ASTEC: adaptive segmentation and tracking of embryonic cells

January, 2020

Contents

Chapter 1

Installation

ASTEC (acronym of "adaptive segmentation and tracking of embryonic cells" [Gui15]) has been designed for unix-like systems (e.g. Linux, or MacOS). It has been developed with `python2.7` and was not tested for `python3.0`. It is a set of `python` scripts, built over a set of `C` commands.

There are two distributions. The first one can be retrieved from `github.com/astec-segmentation/astec-2019-published` and includes both the `python` and the `C` codes. The installation procedure is dedicated to this distribution.

The second one is devoluted to more advanced users that may want to benefit from future developements of the ASTEC distribution:

- `python` scripts can be retrieved from `github.com/astec-segmentation/astec`
- `C` code can be retrieved from `gitlab.inria.fr/morpheme/vt`
- optional third-party librairies can be retrieved from `gitlab.inria.fr/morpheme/vt-third-party`

Both `github.com/astec-segmentation/astec-2019-published` and `github.com/astec-segmentation/astec` contains the following 4 sub-directories

```
astec[-2019-published/]
├── documentation/
├── src/
└── tutorial/
```

- `documentation/` contains this documentation.
- `src/` contains the `python` scripts and files (as well as the `C` codes for the `astec-2019-published` distribution).
- `tutorial/` contains a toy data set and the associated parameter files. See chapter 2.

1.1 `github.com/astec-segmentation/astec`

1.1.1 Linux system

This section describes the required command to install the ASTEC distribution on a Linux system (was tested on a Ubuntu system (18.04.2, 64 bits) installed on a virtual machine[1]) so the tutorial (chapter 2) can be run.

[1] `virtualbox.org`

1. Get the distribution. It is recommended (but not necessary) to use `git`, so keeping up to date with the distribution will be easier. `git` can be installed with

   ```
   $ sudo apt install git
   ```

   Then, choose the directory where to install the ASTEC distribution, and download it

   ```
   $ cd /wherever/one/wants/
   $ git clone https://github.com/astec-segmentation/astec-2019-published.git
   ```

   It creates the directory `/wherever/one/wants/astec-2019-published/` that will be denoted `/path/to/astec/` from now on.

2. Prepare the compilation of the `C` code. Compilation is done within the `cmake`[2] framework. The standard Ubuntu distribution comes with a `C` compiler but not with a `C++` one. Last a development version of the `zlib` is required. The next few lines allow to install the required components.

   ```
   $ sudo apt install cmake
   $ sudo apt install cmake-curses-gui
   $ sudo apt install g++
   $ sudo apt install zlib1g-dev
   ```

3. Compile the `C` code.

   ```
   $ cd /path/to/astec/
   $ cd src/ASTEC/CommunFunctions/cpp/vt/
   $ mkdir build
   $ cd build
   $ cmake ../
   $ make
   ```

4. Install the required `python` libraries. As mentioned, ASTEC has been developed with `python2.7`. `pip` is here used for the installation of the `python` libraries. Required libraries are `numpy`, `scipy`, `libtiff`, and `h5py`.

   ```
   $ sudo apt install python2.7
   $ sudo apt install python-pip
   $ sudo pip install numpy
   $ sudo pip install scipy
   $ sudo pip install libtiff
   $ sudo pip install h5py
   ```

5. Make the ASTEC scripts/commands available as on-line commands. It can be done in a terminal (but will be valid only for this terminal)

   ```
   $ export PATH=$PATH:/path/to/astec/src
   ```

   or by adding the above line in the right setup file (e.g. `.bashrc`, `.profile`, ...).

[2] `cmake.org`

Chapter 2

Tutorial

Before starting

It is advised to add to your `PATH` environment variable the paths to both the python and the C executable commands (the latter is important in case of non-standard installation). So, Astec commands can be launched without specifying the complete path to the command.

It can be done in a terminal (and will be valid only for this terminal)

```
$ export PATH=$PATH:/path/to/astec/src
$ export PATH=$PATH:/path/to/astec/src/ASTEC/CommunFunctions/cpp/vt/build/bin
```

or by modifying a setup file (e.g. `bashrc`, `.profile`, ...).

2.1 Tutorial data

The directory `/path/to/astec/tutorial/tuto-astec1/`, also denoted by `path/to/experiment/` or `<EXPERIMENT>`, contains the `RAWDATA/` and `parameters/` sub-directories and a `README` file

```
path/to/tuto-astec1/
├── RAWDATA/
├── README
└── parameters/
```

The `RAWDATA/` contains 21 time points (indexed from 0 to 20) of subsampled (for file size consideration) raw data from a 3D+t movie acquired by a MuViSPIM microscope.

```
RAWDATA/
├── LC/
│   ├── Stack0000/
│   │   ├── Time000000_00.mha.gz
│   │   ├── ...
│   │   └── Time000020_00.mha.gz
│   └── Stack0001/
│       ├── Time000000_00.mha.gz
│       ├── ...
│       └── Time000020_00.mha.gz
└── RC/
    └── Stack0000/
```

```
 1 PATH_EMBRYO = '.'
 2
 3 EN = '2019-Tutorial100'
 4
 5 begin = 0
 6 end = 20
 7
 8 DIR_LEFTCAM_STACKZERO = 'LC/Stack0000'
 9 DIR_RIGHTCAM_STACKZERO = 'RC/Stack0000'
10 DIR_LEFTCAM_STACKONE = 'LC/Stack0001'
11 DIR_RIGHTCAM_STACKONE = 'RC/Stack0001'
12
13 acquisition_orientation = 'right'
14 acquisition_mirrors = False
15 acquisition_resolution = (1., 1., 1.)
16
17 target_resolution = 1.0
```

Figure 2.1: Tutorial parameter file for the fusion step (lines are numbered).

```
    ├── Time000000_00.mha.gz
    ├── ...
    └── Time000020_00.mha.gz
└── Stack0001/
    ├── Time000000_00.mha.gz
    ├── ...
    └── Time000020_00.mha.gz
```

where `LC/` and `LC/` stand respectively for the left and the right cameras.

2.2 Fusion

We assume that we are located in the directory `/path/to/astec/tutorial/tuto-astec1/`. Running the fusion is done with

```
$ 1-fuse.py -p parameters/1-fuse-tutorial-parameters.py
```

`1-fuse-tutorial-parameters.py` being the dedicated parameter file (figure 2.1).

- The variable `PATH_EMBRYO` is the path to the directory where the directory `RAWDATA/` is located. It can be either relative (as in the above example) or global (it could have been `/path/to/astec/tutorial/tuto-astec1/`).
- The variable `EN` is the prefix after which the result fusion images will be named.
- The variables `begin` and `end` set respectively the first and the last index of the input time points to be processed.
- The variables `DIR_LEFTCAM_STACKZERO`, `DIR_RIGHTCAM_STACKZERO`, `DIR_LEFTCAM_STACKONE` and `DIR_RIGHTCAM_STACKONE` set the sub-directories of the `RAWDATA/` directory, where the 4 acquisitions of the SPIM microscope are located.
- The variables `acquisition_orientation` and `acquisition_mirrors` are parameters describing the acquisition geometry.

```
1 PATH_EMBRYO = '.'
2
3 EN = '2019-Tutorial100'
4
5 begin = 0
6 end = 20
```

Figure 2.2: Tutorial parameter file for the sequence intra-registration step.

- The variable `acquisition_resolution` is the voxel size (along the 3 dimensions X, Y and Z).
- The variable `target_resolution` is the desired isotropic (the same along the 3 dimensions) voxel size for the result fusion images.

After processing, a `FUSE/` directory has been created

```
path/to/tuto-astec1/
├── FUSE/
├── RAWDATA/
├── README
└── parameters/
```

The `FUSE/` directory contains

```
FUSE/
└── FUSE_RELEASE/
    ├── 2019-Tutorial100_fuse_t000.mha
    ├── ...
    ├── 2019-Tutorial100_fuse_t020.mha
    └── LOGS/
```

The fused images are named after `<EN>_fuse<XXX>.mha` (where `<XXX>` denotes the value of the variable `XXX`) and indexed from `<begin>` to `<end>` (as the input data).

The directory `LOGS/` contains a copy of the parameter file (stamped with date and hour) as well as a log file (also stamped with date and hour) reporting information about the processing.

2.3 Sequence intra-registration (or drift compensation) [1]

We assume that we are located in the directory `/path/to/astec/tutorial/tuto-astec1/`. Running the sequence intra-registration is done with

```
$ 1.5-intraregistration.py -p parameters/1.5-intraregistration-tutorial-parameters-fuse.py
```

`1.5-intraregistration-tutorial-parameters-fuse.py` being the dedicated parameter file (figure 2.2).

- The variable `PATH_EMBRYO` is the path to the directory where the directory `FUSE/` is located. It can be either relative (as in the above example) or global (it could have been `/path/to/astec/tutorial/tuto-astec1/`).
- The variable `EN` is the prefix after which the images are named.
- The variables `begin` and `end` set respectively the first and the last index of the input time points to be processed.

After processing, a `INTRAREG/` directory has been created

```
path/to/tuto-astec1/
├── FUSE/
├── INTRAREG/
├── RAWDATA/
├── README
└── parameters/
```

The `INTRAREG/` directory contains

```
INTRAREG/
└── INTRAREG_RELEASE/
    ├── CO-TRSFS/
    │   ├── 2019-Tutorial100_intrareg_flo000_ref001.trsf
    │   ├── ...
    │   └── 2019-Tutorial100_intrareg_flo019_ref020.trsf
    ├── FUSE/
    │   └── FUSE_RELEASE/
    │       ├── 2019-Tutorial100_intrareg_fuse_t000.mha
    │       ├── ...
    │       └── 2019-Tutorial100_intrareg_fuse_t020.mha
    ├── LOGS/
    ├── MOVIES/
    │   └── FUSE/
    │       └── FUSE_RELEASE/
    │           └── 2019-Tutorial100_intrareg_fuse_t000-020_xy0205.mha
    └── TRSFS_t0-20/
        ├── 2019-Tutorial100_intrareg_t000.trsf
        ├── ...
        ├── 2019-Tutorial100_intrareg_t020.trsf
        └── template_t0-20.mha
```

- The directory `CO-TRSF/` contains the co-registration transformations.
- The directory `FUSE/FUSE_RELEASE/` contains the resampled fused images in the same geometry (images have the same dimensions along X, Y and Z), with drift compensation (the eventual motion of the sample under the microscope has been compensated).
- The directory `MOVIES/FUSE/FUSE_RELEASE/` contains a 3D (which is a 2D+t) image, here `2019-Tutorial100_intrareg_` which the #205 XY-section of the resampled fused images for all the time points.
- The directory `TRSFS/` contains the transformation of every fused image towards the reference one as well as the template image (an image large enough to including each fused images after resampling).

  The template image `template_t0-20.mha` is of size $422 \times 365 \times 410$ with a voxel size of 0.6 (the voxel size can be set by the variable `intra_registration_resolution`)

2.4 Segmentation of the first time point

We assume that we are located in the directory `/path/to/astec/tutorial/tuto-astec1/`. Segmenting the first time point is done with

```
$ 2-mars.py -p parameters/2-mars-tutorial-parameters.py
```

`2-mars-tutorial-parameters.py` being the dedicated parameter file (figure 2.3).

```
1 PATH_EMBRYO = '.'
2
3 EN = '2019-Tutorial100'
4
5 begin = 0
```

Figure 2.3: Tutorial parameter file for the segmentation of the first time point.

- The variable `PATH_EMBRYO` is the path to the directory where the directory `FUSE/` is located. It can be either relative (as in the above example) or global (it could have been `/path/to/astec/tutorial/tuto-astec1/`).
- The variable `EN` is the prefix after which the images are named.
- The variable `begin` sets the index of the first input time point (to be processed).

After processing, a `SEG/` directory has been created

```
path/to/tuto-astec1/
├── FUSE/
├── INTRAREG/
├── RAWDATA/
├── README
├── SEG/
└── parameters/
```

The `SEG/` directory contains

```
SEG/
├── SEG_RELEASE/
│   ├── 2019-Tutorial100_mars_t000.mha
│   └── LOGS/
```

`2019-Tutorial100_mars_t000.mha` is the segmented first time point of the sequence.

2.5 Correction of the first time point segmentation

We assume that we are located in the directory `/path/to/astec/tutorial/tuto-astec1/`. Correcting the first time point segmentation is done with

```
$ 3-manualcorrection.py -p parameters/3-manualcorrection-tutorial-parameters.py
```

`3-manualcorrection-tutorial-parameters.py` being the dedicated parameter file (figure 2.4).

- The variable `PATH_EMBRYO` is the path to the directory where the directory `SEG/` is located. It can be either relative (as in the above example) or global (it could have been `/path/to/astec/tutorial/tuto-astec1/`).
- The variable `EN` is the prefix after which the images are named.
- The variable `begin` set the index of the first input time point (to be processed).
- The variable `mancor_mapping_file` gives the file name containing the correction to be applied.

After processing, the `SEG/` directory contains

```
1 PATH_EMBRYO = '.'
2
3 EN = '2019-Tutorial100'
4
5 begin = 0
6
7 mancor_mapping_file='parameters/3-manualcorrection-tutorial.txt'
```

Figure 2.4: Tutorial parameter file for the segmentation correction of the first time point. See figure 2.5 for the `<mancor_mapping_file>` file.

```
10 6
20 13
9 4
26 11
21 11
27 15
32 23
39 31
35 20
38 43
46 45
52 42
58 62
63 60
78 67
74 66
68 66
83 75
82 77
```

Figure 2.5: The segmentation correction file `3-manualcorrection-tutorial.txt` for the first time point. The first number is the line index (lines are numbered).

```
1 PATH_EMBRYO = '.'
2
3 EN = '2019-Tutorial100'
4
5 begin = 0
6 end = 20
7
8 result_lineage_suffix = 'pkl'
```

Figure 2.6: Tutorial parameter file for the segmentation propagation.

```
SEG/
└── SEG_RELEASE/
    ├── 2019-Tutorial100_mars_t000.mha
    ├── 2019-Tutorial100_seg_t000.mha
    ├── LOGS/
    └── RECONSTRUCTION/
```

`2019-Tutorial100_seg_t000.mha` is the corrected version of the segmentation obtained at the previous step.

2.6 Segmentation propagation

We assume that we are located in the directory `/path/to/astec/tutorial/tuto-astec1/`. Segmenting the first time point is done with

```
$ 4-astec.py -p parameters/4-astec-tutorial-parameters.py
```

`4-astec-tutorial-parameters.py` being the dedicated parameter file (figure 2.6).

After processing, the `SEG/` directory contains

```
SEG/
└── SEG_RELEASE/
    ├── 2019-Tutorial100_mars_t000.mha
    ├── 2019-Tutorial100_seg_lineage.pkl
    ├── 2019-Tutorial100_seg_t000.mha
    ├── 2019-Tutorial100_seg_t001.mha
    ├── ...
    ├── 2019-Tutorial100_seg_t020.mha
    └── LOGS/
```

`2019-Tutorial100_seg_lineage.pkl` is a pickle python file containing a dictionary (in the python sense). It can be read by

```
 1 PATH_EMBRYO = '.'
 2
 3 EN = '2019-Tutorial100'
 4
 5 begin = 0
 6 end = 20
 7
 8 EXP_INTRAREG = 'SEG'
 9
10 intra_registration_template_type = "SEGMENTATION"
11 intra_registration_template_threshold = 2
12 intra_registration_margin = 20
13
14 intra_registration_resample_segmentation_images = True
15 intra_registration_movie_segmentation_images = True
```

Figure 2.7: Tutorial parameter file for the sequence intra-registration step, segmentation images being used to build the template.

```
$ python
...
>>> import cPickle as pkl
>>> f = open('2019-Tutorial100_seg_lineage.pkl', 'r')
>>> d = pkl.load(f)
>>> f.close()
>>> d.keys()
['cell_lineage', 'cell_volume']
```

In this pickle file, cells have an unique identifier $i * 1000 + c$, which is made of both the image index i and the cell identifier c within a segmentation image (recall that, within an image, cells are numbered from 2, 1 being the background label).

2.7 Sequence intra-registration (or drift compensation) [2]

We assume that we are located in the directory `/path/to/astec/tutorial/tuto-astec1/`. Running the sequence intra-registration is done with

```
$ 1.5-intraregistration.py -p parameters/1.5-intraregistration-tutorial-parameters-seg.py
```

`1.5-intraregistration-tutorial-parameters-seg.py` being the dedicated parameter file (figure 2.7).

- The variable `PATH_EMBRYO` is the path to the directory where the directory `FUSE/` is located. It can be either relative (as in the above example) or global (it could have been `/path/to/astec/tutorial/tuto-astec1/`).
- The variable `EN` is the prefix after which the images are named.
- The variables `begin` and `end` set respectively the first and the last index of the input time points to be processed.
- the variable `EXP_INTRAREG` set the suffix of the sub-directory of the `INTRAREG/` directory to be created.
- the variable `intra_registration_template_type` set the images to be used to build the template. Here, since it is equal to `'SEGMENTATION'`, they are the segmentation images obtained at the previous step.

The variable `intra_registration_template_threshold` set a threshold to be applied to the template images to define the information to be kept: we want all the points with a value equal or greater than 2 to be contained in the template after resampling. Since cells are labeled from 2 and above, the template is designed to contain all labeled cells after resampling, so it is built as small as possible.

The variable `intra_registration_margin` allows to add margins (in the 3 dimensions) to the built template.

- The variable `intra_registration_resample_segmentation_images` indicates whether the segmentation images are to be resampled in the template geometry.
- The variable `intra_registration_movie_segmentation_images` indicates whether 2D+t movies have to be built from the resampled segmentation images.

After processing, a `INTRAREG/INTRAREG_SEG/` directory has been created and the `INTRAREG/` directory now contains

```
INTRAREG/
├── INTRAREG_RELEASE/
│   └── ...
└── INTRAREG_SEG/
    ├── CO-TRSFS/
    │   ├── 2019-Tutorial100_intrareg_flo000_ref001.trsf
    │   ├── ...
    │   └── 2019-Tutorial100_intrareg_flo019_ref020.trsf
    ├── FUSE/
    │   └── FUSE_RELEASE/
    │       ├── 2019-Tutorial100_intrareg_fuse_t000.mha
    │       ├── ...
    │       └── 2019-Tutorial100_intrareg_fuse_t020.mha
    ├── LOGS/
    ├── MOVIES/
    │   ├── FUSE/
    │   │   └── FUSE_RELEASE/
    │   │       └── 2019-Tutorial100_intrareg_fuse_t0-20_xy174.mha
    │   └── SEG/
    │       └── SEG_RELEASE/
    │           └── 2019-Tutorial100_intrareg_seg_t0-20_xy174.mha
    ├── SEG/
    │   └── SEG_RELEASE/
    │       ├── 2019-Tutorial100_intrareg_seg_t000.mha
    │       ├── ...
    │       └── 2019-Tutorial100_intrareg_seg_t020.mha
    └── TRSFS_t0-20/
        ├── 2019-Tutorial100_intrareg_t000.trsf
        ├── ...
        ├── 2019-Tutorial100_intrareg_t020.trsf
        └── template_t0-20.mha
```

In addition to directories already described in section 2.3, the `INTRAREG_SEG/` directory contains

- The directory `SEG/SEG_RELEASE` contains the resampled segmentation images in the same geometry (images have the same dimensions along X, Y and Z), with drift compensation (the eventual motion of the sample under the microscope has been compensated).

```
1 PATH_EMBRYO = '.'
2
3 EN = '2019-Tutorial100'
4
5 begin = 0
6 end = 20
7
8 EXP_INTRAREG = 'SEG'
```

Figure 2.8: Tutorial parameter file for the sequence properties from the co-registered segmentation images.

- In addition to a 2D+t movie made from the resampled fusion images, the directory `MOVIES/` contains a 2D+t movie made from the resampled segmentation images in the sub-directory `SEG/SEG_RELEASE`.
- The template image `template_t0-20.mha` in the directory `TRSFS/` is now of size $323 \times 265 \times 348$ with a voxel size of 0.6, which is smaller than the one computed in section 2.3, even with the added margins.

  Note that all resampled images (in both the `FUSE/FUSE_RELEASE` and the `SEG/SEG_RELEASE` directories have the same geometry than the template image.

2.8 Sequence properties computation [1]

We assume that we are located in the directory `/path/to/astec/tutorial/tuto-astec1/`. Computing cell properties as well as lineage assumes that segmentation or post-corrected segmentation (see section 2.11) images have been co-registered (see sections 2.7 and 2.10). Extracting the sequence properties from the co-registered segmentation images is done with

```
$ X-embryoproperties.py -p parameters/X-embryoproperties-tutorial-parameters-seg.py
```

`X-embryoproperties-tutorial-parameters-seg.py` being the dedicated parameter file (figure 2.8).

- The variable `PATH_EMBRYO` is the path to the directory where the directory `FUSE/` is located. It can be either relative (as in the above example) or global (it could have been `/path/to/astec/tutorial/tuto-astec1/`).
- The variable `EN` is the prefix after which the images are named.
- The variables `begin` and `end` set respectively the first and the last index of the input time points to be processed.
- the variable `EXP_INTRAREG` set the suffix of the sub-directory of the `INTRAREG/` directory where to search post-corrected segmentation or segmentation images.

  Since the directory `INTRAREG/INTRAREG_SEG/` only contains the co-registered segmentation images (in the `SEG/SEG_RELEASE/` sub-directory), properties will be computed from these images.

After processing, some files appears in the `INTRAREG/INTRAREG_SEG/SEG/SEG_RELEASE/` sub-directory

```
INTRAREG/
├── INTRAREG_RELEASE/
│   └── ...
└── INTRAREG_SEG/
    ├── CO-TRSFS/
    │   └── ...
    └── FUSE/
```

```
│   ├── ...
├── LOGS/
├── MOVIES/
│   ├── ...
├── SEG/
│   ├── SEG_RELEASE/
│   │   ├── 2019-Tutorial100_intrareg_seg_lineage.pkl
│   │   ├── 2019-Tutorial100_intrareg_seg_lineage.tlp
│   │   ├── 2019-Tutorial100_intrareg_seg_lineage.txt
│   │   ├── 2019-Tutorial100_intrareg_seg_lineage.xml
│   │   ├── 2019-Tutorial100_intrareg_seg_t000.mha
│   │   ├── ...
│   │   └── 2019-Tutorial100_intrareg_seg_t020.mha
├── TRSFS_t0-20/
│   ├── ...
```

`2019-Tutorial100_intrareg_seg_lineage.pkl` is a pickle python file containing a dictionary (in the python sense). It can be read by

```
$ python
...
>>> import cPickle as pkl
>>> f=open('2019-Tutorial100_intrareg_seg_lineage.pkl')
>>> d=pkl.load(f)
>>> f.close()
>>> d.keys()
['all_cells', 'cell_barycenter', 'cell_contact_surface',
'cell_principal_vectors', 'cell_principal_values', 'cell_volume',
'cell_compactness', 'cell_surface', 'cell_lineage']
```

In this pickle file (as in the one computed at section 2.6), cells have an unique identifier $i * 1000 + c$, which is made of both the image index i and the cell identifier c within a segmentation image (recall that cells are numbered from 2, 1 being the background label).

`2019-Tutorial100_intrareg_seg_lineage.xml` contains the same information than the pickle file, but in xml format (see figure 2.9).

`2019-Tutorial100_intrareg_seg_lineage.tst` contains some *diagnosis* information (smallest and largest cells, weird lineages, etc.).

2.9 Segmentation post-correction

We assume that we are located in the directory `/path/to/astec/tutorial/tuto-astec1/`. Segmentation post-correction is done with

```
$ 5-postcorrection.py -p parameters/5-postcorrection-tutorial-parameters.py
```

`5-postcorrection-tutorial-parameters.py` being the dedicated parameter file (figure 2.10).

After processing, a `POST/` directory has been created

```
path/to/tuto-astec1/
├── FUSE/
├── INTRAREG/
├── POST/
├── RAWDATA/
└── README
```

```
<data>
  <cell_volume>
    ...
  </cell_volume>
  <cell_surface>
     ...
  </cell_surface>
  <cell_compactness>
     ...
  </cell_compactness>
  <cell_barycenter>
     ...
  </cell_barycenter>
  <cell_principal_values>
     ...
  </cell_principal_values>
  <cell_principal_vectors>
     ...
  </cell_principal_vectors>
  <cell_contact_surface>
     ...
  </cell_contact_surface>
  <all_cells>[2, 3, 4, 5, 6, 7, 8, 11, 12, 13,
     ...
     200097, 200099, 200100, 200101, 200102]</all_cells>
  <cell_lineage>
     ...
  </cell_lineage>
</data>
```

Figure 2.9: XML output properties file from the co-registered segmentation image.

```
1 PATH_EMBRYO = '.'
2
3 EN = '2019-Tutorial100'
4
5 begin = 0
6 end = 20
7
8 result_lineage_suffix = 'pkl'
```

Figure 2.10: Tutorial parameter file for the segmentation post-correction.

```
 1 PATH_EMBRYO = '.'
 2
 3 EN = '2019-Tutorial100'
 4
 5 begin = 0
 6 end = 20
 7
 8 EXP_INTRAREG = 'POST'
 9
10 intra_registration_template_type = "POST-SEGMENTATION"
11 intra_registration_template_threshold = 2
12 intra_registration_margin = 20
13
14 intra_registration_resample_post_segmentation_images = True
15 intra_registration_resample_segmentation_images = True
16 intra_registration_movie_post_segmentation_images = True
17 intra_registration_movie_segmentation_images = True
```

Figure 2.11: Tutorial parameter file for the sequence intra-registration step, post-segmentation images being used to build the template.

```
├── SEG/
└── parameters/
```

The `POST/` directory contains

```
POST/
└── POST_RELEASE/
    ├── 2019-Tutorial100_post_lineage.pkl
    ├── 2019-Tutorial100_post_t000.mha
    ├── ...
    ├── 2019-Tutorial100_post_t020.mha
    └── LOGS/
```

2.10 Sequence intra-registration (or drift compensation) [3]

We assume that we are located in the directory `/path/to/astec/tutorial/tuto-astec1/`. Running the sequence intra-registration is done with

```
$ 1.5-intraregistration.py -p parameters/1.5-intraregistration-tutorial-parameters-post.py
```

`1.5-intraregistration-tutorial-parameters-post.py` being the dedicated parameter file (figure 2.11).

- The variable `PATH_EMBRYO` is the path to the directory where the directory `FUSE/` is located. It can be either relative (as in the above example) or global (it could have been `/path/to/astec/tutorial/tuto-astec1/`).
- The variable `EN` is the prefix after which the images are named.
- The variables `begin` and `end` set respectively the first and the last index of the input time points to be processed.

- the variable `EXP_INTRAREG` set the suffix of the sub-directory of the `INTRAREG/` directory to be created.
- the variable `intra_registration_template_type` set the images to be used to build the template. Here, since it is equal to `"POST-SEGMENTATION''`, they are the post-corrected segmentation images obtained at the previous step.

  The variable `intra_registration_template_threshold` set a threshold to be applied to the template images to define the information to be kept: we want all the points with a value equal or greater than 2 to be contained in the template after resampling. Since cells are labeled from 2 and above, the template is designed to contain all labeled cells after resampling, so it is built as small as possible.

  The variable `intra_registration_margin` allows to add margins (in the 3 dimensions) to the built template.
- The variable `intra_registration_resample_post_segmentation_images` indicates whether the post-corrected segmentation images are to be resampled in the template geometry.
- The variable `intra_registration_resample_segmentation_images` indicates whether the segmentation images are to be resampled in the template geometry.
- The variable `intra_registration_movie_post_segmentation_images` indicates whether 2D+t movies have to be built from the resampled post-corrected segmentation images.
- The variable `intra_registration_movie_segmentation_images` indicates whether 2D+t movies have to be built from the resampled segmentation images.

After processing, a `INTRAREG/INTRAREG_POST/` directory has been created and the `INTRAREG/` directory now contains

```
INTRAREG/
└── INTRAREG_POST/
    ├── CO-TRSFS/
    │   ├── 2019-Tutorial100_intrareg_flo000_ref001.trsf
    │   ├── ...
    │   └── 2019-Tutorial100_intrareg_flo019_ref020.trsf
    ├── FUSE/
    │   └── FUSE_RELEASE/
    │       ├── 2019-Tutorial100_intrareg_fuse_t000.mha
    │       ├── ...
    │       └── 2019-Tutorial100_intrareg_fuse_t020.mha
    ├── LOGS/
    ├── MOVIES/
    │   ├── FUSE/
    │   │   └── FUSE_RELEASE/
    │   │       └── 2019-Tutorial100_intrareg_fuse_t0-20_xy174.mha
    │   ├── POST/
    │   │   └── POST_RELEASE/
    │   │       └── 2019-Tutorial100_intrareg_post_t0-20_xy174.mha
    │   └── SEG/
    │       └── SEG_RELEASE/
    │           └── 2019-Tutorial100_intrareg_seg_t0-20_xy174.mha
    ├── POST/
    │   └── POST_RELEASE/
    │       ├── 2019-Tutorial100_intrareg_post_t000.mha
    │       ├── ...
    │       └── 2019-Tutorial100_intrareg_post_t020.mha
    └── SEG/
        └── SEG_RELEASE/
```

```
1 PATH_EMBRYO = '.'
2
3 EN = '2019-Tutorial100'
4
5 begin = 0
6 end = 20
7
8 EXP_INTRAREG = 'POST'
```

Figure 2.12: Tutorial parameter file for the sequence properties from the co-registered post-corrected segmentation images.

```
        ├── 2019-Tutorial100_intrareg_seg_t000.mha
        ├── ...
        └── 2019-Tutorial100_intrareg_seg_t020.mha
    ├── TRSFS_t0-20/
    │   ├── 2019-Tutorial100_intrareg_t000.trsf
    │   ├── ...
    │   ├── 2019-Tutorial100_intrareg_t020.trsf
    │   └── template_t0-20.mha
├── INTRAREG_RELEASE/
│   └── ...
├── INTRAREG_SEG/
│   └── ...
```

In addition to directories already described in section 2.3, the `INTRAREG_POST/` directory contains

- The directory `POST/POST_RELEASE/` contains the resampled post-corrected segmentation images in the same geometry (images have the same dimensions along X, Y and Z), with drift compensation (the eventual motion of the sample under the microscope has been compensated).
- In addition to a 2D+t movie made from the resampled fusion and the segmentation images, the directory `MOVIES/` contains a 2D+t movie made from the resampled post-corrected segmentation images in the sub-directory `POST/POST_RELEASE/`.
- The template image `template_t0-20.mha` in the directory `TRSFS/` is now of size $323 \times 265 \times 348$ with a voxel size of 0.6, has the same size than the one computed in section 2.7, which is expected since the post-correction does not change the background. Note that all resampled images (in the `FUSE/FUSE_RELEASE/`, the `POST/POST_RELEASE/`, and the `SEG/SEG_RELEASE/` directories have the same geometry than the template image.

2.11 Sequence properties computation [2]

We assume that we are located in the directory `/path/to/astec/tutorial/tuto-astec1/`. Computing cell properties as well as lineage assumes that segmentation or post-corrected segmentation (see section 2.11) images have been co-registered (see sections 2.7 and 2.10). Extracting the sequence properties from the co-registered segmentation images is done with

```
$ X-embryoproperties.py -p parameters/X-embryoproperties-tutorial-parameters-post.py
```

`X-embryoproperties-tutorial-parameters-post.py` being the dedicated parameter file (figure 2.12).

- The variable `PATH_EMBRYO` is the path to the directory where the directory `FUSE/` is located. It can be either relative (as in the above example) or global (it could have been `/path/to/astec/tutorial/tuto-astec1/`).
- The variable `EN` is the prefix after which the images are named.
- The variables `begin` and `end` set respectively the first and the last index of the input time points to be processed.
- the variable `EXP_INTRAREG` set the suffix of the sub-directory of the `INTRAREG/` directory where to search post-corrected segmentation or segmentation images.

  Since the directory `INTRAREG/INTRAREG_POST/` contains the co-registered post-corrected segmentation images (in the `POST/POST_RELEASE/` sub-directory), properties will be computed from these images preferably to the co-registered segmentation images (in the `SEG/SEG_RELEASE/` sub-directory).

After processing, some files appears in the `INTRAREG/INTRAREG_POST/POST/` sub-directory

```
INTRAREG/
├── INTRAREG_RELEASE/
│   └── ...
└── INTRAREG_SEG/
    ├── CO-TRSFS/
    │   └── ...
    ├── FUSE/
    │   └── ...
    ├── LOGS/
    ├── MOVIES/
    │   └── ...
    ├── POST/
    │   └── POST_RELEASE/
    │       ├── 2019-Tutorial100_intrareg_post_lineage.pkl
    │       ├── 2019-Tutorial100_intrareg_post_lineage.txt
    │       ├── 2019-Tutorial100_intrareg_post_lineage.xml
    │       ├── 2019-Tutorial100_intrareg_post_t000.mha
    │       ├── ...
    │       └── 2019-Tutorial100_intrareg_post_t020.mha
    ├── SEG/
    │   └── ...
    └── TRSFS_t0-20/
        └── ...
```

Those files have the same content than the ones already presented in section 2.8.

Chapter 3

User guide: command line interfaces

The Astec distribution contains 4 sub-directories

```
path/to/astec/
├── documentation/
├── src/
└── tutorial/
```

- `documentation/` contains this documentation
- `src/` contains the command line interfaces (CLIs) and the code.
- `tutorial/` contains a tutorial (see chap. 2) along with a toy example.

3.1 Data organization

It is assumed that there will be one directory per experiment. This directory contains the acquired data, but will also contain the result data as depicted below. See section 3.11.3 for more details.

```
/path/to/experiment/
├── RAWDATA/
│   └── ...
├── FUSE/
│   └── ...
├── SEG/
│   └── ...
└── POST/
    └── ...
```

`RAWDATA/` is assumed to contain the raw data (ie acquired images from the MuViSPIM microscope), while the other subdirectories will contain processing results.

3.2 Command line interfaces common options

`-h, --help`
: prints a help message

`-p file, --parameters file`
: set the parameter file to be parsed

`-e path, --embryo-rep path`
: set the `path` to the directory where the `RAWDATA/` directory is located. Can also be given in the parameter file by the variable `PATH_EMBRYO`.

`-k, --keep-temporary-files`
: allows to keep the temporary files. Not to be routinely used.

`-f, --force`
: forces execution, even if (temporary) result files are already existing

`-v, --verbose`
: increases verboseness (both at console and in the log file)

`-nv, --no-verbose`
: no verboseness

`-d, --debug`
: increases debug information (in the log file)

`-nd, --no-debug`
: no debug information

`-pp, --print-param`
: print parameters in console and exit. A parameter file has to be provided (`-p` option). Allows to check the parameters that will be used before any processing; it is also a means to have access to the whole parameter list.

3.3 1-fuse.py

3.3.1 Fusion method overview

The fusion is made of the following steps.

1. Optionally, a slit line correction. Some Y lines may appear brighter in the acquisition and causes artifacts in the reconstructed (i.e. fused) image. By default, it is not done.
2. A change of resolution in the X and Y directions only (Z remains unchanged). It allows to decrease the data volume (and then the computational cost) if the new pixel size (set by `target_resolution`) is larger than the acquisition one.
3. Optionally, a crop of the resampled acquisitions. It allows to decrease the volume of data, hence the computational cost. The crop is based on the analysis of a MIP view (in the Z direction) of the volume, and thus is sensitive to hyper-intensities if any. By default, it is done.
4. Optionally, a mirroring of the images:
   - if the `acquisition_mirrors` variable is set to `False`, a mirroring along the X axis of the 'right camera' images (see also section 3.3.2), and
   - if the `acquisition_leftcamera_z_stacking` variable is set to `'inverse'`, a mirroring along the Z axis of both 'left camera' and 'right camera' images (see also section 3.3.2).
5. Co-registration of the 3 last images onto the first one (the acquisition from the left camera for stack #0) considered as a reference. The reference image is resampled again, to get an isotropic voxel (whose size is given by `target_resolution`), i.e. the voxel size is the same along the 3 directions: X, Y, Z. There are two alternative methods.
   (a) The direct fusion method. Each of the 3 last images is *linearly* co-registered onto the reference image.
   (b) The hierarchical method. Each stack is first reconstructed (with the acquisition couple of both left and right cameras), then stack #1 is *non-linearly* co-registered onto stack #0. From this last registration, non-linear co-registrations are deduced for the stack #1 acquisitions, while linear co-registration is still considered for the right camera acquisition of stack #0.

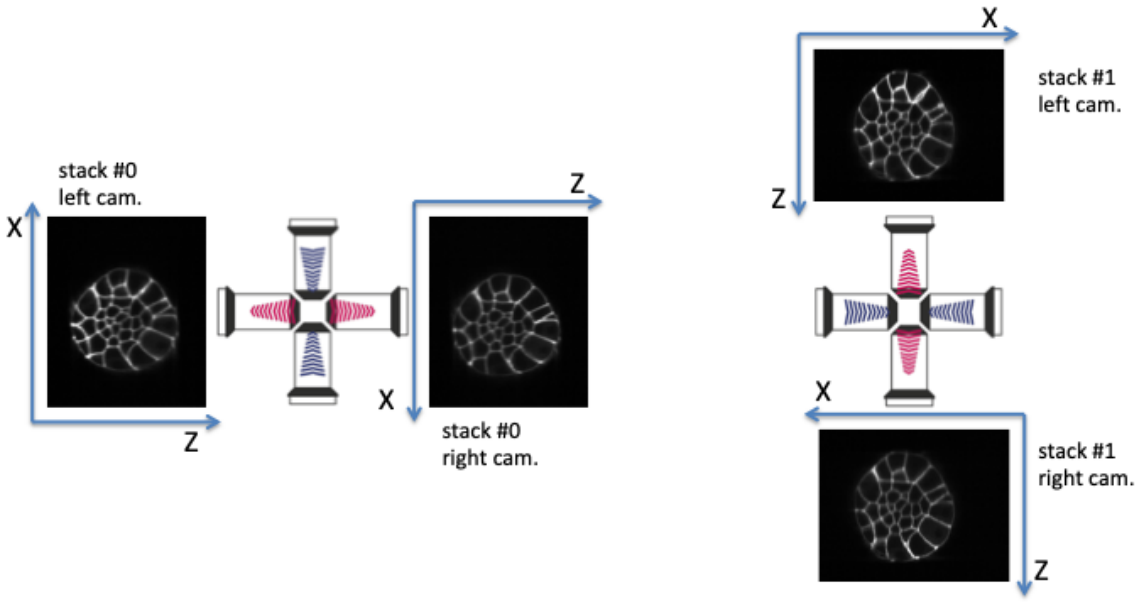

Figure 3.1: Multiview lightsheet microscope acquisition: at a time point, two acquisitions (stack #0 and stack #1) are sequentially performed, the second one orthogonal to the first. For each acquisition, two 3D intensity image stacks are acquired, respectively by the left and the right cameras. It yields four image stacks to be fused. The frame $(\mathbf{X}, \mathbf{Z})$ of the left camera of stack #0 needs to be rotated clockwise (90 degrees along the $\mathbf{Y}$ axis) to correspond to the frame of the left camera of stack #1: `acquisition_orientation` has to be set to `'right'` if `acquisition_leftcamera_z_stacking` is set to `'direct'`.

6. Weighted linear combination of images.
7. Optionally, a crop of the fused image, still based on the analysis of a MIP view (in the Z direction). By default, it is done.

3.3.2 Important parameters in the parameter file

A simple parameter file for fusion is described in the tutorial section 2.2.

Indicating the right values of the acquisition parameters is crucial; these parameters are

- `acquisition_mirrors` (or `raw_mirrors`) is a parameter indicating whether the right camera images have already been mirrored along the X axis (so that the X axis direction is the one of the left cameras) or not. Its value is either `False` or `True`. Such a parameter should depend on the acquisition apparatus (ie the microscope) and the should be identical for all acquisitions.

  In acquisitions depicted in figures 3.1 and 3.2, it can be seen that the X-axis of the right camera image is inverted with respect to the left camera image. `acquisition_mirrors` has to be set to `'False'`
- `acquisition_orientation` (or `raw_ori`) is a parameter describing the acquisition orientation of the acquisition of the stack #1 images with respect to the stack #0 ones.

  - `'right'`: the frame $(\mathbf{X}, \mathbf{Z})$ of the left camera of stack #0 needs to be rotated clockwise (90 degrees along the $\mathbf{Y}$ axis) to correspond to the left camera of stack #1 (see figure 3.1).
  - `'left'`: the frame $(\mathbf{X}, \mathbf{Z})$ of the left camera of stack #0 needs to be rotated counterclockwise (-90 degrees along the $\mathbf{Y}$ axis) to correspond to the left camera of stack #1 (see figure 3.2).

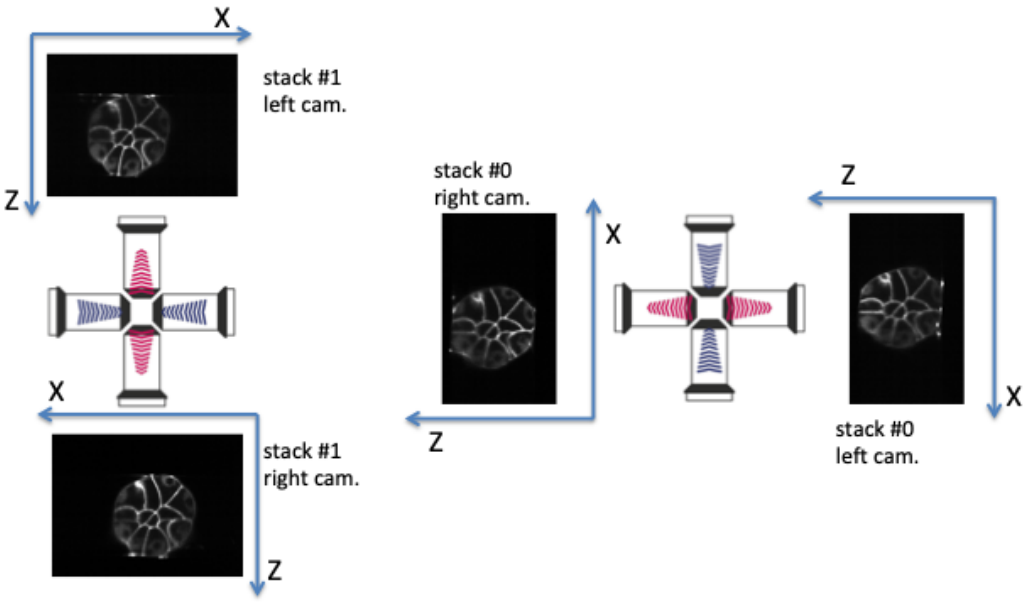

Figure 3.2: The frame $(\mathbf{X}, \mathbf{Z})$ of the left camera of stack #0 needs to be rotated counterclockwise (-90 degrees along the $\mathbf{Y}$ axis) to correspond to the frame of the left camera of stack #1: `acquisition_orientation` has to be set to `'left'` if `acquisition_leftcamera_z_stacking` is set to `'direct'`.

- `acquisition_leftcamera_z_stacking` gives the order of stacking of in the Z direction for the left camera images.
  - `'direct'`: **z** increases from the high-contrasted images to the blurred ones (see figure 3.1).
  - `'inverse'`: **z** increases from the blurred images to the high-contrasted ones (see figure 3.2).

  Looking at XZ-sections of the registered images (see figures 3.5, 3.6, 3.7, and 3.8) provides an efficient means to check whether this parameter is correctly set (see also section 3.3.7).
- `acquisition_resolution` (or `raw_resolution`) is the voxel size (along the 3 dimensions X, Y and Z) of the acquired images.
- `target_resolution` is the desired isotropic (the same along the 3 dimensions) voxel size for the result fusion images.
- `begin` gives the index of the first time point to be processed.
- `end` gives the index of the last time point to be processed.

When one may not be sure of the `raw_ori`, `raw_mirrors`, and `acquisition_leftcamera_z_stacking`right values, it is advised to perform the fusion on only one time point (by indicating the same index for both `begin` and `end`), e.g. with the four possibilities for the variable couple (`raw_ori`, `raw_mirrors`), i.e. (`'left'`, `False`), (`'left'`, `True`), (`'right'`, `False`), and (`'right'`, `True`). It comes to write four parameter files that differ only for the parameters `raw_ori`, `raw_mirrors`, and `EXP_FUSE` (to store the fusion result in different directories, see section 3.3.4). For these first experiments, it is advised

- to set `target_resolution` to a large value, in order to speed up the calculations, and
- to set `fusion_xzsection_extraction` to `True`, in order to check whether `acquisition_leftcamera_z_stacking` was correctly set (see also section 3.3.7).

Please recall that `raw_ori` should depend on the acquisition apparatus (ie the microscope), and should not change for all the other acquisitions on the same microscope (unless the microscope settings change). Then, for most experiments, one change only to test the value of `raw_ori`.

Please note that changing the value of `acquisition_leftcamera_z_stacking` implies to change also the value of `acquisition_orientation`.

3.3.3 Input data

Input data (acquired images from the MuViSPIM microscope, see figures 3.1 and 3.2) are assumed to be organized in a separate `RAWDATA/` directory in the `/path/to/experiment/` directory as depicted below.

- `RAWDATA/LC/Stack000` contains the images acquired at the first angulation by the left camera.
- `RAWDATA/LC/Stack001` contains the images acquired at the second angulation by the left camera.
- `RAWDATA/RC/Stack000` contains the images acquired at the first angulation by the right camera.
- `RAWDATA/RC/Stack001` contains the images acquired at the second angulation by the right camera.

```
/path/to/experiment/
├── RAWDATA/
│   ├── LC/
│   │   ├── Stack0000/
│   │   │   ├── Time000xxx_00.zip
│   │   │   ├── ...
│   │   │   └── Time000xxx_00.zip
│   │   └── Stack0001/
│   │       ├── Time000xxx_00.zip
│   │       ├── ...
│   │       └── Time000xxx_00.zip
│   └── RC/
│       ├── Stack0000/
│       │   ├── Time000xxx_00.zip
│       │   ├── ...
│       │   └── Time000xxx_00.zip
│       └── Stack0001/
│           ├── Time000xxx_00.zip
│           ├── ...
│           └── Time000xxx_00.zip
└── ...
```

where `xxx` denotes a three digit number (e.g. 000, 001, ...) denoting the time point of each acquisition. The range of time points to be fused are given by the variables `begin` and `end`, while the path `/path/to/experiment/` has to be assigned to the variable `PATH_EMBRYO`

Hence a parameter file containing

```
PATH_EMBRYO = /path/to/experiment/
begin = 0
end = 10
```

indicates that time points in $[0, 10]$ of the `RAWDATA/` subdirectory of `/path/to/experiment/` have to be fused.

3.3.3.1 Input data directory names

However, directories may be named differently. The variables `DIR_RAWDATA`, `DIR_LEFTCAM_STACKZERO`, `DIR_RIGHTCAM_STACKZER` `DIR_LEFTCAM_STACKONE`, and `DIR_RIGHTCAM_STACKONE` allow a finer control of the directory names. The images acquired at the first angulation by the left and the right cameras are searched in the directories

<PATH_EMBRYO>/<DIR_RAWDATA>/<DIR_LEFTCAM_STACKZERO>
<PATH_EMBRYO>/<DIR_RAWDATA>/<DIR_RIGHTCAM_STACKZERO>

while the images acquired at the second angulation by the left and the right cameras are searched in the directories

<PATH_EMBRYO>/<DIR_RAWDATA>/<DIR_LEFTCAM_STACKONE>
<PATH_EMBRYO>/<DIR_RAWDATA>/<DIR_RIGHTCAM_STACKONE>

where <XXX> denotes the value of the variable XXX. Then, to parse the following data architecture

```
/path/to/experiment/
├── my_raw_data/
│   ├── LeftCamera/
│   │   ├── FirstStack/
│   │   │   └── ...
│   │   └── SecondStack/
│   │       └── ...
│   └── RightCamera/
│       ├── FirstStack/
│       │   └── ...
│       └── SecondStack/
│           └── ...
└── ...
```

one has to add the following lines in the parameter file

```
DIR_RAWDATA = 'my_raw_data'
DIR_LEFTCAM_STACKZERO = 'LeftCamera/FirstStack'
DIR_RIGHTCAM_STACKZERO = 'RightCamera/FirstStack'
DIR_LEFTCAM_STACKONE = 'LeftCamera/SecondStack'
DIR_RIGHTCAM_STACKONE = 'RightCamera/SecondStack'
```

It has to be noted that, when the stacks of a given time point are in different directories, image file names are tried to be guessed from the directories parsing. It has to be pointed out that indexes have to be encoded with a 3-digit integer with 0 padding (i.e. 000, 001, ...) and that has to be the only variation in the file names (within each directory).

3.3.3.2 Input data image file names

Images acquired from the left and the right cameras may be stored in the same directory, but obviously with different names as in

```
/path/to/experiment/
└── RAWDATA/
    ├── stack_0_channel_0
    │   ├── Cam_Left_00xxx.zip
    │   ├── ...
    │   ├── Cam_Right_00xxx.zip
    │   └── ...
    └── stack_1_channel_0
        ├── Cam_Left_00xxx.zip
        ├── ...
        └── Cam_Right_00xxx.zip
```

The parameter file has then to contain the following lines to indicate the directory names.

```
DIR_LEFTCAM_STACKZERO = 'stack_0_channel_0'
DIR_RIGHTCAM_STACKZERO = 'stack_0_channel_0'
DIR_LEFTCAM_STACKONE = 'stack_1_channel_0'
DIR_RIGHTCAM_STACKONE = 'stack_1_channel_0'
```

In addition, to distinguish the images acquired by the left camera to those acquired by the right one, one has to give the image name prefixes, i.e. the common part of the image file names before the 3-digit number that indicates the time point. This is the purpose of the variables `acquisition_leftcam_image_prefix` and `acquisition_rightcam_image_prefix`. The parameter file has then to contain the following lines not only to indicate the directory names but also the image file name prefixes.

```
DIR_LEFTCAM_STACKZERO = 'stack_0_channel_0'
DIR_RIGHTCAM_STACKZERO = 'stack_0_channel_0'
DIR_LEFTCAM_STACKONE = 'stack_1_channel_0'
DIR_RIGHTCAM_STACKONE = 'stack_1_channel_0'
acquisition_leftcam_image_prefix = 'Cam_Left_00'
acquisition_rightcam_image_prefix = 'Cam_Right_00'
```

3.3.3.3 Multichannel acquisition

In case of multichannel acquisition, the fusion is computed for the first channel, and the computed parameters (e.g. transformations, etc.) are also used for the other channels.

For a second channel, the images acquired at the first angulation by the left and the right cameras are searched in the directories

```
<PATH_EMBRYO>/<DIR_RAWDATA>/<DIR_LEFTCAM_STACKZERO_CHANNEL_2>
<PATH_EMBRYO>/<DIR_RAWDATA>/<DIR_RIGHTCAM_STACKZERO_CHANNEL_2>
```

while the images acquired at the second angulation by the left and the right cameras are searched in the directories

```
<PATH_EMBRYO>/<DIR_RAWDATA>/<DIR_LEFTCAM_STACKONE_CHANNEL_2>
<PATH_EMBRYO>/<DIR_RAWDATA>/<DIR_RIGHTCAM_STACKONE_CHANNEL_2>
```

For a third channel, the images acquired at the first angulation by the left and the right cameras are searched in the directories

```
<PATH_EMBRYO>/<DIR_RAWDATA>/<DIR_LEFTCAM_STACKZERO_CHANNEL_3>
<PATH_EMBRYO>/<DIR_RAWDATA>/<DIR_RIGHTCAM_STACKZERO_CHANNEL_3>
```

while the images acquired at the second angulation by the left and the right cameras are searched in the directories

```
<PATH_EMBRYO>/<DIR_RAWDATA>/<DIR_LEFTCAM_STACKONE_CHANNEL_3>
<PATH_EMBRYO>/<DIR_RAWDATA>/<DIR_RIGHTCAM_STACKONE_CHANNEL_3>
```

3.3.4 Output data

The variable `target_resolution` allows to set the desired isotropic (the same along the 3 dimensions) voxel size for the result fusion images.

3.3.4.1 Output data directory names

The resulting fused images are stored in sub-directory `FUSE/FUSE_<EXP_FUSE>` under the `/path/to/experiment/` directory

```
/path/to/experiment/
├── RAWDATA/
│   └── ...
└── FUSE/
    └── FUSE_<EXP_FUSE>/
        └── ...
```

where `<EXP_FUSE>` is the value of the variable `EXP_FUSE` (its default value is '`RELEASE`'). Hence, the line

```
EXP_FUSE = 'TEST'
```

in the parameter file will create the directory `FUSE/FUSE_TEST/` in which the fused images are stored. For instance, when testing for the values of the variable couple (`raw_ori`, `raw_mirrors`), a first parameter file may contain

```
raw_ori = 'left'
raw_mirrors = False
begin = 1
end = 1
EXP_FUSE = 'TEST-LEFT-FALSE'
```

a second parameter file may contain

```
raw_ori = 'left'
raw_mirrors = True
begin = 1
end = 1
EXP_FUSE = 'TEST-LEFT-TRUE'
```

etc. The resulting fused images will then be in different directories

```
/path/to/experiment/
├── RAWDATA/
│   └── ...
└── FUSE/
    ├── FUSE_TEST-LEFT-FALSE/
    │   └── ...
    ├── FUSE_TEST-LEFT-TRUE/
    │   └── ...
    └── ...
```

This will ease their visual inspection to decide which values of the variable couple (`raw_ori`, `raw_mirrors`) to use for the fusion.

3.3.4.2 Output data file names

Fused image files are named after the variable `EN`: `<EN>_fuse_t<xxx>.inr` where `<xxx>` is the time point index encoded by a 3-digit integer (with 0 padding).

3.3.4.3 Multichannel acquisition

If a single name is given in the variable `EXP_FUSE`, this name will be used to build the directory name for the resulting fused images of the first channel, and the other directory names are built after this first name by adding a suffix `_CHANNEL_2` for the 2nd channel, `_CHANNEL_3` for the 3rd channel, etc.

If the parameter file contains

```
EXP_FUSE = 'MULTI'
```

The resulting fused images will then be the following directories

```
/path/to/experiment/
├── RAWDATA/
│   └── ...
└── FUSE/
    ├── FUSE_MULTI/
    │   └── ...
    ├── FUSE_MULTI_CHANNEL_2/
    │   └── ...
    └── ...
```

Alternatively, a list of names can be specified in the variable `EXP_FUSE`, these names will be used to build the directory names for the resulting fused images of the corresponding channels (the first name of the list for the first channel, etc.).

If the parameter file contains

```
EXP_FUSE = ['1CHANNEL', '2CHANNEL']
```

The resulting fused images will then be the following directories

```
/path/to/experiment/
├── RAWDATA/
│   └── ...
└── FUSE/
    ├── FUSE_1CHANNEL/
    │   └── ...
    ├── FUSE_2CHANNEL/
    │   └── ...
    └── ...
```

3.3.5 Step 3 parameters: raw data cropping

For computational cost purposes, raw data (images acquired by the MuViSPIM microscope) are cropped (only in X and Y dimensions) before co-registration. A threshold is computed with Otsu's method [Ots79] on the maximum intensity projection (MIP) image. The cropping parameters are computed to keep the above-threshold points in the MIP image, plus some extra margins. Hyper-intense areas may biased the threshold computation, hence the cropping.

To desactivate this cropping, the line

```
raw_crop = False
```

has to be added in the parameter file.

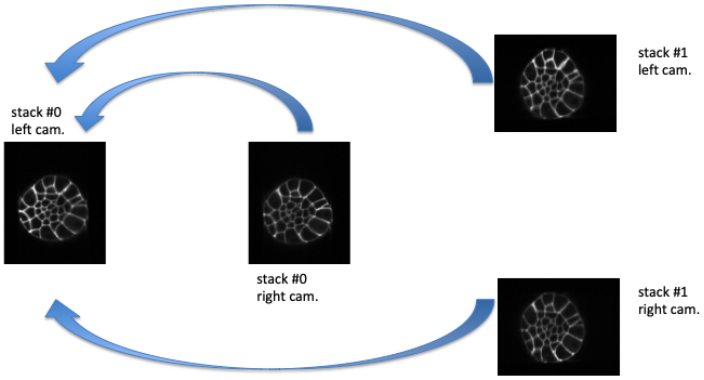

Figure 3.3: Fusion *direct* strategy: each 3D image is co-registered on the reference one, chosen here as the left camera image of stack #0.

3.3.6 Step 5 parameters: image co-registration

To fuse the images, they are co-registered onto a reference one. Co-registration are conducted only on the first channel (in case of multiple channel acquisitions), and the computed transformations are also applied onto the other channels. The reference image is chosen as being the acquisition from the left camera for the first stack (also denoted stack #0). The co-registration strategy is given by the variable `fusion_strategy` in the parameter file.

3.3.6.1 Fusion *direct* strategy

In the parameter file, the line

```
fusion_strategy = 'direct-fusion'
```

will set the co-registration strategy to the one described in [Gui15, GFL+20]: each acquisition image is linearly co-registered with the reference one, i.e. the one from the left camera and for the first stack.

Let us denote by I^0_{LC} the left camera image of stack#0, the three other images are I^0_{RC}, I^1_{LC}, and I^1_{RC}. By (linear) co-registration (see section 3.3.6.3) of these image with I^0_{LC}, the 3 transformations $T_{I^0_{RC}\leftarrow I^0_{LC}}$, $T_{I^1_{LC}\leftarrow I^0_{LC}}$, and $T_{I^1_{RC}\leftarrow I^0_{LC}}$ are computed. $T_{I^0_{RC}\leftarrow I^0_{LC}}$ is the transformation that allows to resample I^0_{RC} in the same frame than I^0_{LC}: this transformation goes from the frame of I^0_{LC} towards the frame of I^0_{RC} (hence the direction of the arrow). $I^0_{RC} \circ T_{I^0_{RC}\leftarrow I^0_{LC}}$ denotes this resampled image.

3.3.6.2 Fusion *hierarchical* strategy

In the parameter file, the line

```
fusion_strategy = 'hierarchical-fusion'
```

defines a hierarchical co-registration strategy. First, the right camera image of each stack is linearly co-registered (see section 3.3.6.3) on its left camera counterpart, yielding the transformations $T_{I^0_{RC}\leftarrow I^0_{LC}}$ and $T_{I^1_{RC}\leftarrow I^1_{LC}}$. According that the left and right camera images of a stack are acquired simultaneously, a linear transformation is then completely adequate to co-register them.

This allows to fuse (see section 3.3.7) the two acquisition of the corresponding left and right cameras into a single stack:

$$\begin{aligned} I^0 &= \omega^0_{LC} I^0_{LC} + \omega^0_{RC} I^0_{RC} \circ T_{I^0_{RC}\leftarrow I^0_{LC}} \quad \text{and} \\ I^1 &= \omega^1_{LC} I^1_{LC} + \omega^1_{RC} I^1_{RC} \circ T_{I^1_{RC}\leftarrow I^1_{LC}} \end{aligned}$$

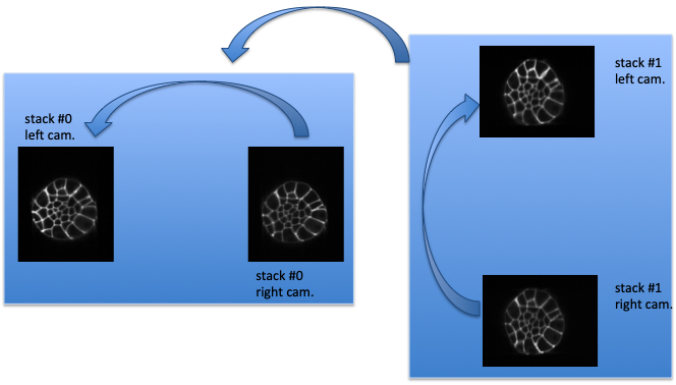

Figure 3.4: Fusion *hierarchical* strategy. Stacks #0 and #1 are reconstructed independently: right camera images are co-registered on the left camera ones, and stacks #0 and #1 are reconstructed by fusing left and right camera images. Fused image of stack #1 is co-registered on fused image of stack #0: by transformation composition, it allows to compute the transformations of left and right camera images of stack #1 onto the left camera image of stack #0.

The reconstructed stacks are then (potentially non-linearly, see section 3.3.6.4) co-registered together, yielding the transformation $T_{I^1 \leftarrow I^0}$. This allows to get the $T_{I^1_{RC} \leftarrow I^0_{RC}}$ and $T_{I^1_{LC} \leftarrow I^0_{RC}}$ transformations

$$\begin{aligned} T_{I^1_{LC} \leftarrow I^0_{LC}} &= T_{I^1 \leftarrow I^0} \quad \text{and} \\ T_{I^1_{RC} \leftarrow I^0_{LC}} &= T_{I^1_{RC} \leftarrow I^1_{LC}} \circ T_{I^1 \leftarrow I^0} \end{aligned}$$

Using a non-linear registration in this last step allows to compensate for some distortions that may occur between the two stacks #0 and #1. Please note that stack #0 is then assumed to be the non-distorted reference while left and right camera image of stack #1 will be deformed before fusion.

3.3.6.3 Acquisitions linear co-registration

The linear co-registrations are either used to co-registered each acquisition onto the reference one in the `'direct-fusion'` strategy, or to build stacks from the left and right cameras in the `'hierarchical-fusion'` strategy. Variables that controls the linear co-registrations are either prefixed by `fusion_preregistration_` or by `fusion_registration_`.

To verify whether a good quality registration can be conducted, the searched transformation type can be changed for a simpler one than affine. Adding the following line in the parameter file.

```
fusion_registration_transformation_type = translation
```

will search for a translation which could be supposed to be sufficient, according that only translations relates the 4 acquisitions of the MuViSPIM microscope (in a perfect setting). If the search for an affine transformation (the default behavior) failed (the fusion looks poor) while the search for a translation is successful (the fusion looks good), a two-steps registration may help to refine the found translation by a subsequent affine transformation as explained below.

Hyper-intensities areas may bias the threshold calculation used for the automatic crop (step 3 of fusion). In such cases, the iterative registration method may find a local minimum that is not the desired one, because the relative positions of the two images to be co-registered are too far apart. To circumvent such a behavior, a two-steps registration can be done. It consists on a first pre-registration with a transformation with fewer degrees of freedom (i.e. a 3D translation).

This pre-registration can be activated by adding the following line in the parameter file.

```
fusion_preregistration_compute_registration = True
```

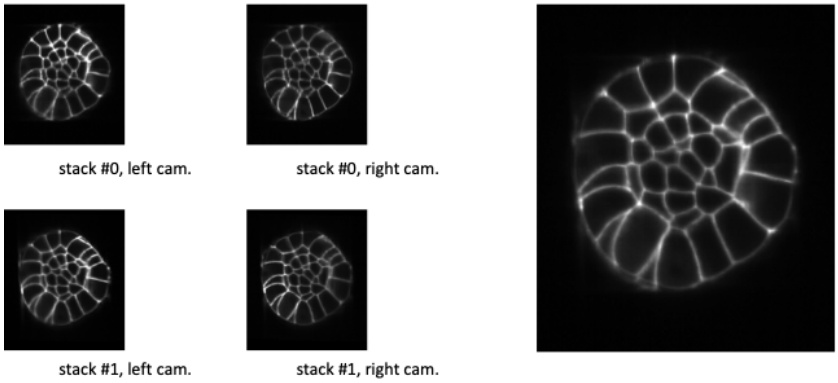

Figure 3.5: At the left, XZ-sections of 4 co-registered stacks. At the right, the linear combination of the 4 co-registered stacks with an uniform (or constant) weighting function. It comes to make an average of the 4 co-registered stacks.

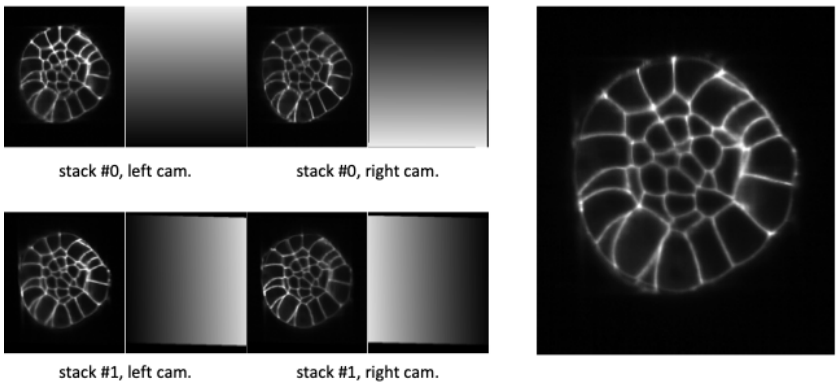

Figure 3.6: At the left, XZ-sections of 4 co-registered stacks together with their ramp weighting function. At the right, the linear combination of the 4 co-registered stacks with this ramp weighting function.

3.3.6.4 Stacks non-linear co-registration

Variables that controls the non-linear co-registrations are either prefixed by `fusion_stack_preregistration_` or by `fusion_stack_registration_`. They are defined similarly as the one of acquisitions co-registration.

3.3.7 Step 6: linear combination of co-registered image stacks

The resampled co-registered image stacks are fused together by the means of a weighted linear combination.

$$I_{fuse} = \omega^0_{LC} I^0_{LC} + \omega^0_{RC} I^0_{RC} \circ T_{I^0_{RC} \leftarrow I^0_{LC}} + \omega^1_{LC} I^1_{LC} \circ T_{I^1_{LC} \leftarrow I^0_{LC}} + \omega^1_{RC} I^1_{RC} \circ T_{I^1_{RC} \leftarrow I^0_{LC}}$$

The choice of the weighting function is controlled by the variable `fusion_weighting`, eventually suffixed by `_channel_[1,2,3]` if one wants to use different weighting schemes for the different channels to be fused.

The variable `fusion_weighting` can be set to

- `'uniform'`: it comes to the average of the resampled co-registered stacks (see figure 3.5). Such a weighting does not depend on the stacks to be fused.
- `'ramp'`: the weights are linearly increasing along the **Z** axis (see figure 3.6).

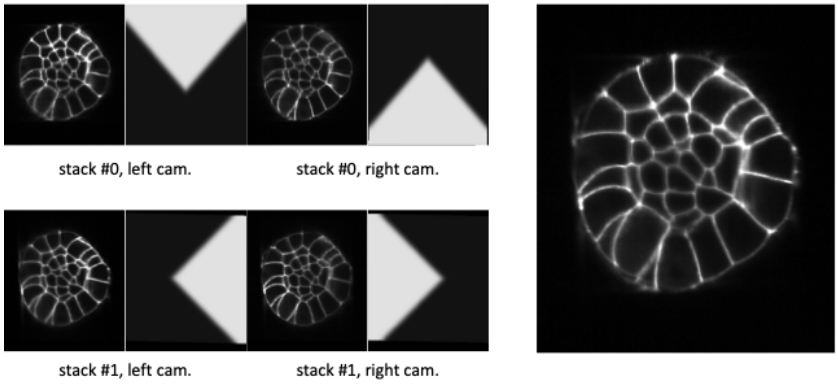

Figure 3.7: At the left, XZ-sections of 4 co-registered stacks together with their corner weighting function. At the right, the linear combination of the 4 co-registered stacks with this corner weighting function.

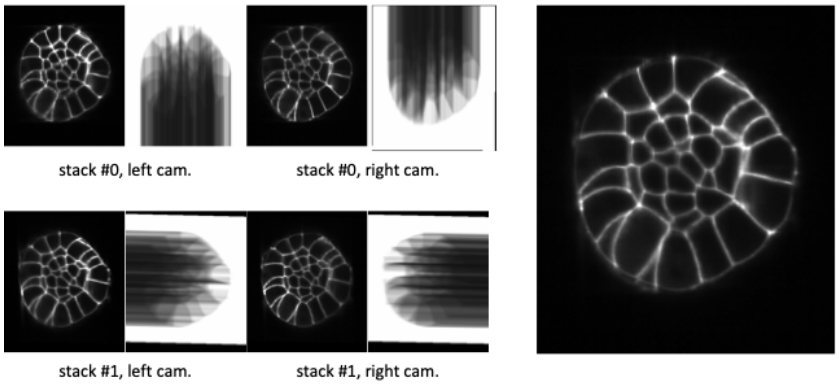

Figure 3.8: At the left, XZ-sections of 4 co-registered stacks together with their Guignard's weighting function. At the right, the linear combination of the 4 co-registered stacks with this weighting function.

- `'corner'`: the weights are constant in a corner portion of the stack, defined by two diagonals in the **XZ**-section (see figure 3.7). It somehow mimics a stitching of the 4 resampled co-registered image stacks, where the information is kept from the most informative image.
- `'guignard'`: the weighting function is the one described in [Gui15]. More weight are given to sections close to the camera and it also takes into account the traversed material (see figure 3.8).

Weighting functions are designed so that the weights decrease with **Z** for the left camera images and increase with **Z** for the left camera images. So, setting the `acquisition_leftcamera_z_stacking` variable to the wrong value (`'direct'` instead of `'inverse'`, or vice-versa) may then decrease the fusion quality.

Looking at XZ-sections of the co-registered image stacks, as well as the weighting function images, (see figures 3.5, 3.6, 3.7, and 3.8) provides a direct and efficient means to check whether this parameter is correctly set. Such sections can be extracted by setting the `fusion_xzsection_extraction` parameter to `True`. It creates `XZSECTION_<xxx>/` subdirectories (one par time point, `<xxx>` being the time point index) in the `FUSE/FUSE_<EXP_FUSE>/` directory.

```
/path/to/experiment/
├── RAWDATA/
│   └── ...
└── FUSE/
    └── FUSE_<EXP_FUSE>/
        ├── ...
        ├── XZSECTION_<xxx>/
        └── ...
```

When using the variable `fusion_weighting`, the same weights (computed on the first channel to be processed) are used for all fusion. However, different weighting functions can be used for the channels to be fused by using the variables `fusion_weighting_channel_[1,2,3]`, eg

```
fusion_weighting_channel_1 = 'guignard'
fusion_weighting_channel_2 = 'uniform'
```

3.3.8 Step 7: fused data cropping

To save disk storage, fused images are cropped at the end of the fusion stage. To desactivate this cropping, the line

```
fusion_crop = False
```

has to be added in the parameter file.

3.3.9 Troubleshooting

- The fused images are obviously wrong.
  1. Are the values of the variable couple (`raw_ori`, `raw_mirrors`) the right ones? Conduct experiments as suggested in section 3.3.2 (see also section 3.3.4) to get the right values.
  2. The registration may have failed.
     (a) Try to register with a simpler transformation type (i.e. translation) and/or with a two-steps registration (see section 3.3.6).
- The imaged sample is cropped by the image border in the fused image.
  1. Check whether the imaged sample was not already cropped in the raw data.

2. The automated cropping may have failed. It is more likely to happen when cropping the raw data, so deactivate it (see section 3.3.5). If it still happens, try to deactivate also the fused image cropping (see section 3.3.8).

3.4 1.5-intraregistration.py

3.4.1 Intra-registration procedure overview

The sequence intra-registration procedure can be done either after the fusion step, or after the (post-)segmentation step. It aims at

- compensating for the eventual motion of the imaged sample with respect to the microscope
- resampling the fusion and/or the segmentation images into a common frame/geometry, so they can better be compared, and
- building 2D+t images made of 2D sections from either the fusion and/or the segmentation images, so that the quality of the fusion and/of the tracking step can be visually assessed.

The intra-registration procedure is made of the following steps:

1. Co-registration of pairs of successive fused images (section 3.4.5). This yields the transformations $T_{t+1\leftarrow t}$. Fused images are located in `<EMBRYO>/FUSE/FUSE_<EXP_FUSE>`: the parameter `EXP_FUSE` is either set in the parameter file or is set at `RELEASE`. This step may be long.
2. Composition of transformations issued from the co-registration step. This step computes the transformations $T_{ref\leftarrow t}$. towards a reference image `ref` given by the parameter `intra_registration_reference_index`.
3. Computation of the *template* image (section 3.4.6). This *template* image dimension are computed so that the useful information of all resampled images fits into it. Useful information can be issued from either the fused sequence, the segmentation sequence or the post-segmentation sequence. It is indicated by the `intra_registration_template_type` which value can be either `'FUSION'`, `'SEGMENTATION'`, or `'POST-SEGMENTATION'`. This step may be long.
4. Resampling of either the fused or the segmentation images (section 3.4.7). Note that changing the parameters for this step will not require to re-compute the first steps.
5. Extraction of 2D+t images from the resampled sequences (section 3.4.8). Note that changing the parameters for this step (i.e. requiring extra movies) will not require to re-compute the first steps, with an eventual exception for the resampling step.
6. Computation of a maximum image from the resampled images (section 3.4.9). Computing the maximum over the resampled fusion images may be useful to define a common cropping area for the sequence. Note that changing the parameters for this step will not require to re-compute the first steps.

3.4.2 1.5-intraregistration.py additional options

The following options are available:

`-t file`
: set the resampling transformation file for the reference image (see section 3.4.6)

`-a string`
: set the resampling transformation angles for the reference image (see section 3.4.6)

3.4.3 Input data

3.4.3.1 Multichannel acquisition

The co-registration transformations are computed from one series of fused images, issued from the `/path/to/experiment/FUSE` directory.

In case of multi-channel acquisition, all fused image directories listed in the EXP_FUSE variable will be transformed by the transformations computed on the *first* fused image directory of the list.

As detailed in section 3.4.5 Specifying

```
EXP_FUSE = ['MEMBRANES', 'NUCLEI']
```

in the parameter file implies that co-registrations will be computed on the fused images from FUSE/FUSE_MEMBRANES/, but both fused image series will be transformed.

The same stands for segmentation and post-segmentation series: multiple directories can be specified in either EXP_SEG or EXP_POST.

3.4.4 Output data

The results are stored in sub-directories INTRAREG/INTRAREG_<EXP_INTRAREG> under the /path/to/experiment/ directory where <EXP_INTRAREG> is the value of the variable EXP_INTRAREG (its default value is 'RELEASE').

```
/path/to/experiment/
├── ...
├── INTRAREG/
│   └── INTRAREG_<EXP_INTRAREG>/
│       ├── CO-TRSFS/
│       ├── [FUSE/]
│       ├── LOGS/
│       ├── [MAXIMUM/]
│       ├── [MOVIES/]
│       ├── [POST/]
│       ├── [SEG/]
│       └── TRSFS_t<begin>-<end>/
└── ...
```

Output data are of two kinds: image series (fused images, segmentation images, post-corrected segmentation images) can be resampled in the same common geometry (also known as the *template*), see section 3.4.7, and 3D (ie 2D+t) images of the evolution (with respect to time) of one section (XY, XZ, or YZ) of the images of the series can be built, see section 3.4.8.

3.4.5 Step 1: co-registration

Default registration parameters for the co-registration are set by:

```
# intra_registration_compute_registration = True
# intra_registration_transformation_type = 'rigid'
# intra_registration_transformation_estimation_type = 'wlts'
# intra_registration_lts_fraction = 0.55
# intra_registration_pyramid_highest_level = 6
# intra_registration_pyramid_lowest_level = 3
# intra_registration_normalization = True
```

Computed transformations are stored in INTRAREG/INTRAREG_<EXP_INTRAREG>/CO-TRSFS. It may be advised to set the pyramid lowest level value to some higher value to speed up the co-registrations (recall that all pairs of successive images will be co-registered, i.e.

```
intra_registration_pyramid_lowest_level = 4
```

Co-registration are computed using the fused images of /path/to/experiment/FUSE/FUSE_<EXP_FUSE>. If EXP_FUSE is a list of strings (ie indicates a list a directories) rather than a single string, the fused image from the first directory are used for the co-registration computation.

Typically, if there are several fused series (eg, in case of multi-channel acquisition) as in

```
/path/to/experiment/
├── ...
├── FUSE/
│   ├── FUSE_MEMBRANES/
│   └── FUSE_NUCLEI/
└── ...
```

Specifying

```
EXP_FUSE = ['MEMBRANES', 'NUCLEI']
```

in the parameter file implies that co-registrations will be done on the fused images from `FUSE/FUSE_MEMBRANES/`.

3.4.6 Step 3: template building

```
# intra_registration_reference_index = None
# intra_registration_reference_resampling_transformation_file = None
# intra_registration_reference_resampling_transformation_angles = None
#
# intra_registration_template_type = "FUSION"
# intra_registration_template_threshold = None
# intra_registration_margin = None
#
# intra_registration_resolution = 0.6
#
# intra_registration_rebuild_template = False
```

- The `intra_registration_reference_index` allows to choose the reference image (the one which remains still, i.e. up to a translation), by default it is the first image image of the series (associated to `begin`). However, it may happen that this image has to be reoriented to fit the user's expectation. The resampling transformation[1], that re-orient the reference image, can then be given and will be applied to the whole series.
  - `intra_registration_reference_resampling_transformation_file` can be given a resampling transformation file name.
  - `intra_registration_reference_resampling_transformation_angles` can be given a string describing the successive rotations (with respect to the frame axis) to be applied. E.g. the string `"X 30 Y 50"` defines a resampling transformation equal to $R_X(30) \circ R_Y(50)$ where $R_X(30)$ is a rotation of 30 degrees around the X axis and $R_Y(50)$ is a rotation of 50 degrees around the Y axis.
- Depending on `intra_registration_template_type` (`'FUSION'`, `'SEGMENTATION'` or `'POST-SEGMENTATION'`), the two latter assume obviously that the segmentation has been done), the *template* image can be built either after the fusion or the segmentation images. If no threshold is given by `intra_registration_template_threshol` the built template will be large enough to include all the transformed fields of view (in this case, the template is the same whatever `intra_registration_template_type` is).

  If `intra_registration_template_type='FUSION'` (respectively `'SEGMENTATION'` and `'POST-SEGMENTATION'`), the template is built from the images of the first directory indicated by `EXP_FUSE` (respectively `EXP_SEG` and `EXP_POST`) in case of `EXP_FUSE` contains a list of strings.

[1] The resampling transformation is the one that goes from the destination image towards the input image.

If a threshold is given, the built template will be large enough to include all the transformed points above the threshold. E.g., the background is labeled with either '1' or '0' in segmentation images, then a threshold of '2' ensures that all the embryo cells will not be cut by the resampling stage. In this case, adding an additional margin (with `intra_registration_margin`) to the template could be a good idea for visualization purpose.

- Specifying using a different resolution for the drift-compensated series than the `target_resolution` (the resolution of the fused images) allows to decrease the resampled images volume. This can be achieved by setting `intra_registration_resolution` to the desired value (default is 0.6).
- Last, co-registrations may have been computed during a first computation, fused images being used to compute the template. However, if a subsequent segmentation has been conducted, a smaller template is likely to be computed (with the segmentation images to build the template), without recomputing the co-registration. This is the purpose of the variable `intra_registration_rebuild_template`. If set to `True`, it forces to recompute the template as well as the transformations from the co-registrations (that are not re-computed). Obviously, resampling as well as 2D+t movies are also re-generated.

As an example, building a *template* image after the segmentation images can be done with

```
# intra_registration_reference_index = None
intra_registration_template_type = "SEGMENTATION"
intra_registration_template_threshold = 2
# intra_registration_resolution = 0.6
intra_registration_margin = 10
```

Computed transformations from the *template* image as well as the *template* image itself are stored in `INTRAREG/INTRAREG<EXP_INTRAREG>/TRSFS_t<F>-<L>/` where `<F>` and `L` are the first and the last index of the series (specified by `begin` and `end` from the parameter file).

3.4.7 Step 4: resampling fusion/segmentation images

The resampling of the fused and/or segmentation images are done depending on the value of the following variables (here commented). Resampling is done either if the following parameters are set to `True` or if movies are requested to be computed (section 3.4.8).

```
# intra_registration_resample_fusion_images = True
# intra_registration_resample_segmentation_images = False
# intra_registration_resample_post_segmentation_images = False
```

This default behavior implies that the fusion images will be resampled while the segmentation and the post-corrected segmentation images are not.

Resampled images will be stored in the `INTRAREG/INTRAREG_<EXP_INTRAREG/>` directory, with the same hierarchy than under `/path/to/experiment`. E.g.

```
/path/to/experiment/
├── ...
├── FUSE/
│   ├── FUSE_1/
│   └── FUSE_2/
└── ...
```

Specifying

```
EXP_FUSE = ['1', '2']
```

in the parameter file causes the resampling of both fused image series (`FUSE/FUSE_1/` and `FUSE/FUSE_2/`)

```
/path/to/experiment/
├── ...
├── FUSE/
│   ├── FUSE_1/
│   └── FUSE_2/
├── INTRAREG/
│   └── INTRAREG_<EXP_INTRAREG>/
│       ├── FUSE/
│       │   ├── FUSE_1/
│       │   └── FUSE_2/
│       └── ...
└── ...
```

The same behavior stands for `EXP_SEG` and `EXP_POST`.

3.4.8 Step 5: 2D+t movies

For either visual assessment or illustration purposes, 2D+t (i.e. 3D) images can be built from 2D sections extracted from the resampled temporal series. This is controlled by the following parameters:

```
# intra_registration_movie_fusion_images = True
# intra_registration_movie_segmentation_images = False
# intra_registration_movie_post_segmentation_images = False

# intra_registration_xy_movie_fusion_images = [];
# intra_registration_xz_movie_fusion_images = [];
# intra_registration_yz_movie_fusion_images = [];

# intra_registration_xy_movie_segmentation_images = [];
# intra_registration_xz_movie_segmentation_images = [];
# intra_registration_yz_movie_segmentation_images = [];

# intra_registration_xy_movie_post_segmentation_images = [];
# intra_registration_xz_movie_post_segmentation_images = [];
# intra_registration_yz_movie_post_segmentation_images = [];
```

If `intra_registration_movie_fusion_images` is set to `True`, a movie is made with the XY-section located at the middle of each resampled fusion image (recall that, after resampling, all images have the same geometry). Additional XY-movies can be done by specifying the wanted Z values in `intra_registration_xy_movie_fusion_im` E.g.

```
intra_registration_xy_movie_fusion_images = [100, 200];
```

will build two movies with XY-sections located respectively at Z values of 100 and 200. The same stands for the other orientation and for the resampled segmentation images.

Movies will be stored in the `INTRAREG/INTRAREG_<EXP_INTRAREG>/MOVIES/` directory, with the same hierarchy than under `/path/to/experiment`. E.g.,

```
EXP_FUSE = ['1', '2']
```

in the parameter file results in

```
/path/to/experiment/
```

```
├── ...
├── FUSE/
│   ├── FUSE_1/
│   └── FUSE_2/
├── INTRAREG/
│   └── INTRAREG_<EXP_INTRAREG>/
│       ├── FUSE/
│       │   ├── FUSE_1/
│       │   └── FUSE_2/
│       ├── MOVIES/
│       │   └── FUSE/
│       │       ├── FUSE_1/
│       │       └── FUSE_2/
│       └── ...
└── ...
```

The same behavior stands for `EXP_SEG` and `EXP_POST`.

3.4.9 Step 6: 3D maximum over the 3D+t sequence

To set a cropping area valid for the whole resampled sequence, a maximum image can be built from the resampled temporal series. This is controlled by the following parameters:

```
# intra_registration_maximum_fusion_images = False
# intra_registration_maximum_segmentation_images = False
# intra_registration_maximum_post_segmentation_images = False
```

If `intra_registration_maximum_fusion_images` is set to `True`, a maximum image is computed over the sequence of resampled fusion images (recall that, after resampling, all images have the same geometry). The value of a voxel in this maximum image is the maximum value (over time) of this voxel in the sequence.

The maximum image will be stored in the `INTRAREG/INTRAREG_<EXP_INTRAREG>/MAXIMUM/` directory, with the same hierarchy than under `/path/to/experiment`. E.g.,

```
EXP_FUSE = ['1', '2']
```

in the parameter file results in

```
/path/to/experiment/
├── ...
├── FUSE/
│   ├── FUSE_1/
│   └── FUSE_2/
├── INTRAREG/
│   └── INTRAREG_<EXP_INTRAREG>/
│       ├── FUSE/
│       │   ├── FUSE_1/
│       │   └── FUSE_2/
│       ├── MAXIMUM/
│       │   └── FUSE/
│       │       ├── FUSE_1/
│       │       └── FUSE_2/
│       └── MOVIES/
│           └── FUSE/
```

```
        ├── FUSE_1/
        ├── FUSE_2/
    ├── ...
├── ...
```

The same behavior stands for `EXP_SEG` and `EXP_POST`.

3.5 2-mars.py

3.5.1 Mars method overview

The name `mars` comes from [FDM+10] where `MARS` is the acronym of *multiangle image acquisition, 3D reconstruction and cell segmentation*.

This method aims at producing a segmentation of a membrane cell image (e.g. a fused image) into a segmentation image. This segmentation image is a integer-valued image where each integer labeled an unique cell in the image. By convention, '1' is the background label, while cells have labels greater than 2. It is made of the following steps:

1. Pre-processing of the input image to produce the input seed image for seed computation. This is described in section 3.10. The parameters that governed the pre-processing are described in section 3.11.6 and prefixed by `seed_`.
2. Seed extraction through the computation of the h-minima of the input seed image
3. Eventually seed correction
4. Pre-processing of the input image to produce the input membrane image for the seeded watershed. This is described in section 3.10. The parameters that governed the pre-processing are described in section 3.11.6 and prefixed by `membrane_`.
5. A seeded watershed.

3.5.2 Output data

The results are stored in sub-directories `SEG/SEG_<EXP_SEG>` under the `/path/to/experiment/` directory where where `<EXP_SEG>` is the value of the variable `EXP_SEG` (its default value is '`RELEASE`').

```
/path/to/experiment/
├── ...
├── SEG/
│   └── SEG_<EXP_SEG>/
│       ├── <EN>_mars_t<begin>.inr
│       ├── LOGS/
│       └── RECONSTRUCTION/
└── ...
```

3.5.3 Steps 1 and 4: input image pre-processing

The input image (typically the fused image representing the cell membranes/walls) can be pre-processed before use in the seeded watershed. The pre-processing can be different for the seed input image (the one that will be used to extract the seeds) and the membrane input image (the one that will be used as the height image for the seeded watershed). Details about the pre-processing can be found in section 3.10.

Default settings are

```
intensity_transformation = 'Identity'
intensity_enhancement = None
```

meaning that the original fused image is used for both inputs. Different pre-processing can be done. E.g.

```
seed_intensity_transformation = 'Identity'
membrane_intensity_transformation = 'normalization_to_u8'
intensity_enhancement = None
```

comes to use the original image for the seed extraction, but its normalization into 8 bits as the height image for the seeded watershed.

If the input image is transformed before segmented, the transformed images can be saved in the directory `SEG/SEG_<EXP_SEG>/RECONSTRUCTION/` if the value of the variable `keep_reconstruction` is set to `True`.

3.5.4 Step 2: seed extraction

The seed extraction is made of the following steps:

1. Gaussian smoothing of the input image, the gaussian standard deviation being given by the variable `seed_sigma`.
2. Extraction of the h-minima of the previous image, h being given by the variable `seed_hmin`.
3. Hysteresis thresholding (and labeling) of the h-minima image, with a high threshold equal to `seed_high_threshold` (default is h) and and a low threshold equal to 1. It then only selects the h-minima that have an actual depth of h.

3.5.5 Step 3: seed correction

Several rounds of correction of the computed seeds can be done. At each round, different seeds can be assigned the same label (and this will fuse the further reconstructed cells) or new seeds (each new seed is a single voxel) can be added. See the `'seed_edition_files'` variable for details.

When correcting seeds, it is advised to launch `2-mars.py` with the `'-k'` option. Indeed, temporary files, as the seed image, are kept in a temporary directory located in the `SEG/SEG_'EXP_SEG'/` directory and then re-used, and not recomputed at each `2-mars.py` use.

3.5.6 Step 5: seeded watershed

Given the seeds, the watershed is performed on the smoothed input membrane image (gaussian standard deviation being given by the variable `membrane_sigma`).

3.6 `3-manualcorrection.py`

3.6.1 Manual correction overview

The seeded watershed is likely to produce segmentation errors, even with a careful choice of parameters. It is advised to set the parameters to favour over-segmentations insted of under-segmentations since the former are much more easier to correct, which is the purpose of `3-manualcorrection.py`. Note that the segmentation error correction could also be done at the seed correction step of the `2-mars.py` stage, see section 3.5.5.

3.6.2 `3-manualcorrection.py` additional options

The following options are available:

`-i input_image` set the `input_image` file to be corrected. Allows to skip the automated naming of files.

`-o output_image` set the resulting `ouput_image` file to be saved. Allows to skip the automated naming of files.

`-m mapping_file` set the `mapping_file` to be used for the correction.

`-nsc smallest_cells` set the number of the smallest cells to be displayed after correction. The smallest cells are the most likely to be issued from an over-segmentation.

`-nlc largest_cells` set the number of the largest cells to be displayed after correction. The largest cells are the most likely to be issued from an under-segmentation.

3.6.3 Output data

The results are stored in sub-directories `SEG/SEG_<EXP_SEG>` under the `/path/to/experiment/` directory where `<EXP_SEG>` is the value of the variable `EXP_SEG` (its default value is 'RELEASE'). `<EN>_seg_t<begin>.inr` is the correction of the segmentation image `<EN>_mars_t<begin>.inr`.

```
/path/to/experiment/
├── ...
├── SEG/
│   └── SEG_<EXP_SEG>/
│       ├── <EN>_mars_t<begin>.inr
│       ├── <EN>_seg_t<begin>.inr
│       ├── LOGS/
│       └── RECONSTRUCTION/
└── ...
```

3.6.4 Segmentation correction parameters

`3-manualcorrection.py` parses a correction file whose name is given by the variable `mancor_mapping_file`. The syntax of this file is very simple. Lines beginning with `#` are ignored (and can be used to insert comments in the files). Non-empty lines should contain two numbers separated by a space, and `3-manualcorrection.py` will replace the first number by the second in the segmentation file.

E.g. a cell c is recognized to be over-segmented, and then is represented by two labels, says 9 and 10. Thus the line

```
10 9
```

will replace all 10's by 9's in the segmentation image, thus c will only be represented by 9's after correction. See also the tutorial section 2.5 for an other example.

3.7 `4-astec.py`

The name `astec` comes from the Phd work of L. Guignard [Gui15] where `ASTEC` is the acronym of *adaptive segmentation and tracking of embryonic cells*.

This method aims at producing a segmentation of each membrane cell image (e.g. a fused image) temporal sequence. This is a method of segmentation by propagation: it used the segmentation of the previous timepoint (say $t-1$) to constraint the segmentation at the aimed timepoint (say t).

3.7.1 Astec method overview

Astec principle is to guide the segmentation of the t timepoint image I_t with the segmentation $S^{\star}_{t-1}$ of the $t-1$ timepoint image I_{t-1}.

1. A first segmentation of I_t, $\tilde{S}_t$, is computed by a seeded watershed, where the seeds are the eroded cells of $S^{\star}_{t-1}$, projected onto I_t. By construction, no cell division can occur.

2. h-minima (see also section 3.5.4) are computed over a range of h values. Studying the numbers of h-minima located in each cell of $\tilde{S}_t$ gives an indication whether there might be a cell division or not. From this study a seed image $Seeds_t$ is computed, and then a new segmentation image $\hat{S}_t$.

3. Some potential errors are detected by checking whether there is a significant volume decrease from a cell of $S^\star_{t-1}$ and its corresponding cells in $\hat{S}_t$. For such cells, seeds may be recomputed, as well as the $\hat{S}_t$ segmentation.

4. It may occur, in step 3, that some cell from $S^\star_{t-1}$ correspond to 3 cells in $\hat{S}_t$. This step aims at correcting this.

5. Some other potential errors are detected by checking whether there is a significant volume decrease from a cell of $S^\star_t$ and its corresponding cells in $\hat{S}_t$ due to a background invasion. For such cells, morphosnakes [MNBA14] are computed to try to recover cell loss.

6. The morphosnake correction operated in step 5 may invade the background too much. This step aims at correcting this.

3.7.2 Output data

The results are stored in sub-directories `SEG/SEG_<EXP_SEG>` under the `/path/to/experiment/` directory where where `<EXP_SEG>` is the value of the variable `EXP_SEG` (its default value is '`RELEASE`').

```
/path/to/experiment/
├── ...
├── SEG/
│   └── SEG_<EXP_SEG>/
│       ├── <EN>_seg_lineage.xml
│       ├── <EN>_seg_t<begin>.mha
│       ├── <EN>_seg_t<...>.inr
│       ├── <EN>_seg_t<end>.mha
│       ├── LOGS/
│       └── RECONSTRUCTION/
└── ...
```

3.7.3 Input image pre-processing

The input image (typically the fused image representing the cell membranes/walls) can be pre-processed before use in the astec stage (as for `2-mars.py`, see section 3.5.3)/ The pre-processing can be different for the

- the seed input image (the one that will be used to compute the h-minima),
- the membrane input image (the one that will be used as the height image for the seeded watersheds), and
- the morphosnake input image (the one that will be used to define the morphosnake energy).

Pre-processing parameters, described in section 3.11.6, and prefixed respectively by `seed_`, `membrane_` and `morphosnake_` allow to tune these pre-processing. Hence, the lines

```
seed_intensity_transformation = 'Identity'
membrane_intensity_transformation = 'normalization_to_u8'
morphosnake_intensity_transformation = 'Identity'
intensity_enhancement = None
```

come to choose the original image for both the seed extraction and the morphosnake stage, but its normalization on 8 bits for the seeded watershed (this corresponds to the choice of the historical version of astec).

3.7.4 Step 1: $\tilde{S}_t$

A first segmentation of I_t, $\tilde{S}_t$, is computed by a seeded watershed, where the seeds are built from the eroded cells of $S^\star_{t-1}$.

- `previous_seg_method = 'erode_then_deform'`
  The cells of $S^\star_{t-1}$ are first eroded, yielding the image S^e_{t-1}, then this image is mapped onto I_t frame thanks to the transformation $\mathcal{T}_{t-1\leftarrow t}$, resulting in the eroded seed image $S^e_{t-1\leftarrow t} = S^e_{t-1} \circ \mathcal{T}_{t-1\leftarrow t}$. This is the historical astec behavior.

- `previous_seg_method = 'deform_then_erode'`
  $S^\star_{t-1}$ is first mapped onto I_t frame thanks to the transformation $\mathcal{T}_{t-1\leftarrow t}$, resulting in the image $S^\star_{t-1\leftarrow t} = S^\star_{t-1} \circ \mathcal{T}_{t-1\leftarrow t}$. Cells of $S^\star_{t-1\leftarrow t}$ are then eroded to get $S^e_{t-1\leftarrow t}$

This seed image, $S^e_{t-1\leftarrow t}$, plus the membrane input image are used as input for a seeded watershed, and yield $\tilde{S}_t$. By construction, no cell division can occur in $\tilde{S}_t$ with respect to $S^\star_{t-1}$.

If the variable `propagation_strategy` is set to `'seeds_from_previous_segmentation'`, the segmentation propagation stops and $\tilde{S}_t$ is the final result.

3.7.5 Step 2: $\hat{S}_t$

The h-minima are computed in the seed input image for a range of $h \in [h_{min}, h_{max}]$, with a step of δh.

h_{min}, h_{max} and δh are set respectively by the variables `watershed_seed_hmin_min_value`, `watershed_seed_hmin_max_valu` and `watershed_seed_hmin_delta_value`.

For a given cell of $\tilde{S}_t$, if there is no cell division, and if the h-minima are well detected, ther should be only one h-minima included in the cell for all values of h. However, if a cell division occurs, there should be mostly teo h-minima included in the cell. Then, the study of the number of h-minima strictly included allows to decide whether a cell division has occur (see [Gui15, GFL+20] for details).

This step results in the image $\hat{S}_t$.

If the variable `propagation_strategy` is set to `'seeds_selection_without_correction'`, the segmentation propagation stops and $\hat{S}_t$ is the final result.

3.7.6 Steps 3 and 4: volume checking

Some potential errors are detected by checking whether there is a large volume decrease from a cell of $S^\star_{t-1}$ and its corresponding cells in $\hat{S}_t$. For such cells, seeds are recomputed, as well as the $\hat{S}_t$ segmentation.

It may occur, in step 3, that some cell from $S^\star_{t-1}$ correspond, after correction, to 3 cells in $\hat{S}_t$. A second step aims at correcting this.

3.7.7 Steps 5 and 6: morphosnake correction

This step is performed if `morphosnake_correction` is set to `True`.

Some other potential errors are detected by checking whether there is a significant volume decrease from a cell of $S^\star_t$ and its corresponding cells in $\hat{S}_t$ due to a background invasion. For such cells, morphosnakes [MNBA14] are computed to try to recover cell loss.

3.8 `5-postcorrection.py`

3.8.1 Post-correction overview

The Astec segmentation procedure yields a series of segmented images $\{S^\star_t\}_t$, where each segmented image $S^\star_{t+1}$ takes advantage of the knowledge of the previously segmented image $S^\star_t$ to decide, at the cell level,

whether a cell division may occur. However, there are still segmentation errors, that can be detected from the study of the overall lineage (see [Gui15, section 2.3.3.7, page 74] and [GFL+20, supp. mat.]).

As suggested by its name, the post-correction will try to a posteriori correct the segmentation resulting from the `4-astec.py` stage (see section 3.7).

The post-correction is made of the following steps.

1. Lineage pruning: it goes through the end branches (a branch does not have any cell division; an end branch finishes either at the end of the sequence or the cell vanishes between two time points) of the lineage tree. Some lineage end branches are deleted (see section 3.8.4 for details), meaning that the corresponding cells are fused with other cells of the embryo.

2. Division postponing: some divisions are postponed.

3.8.2 Input data

Input data are the result of the `4-astec.py` stage (see section 3.7) and will be searched in the directory `SEG/SEG\_<EXP\_SEG>/` (see section 3.7.2).

```
/path/to/experiment/
├── ...
├── SEG/
│   └── SEG_<EXP_SEG>/
│       ├── <EN>_seg_lineage.xml
│       ├── <EN>_seg_t<begin>.mha
│       ├── <EN>_seg_t<..>.mha
│       ├── <EN>_seg_t<end>.mha
│       └── ...
└── ...
```

3.8.3 Output data

The results are stored in sub-directories `POST/POST_<EXP_POST>` under the `/path/to/experiment/` directory where where `<EXP_POST>` is the value of the variable `EXP_POST` (its default value is '`RELEASE`').

```
/path/to/experiment/
├── ...
├── POST/
│   └── POST_<EXP_POST>/
│       ├── <EN>_post_lineage.xml
│       ├── <EN>_post_t<begin>.mha
│       ├── <EN>_post_t<..>.mha
│       ├── <EN>_post_t<end>.mha
│       └── LOGS/
└── ...
```

The image format to be used (here `mha`) is given by the variable `result_image_suffix`, while the lineage format to be used (here `xml`) is given by the variable `result_lineage_suffix`.

3.8.4 Step 1: lineage pruning

Bifurcations of the lineage tree correspond to cell division, while branches (between two bifurcations or between a bifurcation and a leaf) corresponds to the lifespan of a cell. The purpose of this step is to detect suspicious end branches (terminating by a leaf) that may correspond to an over-segmentation error.

An end branch is candidate for deletion if

- either it terminates before the last time point (it corresponds then to a cell without daughter cell in the next time point),

- or the volume of its last cell is too small (threshold given by the variable `postcorrection_volume_minimal_value`).

An end branch candidate for deletion is deleted if

- either it is too short (threshold given by the variable `postcorrection_lifespan_minimal_value`),
- or (if the variable `postcorrection_test_early_division` is set to `True`) either its sister branch (which may not be an end branch) or its mother branch is too short, meaning that there are two divisions too close, (thresholds still given by the variable `postcorrection_lifespan_minimal_value`),
- or if the Pearson correlation coefficient between the volumes of the candidate end branch and its sister branch is less than -`postcorrection_correlation_threshold`, meaning that the volumes are anti-correlated (typically the volumes of the candidate end branch are decreasing while the ones of the sister branch are increasing, indicating a fake division detection).

3.8.5 Step 2: division postponing

- `postcorrection_volume_minimal_value` branch ending with leaf cell below this value are candidate for deletion. Expressed in voxel unit.
- `postcorrection_lifespan_minimal_value`
- `postcorrection_test_early_division`
- `postcorrection_test_volume_correlation`
- `postcorrection_correlation_threshold`
- `postcorrection_lineage_diagnosis` performs a kind of diagnosis on the lineage before and after the post-correction.

3.9 `X-embryoproperties.py`

`X-embryoproperties.py` can be used either to extract cell properties as well as cell lineage from a co-registered image sequence or to handle a property file (pkl or xml).

3.9.1 `X-embryoproperties.py` additional options

The following options are available:

`-i files ...` input files (pkl or xml) to be read

`-o files ...` output files (pkl or xml) to be read

`-c files ...` files (pkl or xml) to be compared to those given by `-i`

`-feature features ...` features to be extracted from the input files, that are to be written in the output files. Features have to be chosen in 'lineage', 'h_min', 'volume', 'surface', 'sigma', 'label_in_time', 'barycenter', 'fate', 'fate2', 'fate3', 'fate4', 'all-cells', 'principal-value', 'name', 'contact', 'history', 'principal-vector', 'name-score', 'cell-compactness'

`-property features ...` same as `-feature`

`--diagnosis` performs some test on the read properties

`--diagnosis-minimal-volume DIAGNOSIS_MINIMAL_VOLUME` displays all cells with volume smaller than DIAGNOSIS_MINIM

`--diagnosis-items DIAGNOSIS_ITEMS` minimal number of items to be displayed

`--print-content` print the keys of the input file(s) (read as python dictionary)

`--print-keys` same as `--print-content`

`--print-types` print types of read features (for debug purpose)

3.9.2 Extracting properties from a co-registered image sequence

When a parameter file is passed after the `-p` option, `X-embryoproperties.py` will compute image sequence properties. Computing cell related informations as well as the lineage tree requires that the (post-corrected) segmentation images have already been co-registered (with `1.5-intraregistration.py` see section 3.4). `X-embryoproperties.py` will parse the `INTRAREG/INTRAREG_<EXP_INTRAREG>/` directory, and will compute the properties from the images in the `POST/POST_<EXP_POST>/` sub-directory, if existing, else of from the `SEG/SEG_<EXP_SEG>/` sub-directory.

3.9.3 Output data

The results are stored in the `POST/POST_<EXP_POST>/` or `SEG/SEG_<EXP_SEG>/` sub-directory under the `INTRAREG/INTRAREG_<EXP_INTRAREG>` where `<EXP_INTRAREG>` is the value of the variable `EXP_INTRAREG` (its default value is '`RELEASE`'). The resulting properties will be stored in the same directory than the images they are issued. It will be stored as a pickle python file, and also as a XML file. Both files contain exactly the same information.

According that the `POST/POST_<EXP_POST>/` sub-directory exists (that post-corrected segmentation images have been co-registered), 3 files will be created, named after `<EN>`

```
/path/to/experiment/
├── ...
├── INTRAREG/
│   └── INTRAREG_<EXP_INTRAREG>/
│       ├── ...
│       ├── POST/
│       │   ├── POST_<EXP_POST>/
│       │   │   ├── <EN>_intrareg_post_lineage.pkl
│       │   │   ├── <EN>_intrareg_post_lineage.txt
│       │   │   └── <EN>_intrareg_post_lineage.xml
│       │   └── ...
│       └── ...
└── ...
```

The computed information are

`all_cells` All the cell identifiers. Each cell (in a segmentation image) has a given label (ranging from 2 and above, 1 being used for the background) in each image. To uniquely identify a cell in the sequence, it has been given an unique identifier computed by $i * 1000 + c$, i and c denoting respectively the image index (ranging in [`<begin>`, `<end>`]) and the cell label.

`cell_barycenter` Cell center of mass (in voxel coordinates)

`cell_contact_surface` For each cell, give for each neighboring cell the contact surface. The sum of these contact surfaces is the cell surface.

`cell_principal_vectors` The cell principal vectors are issued from the diagonalization of the cell covariance matrix (in voxel unit).

`cell_principal_values` The cell principal value are issued from the diagonalization of the cell covariance matrix (in voxel unit).

`cell_volume` Cell volume (in voxel unit)

`cell_compactness` The cell compactness is defined by $\mathcal{C} = \frac{\sqrt[3]{\mathcal{V}}}{\sqrt[2]{\mathcal{S}}}$ where $\mathcal{V}$ is the volume of the cell and $\mathcal{S}$ is its surface.

`cell_surface` Cell surface (in pixel unit). For this computation, is mandatory that the co-registered images are isotropic (the same voxel size along the 3 dimensions X, Y, and Z).

`cell_lineage`

The text file <EN>_intrareg_post_lineage.txt contains diagnosis information about the sequence. It lists

- the cell with the smallest sizes as well as the ones with the largest sizes
- the cell with a weird lineage: cells without a mother cell, or cells without daughter cells or having more than 2 daughter cells
- cells having a small intersection with its mother cell with respect to either the mother cell volume or the cell volume.

3.9.4 Handling properties files

X-embryoproperties.py can also help managing property files.

- Converting from xml to pkl and the other way around.

  ```
  $ X-embryoproperties.py -i file.pkl -o file.xml
  ```

  convert the pickle file file.pkl into the xml file file.xml
- Converting the lineage information from either an xml or an pkl file to a tlp[2] file for lineage visualization

  ```
  $ X-embryoproperties.py -i file.pkl -o file.tlp
  ```

  convert the pickle file file.pkl into the tlp file file.tlp
- Merging files.

  ```
  $ X-embryoproperties.py -i file1.pkl file2.xml ...filen.pkl -o
  merge.xml merge.pkl
  ```

  will merge the files file1.pkl, file2.xml , ..., filen.pkl (note that they can be either xml or pkl) and write the result both in xml and pkl formats.
- Extracting properties.

  ```
  $ X-embryoproperties.py -i file.pkl -feature volume surface -o file.xml
  ```

  will extract the cell volume and surface information from the pickle file file.pkl and write them into the xml file file.xml

3.10 Image preprocessing

The segmentation of membranes images is based on a seeded watershed. Seeds are computed from either one single regional minima image (segmentation of the first time point, see section 3.5) or several ones (segmentation by propagation of the other time points, see section 3.7).

The regional minima operation, as well as the watershed operation, are conducted on the pre-processed version of the fused image. More precisely, the fused image may undergo two kinds of pre-processing, one denoted 'intensity_transformation' (and transform the image values based on its histogram) and the other 'intensity_enhancement' (and transform the image based on a membrane dedicated process). The image used for segmentation is the fusion (by the maximum) of these two pre-processing results (see figure 3.9).

If the fused image is transformed before being segmented, the transformed image is named <EN>_fuse_t<timepoint>_membr and stored in the directory SEG/SEG_<EXP_SEG>/RECONSTRUCTION/ if the value of the variable 'keep_reconstruction' is set to True.

Note that specifying

```
intensity_transformation = 'identity'
intensity_enhancement = None
```

[2]Tulip is a Data Visualization Software, see tulip.labri.fr.

Figure 3.9: The input for segmentation (ie h-minima computation, seeded watershed) is built from (eventually) two images derived from the fusion image.

in the parameter file comes to use the unprocessed fused image as input image for the segmentation.

3.10.1 Histogram based image value transformation

The option `'intensity_transformation'` can be set to one out the three (segmentation of the first time point, see section 3.5) or four (segmentation by propagation of the other time points, see section 3.7) values.

- `None`: this pre-processing channel is not used, meaning that only the membrane dedicated process will produce the input for the segmentation.
- `'identity'`: there is no transformation of the fused image.
- `'normalization_to_u8'`: input images are usually encoded on 2 bytes. However, it is of some interest to have input images of similar intensity distribution, so the segmentation parameters (eg the h's for the regional minima computation) do not have to be tuned independently for each image or sequence.

  This choice casts the input image on a one-byte image (ie into the value range $[0, 255]$) by linearly mapping the fused image values from $[I_{min}, I_{max}]$ to $[0, 255]$. I_{min} and I_{max} correspond respectively to the 1% and to the 99% percentiles of the fused image cumulative histogram. This allows to perform a robust normalization into $[0, 255]$ without being affected by extremely low or high intensity values. Values below I_{min} are set to 0 while values above I_{max} are set to 255.

  The percentiles used for the casting can be tuned by the means of two variables

  ```
  normalization_min_percentile = 0.01
  normalization_max_percentile = 0.99
  ```

- `'cell_normalization_to_u8'`: this choice can only be used for the segmentation propagation (see section 3.7). It has been developed (and kept) for historical reasons but has not proven to be useful yet.

  The segmentation (the image of cell labels) at time point t, $S_t^\star$, is first deformed onto the image at time $t+1$ thanks to the transformation $\mathcal{T}_{t \leftarrow t+1}$ from the image I_{fuse}^{t+1} at time $t+1$ towards to image I_{fuse}^{t} at time t (this transformation is computed with the fused images). The deformed segmentation can be denoted by $S_t^\star \circ \mathcal{T}_{t \leftarrow t+1}$. According that the co-registration of the image I_{fuse}^{t+1} and I_{fuse}^{t} is successful, this deformed segmentation is an estimated segmentation (without any cell division) of I_{fuse}^{t+1}.

  Instead of computing one histogram for the whole image as in the `'normalization_to_u8'`, and thus having one I_{min} and one I_{max} value for the whole image, histogram are here computed on a cell basis,

and a couple (I_{min}, I_{max}) is computed for each label of $S_t^\star \circ \mathcal{T}_{t \leftarrow t+1}$, yielding images of values I_{min} and I_{max}. Since this induces discontinuities at cell borders, these two images are smoothed (with a Gaussian filter of standard deviation `'cell_normalization_sigma'` before casting into $[0, 255]$.

For each cell, different histogram can be used for the computation of I_{min} and I_{max}.

- `'cell_normalization_max_method'` sets the cell area where to compute the histogram for the I_{max} value, while
- `'cell_normalization_min_method'` sets the cell area where to compute the histogram for the I_{min} value.

Cell areas can be defined as

- `'cell'`: all the values of I_{fuse}^{t+1} below the aimed cell defined in $S_t^\star \circ \mathcal{T}_{t \leftarrow t+1}$ are used for the histogram computation,
- `'cellborder'`: only the values of I_{fuse}^{t+1} at the aimed cell border defined in $S_t^\star \circ \mathcal{T}_{t \leftarrow t+1}$ are used for the histogram computation, and
- `'cellinterior'`: all the value of I_{fuse}^{t+1} in the aimed cell interior (the border is excluded) defined in $S_t^\star \circ \mathcal{T}_{t \leftarrow t+1}$ are used for the histogram computation.

Default values are

```
cell_normalization_max_method = 'cellborder'
cell_normalization_min_method = 'cellinterior'
```

meaning that I_{max}'s are computed at the cells' borders while I_{min}'s are computed in the cells' interiors.

3.10.2 Membrane dedicated enhancement

The option `'intensity_transformation'` can be set to one out the two (segmentation of the first time point, see section 3.5) or three (segmentation by propagation of the other time points, see section 3.7) values.

- `None`: this pre-processing channel is not used, meaning that only the histogram based image value transformation will produce the input for the segmentation.
- `'GACE'` stands for *Global Automated Cell Extractor*. This is the method described in [MGFM14, Mic16].
- `'GLACE'` stands for *Grouped Local Automated Cell Extractor*. It differs from one step from `GACE`: the threshold of extrema image is not computed globally (as in `GACE`), but one threshold is computed per cell of $S_{t-1}^\star \circ \mathcal{T}_{t-1 \leftarrow t}$, from the extrema values of the cell bounding box.

`GACE` and `GLACE` consist both of the following steps.

1. Membrane dedicated response computation. The Hessian is computed by convolution with the second derivatives of a Gaussian kernel (whose standard deviation is given by `'mars_sigma_membrane'`). The analysis of eigenvalues and vectors of the Hessian matrix allows to recognize the normal direction of an eventual membrane. A response is then computed based on a contour detector in the membrane normal direction.
2. Directional extrema extraction. Extrema of the response in the direction of the membrane normal are extracted. It yields a valued image of membrane centerplanes.
3. Direction dependent automated thresholding.

   It has been observed that the membrane contrast depends on the membrane orientation with respect to the microscope apparatus. Directional response histogram are built and a threshold is computed for each of them, which allows to compute a direction-dependent threshold.

Thresholds are computing by fitting known distribution on histograms. Fitting is done by the means of an iterative minimization, after an automated initialization. The `'mars_sensitivity'` option allows to control the threshold choice after the distribution fitting.

Setting the `'mars_manual'` to `True` allows to manually initialize the distribution before minimization thanks to the `'mars_manual_sigma'` option.

Last, the user can directly give the threshold to be applied (this is then a global threshold that did not depend on the membrane direction) by setting the `'mars_hard_thresholding'` option at `True`: the threshold to be applied has to set at the `'mars_hard_threshold'` option.

4. Sampling. Points issued from the previous binarization step will be further used for a tensor voting procedure. To decrease the computational cost, only a fraction of the binary membrane may be retained. This fractions is set by the `'mars_sample'` option.

   Sampling is performed through pseudo-random numbers. To reproduce a segmentation experiment by `2-mars.py`, the random seed can be set thanks to the `'mars_sample_random_seed'` option.

   If one want to reproduce segmentation experiments, the verboseness of the experiments has to be increased by adding at least one `'-v'` in the command line of `2-mars.py`. This ensures that the necessary information will be written into the `.log` file. Then, to reproduce one given experiment, one has to retrieve the used random seed `'RRRRRRRRRR'` from the line

   ```
   Sampling step : random seed = RRRRRRRRRR
   ```

   in the log file `SEG/SEG_<EXP_SEG>/LOGS/2-mars-XXXX-XX-XX-XX-XX-XX.log`, and then to add the line

   ```
   mars_sample_random_seed = 'RRRRRRRRRR'
   ```

   in the parameter file to get the same sampling.
5. Tensor voting. Each retained point of the binary image (together with its membrane normal direction) generates a tensor voting field, whose extent is controlled by the `'mars_sigma_TV'` option (expressed in voxel units). These fields are added to yield a global tensor image, and a membraness value is computed at each point, resulting in a scalar image.
6. Smoothing. An eventual last smoothing of this scalar image may be done, controlled by the `'mars_sigma_LF'` option.

3.10.3 Parameter list

General parameters governing the segmentation pre-processing:

- `astec_intensity_enhancement`: equivalent to `intensity_enhancement`
- `astec_intensity_transformation`: equivalent to `intensity_transformation`
- `astec_keep_reconstruction`: equivalent to `keep_reconstruction`
- `intensity_enhancement`
- `intensity_transformation`
- `keep_reconstruction`
- `mars_intensity_enhancement`: equivalent to `intensity_enhancement`
- `mars_intensity_transformation`: equivalent to `intensity_transformation`
- `mars_keep_reconstruction`: equivalent to `keep_reconstruction`

Parameters for the histogram based image value transformation:

- `astec_cell_normalization_max_method`: equivalent to `cell_normalization_max_method`
- `astec_cell_normalization_min_method`: equivalent to `cell_normalization_min_method`

- astec_cell_normalization_sigma: equivalent to cell_normalization_sigma
- astec_normalization_max_percentile: equivalent to normalization_max_percentile
- astec_normalization_min_percentile: equivalent to normalization_min_percentile
- cell_normalization_max_method
- cell_normalization_min_method
- cell_normalization_sigma
- mars_normalization_max_percentile: equivalent to normalization_max_percentile
- mars_normalization_min_percentile: equivalent to normalization_min_percentile
- normalization_max_percentile
- normalization_min_percentile

Parameters for the membrane dedicated enhancement;

- astec_hard_threshold: equivalent to mars_hard_threshold
- astec_hard_thresholding: equivalent to mars_hard_thresholding
- astec_manual: equivalent to mars_manual
- astec_manual_sigma: equivalent to mars_manual_sigma
- astec_sample: equivalent to mars_sample
- astec_sample_random_seed: equivalent to mars_sample_random_seed
- astec_sensitivity: equivalent to mars_sensitivity
- astec_sigma_LF: equivalent to mars_sigma_LF
- astec_sigma_TV: equivalent to mars_sigma_TV
- astec_sigma_membrane: equivalent to mars_sigma_membrane
- mars_hard_threshold
- mars_hard_thresholding
- mars_manual
- mars_manual_sigma
- mars_sample: this parameter sets the fraction of the binary centerplanes that will be used for tensor voting (step 5). Points being randomly drawn, results are not strictly reproducible if the code is re-run with the same sets of parameters. Using a larger value (smaller than or equal to 1.0) increases the reproductibility but induces a larger computational cost.
- mars_sample_random_seed: allows to set the random seed for reproductibility of the sampling step
- mars_sensitivity: this parameter sets the sensitivity for the centerplanes thresholding of step 3. It is set to 0.99 by default. Using larger value (smaller than or equal to 1.0, say 0.9999) allows to extract less-contrasted membranes (for instance cell/background membranes).
- mars_sigma_LF: expressed in real units
- mars_sigma_TV: expressed in voxel units
- mars_sigma_membrane: expressed in real units

3.11 Parameters

The different command line interfaces, or CLIs, (`1-fuse.py`, `2-mars.py`, etc.) requires a parameter file (which is nothing but a `python` file) that contains both information on the experiment (path to the experiment directory, on the sub-directory names – see section 3.11.3) as well as specific parameters for the CLIs.

3.11.1 Prefixed parameters

Some of the parameter sets are said to be *prefixed*, such as the two sets of pre-processing parameters for the `2-mars.py` CLI (see section 3.11.12). Indeed, the pre-processing can be set differently for the seed input image and the membrane input image (eg see section 3.5).

Prefixing parameters allows to either set *all* the parameters with the same name together or set them *independently*.

As exemplified in section 3.5.3, the parameter file lines (where the variables are not prefixed)

```
intensity_transformation = 'normalization_to_u8'
intensity_enhancement = None
```

will set the corresponding pre-processing parameters for both the seed and the membrane image pre-processing. However, using prefixes, as in the lines

```
seed_intensity_transformation = 'Identity'
membrane_intensity_transformation = 'normalization_to_u8'
intensity_enhancement = None
```

allows to set them independently.

This mechanism is designed to simplify the parameter file, but may have undesired consequences. Indeed, using the basic variable names of the registration parameters (see section 3.11.7) for the `4-astec.py` CLI will change all registration parameters included in the pre-processing parameters.

To check whether the parameters have been set correctly, one can either use the `--print-param` CLI option (see section 3.2) beforehand, or to a posteriori check the used parameter in the `log` file.

3.11.2 Common parameters

- `begin`: first time point to be processed (`1-fuse.py`, `4-astec.py` or `5-postcorrection.py`) or single time point to be processed (`2-mars.py` or `3-manualcorrection.py`).
- `end`: last time point to be processed (`1-fuse.py`, `4-astec.py` or `5-postcorrection.py`).
- `delta`: interval between two time points to be processed. Set to 1 by default. Fragile.
- `raw_delay`: Delay to be added to the time points to build the file names. Fragile.
- `time_digits_for_filename`: number of digits used to build the file names.
- `time_digits_for_cell_id`: number of digits used to define unique cellule id. in lineage file. The unique id of cell c at time t is $t \times 10^d + c$ where d is set by `time_digits_for_cell_id`.
- `default_image_suffix`: used for both the result and the temporary data.
  - `'inr'`: Inrimage format, historical choice.
  - `'mha'`: MetaImage format, readable by Fiji.
  - `'tif'`: not advised, since the tiff format does not allow to keep the voxel size along the z direction (at least in a standardized way).
- `result_image_suffix`: used for both the result data.
- `result_lineage_suffix`:
  - `'pkl'`:
  - `'xml'`:

```
<PATH_EMBRYO>/
├── <DIR_RAWDATA>/
│   ├── <DIR_LEFTCAM_STACKZERO>/
│   │   ├── <acquisition_leftcam_image_prefix>000.zip
│   │   ├── <acquisition_leftcam_image_prefix>001.zip
│   │   └── ...
│   ├── <DIR_RIGHTCAM_STACKZERO>/
│   │   ├── <acquisition_rightcam_image_prefix>000.zip
│   │   ├── <acquisition_rightcam_image_prefix>001.zip
│   │   └── ...
│   ├── <DIR_LEFTCAM_STACKONE>/
│   │   ├── <acquisition_leftcam_image_prefix>000.zip
│   │   ├── <acquisition_leftcam_image_prefix>001.zip
│   │   └── ...
│   └── <DIR_RIGHTCAM_STACKONE>/
│       ├── <acquisition_rightcam_image_prefix>000.zip
│       ├── <acquisition_rightcam_image_prefix>001.zip
│       └── ...
└── ...
```

Figure 3.10: Typical organisation of mono-channel data. See also section 3.3.3.

3.11.3 Data organisation parameters

- `DIR_LEFTCAM_STACKONE` see section 3.3.3, see figures 3.10, 3.11, and 3.12.
- `DIR_LEFTCAM_STACKONE_CHANNEL_2` see section 3.3.3
- `DIR_LEFTCAM_STACKONE_CHANNEL_3` see section 3.3.3
- `DIR_LEFTCAM_STACKZERO` see section 3.3.3, see figures 3.10, 3.11, and 3.12.
- `DIR_LEFTCAM_STACKZERO_CHANNEL_2` see section 3.3.3
- `DIR_LEFTCAM_STACKZERO_CHANNEL_3` see section 3.3.3
- `DIR_RAWDATA` see section 3.3.3, see figures 3.10, 3.11, and 3.12.
- `DIR_RAWDATA_CHANNEL_2` see section 3.3.3
- `DIR_RAWDATA_CHANNEL_3` see section 3.3.3
- `DIR_RIGHTCAM_STACKONE` see section 3.3.3, see figures 3.10, 3.11, and 3.12.
- `DIR_RIGHTCAM_STACKONE_CHANNEL_2` see section 3.3.3
- `DIR_RIGHTCAM_STACKONE_CHANNEL_3` see section 3.3.3
- `DIR_RIGHTCAM_STACKZERO` see section 3.3.3, see figures 3.10, 3.11, and 3.12.
- `DIR_RIGHTCAM_STACKZERO_CHANNEL_2` see section 3.3.3
- `DIR_RIGHTCAM_STACKZERO_CHANNEL_3` see section 3.3.3
- `EN`: the so=called *embryo* name. All files will be named after this name. E.g. see section 3.3.4. see section 3.3.4, see figure 3.13.
- `EXP_FUSE`: String (`str` type) or list (`list` type) of strings. It indicates what are the fused images directories, of the form `<PATH_EMBRYO>/FUSE/FUSE_<EXP_FUSE>`.

  ```
  EXP_FUSE = 'exp1'
  EXP_FUSE = ['exp1', 'exp2']
  ```

  are then both valid. Default value of `EXP_FUSE` is `'RELEASE'`. See section 3.3.4, see figure 3.13.

```
<PATH_EMBRYO>/
├── <DIR_RAWDATA>/
│   ├── <DIR_LEFTCAM_STACKZERO>/
│   │   ├── <acquisition_leftcam_image_prefix>000.zip
│   │   └── ...
│   ├── <DIR_RIGHTCAM_STACKZERO>/
│   │   ├── <acquisition_rightcam_image_prefix>000.zip
│   │   └── ...
│   ├── <DIR_LEFTCAM_STACKONE>/
│   │   ├── <acquisition_leftcam_image_prefix>000.zip
│   │   └── ...
│   ├── <DIR_RIGHTCAM_STACKONE>/
│   │   ├── <acquisition_rightcam_image_prefix>000.zip
│   │   └── ...
│   ├── <DIR_LEFTCAM_STACKZERO_CHANNEL_2>/
│   │   ├── <acquisition_leftcam_image_prefix>000.zip
│   │   └── ...
│   ├── <DIR_RIGHTCAM_STACKZERO_CHANNEL_2>/
│   │   ├── <acquisition_rightcam_image_prefix>000.zip
│   │   └── ...
│   ├── <DIR_LEFTCAM_STACKONE_CHANNEL_2>/
│   │   ├── <acquisition_leftcam_image_prefix>000.zip
│   │   └── ...
│   └── <DIR_RIGHTCAM_STACKONE_CHANNEL_2>/
│       ├── <acquisition_rightcam_image_prefix>000.zip
│       └── ...
└── ...
```

Figure 3.11: Typical organisation of multi-channel data. See also section 3.3.3.

```
<PATH_EMBRYO>/
├── <DIR_RAWDATA>/
│   ├── <DIR_LEFTCAM_STACKZERO>/
│   │   ├── <acquisition_leftcam_image_prefix>000.zip
│   │   └── ...
│   ├── <DIR_RIGHTCAM_STACKZERO>/
│   │   ├── <acquisition_rightcam_image_prefix>000.zip
│   │   └── ...
│   ├── <DIR_LEFTCAM_STACKONE>/
│   │   ├── <acquisition_leftcam_image_prefix>000.zip
│   │   └── ...
│   └── <DIR_RIGHTCAM_STACKONE>/
│       ├── <acquisition_rightcam_image_prefix>000.zip
│       └── ...
├── <DIR_RAWDATA_CHANNEL_2>/
│   ├── <DIR_LEFTCAM_STACKZERO>/
│   │   ├── <acquisition_leftcam_image_prefix>000.zip
│   │   └── ...
│   ├── <DIR_RIGHTCAM_STACKZERO>/
│   │   ├── <acquisition_rightcam_image_prefix>000.zip
│   │   └── ...
│   ├── <DIR_LEFTCAM_STACKONE>/
│   │   ├── <acquisition_leftcam_image_prefix>000.zip
│   │   └── ...
│   └── <DIR_RIGHTCAM_STACKONE>/
│       ├── <acquisition_rightcam_image_prefix>000.zip
│       └── ...
└── ...
```

Figure 3.12: Alternative organisation of multi-channel data. See also section 3.3.3.

```
<PATH_EMBRYO>/
├── <DIR_RAWDATA>/
│   └── ...
├── <FUSE>/
│   └── FUSE_<EXP_FUSE>/
│       ├── <EN>_fuse_t000.<result_image_suffix>
│       ├── <EN>_fuse_t001.<result_image_suffix>
│       └── ...
└── ...
```

Figure 3.13: Typical organisation of fused images. See also section 3.3.4.

- `EXP_FUSE_CHANNEL_2` see section 3.3.4
- `EXP_FUSE_CHANNEL_3` see section 3.3.4
- `PATH_EMBRYO`: path to the *experiment*. If not present, the current directory is used. See section 3.3.3, see figures 3.10, 3.11, 3.12, and 3.13
- `acquisition_leftcam_image_prefix` see section 3.3.3, see figures 3.10, 3.11, and 3.12.
- `acquisition_rightcam_image_prefix` see section 3.3.3, see figures 3.10, 3.11, and 3.12.

3.11.4 Ace parameters

Ace stand for *Automated Cell Extractor*. [G[L]]ACE methods aim at detecting and enhancing membranes in a 3D images (see also section 3.10.2).

1. Hessian-based detection of 2-D manifolds, computation of a center-membrane image.
2. Thresholding of the center-membrane image to get a binary image.
3. Reconstruction of a membrane images from the binary image through tensor voting.

- `sigma_membrane`: this is the gaussian sigma that is used to compute image derivatives (in real units), used in step 1.
- `hard_thresholding`: `True` or `False`. If set to `True`, a hard threshold (set by variable `hard_threshold`) is used instead of an automated threshold.
- `hard_threshold`
- `manual`: `True` or `False`. By default, this parameter is set to False. If failure, (meaning that thresholds are very bad, meaning that the binarized image is very bad), set this parameter to True and relaunch the computation on the test image. If the method fails again, "play" with the value of `manual_sigma` ... and good luck.
- `manual_sigma`: Axial histograms fitting initialization parameter for the computation of membrane image binarization axial thresholds (this parameter is used if `manual` is set to `True`). One may need to test different values of `manual_sigma`. We suggest to test values between 5 and 25 in case of initial failure. Good luck.
- `sensitivity`: membrane binarization parameter. Use larger values (smaller than or equal to 1.0) to increase the quantity of binarized membranes to be used for tensor voting.
- `sigma_TV`: parameter which defines the voting scale for membrane structures propagation by tensor voting method (real coordinates). This parameter should be set between 3 μm (little cells) and 4.5 μm (big gaps in the binarized membrane image).
- `sigma_LF`: Additional smoothing parameter for reconstructed image (in real coordinates). It seems that the default value = 0.9 μm is ok for standard use.
- `sample`: Set the fraction of the binarized membranes (obtained at step 2) further used for tensor voting. It allows tensor voting computation speed optimisation (do not touch if not bewared): the more sample, the higher the cost.
- `sample_random_seed`: Drawing a sample from the binarized membranes (see parameter `sample`) is a stochastic process. Setting this parameter to some `int` value allows to make this stochastic process reproducible.
- `bounding_box_dilation`
- `default_image_suffix`

3.11.5 Morphosnake parameters

- `dilation_iterations`: dilation of the cell bounding box for computation purpose.
- `iterations`: maximal number of morphosnake iterations.
- `delta_voxel`: error on voxel count to define a stopping criteria.

- `energy`:
  - `'gradient'`: uses the same formula as in [MNBA14], as in the historical astec version. But seems to be a poor choice.
  - `'image'`: uses directly the image as the energy image.
- `smoothing`: internal parameter for the morphosnake.
- `balloon`: internal parameter for the morphosnake.
- `processors`: number of processors used for the morphosnake correction.
- `mimic_historical_astec`: `True` or `False`. If set to `True`, same implementation than the historical astec version. Kept for comparison purpose.

3.11.6 Preprocessing parameters

The input image may be pre-processed before being used as

- either the *membrane* image (ie the height image) for watershed segmentation,
- or the *seed* image (ie the image with which the regional minima are computed),
- or the *morphosnake* image (ie the image with which the morphosnake energy is computed).

For more details, see section 3.10.

- Ace parameters (see section 3.11.4)
- `intensity_transformation`: set the (histogram based) intensity transformation of the original image (see section 3.10.1)
  - `None`: no intensity transformation of the original image is used to pre-process the input image.
  - `'identity'`: the input image is used without any transformation.
  - `'normalization_to_u8'`: the input image (usually encoded on 16 bits) is normalized onto 8 bits. The values corresponding to percentiles given by the variables `normalization_min_percentile` and `normalization_max_percentile` are mapped respectively on 0 and 255.
  - `'cell_normalization_to_u8'`: same principle than `'normalization_to_u8'` but values mapped on 0 and 255 are computed on a cell basis (cells are the ones of $S^{\star}_{t-1} \circ \mathcal{T}_{t-1 \leftarrow t}$ – see [Gui15] for notations –, ie the segmentation obtained for the previous time point $t-1$ and deformed onto the frame at the current time point t). This can be used only with `4-astec.py` (section 3.7). This feature has been added for tests, but has not demonstrated yet any benefit.
- `intensity_enhancement` set the membrane enhancement transformation of the original image (see section 3.10.2)
  - `None`: no membrane enhancement of the original image is used to pre-process the input image.
  - `'GACE'`: stands for *Global Automated Cell Extractor*. It tries to reconstructed a membrane image through a membrane detector, an automated thresholding and a tensor voting step. The automated thresholding is computed once for the whole image.
  - `'GLACE'`: stands for *Grouped Local Automated Cell Extractor*. It differs from one step from `GACE`: the threshold of extrema image is not computed globally (as in `GACE`), but one threshold is computed per cell of $S^{\star}_{t-1} \circ \mathcal{T}_{t-1 \leftarrow t}$, from the extrema values of the cell bounding box. This can be used only with `4-astec.py` (section 3.7).
- `outer_contour_enhancement`
- `reconstruction_images_combination`:
  - `'addition'`
  - `'maximum'`

- `cell_normalization_min_method`: set the cell area where is computed the percentile value that will give the 0 value in the normalized image
  - `'cell'`
  - `'cellborder'`
  - `'cellinterior'`
- `cell_normalization_max_method`: set the cell area where is computed the percentile value that will give the 255 value in the normalized image
  - `'cell'`
  - `'cellborder'`
  - `'cellinterior'`
- `normalization_min_percentile`
- `normalization_max_percentile`
- `cell_normalization_sigma`: the `'cell_normalization_to_u8'` method computes a couple (I_{min}, I_{max}) for each cell of $S^{\star}_{t-1} \circ \mathcal{T}_{t-1 \leftarrow t}$, yielding discontinuities in the I_{min} and I_{max} from cell to cell. To normalize the whole image, images of I_{min} and I_{max} are built and then smoothed with a gaussian kernel (sigma given by the variable `cell_normalization_sigma`.
- `intensity_transformation`
- Registration parameters (see section 3.11.7) prefixed by `linear_registration_`
- Registration parameters (see section 3.11.7) prefixed by `nonlinear_registration_`
- `keep_reconstruction`: `True` or `False`. Is set to `True`, pre-processed images are kept in a `RECONSTRUCTION/` directory.

3.11.7 Registration parameters

- `compute_registration`
- `pyramid_highest_level`: highest level of the pyramid image for registration. Registration is done hierarchically with a pyramid of images. At each pyramid level, image dimensions are divided by 2. Setting this variable to 6 means that registration starts with images whose dimensions are 1/64th of the original image.
- `pyramid_lowest_level`: lowest level of the pyramid image for registration. Setting it to 0 means that the lowest level is with the image itself. Setting it to 1 or even 2 allows to gain computational time.
- `gaussian_pyramid`
- `transformation_type`
- `elastic_sigma`
- `transformation_estimation_type`
- `lts_fraction`
- `fluid_sigma`
- `normalization`

3.11.8 Seed edition parameters

- `seed_edition_dir`:
- `seed_edition_file`: if run with `'-k'`, temporary files, including the computed seeds are kept into a temporary directory, and can be corrected in several rounds

```
seed_edition_file = [['seeds_to_be_fused_001.txt', 'seeds_to_be_created_001.txt'], \
                     ['seeds_to_be_fused_002.txt', 'seeds_to_be_created_002.txt'],
                     ...
                     ['seeds_to_be_fused_00X.txt', 'seeds_to_be_created_00X.txt']]
```

Each line of a `seeds_to_be_fused_00x.txt` file contains the labels to be fused, e.g. "10 4 2 24". A same label can be found in several lines, meaning that all the labels of these lines will be fused. Each line of `seeds_to_be_created_00x.txt` contains the coordinates of a seed to be added.

3.11.9 Watershed parameters

- `seed_sigma`: gaussian sigma for smoothing of initial image for seed extraction (real coordinates).
- `seed_hmin`: h value for the extraction of the h-minima,
- `seed_high_threshold`: regional minima thresholding.
- `membrane_sigma`: gaussiab sigma for smoothing of reconstructed image for image regularization prior to segmentation (real coordinates).

3.11.10 `1-fuse.py` parameters

- `acquisition_orientation`: image orientation (`'right'` or `'left'`) gives the rotation (with respect to the Y axis) of the left camera frame of stack #0 to be aligned with the the left camera frame of stack #1.
  - `'right'`: +90 degrees
  - `'left'`: -90 degrees

  See section 3.3.2.
- `acquisition_mirrors`: mirroring of the right camera image along the X-axis. Right camera images may have to be mirrored along the X-axis to be aligned with the left camera images.
  - `True`: +90 degrees
  - `False`: -90 degrees

  Since it should depend on the apparatus, this parameter should not change for all acquisitions performed by the same microscope. See section 3.3.2.
- `acquisition_resolution`: acquisition voxel size e.g.

  ```
  raw_resolution = (.21, .21, 1.)
  ```

  see section 3.3.2
- `acquisition_stack0_leftcamera_z_stacking`: see `acquisition_leftcamera_z_stacking`.
- `acquisition_stack1_leftcamera_z_stacking`: see `acquisition_leftcamera_z_stacking`.
- `acquisition_slit_line_correction`: `True` or `False`. See section 3.3.1.
- `target_resolution`: isotropic voxel size of the fusion result (fused images). See section 3.3.4.
- `fusion_strategy`:
  - `'direct-fusion'`: each acquisition is linearly co-registered with the first acquisition (stack #0, left camera). Used registration parameters are the ones prefixed by `fusion_preregistration_` and `fusion_registration_`. Then weights and images are transformed thanks to the computed transformations.

  - 'hierarchical-fusion': from the couple (left camera, right camera), each stack is reconstructed (with the registration parameters prefixed by fusion_preregistration_ and fusion_registration_), following the same scheme than the direct fusion but with only 2 images. Then stack#1 is (non-)linearly co-registered with stack #0 with the registration parameters prefixed by fusion_stack_preregistration_ and fusion_stack_registration_. Images and weights associated with stack#1 are then (non-)linearly transformed. Finally a weighted linear combination gives the result.

  See section 3.3.6
- acquisition_cropping: True or False. If set to True, the acquisitions stacks are cropped before fusion. See section 3.3.5
- acquisition_cropping_margin_x_0: extra margin for the left side of the X direction.
- acquisition_cropping_margin_x_1: extra margin for the right side of the X direction.
- acquisition_cropping_margin_y_0: extra margin for the left side of the Y direction.
- acquisition_cropping_margin_y_1: extra margin for the right side of the Y direction.
- acquisition_cropping_margin_x: allows to set both acquisition_cropping_margin_x_0 and acquisition_cropping_
- acquisition_cropping_margin_y: allows to set both acquisition_cropping_margin_y_0 and acquisition_cropping_
- acquisition_cropping_margin: allows to set the four margin variables.
- Registration parameters (see section 3.11.7) prefixed by fusion_preregistration_
- Registration parameters (see section 3.11.7) prefixed by fusion_registration_
- Registration parameters (see section 3.11.7) prefixed by fusion_stack_preregistration_
- Registration parameters (see section 3.11.7) prefixed by fusion_stack_registration_
- xzsection_extraction: True or False. Setting xzsection_extraction to True allows to extract XZ-sections of the 4 co-registered stacks as well as the weighting function images. It provides an efficient way to check whether the acquisition_leftcamera_z_stacking variable was correcly set. See section 3.3.7
- fusion_cropping: True or False. If set to True, the fusion result is cropped. see section 3.3.8
- fusion_cropping_margin_x_0
- fusion_cropping_margin_x_1
- fusion_cropping_margin_y_0
- fusion_cropping_margin_y_1
- fusion_cropping_margin_x: allows to set both fusion_cropping_margin_x_0 and fusion_cropping_margin_x_1
- fusion_cropping_margin_y: allows to set both fusion_cropping_margin_y_0 and fusion_cropping_margin_y_1
- fusion_cropping_margin: allows to set the four margin variables.
- acquisition_leftcamera_z_stacking: allows to set both acquisition_stack0_leftcamera_z_stacking and acquisition_stack1_leftcamera_z_stacking. Gives the order of stacking of in the Z direction

  - 'direct': from the high-contrasted images (small values of z) to the fuzzy/blurred ones (large values of z)
  - 'inverse': the other way around.

  See section 3.3.2.
- fusion_weighting: set the weighting function for the weighted sum of the registered acquisition stacks (for all channels to be processed).

  - 'uniform': uniform (or constant) weighting, it comes to the average of the resampled co-registered stacks
  - 'ramp': the weights are linearly increasing or decreasing along the Z axis
  - 'corner': the weights are constant in a corner portion of the stack, defined by two diagonals in the XZ-section

  - 'guignard': original historical weighting function, described in Leo Guignard's Phd thesis [Gui15], that puts more weight to sections close to the camera and take also account the traversed material.

  See section 3.3.7.
- fusion_weighting_channel_1: set the weighting function for the weighted sum of the registered acquisition stacks for the first channel (in case of multi-channel acquisition).
- fusion_weighting_channel_2: set the weighting function for the weighted sum of the registered acquisition stacks for the second channel (in case of multi-channel acquisition).
- fusion_weighting_channel_3: set the weighting function for the weighted sum of the registered acquisition stacks for the third channel (in case of multi-channel acquisition).

The following parameters are kept for backward compatibility:

- fusion_crop same as fusion_cropping
- fusion_margin_x_0 same as fusion_cropping_margin_x_0
- fusion_margin_x_1 same as fusion_cropping_margin_x_1
- fusion_margin_y_0 same as fusion_cropping_margin_y_0
- fusion_margin_y_1 same as fusion_cropping_margin_y_1
- fusion_xzsection_extraction same as xzsection_extraction
- raw_crop same as acquisition_cropping
- raw_margin_x_0 same as acquisition_cropping_margin_x_0
- raw_margin_x_1 same as acquisition_cropping_margin_x_1
- raw_margin_y_0 same as acquisition_cropping_margin_y_0
- raw_margin_y_1 same as acquisition_cropping_margin_y_1
- raw_mirrors same as acquisition_mirrors
- raw_ori same as acquisition_orientation
- raw_resolution same as acquisition_resolution

- begin see section 3.3.2
- delta
- end see section 3.3.2
- fusion_weighting
- fusion_weighting_channel_1
- fusion_weighting_channel_2
- fusion_weighting_channel_3
- raw_delay

3.11.11 1.5-intraregistration.py parameters

These parameters are prefixed by intra_registration_.

- Registration parameters (see section 3.11.7)
- reference_index: defines the still image after transformation compositions it will only translated, except if reference_transformation_file or reference_transformation_angles are set. See section 3.4.6.
- reference_transformation_file: resampling transformation to be applied to the reference image (and to the whole serie) after transformation compositions. See section 3.4.6.
- reference_transformation_angles: list of rotations wrt the X, Y,or Z axis that defines the resampling transformation.

```
reference_transformation_angles = 'X 30 Y 50'
```

represents a rotation of 30 degree around the X axis followed by a rotation of 50 degrees around the Y axis.

Beware: rotation composition depends on the order, so `'X 30 Y 50'` is not equivalent to `'Y 50 X 30'`.

- `template_type`
- `template_threshold`
- `margin`
- `resolution`
- `rebuild_template`: `True` or `False`. If set to `True`, force to recompute the template as well as the transformations from existing co-registrations (that are not re-computed). It is useful when a first intra-registration has been done with only the fusion images: a second intra-registration with the segmentation images as template can be done without recomputing the co-registrations.
- `sigma_segmentation_images`
- `resample_fusion_images`
- `resample_segmentation_images`
- `resample_post_segmentation_images`
- `movie_fusion_images`
- `movie_segmentation_images`
- `movie_post_segmentation_images`
- `xy_movie_fusion_images`
- `xz_movie_fusion_images`
- `yz_movie_fusion_images`
- `xy_movie_segmentation_images`
- `xz_movie_segmentation_images`
- `yz_movie_segmentation_images`
- `xy_movie_post_segmentation_images`
- `xz_movie_post_segmentation_images`
- `yz_movie_post_segmentation_images`
- `maximum_fusion_images`
- `maximum_segmentation_images`
- `maximum_post_segmentation_images`

3.11.12 `2-mars.py` parameters

These parameters are prefixed by `mars_`.

- `first_time_point`: first time point to be segmented by the mars method. Overrides the value of the `begin` variable.
- `last_time_point`: last time point to be segmented by the mars method.
- Watershed parameters (see section 3.11.9)
- Seed edition parameters (see section 3.11.8)
- Preprocessing parameters (see section 3.11.6) prefixed by `seed_`
- Preprocessing parameters (see section 3.11.6) prefixed by `membrane_`

3.11.13 `3-manualcorrection.py` parameters

- `first_time_point`: first time point to be corrected. Overrides the value of the `begin` variable.
- `last_time_point`: lats time point to be corrected.
- `input_image`: defines the input file names (to be used when correcting other files than the `2-mars.py` output file.
- `output_image`: defines the output file names (to be used when correcting other files than the `2-mars.py` output file.
- `mapping_file`: path to mapping file for manual correction of a segmentation (ie label) image. See above the syntax of this file.
  - 1 line per label association
  - background label has value 1
  - the character `#` denotes commented lines

  Example of `mapping_file`:

```
# here the input label 8 will be mapped with new value 7, etc...
8 7
9 2
4 64
29 23
# ... etc ...
# background labels
30 1
89 1
```

3.11.14 `4-astec.py` parameters

These parameters are prefixed by `astec_`.

- Watershed parameters (see section 3.11.9)
- Preprocessing parameters (see section 3.11.6) prefixed by `seed_`
- Preprocessing parameters (see section 3.11.6) prefixed by `membrane_`
- Preprocessing parameters (see section 3.11.6) prefixed by `morphosnake_`
- Morphosnake parameters (see section 3.11.5)
- `propagation_strategy`:
  - `'seeds_from_previous_segmentation'`
  - `'seeds_selection_without_correction'`
- `previous_seg_method`: how to build the seeds $S^e_{t-1\leftarrow t}$ for the computation of $\tilde{S}_t$
  - `'deform_then_erode'`: $S^\star_{t-1}$ is transformed towards I_t frame through $\mathcal{T}_{t-1\leftarrow t}$, and then the cells and the background are eroded.
  - `'erode_then_deform'`: historical method. The cells and the background of $S^\star_{t-1}$ are eroded, and then transformed towards I_t frame through $\mathcal{T}_{t-1\leftarrow t}$.
- `previous_seg_erosion_cell_iterations`: set the cell erosion size for $S^e_{t-1\leftarrow t}$ computation.
- `previous_seg_erosion_background_iterations`: set the background erosion size for $S^e_{t-1\leftarrow t}$ computation.
- `previous_seg_erosion_cell_min_size`: size threshold. Cells whose size is below this threshold will be discarded seeds in $S^e_{t-1\leftarrow t}$

- `watershed_seed_hmin_min_value`: set the h_{min} value of the $[h_{min}, h_{max}]$ interval.
- `watershed_seed_hmin_max_value`: set the h_{max} value of the $[h_{min}, h_{max}]$ interval.
- `watershed_seed_hmin_delta_value` set the δh to go from one h to the next in the $[h_{min}, h_{max}]$ interval.
- `background_seed_from_hmin`: `True` or `False`. Build the background seed at time point t by cell propagation.
- `background_seed_from_previous`: `True` or `False`. Build the background seed at time point t by using the background seed from $S^e_{t-1 \leftarrow t}$. Fragile.
- `seed_selection_tau`: Set the τ value for division decision (seed selection step).
- `minimum_volume_unseeded_cell`: Volume threshold for cells without found seeds in the seed selection step. Cells with volume (in $\tilde{S}_t$) whose size is below this threshold and for which no seed was found are discarded.
- `volume_ratio_tolerance`: Ratio threshold to decide whether there is a volume decrease (due to the background) for morphosnake correction.
- `volume_ratio_threshold`: Ratio threshold to decide whether there is a large volume decrease for segmentation consistency checking.
- `volume_minimal_value`: Size threshold for seed correction step. For a given cell at time point $t-1$, if the corresponding cell(s) at time point t has(ve) volume below this threshold, they are discarded (and the cell at time point $t-1$ has no lineage.
- `morphosnake_correction`: `True` or `False`.
- `outer_correction_radius_opening`

3.11.15 `5-postcorrection.py` parameters

These parameters are prefixed by `postcorrection_`.

- `volume_minimal_value` branch ending with leaf cell below this value are candidate for deletion. Expressed in voxel unit.
- `lifespan_minimal_value`
- `test_early_division`
- `test_volume_correlation`
- `correlation_threshold`
- `test_postponing_division`
- `postponing_correlation_threshold`
- `postponing_minimal_length`
- `postponing_window_length`
- `lineage_diagnosis` performs a kind of diagnosis on the lineage before and after the post-correction.

Bibliography

[FDM+10] R. Fernandez, Pradeep Das, V. Mirabet, E. Moscardi, Jan Traas, J.-L. Verdeil, Grégoire Malandain, and Christophe Godin. Imaging plant growth in 4D: robust tissue reconstruction and lineaging at cell resolution. *Nature Methods*, 7:547–553, 2010.

[GFL+20] Léo Guignard, Ulla-Maj Fiuza, Bruno Leggio, Julien Laussu, Emmanuel Faure, Gaël Michelin, Kilian Biasuz, Lars Hufnagel, Grégoire Malandain, Christophe Godin, and Patrick Lemaire. Contact area-dependent cell communication and the morphological invariance of ascidian embryogenesis. *Science*, July 2020.

[Gui15] Léo Guignard. *Quantitative analysis of animal morphogenesis: from high-throughput laser imaging to 4D virtual embryo in ascidians.* Theses, Université Montpellier, December 2015.

[MGFM14] Gaël Michelin, Léo Guignard, Ulla-Maj Fiuza, and Grégoire Malandain. Embryo Cell Membranes Reconstruction by Tensor Voting. In *ISBI - International Symposium on Biomedical Imaging*, Beijing, China, April 2014. IEEE.

[Mic16] Gaël Michelin. *Image analysis tools and inter-individual registration for the study of animal and plant morphogenesis.* Theses, Université Côte d'Azur, October 2016.

[MNBA14] P. Màrquez-Neila, L. Baumela, and L. Alvarez. A morphological approach to curvature-based evolution of curves and surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(1):2–17, Jan 2014.

[Ots79] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE Trans. Sys., Man., Cyber.*, 9(1):62–66, 1979.