User manual for the registration

The notebook `registration_notebook.ipynb` presents a user-friendly code to register multiple samples in high-throughput and using multichannel images, and outputs the fused images from the 2 views. The purely registration part comes from the Python package registration-tools, that is itself a wrapping of the blockmatching algorithm published here.
To provide an accurate initial transformation to the code, 3D visualization can be necessary, so we developped a Napari plugin called napari-manualregistration to help the user.

1 Installation

- To use the 3D visualization tools, you will need to install Napari (for troubleshooting, see the napari page). Using pip :

  ```
  python -m pip install "napari[all]"
  ```

- And to install the registration tools :

  ```
  conda activate organoid
  conda install -c morpheme vt
  ```

- Write applyTrsf in your terminal. If you get : Usage: applyTrsf [[-floating—-flo] image-in] [[-result—-res] image-out] [-transformation —-trsf [-voxel-transformation —-voxel-trsf . . . . Then the installation is successful, otherwise check Troubleshooting section.

  ```
  pip install 3D-registration
  ```

Troubleshooting

- When you use conda, if it shows "Conda SSL Error : OpenSSL appears to be unavailable on this matching" : Go to `C:/Users/username/Anaconda3/Library/bin`, copy the files `libcrypto-1_1-x64` and `libssl-1_1-x64` and paste them into `C:/Users/username/Anaconda3/DLLs`

- When you write "conda install -c morpheme vt", if you have error messages on compatibilities, then you have to deactivate your environnement and delete it in the following way :

  ```
  conda deactivate
  conda env remove -n registration-env
  ```

  Then write :

  ```
  conda config --add channels conda-forge
  ```

  And start the installation again, from conda create.

- When running ApplyTrsf, if you get an error like "applyTrsf.exe - System Error. Executing the code is impossible because pthreadVSE2.dll cannot be found.", open this link : `https://www.pconlife.com/download/otherfile/185413/9ec519368844bffd89ed4ff61342b98d/` Complete the captcha, download pthreadvse2.dll using the green button, and put the resulting file in the folder `C:/Users/username/Anaconda3/envs/registration/Library/bin`

2 Usage

2.1 Notebook

In the notebook, the convention is such that one of the view is fixed, called the reference view, and we register the other view, called floating view, onto the first one. The code works on 3D images, multichannels or not, but no timesequence. If the image is multichannel, the code will register first what we call a reference channel, generally a marker expressed ubiquitously, and then use this transformation to register the other view using the exact same rotation and translation.

Data structure Before using the notebook, ensure that your data is organized correctly: Each sample should have two TIFF files, representing the twp different views. These files should be stored in the same folder. The TIFF files must be in one of the following formats: int16, uint16, or float32. If your files are in a different format, they will not be processed, and no output will be saved.
The code will take your 3D or 3D+multichannel data, create a folder that has the name of the reference image. In that folder :

- the subfolder 'raw' will contain the raw 3D images, one per channel and per view.
- the subfolder 'trsf' will contain the transformations used to register the data, that the algorithm will compute, save and apply.
- the subfolder 'registered' will contain the individual images after registration, again one per channel and per view.
- the subfolder 'fused' will contain one file per channel, after fusion of the two views. It will also contain a multichannel hyperstack combining the channels. That final image will have a different size from the original one.

Notebook Walkthrough Here is a overview list of the functionnalities, for more precision see the parameters list or the source code :

1. Optional: Handling multiposition experiments
   If your dataset contains multiple samples with different positions, and you have saved in the metadata XML files defining these positions, you can use them to automatically assign the 2 views of each sample. This step is crucial if the views were not captured in the same order. If it was assign in the same order, you can manually define `list_ref` and`list_float` as the name of the reference and floating images in the correct order, for example `list_ref`$= ['01','02']$ and `list_float`$= ['03','04']$ with 01 and 03 the reference and floating views of sample 1 and 02 and 04 the reference and floating views of sample 2.
   `associate_positions` pairs the views for each sample, gives as an output the numbers to associate. It uses the string values 'dXPosition' and 'dYPosition' in the XML file.

2. Defining path, names and channels, and creating the folder structure
   You need to define the path to the folder where your data is stored (see Data Structure), `folder_experiment`, and specify the names of the images you are working with, optionally using the number saved by the function `associate_positions`. The function `create_folders` takes the multichannel image and save the individual 3D images in the 'raw' subfolder.

3. Automatic Image Registration
   To register your floating image onto the reference one, you should have an idea of the transformation to apply, e.g 180° in X if you flipped the sample horizontally. From this approximative initial transformation, the algorithm will find the exact transformation to match the 2 sides precisely. Necessary parameters are the approximate transformation (rotation and translation) in pixel, the voxel size of your input and output (you can choose to save an image of a different size), and the reference channel used to compute the registration. The transformation applied can be rigid or affine, the default is rigid because the sample usually does not deform between the 2 views.

4. Visualizing the Registration with Napari
   Napari is a visualization tool that allows you to inspect the results of the registration process. By visualizing the registered images, you can determine if the alignment is satisfactory or if further adjustments are needed. The reference channel is used by default for visualization, but you can choose any other channel if needed. Once you ensured that the registration process has been successful and if you have multiple samples, you can run the registration automatically on other samples.

5. Finding a precise initial transformation
   If your initial parameters gave an output that is not satisfactory, consider providing a more precise initial transformation. You can check what your initial transformation looks like by putting the parameter `test_init` to 1. If you are not sure what initial transformation to provide, you can use the plugin napari-manualregistration to visualize your two views and test different transformations. From the plugin, you can save a parameter file of your approximate transformation and give it as an input in the parameter `input_init_trsf_from_plugin` in the `register` function.
   If you can not use Napari, there is the possibility to manually define landmarks using another tool of your choice, and use the function `manual_registration_fct` to compute the values of rotation and translation as described at the end of the notebook. One landmark is a feature that you recognize in both the reference image and the floating image, that you will need to pinpoint with a marker of given label, this label has to be a ROI that has the same value in the reference and floating image. The size of the ROIs does not matter, the code will match the center of each ROI in the two views. 3 ROIs at least are needed to compute the transformation to match the two sides.

6. Fusing the Images
   After successful registration, the next step is to fuse the two views into a single image. During fusion, a weight profile in the shape of a sigmoid function is applied to blend the views smoothly. You can adjust the slope of this profile, 5 corresponds to a low slope, wide fusion width and 25 to a strong slope, very thin fusion width.

7. Creating a Multichannel Image
   Once the views are fused, if you need an output as a hyperstack, you can combine all the channels into a single multichannel image.

Specific parameters of the registration function

- `compute_trsf`, `[0 | 1]`, default value: `0`, if `1` the transformation has to be computed. If it is `0`, the transformation already exists. If you have multiple channels of the same image, it is recommended to pick one expressed homogeneously as the reference, register this channel using `compute_trsf=1`. Then you can use `compute_trsf=0`, and the algorithm will find the pre-existing transformation to register the other channels onto the reference channel.

- `rot`, `[list]`, default value: `[0,0,0]`, list of the rotations to apply to the floating image. (convention XYZ)
- `trans1`, `[list]`, default value: `[0,0,0]`, list of the translations to apply to the floating image before rotation. (convention XYZ)
- `trans2`, `[list]`, default value: `[0,0,0]`, list of the translations to apply to the floating image after rotation. (convention XYZ)
- `other_trsf`, `[list]`, default value: `[]`, list of transformations to apply to the floating image. If you want to directly give the sequence of transformations, use this parameter. If this argument is not `None`, the values of the parameters `rot`, `trans1`, and `trans2` will be ignored. You can use flipping (`flip`), rotations (`rot`), and translation (`trans`). Specify the axis `X`, `Y`, or `Z` after the sample ID of the transformation. For rotations and translation, specify the value (angle or distance) after the axis. IMPORTANT: Needs 2 sets of brackets. If `ordered_init_trsfs` is `True`, the transformations will be applied in the order they are given in the list `init_trsfs`. Example: `[['flip', 'Z', 'trans', 'Z', -10,'trans','Y',100,'rot','X',-29,'rot','Y',41,'rot','Z',-2]]`
- `input_init_trsf_from_plugin`, `[str]`, optional, path to the JSON file saved by the plugin, containing the transformation. If this parameter is not `None`, the transformation will be extracted from the JSON file no matter the parameters `rot`, `trans1`, `trans2`, or `other_trsf`.
- `test_init`, `[0 | 1]`, default value: `0`, if `1`, only the initial transformation is applied. If `0`, the initial transformation plus the blockmatching algorithm is applied.
- `trsf_type`, `[str]`, default value: `'rigid'`, type of transformation to compute: `'rigid'`, `'affine'`,`'translation'`,`'vectorfield'`
- `depth`, `[int]`, 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
- `bbox`, `[0 | 1]`, default value: `0`, if `1`, the bounding box of the original image can extend, otherwise it cannot.
- `image_interpolation`, `[str]`, default value: `'linear'`, type of interpolation to apply to the image.
- `padding`, `[int]`, default value: `0`, 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.
- `save_json`, `[str]`, optional, if not `None`, save the parameters of the function in a JSON file at the given path. IMPORTANT: saves the parameters of the function, not the parameters of the registration, particularly the initial transformations only. To save the actual transformations, use the function `"compute_transformation_from_trsf_files"`.
- `ordered_init_trsfs`, `[bool]`, default value: `False`, if `True`, the transformations will be applied in the order they are given in the list `init_trsfs`.

JSON files There is a default parameter in the `register` function, saving a json file with the list of parameters you gave as an input : paths, voxel sizes, initial transformations. This file is named like your reference image (.json) and can be used to run the algo in command line (see next section). From the trsf folder, you can compute the values of rotation and translation actually applied by the blockmatching algorithm using the function `compute_transformation_from_trsf_files`.
Another json file, named `initial_transformations.json`, is saved as an output from the manual-registration-plugin with values of rotations and translations only. You can use the same file from multiple samples from the same acquisition.

2.2 Run registration in command line

To run the code on a big batch of data, you can use a loop in the notebook, or you can use the command line. In that case, you need to know the parameters beforehand, and fill them into a json file. Parameters are found here and you can find an example of a json file in the folder "jsonfiles". 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. In your environment, write:

```
spatial-registration
```

and then drop the json file with the registration parameters.

If the configuration file is correctly filled, the script will then compute the registrations and output the registered images (according to what was specified by 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:

```
spatial-registration /path/to/configuration/file.json
```

2.3 Troubleshooting

- If some basic modules cannot be imported when you execute a code in VS Code, check in the bottom of the VS Code window that the interpreter is in the right environment : it should be written "3.10.8 ('registration-env':conda) ". If it is not, click on the button next to Python and select the right environment.

- The warning "pyklb library not found" or "KLB library is not installed" does not prevent the code from running normally.

- When running the code, if you get an error like "applyTrsf.exe - System Error. Executing the code is impossible because pthreadVSE2.dll cannot be found." see Troubleshooting section in the section 'Installation'.