Time registration Manual

Léo Guignard

September 16, 2019

Contents

1 Purpose

This repository has for purpose to wrap the set of tools provided by the `'blockmatching'` algorithm described in [Ourselin et al., 2000] and used in [McDole et al., 2018; Guignard et al., 2017, 2014].

Provided the `'blockmatching'` library installed (see here) this wrapper allows to easily apply it to register in time 3D movies.

2 Requirements and installation, (tested on Linux, Ubuntu 18.04)

To install `'blockmatching'`, it is necessary to have cmake, a c/c++ compiler and zlib installed:

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

Here are the steps to install `'blockmatching'`: First one have to install the external libraries. To do so, one can run the following commands in a terminal from the BlockMatching folder:

To install klb read/write tools:

```
$ cd external/KLBFILE/
$ mkdir build
$ cd build
$ cmake ../keller-lab-block-filetype
$ make -j<desired number of cores>
```

To install tiff read/write tools, from the folder BlockMatching:

```
$ cd external/TIFF
$ mkdir build
$ cd build
$ cmake ../tiff-4.0.6
$ make -j<desired number of cores>
```

Once these are installed one can run the following commands in a terminal from the BlockMatching folder:

```
$ mkdir build
$ cd build
$ ccmake ..
```

press c then e
Then enter the correct absolute paths for the tiff and klb builds (ie `'/path/to/BlockMatching/external/TIFF/build and /path/to/BlockMatching/external/KLBFILE/build'`).
Then c then e then g

```
$ make -j<desired number of cores>
```

Then this newly built binaries (found in BlockMatching/build/bin) have to be accessible from the different scripts that will be ran. To do so one can add the following command to their /.bashrc (/.profile for mac users):

```
export PATH=$PATH:/path/to/BlockMatching/build/bin
```

One 'direct' way to do so is to run the following command:

```
echo 'export PATH=$PATH:/path/to/BlockMatching/build/bin' >>  /.bashrc
```

or add a line to the json configuration file to specify the path to the binaries (see below).

Then, for this wrapper to it work it is first necessary to have installed:

- Python 2.7.x (`https://www.python.org/download/releases/2.7/`)

Some Python libraries are also required:

- Scipy (`https://www.scipy.org/`)
- Numpy (`https://numpy.org/`)
- IO (`https://github.com/leoguignard/IO`)
  - h5py (`https://pypi.python.org/pypi/h5py`)
  - pylibtiff (`https://github.com/pearu/pylibtiff`)
- pyklb (`https://github.com/bhoeckendorf/pyklb`)
- statsmodels (`https://www.statsmodels.org`)

Once everything is installed `'time-registration.py'` is ready to run.

3 Running `'time-registration.py'`

To run the script `'time-registration.py'` one can run the following command from the folder containing it:

```
$ python time-registration.py
```

The script then prompt the user to inform a path to the json configuration files or to a folder containing at least one json configuration file. Some examples are provided in the folder `'json-examples'`.
If the configuration file is correctly filled, the script will then compute the registrations and output the registered images (according to what specified the user in the json file).
In order to skip entering the path to the configuration file(s) one can also run the script as previously appending the name of the configuration file to the command:

```
$ python time-registration.py /path/to/configuration/file.json
```

4 Description of the configuration file

Everything that the script will do is determined by the configuration file. The script being somewhat flexible, the configuration file has a significant number of possible parameters. Some of them are required, others have default values and some are optional depending on the chosen options. The parameter files are written in json (`https://www.json.org/`), please follow the standard format so your configuration files can be read correctly.

4.1 List of all parameters

Here is an exhaustive list of all the possible parameters:
File path format:

- `path_to_data`, mandatory, common path to the image data
- `file_name`, mandatory, image name pattern
- `trsf_folder`, mandatory, output path for the transformations
- `output_format`, default value: `None`, path to the ouput image. Can be a folder, in that case the original image name is kept. Can be a file name, in that case it will be written in the original folder. Can be a folder plus name pattern.
- `suffix`, default value: `None`, suffix to add to the output image if output_format is not specified
- `projection_path`, default value: `None`, path to the output folder for the maximum intensity projection images. If not specified it will be written like the output format with -1 for the time.
- `check_TP`, default value: `None`, whether or not checking of all the required images exist.
- `path_to_bin`, default value: `""`, path to the blockmatching binaries, not necessary if it is in your $PATH

Image information:

- `voxel_size`, mandatory, voxel size of the image

- `first`, mandatory, first time point of the sequence to register
- `last`, mandatory, last time point of th sequence to register
- `not_to_do`, default value: `[]`, list of time points to not process (they will be totally ignored from the process)

Registration parameters:

- `compute_trsf`, default value: `1`, wheter or not computing the transformations (one might not want to do it if the transformations have already been computed for a different channel)
- `ref_TP`, mandatory, time point of the reference image onto which the registration will be made.
- `ref_path`, default value: `None`, path to the reference image, if specified, every image will be directly registered to that particular image
- `trsf_type`, default value: `"rigid"`, choose between `"translation"`, `"rigid"`, `"affine"`, `"vectorfield"`
- `registration_depth`, default value: `3`, Maximum size of the blocks for the registration algorithm (min `0`, max `5`), the lower the value is the more the registration will take into account local details but also the more it will take time to process
- `interpolation`, default value: `"linear"`, choose between `"nearest"` (for label images), `"linear"` and `"cspline"` (the `"cspline"` interpolation has border effects when the signal is too low)
- `padding`, default value: `1`, put to `1` if you want the resulting image bounding boxes to be padded to the union of the transformed bounding boxes. Otherwise (value at `0`) all the images will be written in the reference image bounding box
- `recompute`, default value: `1`, if `1`, it forces to recompute the transformations even though they already exist
- `lowess_interpolation`, default value: `0`, if `1`, smoothes the transformations, so far only works with `"translations"`
- `window_size`, default value: `5`, size of the window for the lowess interpolation
- `step_size`, default value: `100`, if it is not necessary to compute all the time points, this is the step size between the computed times. To be used `lowess_interpolation` has to be `1`
- `sigma`, default value: `2.0`, smoothing parameter for the non-linear registration
- `keep_vectorfield`, default value: `0`, if `1`, will write the transformation vector field (it will be large!)

Apply registration parameter:

- `apply_trsf`, default value: `1`, if `1`, will apply the computed transformations to the images.

4.2 Mandatory parameters

It is mandatory to inform the path to your dataset, the file name pattern, the path to where the transformation will be stored and to inform the format of the output registered image. The path to the data set is specified with the keyword `"path_to_data"`.

References

Guignard, L., Fiuza, U.-M., Leggio, B., Faure, E., Laussu, J., Hufnagel, L., Malandain, G., Godin, C., and Lemaire, P. (2017). Contact-dependent cell communications drive morphological invariance during ascidian embryogenesis.

Guignard, L., Godin, C., Fiuza, U.-M., Hufnagel, L., Lemaire, P., and Malandain, G. (2014). Spatio-temporal registration of embryo images. In *ISBI - International Symposium on Biomedical Imaging*, Pekin, Chine. IEEE.

McDole, K., Guignard, L., Amat, F., Berger, A., Malandain, G., Royer, L. A., Turaga, S. C., Branson, K., and Keller, P. J. (2018). In toto imaging and reconstruction of post-implantation mouse development at the single-cell level. *Cell*, 175(3):859–876.

Ourselin, S., Roche, A., Prima, S., and Ayache, N. (2000). Block matching: A general framework to improve robustness of rigid registration of medical images. In *MICCAI'00*, volume 1935 of *LNCS*, pages 557–566. Springer.