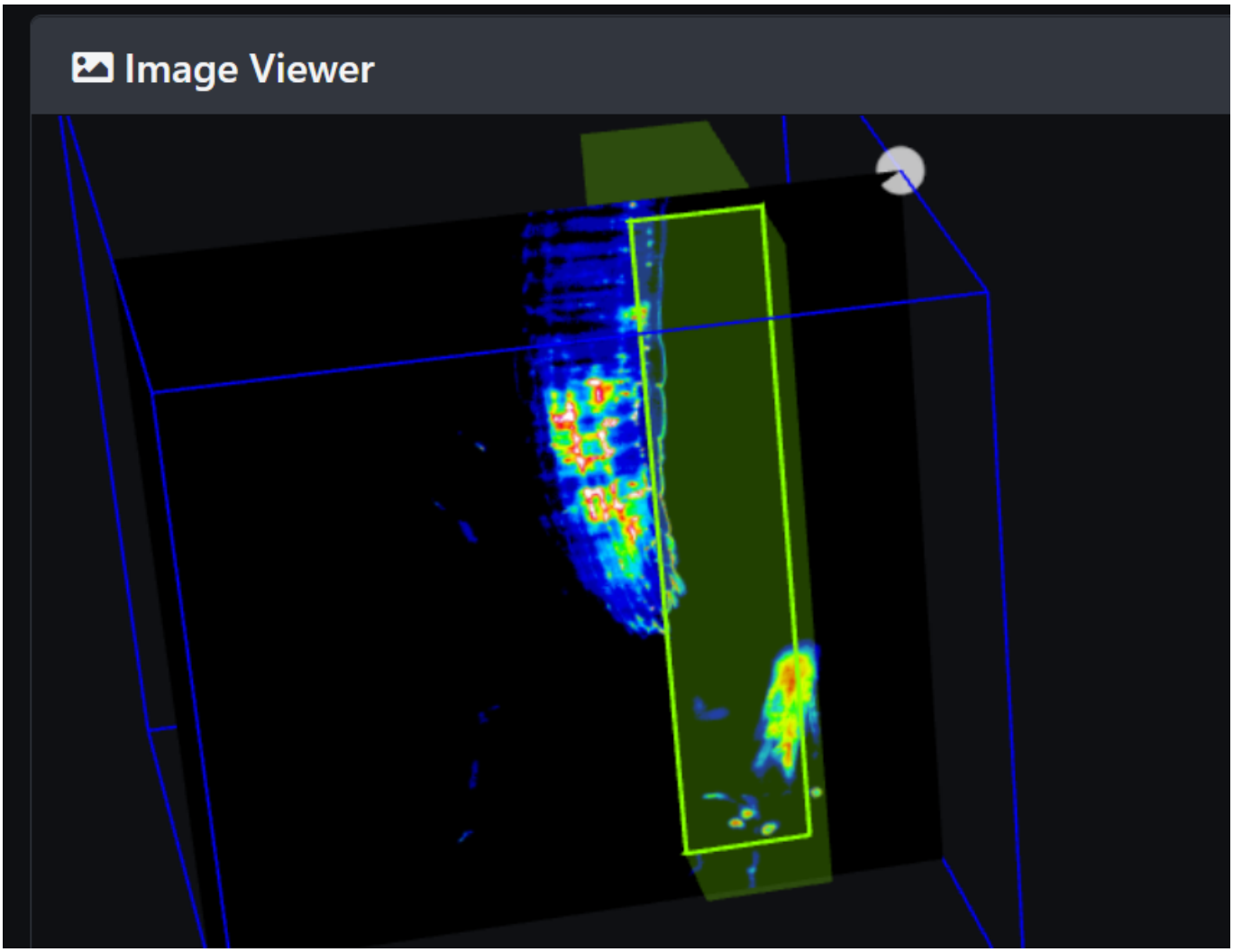
With the different modes in `Output` > `Bounding boxes`, one can choose what bounding boxes (volumes), i.e., which (parts of) stacks, go into the output-image files. The available modes are:

- `UnionAll`: Write the union (enclosing) of all stacks to a single output file (largest possible output volume that fully contains all stacks). Example: stitching one large sample from overlapping tiles.
- `UnionsOverlapping`: The union of each group of overlapping stacks goes to a separate output file. Example: a multi-position acquisition with stitching of several tiles at each position.
- `Intersections`: The intersection (overlap region) of each group of overlapping stacks goes to a separate output file. Example: a multi-position acquisition, where at each position there are different views and only the common region of the views is desired as output (for each position).
- `Separate`: Each stack goes to a separate output file.

For the stitching of tiles (views), either `UnionAll` or `UnionsOverlapping` should be used. Use `Separate`, to only transform the stacks to sample space and generate a separate output file for each stack (also for Fiji BigDataViewer, Imaris, etc). For this, registration between different stacks is usually not needed and can be switched off.

When `UnionAll` or `UnionsOverlapping` is selected, the names of the output `.lux.h5` image files will be prefixed with `uni`. When `Intersections` or `Separate` is selected, then they will be prefixed with `int` or `sep`, respectively. This way, one can easily see from the names of the output files what `Bounding boxes` mode was used.

Use ROI



A ROI (region of interest) that was set in the `Image Viewer` can be used in an Image-Processor task to crop input images. When a ROI is selected by its label in the task configuration, in `Input` > `Use ROI`, then only the image data from the selected ROI of the input images is used as input data for content `Registration` and `Fusion`. That is, the Image Processor behaves just as if the selected ROI is the whole input image; the ROI replaces the input image for all processing, so to say.

In particular:

- The different modes in `Output` > `Bounding boxes` (e.g. `UnionAll`) also apply the same to combinations of ROIs and whole input images.
- In the `Image Viewer`, one can also set ROIs *with the same label* in different stacks. When this label is then selected in the Image Processor in `Use ROI` > `Which ROI`, these different ROIs are applied for cropping in the different stacks. This way, the ROIs of the stacks can be combined for instance by `UnionAll` or `Intersections` to get the corresponding combined region as output image.

Using ROIs, also content registration can be limited to a specific region in the sample:

1. In the `Image Viewer` set ROI(s) for the region(s), to which the content registration should be limited.
2. In a first Image-Processor task "Task 1", use this ROI to crop the input images and content-register only in the corresponding region.
3. In a second task "Task 2", it is now possible to *not use* the ROI (not crop) and use the full images as input, but select `Registration` > `Reuse registrations from other task` > "Task 1". This will

use the alignment transforms from "Task 1", i.e., will effectively apply the ROI also in "Task 2", but only for the content registration, not for the fusion / output.

Apply single ROI to all input images

A single ROI, drawn in the Image Viewer, can also be applied to *all* selected input images, with `Use ROI` > `Apply to all images`.

This can be useful for instance when stitching tiles and the intensity in the margins of the tiles is too low (e.g. for subsequent segmentation of the stitched images). Then, to crop all tiles in the same way before stitching, a ROI only needs to be drawn in *one* tile and can be applied to all tiles (with `Apply to all images`), which will cut away the same margins in all tiles, given the tiles all have the same size. (Please keep in mind though to make the ROI large enough to still have sufficient overlap between the tiles for stitching.)

Flatfield correction

What is it?

Ideally, in the acquired images of a *completely uniform* sample, we would expect a flat intensity across the entire field of view (FOV). In practice, however, properties of the detection optics and/or illumination characteristics can cause local reductions of intensity in parts of the FOV. For instance, the detection objective can cause a significant, noticable decrease of intensity going from the center of the FOV towards the edges. This is shown in the picture below (Fig1), for a Luxendo LCS SPIM at 2x magnification.

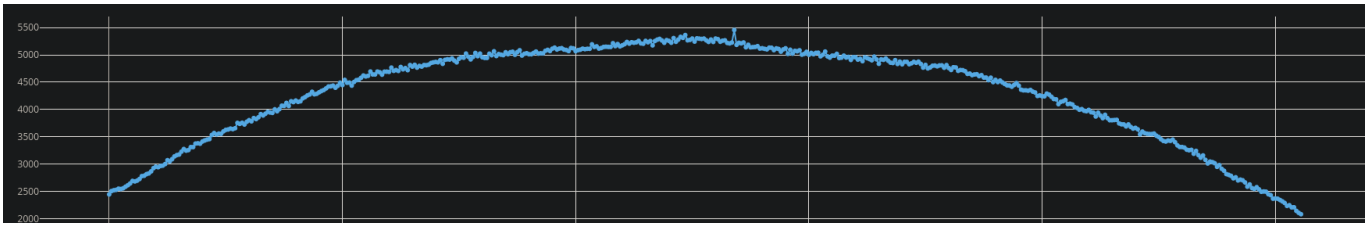


Fig1: Intensity profile across FOV, of mean projection of a stack imaged without sample and open lid, with a Luxendo LCS SPIM at 2x magnification.

Such local reductions of intensity also underlie the acquired images of a normal (non-uniform) sample. For instance, this can cause visible dark-bright-dark patterns in stitched images, consisting of multiple FOVs, when the outer parts of the FOVs are darker than the center parts.

Flatfield correction is a processing step to approximately compensate such local reductions of intensity.

It uses "correction images", typically acquired separately with a uniform-enough sample or without sample, that capture an (averaged) intensity profile, characteristic for the used detection optics and/or illumination. The assumption is then that this is the intensity profile also present in the sample images, that we want to compensate for, by "flattening" it.

Based on these correction images, a 2D "multiplication layer" is constructed such that multiplying this layer pixel-wise onto the *correction images*, would result in a flat intensity profile across the FOV. For instance, in the picture below (Fig2), the profile of the 2D multiplication layer is shown that would approximately flatten the intensity profile in Fig1.

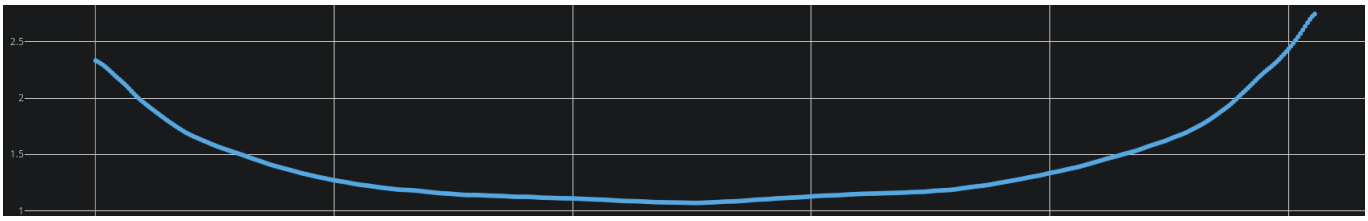


Fig2: Profile across FOV of 2D "multiplication layer" that would approximately flatten the intensity profile in Fig1, such that the resulting profile has roughly the original maximum intensity in Fig1 across the entire FOV (the vertical axis shows the "amplification" factor).

Flatfield correction of the *sample* images is then achieved by multiplying this multiplication layer onto each z plane of the *sample* images.

In the Image Processor, the parameters for the flatfield correction can be set in `Input` > `Flatfield correction`. It is then performed with the following steps:

1. Load 2D mean projection (along z) of the correction-image stacks from TIFFs (if TIFF not present, load full correction-image *stack*, compute mean projection and save it to TIFF, for re-use).
2. Compute 2D "multiplication layer" (see Fig2 for example result):
    - Take mean projection from step 1, and apply `Input` > `Camera-pixel adjustments` (compensate camera-pixel read offsets and gains).
    - Smoothing by convolution with Gaussian kernel: `Smoothing kernel size`.
    - Normalize the resulting 2D layer such that its maximum is one.
    - Take pixel-wise inverse and limit to maximum factor (cutoff): `Maximum factor`.
3. Flatfield-correct *sample* images, by multiplying each of their z planes with the multiplication layer from step 2.

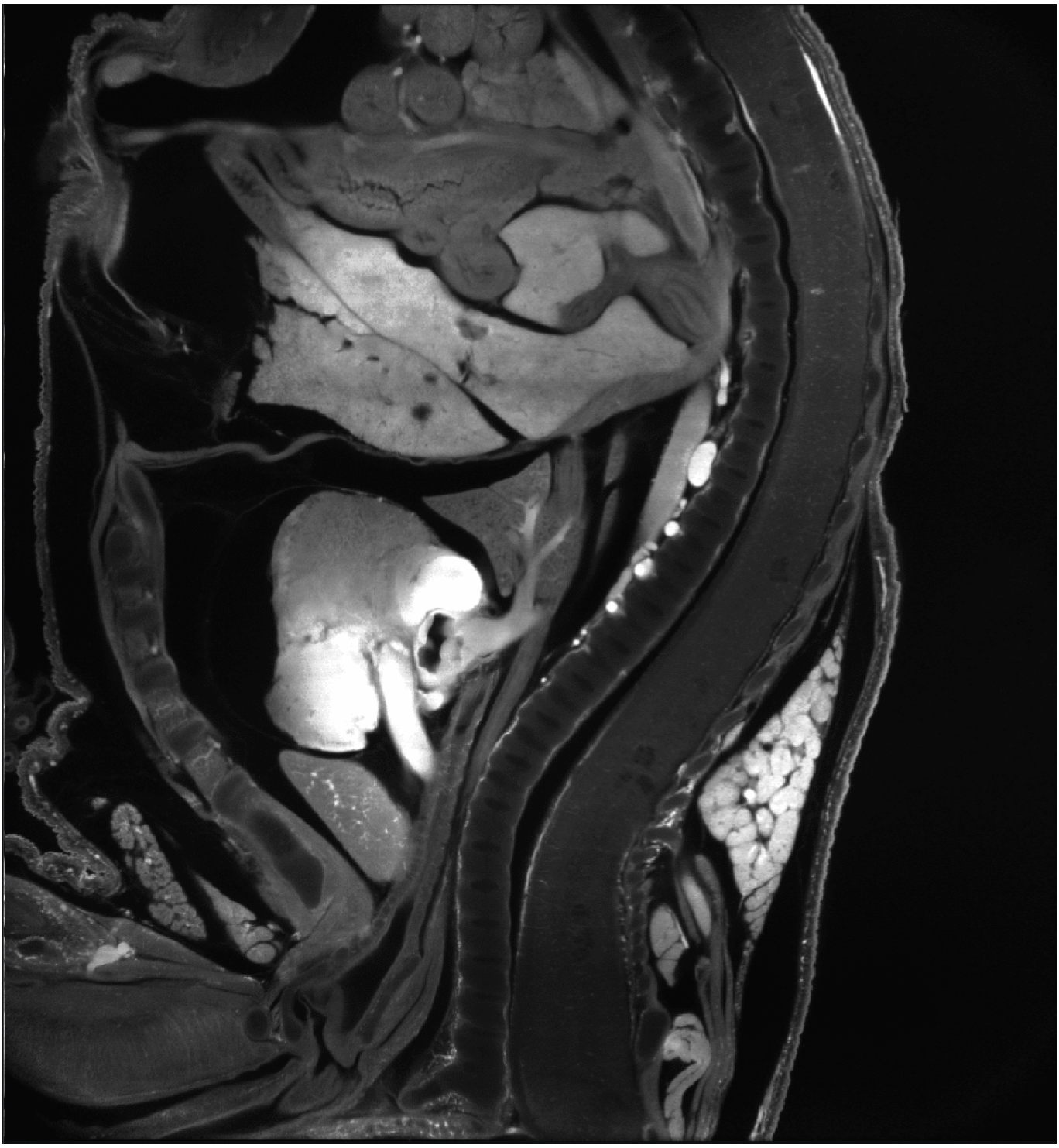
(The used mean projection of the correction images and the calculated multiplication layer can be saved to TIFF in the task-output folder, for evaluation, with `Save total mean projections and multiplication layers`.)

In some cases, where many FOVs (tiles) of the sample were acquired, the total mean projection over all z planes of all tiles could potentially also serve as "correction image", if local features of the sample are sufficiently averaged out this way. (This can be checked by evaluating the resulting TIFF in which the total mean projection is stored.) The sample images are automatically used in this way, i.e., *also* as correction images, when `Correction images` is switched OFF, i.e., no separate correction images are loaded.

Example

A typical case where flatfield correction can be useful is acquisition with a Luxendo LCS SPIM at 2x or 4x magnification.

Below picture (Fig3) shows a comparison of stitched 2x2 tiles, imaged with a Luxendo LCS SPIM at 2x magnification, *without* (left) and *with* flatfield correction (right). One can see that the flatfield correction amplifies the intensity close to the edges of the tiles (for instance the features marked by red arrows), such that the level of intensity there becomes similar to that in the center of the tiles.



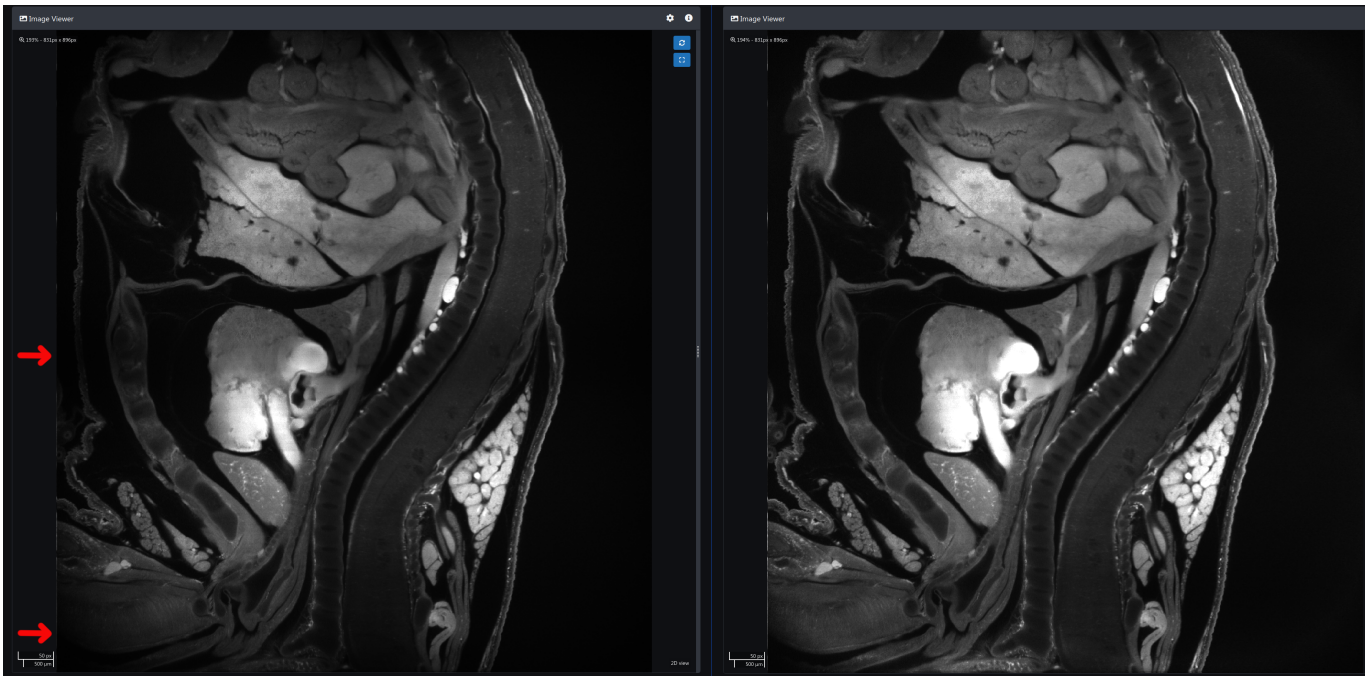


Fig3: Left: Stitched 2x2 tiles, imaged with Luxendo LCS SPIM at 2x magnification (red arrows mark lower intensity close to edges of tiles). Right: With flatfield correction. Note how the intensity close to the edges of the tiles is increased now, to the level in the center of the tiles.

Such amplification of intensity in the outer parts of the FOV, to even out the overall intensity profile, can in many cases greatly improve visual inspection and further analysis of the images, especially for acquisition and stitching without cropping and with a rather small overlap of the tiles.

The settings used for the flatfield correction (in `Input` > `Flatfield correction`) in this example, which produced the result on the right side of Fig3, are:

- `Correction images`: a stack imaged *without* sample with open lid on the same LCS SPIM, at the same magnification as used for the sample images (2x). (Same image stack for which the intensity profile is shown in Fig1.)
- `Smoothing kernel size (pixels)`: the default value of 9 was kept here (this probably doesn't need to be changed in most cases)
- `Maximum factor` in multiplication layer: we set this to a value of 3, to not over-amplify the noise in the corners of the tiles. (This is an important parameter, which may need to be adjusted for the sample and correction images at hand, e.g., when you see that noise is amplified too much in the corners of the FOVs)

Requirements

Requirements when using flatfield correction:

- Sample and correction images must be acquired at the same magnification.
- Pixel size (width, height of pixel, in µm) *must be the same* for sample and correction image.
- Allowed regarding image size / crop:
    - Correction image has same crop (offset, size) as sample image (raw or processed images can be used as sample and correction images).
    - *Raw* correction image full-sized (uncropped), and *raw* sample image can have any cropping (from acquisition).
    - In current processing task, *no* ROI applied to correction image, but ROI applied to sample image.

Tips

Tips when working with flatfield correction in the Image Processor (please also see the tooltips in the GUI, that show up when pointing at the different settings labels in the `Flatfield correction` section):

- If you have correction images that you want to apply to given sample images, but the correction images have a higher sampling (smaller pixel size) than the sample images, because the sample images had been downsampled, then you can simply first run a task where you downsample the correction images such that they then match the sampling of the sample images. Then you can run the task with the flatfield correction afterwards, using the downsampled correction images. (Note that sample and correction images must be acquired with *same* magnification for the flatfield correction to work.)
- To see whether given correction images lead to appropriate mean projections and multiplication layers, you can look at them (e.g. with Fiji) via the TIFFs that are saved to the task-output directory.

Deconvolution

To perform deconvolution of the input images, switch on `Fusion` > `Deconvolution`.

The Image Processor does single-view deconvolution at places in the output image where there is only a single input-image chunk (a single input view), and automatically performs multi-view deconvolution, as a special form of fusion, at places where there are multiple input-image chunks (from multiple input views), e.g., for multiple rotated views, or in the overlap region of tiles.

The point-spread function (PSF) used in the deconvolution is modelled as a 3d Gaussian. Its lateral width and axial width (length), with respect to the axis of the detection objective, in sample (output) space, can be changed with the corresponding sliders in the GUI. When one of the sliders is changed, a separate window appears that shows the resulting PSF in sample space from the three directions of the output-space axes. For this, input images need to be selected first, because the metadata from the images (to-sample-space transforms, etc) is needed to calculate the PSF. The shown PSF is the one that will be used in the deconvolution, and its voxels are in the space of the final output image. Note that the separate window showing the PSF has to be closed before you can continue to interact with the main window of the GUI.

The deconvolution uses an iterative Richardson-Lucy algorithm, and the number of iterations can be set with `Deconvolution` > `Number of iterations`. Usually, the default number of 10 iterations is a reasonable choice.

Some important **tips** for deconvolution are the following:

- Acquire the images with at least two-fold, or better three-fold, oversampling (i.e., choose a small enough Z spacing, depending on the magnification of the used objective).
- Do **not** downsample in the output image, to keep the provided sampling of the data. That is, under `Output` > `Down/up-sample factors`, leave all factors set to 1 (i.e., leave the default settings there).
- To speed up deconvolution test runs, for a faster testing cycle of different parameters (PSF parameters or number of iterations), under `Input` > `Use ROI` choose a ROI that is *as small as possible*. This will usually allow a fast test cycle of just a few minutes, since it significantly decreases the amount of data that needs to be deconvolved.
- To get a rough estimate for the shape of the PSF, before running the deconvolution, you can first do a normal fusion (switch off `Deconvolution`) and then look at a spot in the output image of the sample where there are very small, isolated (spot-like) features: the shape observed there can be used as a

guide for the shape of the PSF that should be targeted, as shown in the PSF-preview window (when changing the PSF width/length sliders). Keep in mind that the PSF is just a rough approximation of the real PSF, and thus fine-tuning its width/length is often not necessary.
- Then try different PSF width/length values in a reasonable range. The number of iterations can usually be kept at 10, but it can make sense to also try e.g. 5 or 15. Compare the respective deconvolved output images from the different test runs, to find the best parameters.
- Once good parameters are found, you can run the deconvolution again with these parameters on the desired full volume for the output image (i.e., disable the small ROI).
- For multi-view deconvolution, a reasonably good registration of the input views is usually necessary to achieve good deconvolution results.

Photomanipulation

If the input folder contains images to which photomanipulation (PM) was applied, the Image Processor supports this in the following ways:

- The images to which PM was applied have a PM "ID" in their metadata (and file name) that uniquely identifies the PM settings that were used (geometrical shape of the PM, laser intensity, etc); and the Image Processor reads this PM ID from the metadata.
- In the `Input` tab, input images can be selected by their PM ID. If PM IDs are absent in entries, this means that no PM was applied for these images.
- Images with *different* PM IDs are separated by the Image Processor:
  - they are *not* content-registered against each other
  - they are *not* fused together
  - for each PM ID, images are shown like a separate "channel" in output files for Imaris, BigDataViewer, etc.
- Content registration (if switched on) is only performed between the images of the one PM ID (or no PM ID) that is selected as `Reference` in the `Input` tab. The resulting correction transforms are then also applied to the *corresponding* images of the other PM IDs. That is, it is assumed that two corresponding images (same stack, objective, time point, channel, camera) that only differ by PM ID need the same transform.

Advanced topics

Only conversion to Imaris or Fiji BigDataViewer formats

To only run a conversion of the input images separately to the formats readable by Imaris or Fiji BigDataViewer, do the following:

1. In the `Input` tab of a new task, select all images to be converted.
2. Switch `Registration` off completely, and switch `Fusion` on.
3. In the `Output` tab, select output resolution etc, and set `Bounding boxes` to `Separate`.
4. Start the task.

This will create a separate Imaris and Fiji BigDataViewer file for each stack.

Manually refine / correct to-sample-space transforms for input images

The Luxendo Images (`.lux.h5`) that are used as input for the Image Processor, carry to-sample-space affine transforms in their metadata. In some cases, it can be useful to refine or correct these transforms manually -- for instance, when some additional alignment calibration transform between multiple color channels is needed, or for manual magnification scaling, or to compensate for a slight rotation of cameras.

The Image Processor provides a mechanism to attach additional arbitrary affine transforms to the input images. These transforms can be either prepended to the original transforms (i.e., applied before them), or appended (i.e., applied after them), or replace the original transforms entirely. For this purpose, a special "multi-transform file" `transforms.mtf.json` can be placed in the parent directory of the `raw` folder (in case of raw input images), or in the directory with the input images (in case of already processed input images). A multi-transform (MTF) file that is in the *same* directory as the images will be considered first for these images. If there is no MTF file present in the same directory, then a MTF file in the directory containing the `raw` folder will be used. There should only be a single MTF file in the directory.

Given a MTF file, the Image Processor will use it when `Input` > `Create in input-image folders` > `New metadata for input images` > `Use multi-transform (MTF) file` is switched on.

Instead of the default paths of the `transforms.mtf.json` file, also a specific MTF file at a specific path can be selected through `Use multi-transform (MTF) file` > `Non-default MTF file path`. If such a path is selected, MTF files at the other locations will be ignored. If, for instance, for a microscope, a specific correction transform should always be applied by default (e.g. to compensate a slight rotation of the left camera), one can select the path to the MTF file that contains the correction transform through the `Non-default MTF file path` field, and then use `Task Configuration` > `Save as default` to have this MTF file selected by default in every new task.

Inside a MTF file, there can be multiple transforms to be prepended and/or appended, or to replace the original transforms. And it is also specified to which images (which time points, channels, stacks, objectives, cameras) the transforms apply. The affine transforms that the MTF file contains will be applied to all input images (of the current task) matched in the file.

Here is an example of the content of a multi-transform file:

```
{
   "affines": [
      {
         "time_points": [0, 1], // <- If empty, apply to all.
         "channels": ["2", "3"],
         "stacks": ["1", "2", "3"],
         "objectives": ["left"],
         "cameras": ["left"],
         "prepend": [
            {
              "matrix": [           // Rows of affine matrix.
                  [1.0, 0.0, 0.0],  // Order: image width, height, depth.
                  [0.0, 1.0, 0.0],  // <- "Identity" transform matrix
                  [0.0, 0.0, 1.0]   // with ones on diagonal: does nothing
              ],
              "translation": [4.12, 0.34, 1.17]  // 3d translation vector
            },                                   // (width, height, depth)
            {
```

```
                "matrix": [
                    [0.5, 0.0, 0.0],  // <- Scaling by factor 0.5 in width
                    [0.0, 0.4, 0.0],  // <-  by factor 0.4 in height
                    [0.0, 0.0, 1.2]   // <-  by factor 1.2 in depth
                ],
                "translation": [3.56, 1.12, -0.45]
            }
        ],
        "replace_by": [
            {
                "matrix": [
                    [1.0, 0.0, 0.0],  // <- "Identity" transform matrix
                    [0.0, 1.0, 0.0],  // with ones on diagonal: does nothing
                    [0.0, 0.0, 1.0]
                ],
                "translation": [5.23, 8.3, -11.07]  // <- Only translation
            }
        ],
        "append": [
            {
                "matrix": [
                    [0.71, -0.71, 0.0], // <- Rotation by 45 degrees
                    [0.71, 0.71, 0.0],
                    [0.0, 0.0, 1.0]   // <- Rotation around depth (z) axis
                ],
                "translation": [0.0, 0.0, 0.0]  // <- zero translation
            },
            {
                "matrix": [
                    [1.0, 0.0, 0.0],  // <- "Identity" transform matrix
                    [0.0, 1.0, 0.0],  // with ones on diagonal: does nothing
                    [0.0, 0.0, 1.0]
                ],
                "translation": [2.23, 0.11, -1.03]  // <- Only translation
            }
        ]
    }
  ]
}
```

In the above example, there is only one item in the list `affines`, but there could also be multiple, if different transforms are to be applied to different `time_points`, `channels`, etc. When a list `time_points`, `channels`, etc, is empty, this means that the corresponding transforms should be applied to *all* time points, channels, etc, found in this directory of input images. To see what the names of the `channels`, `objectives`, `cameras`, etc of the input images are, that can be entered in the filter lists in the MTF file if necessary, you can look at them in the `Input` tab of the GUI after loading the input images.

The list `prepend` contains the transforms to be prepended to the original to-sample-space transforms (the transforms above are just examples). The first transform in the list is the first applied. For instance, correction to the magnification scaling could be applied by a scaling transform that is prepended. On the other hand, the list `append` contains the transforms to be appended, i.e., applied after the original transforms. The lists `prepend` and `append` both can contain multiple transforms that will all be applied one after another. The list

`replace_by` can contain multiple transforms to replace the original transforms. This can be convenient in cases in which it would be complicated to refine/correct the original transforms only by prepending or appending transforms. If `replace_by` is an empty list, then the original transforms are kept. The transforms in `prepend` and `append` will be added *after* replacing the original transforms by the transforms in `replace_by`.

The form

```
{
    "matrix": [
        [0.5, 0.0, 0.0],
        [0.0, 0.4, 0.0],
        [0.0, 0.0, 1.2]
    ],
    "translation": [-14.7, 0.77, 1.32]
}
```

of a single affine transform means: `matrix` contains the rows of the affine matrix, and `translation` contains the 3d translation vector of the affine transform. The order of the elements in `matrix` as well as `translation` is: image width, height, depth.

If you enter affine transforms by hand, please make sure they are valid affine transforms (their matrix is invertible). Otherwise, the Image Processor will throw an error: `Affine transform in multi-transform object is invalid (not invertible)`. For instance, if you enter a pure scale transform with zeros everywhere off the diagonal of the matrix, make sure the values on the diagonal (scale factors) are all non-zero.

An "identity" transform matrix that does nothing, has ones on the diagonal and zeros everywhere off the diagonal. Such an identity-transform matrix is useful when only translation should be applied. Other examples of affine transforms, with the corresponding matrices and images, can be found here.

To deactivate a multi-transform file, you can simply rename its extension, e.g., to `._off_mft.json`.

Run with terminal window

To run the Image-Processor Server with a terminal window, to see additional status, progress, or error messages from the processing tasks, do the following:

1. Open the Windows Task Manager and go to the "Details" tab.
2. Sort the processes according to "Name", and end the task "ips.exe".
3. Open the File Explorer and go to the `C:\Luxendo\Image Processor\<latest version>` folder.
4. In the address bar, enter `cmd` to start a command terminal at this location.
5. In the terminal, type `ips.exe` to start the Image-Processor server.

Now the Image-Processor Server is running with a terminal window, and relevant content (such as error messages) can be read and copied from the window.

Adjust used compute resources

The compute (hardware) resources used by the Image Processor can be adjusted. In the folder `C:\Luxendo\Image Processor\<active version>`, there is a file `config_ips.json` that contains settings to do this. These are settings for the Image-Processor Server (IPS), which manages the different processing tasks, and starts each task with a new instance of an Image-Processor Worker (IPW).

Warning: Changing these settings can lower the performance of the Image Processor, or even cause out-of-memory errors. Also, it can lead to an unintentional amount of system resources being taken away from other software by the Image Processor. Thus, please make such changes with care (especially in a multi-user environment), and backup the configuration file before you change the settings, to be able to revert to the original settings later.

If you want to change the number of tasks that can run simultaneously, you can do this in the `config_ips.json` file through the parameter `max_number_running_tasks`. It can either be set to an integer number (1, 2, ...), or to `null`. `null` (default) means that the number of tasks running simultaneously will be set automatically to the number of available GPUs (with sufficient compute capability). Depending on the hardware used, allowing more tasks to run simultaneously can decrease the total processing time needed by the tasks.

If you encounter out-of-memory errors, you can try to reduce `max_number_running_tasks` and restart LuxControl (to restart the IPS). (Of course, running out of memory could also be caused by other software running simultaneously on the same machine.)

Headless Image Processor

The Image-Processor Worker (IPW) can be run separately as a standalone "headless" program with a command-line interface (CLI). It will then carry out a single processing task, with the settings of the task being passed to the IPW via a `.json` file containing the configuration of the task (paths of the input images, and values for all parameters). As a template for this task-configuration `.json` file, the file `config_task_default.json` can be used, which is located in the subdirectory of the current version of the Image Processor and contains the saved default settings for a task.

This way of running the IPW can be useful in many scenarios, for instance:

- processing many tasks in parallel on multiple machines (e.g. a compute cluster)
- starting the IPW from some other custom script or program, which could then also use the results of the IPW task in an automated fashion (by accessing the output directory of the task)

Going to the subdirectory of the current version of the Image Processor and typing `./ipw --help` in the terminal will print information about the behavior and features of the program and list the available options of the CLI. Some key points are:

- the `--help` output will show how to stop the IPW task gracefully so that output-image data stays intact
- when the IPW is given an output directory of a stopped task, it will automatically try to resume this task

Troubleshooting / FAQ / known issues

- If the output-image files are unexpectedly large, please consider / keep in mind that:

    - the output sampling may accidentally have been set to isotropic (`Output` > `Down/up-sample factor` > `Depth` is not checked)
    - the output-image size is rounded to integer multiples of the chunk size (usually 64 pixels) in each direction
    - the output image usually has multiple resolution layers

- When a task that does `Registration` or `Deconvolution` is started while another application (e.g. TeamViewer) blocks the GPU, this task may hang (i.e., be blocked in the GPU-based step) until the application blocking it is closed.

- Currently, it is *not* supported to run Image-Processor tasks and acquisition on the same PC simultaneously, because these processes may compete over the access to the RAID / hard drive and this way slow each other down. Also sample tracking should currently not be run simultaneously with Image-Processor tasks, because they may compete over the GPU as well as the RAID / hard drive.

- Heavy processing of images in other applications (e.g. Imaris or Fiji) can also negatively interfere with running Image-Processor tasks, when they compete over RAID / hard drive or GPU resources.

- Input images being open in other software, e.g., Fiji, while also being used as input image in a running Image-Processor task, can lead to access problems on the side of the Image Processor, and the processing task can fail. Please close the input images in other applications before starting the task using them as input.

- For each task, the Image Processor estimates the disk space required to store the task's output data. If there is not enough free space available on the target drive for the output data (e.g. the local RAID), it will warn about this and say how much free space is needed. However, third-party software producing data on the same target drive, or copy processes to the same target drive, while Image-Processor tasks are already running, can still cause the tasks to run out of disk space and fail. We thus recommend to stop or pause such external processes that generate data on the same target drive, while Image-Processor tasks are running. This will also lead to better performance of the Image-Processor tasks, since input/output from/to a drive is often a speed bottleneck for the tasks, and so external processes reading from / writing to the same drive, may slow them down significantly.