PyRIS - Python RIvers by Satellite
Usage manual

PyRIS v3.1 (Python 3)

Riccardo Bonanomi *

University of Trento
2023

*University of Trento - Center Agriculture Food Environment (C3A) - riccardo.bonanomi@unitn.it

License

Introduction to this manual

PyRIS is a software developed by Federico Monegaglia[1] (Monegaglia et al., 2018) and it can be used to analyse meandering river from Landasat[2] satellite multi-spectral images. The original code could be found in a GitHub[3] repository, but this is not available anymore, and it was written in Python 2[4]. Monegaglia then developed PyRISdem that adds to PyRIS the ability to extract the slope of the free surface from SRTM[5] data, but it has not yet been published.
Both software were then translated from Python 2 to Python 3 by the author of this manual. Together with Marta Crivellaro [6], the support for mask generated with Google Earth Engine (GEE from here on) was added.
The up-to-date code can be found at `https://github.com/riccardobonanomi/pyris3`.
This manual is under development and it is not written by the original developer of the software. At the moment, is only describing PyRIS and not PyRISdem. Feel free to report any issue or mistake at `riccardo.bonanomi@unitn.it`.

[1] `f.monegaglia@gmail.com`
[2] `https://landsat.gsfc.nasa.gov/`
[3] `https://github.com/fmonegaglia/pyris`
[4] `https://www.python.org/`
[5] `https://www2.jpl.nasa.gov/srtm/`
[6] University of Trento - Center Agriculture Food Environment (C3A)

Contents

1 Introduction

PyRIS - Python RIvers by Satellite - is a open source software and package based on the freely available programming code Python[4]. PyRIS allows for the analysis of meandering and single threaded rivers morphodynamic characteristics using multitemporal satellite imagery (Monegaglia et al., 2018).
PyRIS allows for the extraction of river axis, channel width, axis curvature and inflection angle from binary masks. The mask can be either produced by the software from Landsat[2] imagery or imported from an external source (such as the ones produced by GEE). Moreover, when two or more axis are available, the software is able to correlate the bends through the analysis of the inflection points and to evaluate the migration vectors. Finally, it is able to detect channel bars and track their migration. For a more in depth review of the code capability refer to the original publication (Monegaglia et al., 2018).

2 Version tracking

In Table 1 we find the history of the various version of the software with a brief description of its main features. Note that it is only accurate for version higher than 3.0.0, the ones developed by the author of this guide.

Table 1: PyRIS version and main new features

Version	Date	Author	Main new features
1.0	2018	Federico Monegaglia	PyRIS deployment
3.0.0	2022	Riccardo Bonanomi	Converted to python 3
3.1.0	2022	Riccardo Bonanomi	Added GEE mask analysis capability
3.1.1	2023/03/20	Riccardo Bonanomi	Added ability to skip masks when segmenting Landsat or importing GEE mask
3.1.2	2023/04/19	Riccardo Bonanomi	Added ability to draw blacks mask on gee mask

3 Installation

To install PyRIS we need the installation of GDAL[7] and of the following Python packages:

- numpy,
- scipy,
- matplotlib,
- scikit-image,
- gdal.

The installation procedure depends by the user operating system. We describe here the procedure for Linux and Windows.

[7] `https://gdal.org/`

3.1 Linux

On Linux PyRIS is a software that can be used from the command line and has been thoroughly tested. Moreover, the ability to merge different Landsat images is only possible in Linux at the moment.
To install PyRIS we first need to install pip with

```
sudo apt-get install python-pip
```

and then GDAL, that can be done following this tutorial[8]:

```
sudo apt-get install python-dev
sudo add-apt-repository ppa:ubuntugis/ppa
sudo apt-get update
sudo apt-get install gdal-bin
sudo apt-get install libgdal-dev
export CPLUS_INCLUDE_PATH=/usr/include/gdal
export C_INCLUDE_PATH=/usr/include/gdal
```

after installing GDAL, in order to install the Python package we need to check the GDAL version with

```
ogrinfo --version
```

and then we can install it with

```
sudo pip install GDAL==<GDAL VERSION FROM OGRINFO>
```

note that the package `wheel` could be needed and it can be installed with

```
sudo pip install wheel
```

We need to install also the other python packages needed for PyRIS with

```
sudo pip install numpy scipy matplotlib scikit-image
```

After this we can download the source code from GitHub with

```
git clone https://github.com/riccardobonanomi/pyris3.git
```

if you have Git[9] or simply downloading the zip from the repository[10].
From the `pyris3` folder we need to launch the command

```
sudo python setup.py install
```

and PyRIS will be installed. To test it we can simply call

```
pyris -h
```

to print out the help message or

```
pyris -i <test filename>
```

to output a standard configuration file named `<test filename>`. The latter approach allows to test the PyRIS module, while the help script only test the existence of the software, but not its functioning. PyRIS can also be installed without admin privilege or in a virtual environment, but if you do not have admin privilege to install GDAL this becomes tricky. The procedure is not yet described here, but feel free to contact the author for help with it.

[8] `https://mothergeo-py.readthedocs.io/en/latest/development/how-to/gdal-ubuntu-pkg.html`
[9] `https://git-scm.com/`
[10] `https://github.com/riccardobonanomi/pyris3.git`

3.2 Windows

On Windows we have two main options, using Anaconda[11] or using a standalone Python[4] installation. The author prefers the latter approach, but the former allows for an easier GDAL installation.

3.2.1 Standalone Python

After installing Python 3 from https://www.python.org/downloads/ and adding it to the path (the Python installer should to do it on its own), we need to install GDAL and its Python package. This is somewhat tricky and OSGeo[12] installation of GDAL do not necessarily work.
At `https://sandbox.oarc.ucla.edu/tutorials/installing-gdal-for-windows` the author found a procedure that worked.
From the website `https://www.gisinternals.com/index.html` we download and install first the core installer and then the Python bindings, for the appropriate version of GDAL and Python that we have. We then need to add the GDAL path (e.g. `C:\Program Files\GDAL`) to the Path System Environment Variable and the path need to be higher than the OSGeo installation, if we have one. Moreover we need to create two more Environment Variable, one named `GDAL_DATA`, pointing to the `gdal-data` folder (e.g. `C:\Program Files\GDAL\gdal-data`), and one named `GDAL_DRIVER_PATH`, pointing to the `gdalplugins` folder (e.g. `C:\Program Files\GDAL\gdalplugins`). To test the installation we can just type in the command prompt

```
ogrinfo --version
```

and the output should match the version we installed.
After installing GDAL, we need to install all the Python packages needed using

```
pip install numpy scipy matplotlib scikit-image
```

from the command prompt.
We then download the PyRIS source code from the repository[10] either with

```
git clone https://github.com/riccardobonanomi/pyris3.git
```

or downloading the zip folder.
From the `pyris3` folder we need to launch the command

```
python setup.py install
```

in the command prompt and PyRIS will be installed.
In Windows however PyRIS is not a software like in Linux, the procedure only installs the PyRIS modules and creates the `pyrisw.py` wrapper script, that acts as the Linux software. To use it we just need to add the `Scripts` folder, in the Python executable folder to the Path and then associate Python as the default software to open `.py` files. Doing so, we will be able to use PyRIS just by typing

```
pyrisw.py
```

in the command prompt. If we do not want to choose Python as the default software we just need to type the path of the `pyrisw.py` script, either in the Python path or the `build` folder in the `pyris3` folder. In this case we just need to type

```
python <path-to-script>\pyrisw.py
```

[11] `https://www.continuum.io/`
[12] https://www.osgeo.org/

in the command prompt like a normal Python script. To test, like for the Linux case, we can just use the `-h` flag to print the help message or the `-i` flag with a test filename to output a standard configuration file. As seen for the Linux case, the latter allows to test also the PyRIS module and not only the main script.

3.2.2 Anaconda

After installing Anaconda from `https://www.continuum.io/downloads`, we need to add Python to the Environment Variables.
We need the to install GDAL running from the Anaconda prompt

```
conda install gdal
```

and then we can install the various packages like for the Linux case with

```
pip install numpy scipy matplotlib scikit-image gdal
```

also from the Anaconda prompt.
After this we can download the PyRIS source code as seen for the Python standalone case and run

```
python setup.py install
```

in the Anaconda prompt from the `pyris3` folder and PyRIS will be installed.
To launch it and test it we can refer to the Python standalone case, just using the Anaconda prompt instead of the command prompt.

4 Input data

4.1 Configuration file

The original code requires a configuration file with the following structure:

```
[Data]
input =
output =
flow_from =
channel_width = None

[Segmentation]
method = MIX
thresholding = global
black_masks = []

[Pruning]
prune_iter = 25

[Axis]
reconstruction_method = std
maxiter = 100000
```

that can be created using the `-i` or `--init` flags followed by the name of the file. The file is not needed when only evaluating the migration rate specifying the axis folder or files.
The `Data` section contains information about the input and output data, in particular we need to specify:

- `input`, the path to the Landsat folder, which is not needed when using the GEE options (`-G` or `-C`);
- `output`, with the output folder that will contain all the PyRIS outputs;
- `flow_from`, to specify the direction from which the water flows, it accepts:
  - `t` = top;
  - `b` = bottom;
  - `r` = right;
  - `l` = left;
- `channel_width`, to specify the minimum river width to consider, it can not be left as `None` as it is initialized when creating the config file; we suggest to choose a lower value than the actual width since nothing below this value will be considered.

The `Segmentation` section contains parameters regarding the river mask segmentation, that is the process that extract the river mask from the Landsat images. The data needed are:

- `method`, to specify the indexes used to segment the image, we can use:
  - `NDVI`, the Normalized Difference Vegetation Index;
  - `MNDWI`, the Modified Normalized Difference Water Index;
  - `BAR`, that is the SWIR, the Short Wave InfraRed Band;
  - `MIX`, that is a combination of the previous indexes and the default option;
  - `AWEI`, not yet implemented;
- `thresholding`, the thresholding method for the chosen index, the available option are;
  - `global`, to apply a global Otsu threshold (default);
  - `local`, to apply a local Otsu threshold, more time consuming;
- `black_mask`, that contains an array of coordinates array for each rectangular area of the image to exclude from the analysis. It can be populate through the `--select-black-mask` flag.

The `Pruning` section only contains the `prune_iter` parameter, that sets the maximum pruning iteration counter (the default value is 25). The `Axis` section contains the parameters used to extract the axis from the mask. It needs:

- `reconstruction_method`, to specify with method to use to choose between branches at a junction, among:
  - `std`, standard method, a mix between length and width constraints (default value);
  - `length`, to choose the branch with the highest length;
  - `width`, to choose the branch with the highest average width;
- `maxiter`, maximum iteration for the axis extraction (the default value is 100000).

4.2 Landsat data

The Landsat data folder has to be specified in the configuration file and it will contain a sub-folder for each image, containing the various bands. The data can be downloaded through the USGS's Earth Explorer[13] and the naming of the subfolder for each year shall remain the one you Earth Explorer gives when you download the data.

4.3 Gee mask files

Using the GEE code developed by Marta Crivellaro[14] is possible to generate `.tif` masks in cloud instead of extracting them from the Landasat data with PyRIS. Up until now the mask from GEE were extracted for permafrost rivers and to be able to do so the code uses the Perona-Malik filter to reduce the noise and analyse the summer envelope. Knowing this PyRIS does not know the precise day and assigns automatically as a day 213, while the year is extracted from the mask name.
PyRIS is able to use already clean masks, where the feature choice has been done externally, in a GIS or other software, and non-clean masks, where the feature choice is done in PyRIS with an identical method as for the Landasat case.
Technically PyRIS is able to ingest whatever `.tif` binary mask, not necessarily generated in GEE, provided they have the correct naming as described further down this guide.

4.3.1 Raw masks

The binary raw masks (not cleaned) need to be in the `.tif` with an UTM projection (as the Landasat are). Moreover, the name of the file has to finish with the year of the mask. They need to be put in a `gee` sub-folder in the output folder or they can be put in an external folder that can be specified with the `--gee-dir=<path-to-mask-dir>` option.

4.3.2 Clean masks

The already cleaned mask also need to be in the `.tif` with an UTM projection, but the name has to finish with the year of the mask followed by `_refined`. The folder can either be the `geeclean` subfolder in the output folder or specified with the `--gee-clean-dir=<path-to-mask-dir>` option. The masks need to be already properly cleaned, removing all the extra features that are not of interest for the analysis.

5 Usage

5.1 Help message and configuration file

To launch the software use the command `pyris` on Linux or `pyrisw.py` on Windows. The help page that follows will appear when using the `-h` flag and shows all the software commands.

```
usage: pyrisw.py [-h] [-i] [--select-black-mask SELECT_BLACK_MASK]
                 [--merge MERGE [MERGE ...]] [--output OUTPUT] [-S]
                 [--label LABEL] [--clean-mask [CLEAN_MASK ...]] [-G]
                 [--gee-dir GEE_DIR] [-C] [--gee-clean-dir GEE_CLEAN_DIR] [-K]
                 [-A] [--mask-dir MASK_DIR] [--no-geo] [-M]
                 [--axis-files AXIS_FILES [AXIS_FILES ...]] [--axis-dir AXIS_DIR]
```

[13] `https://earthexplorer.usgs.gov/`
[14] University of Trento - Center Agriculture Food Environment (C3A)

```
                [--axis-columns AXIS_COLUMNS AXIS_COLUMNS]
                [--output-dir OUTPUT_DIR] [--pfreq PFREQ] [-B] [--show]
                [config]

PyRIS v3.1.1 :: Python - RIvers from Satellite

positional arguments:
  config                input configuration file

optional arguments:
  -h, --help            show this help message and exit
  -i, --init            initialize a config file with default settings
  --select-black-mask SELECT_BLACK_MASK
                        interactively draw black masks from landsat data
  --merge MERGE [MERGE ...]
                        landsat folders to be merged together (requires
                        gdal_merge)
  --output OUTPUT       where merged landsat are to be stored
  --label LABEL         if auto, pyris will try to label masks by itself. other
                        options are max and all
  -S, --segmentation    perform segmentation of all the landsat folders
  -G, --gee-mask-import
                        import gee .tif masks obtained outside PyRIS to be cleaned
  --gee-dir GEE_DIR     dir in which to read ordered-by-name gee .tif mask
  -C, --clean-gee-mask-import
                        import clean gee .tif masks obtained outside PyRIS
  --gee-clean-dir GEE_CLEAN_DIR
                        dir in which to read ordered-by-name gee clean .tif mask
  --clean-mask [CLEAN_MASK ...]
                        manually remove some branches
  -K, --skeletonization
                        skeletonize masks
  -A, --axis            extract main channel axis from channel mask
  --mask-dir MASK_DIR   directory containing mask files
  --no-geo              do not georeference mask files
  -M, --migration       compute migration rates of subsequent axis
  --axis-files AXIS_FILES [AXIS_FILES ...]
                        ordered sequence of files for which the migration rates
                        must be computed
  --axis-dir AXIS_DIR   dir in which to read ordered-by-name axis files (if
                        --axis-files is not specified)
  --axis-columns AXIS_COLUMNS AXIS_COLUMNS
                        columns containing axis coordinates
  --output-dir OUTPUT_DIR
                        directory where outputs fof migration rates will be stored
  --pfreq PFREQ         an interpolation step between points to be used as a
                        frequency filter (a few channel widths in pixels).
  -B, --bars            compute the position of channel bars
  --show                show results after computing
```

To create an empty configuration file, as described in the input section, use the `-i` or `--init` flag followed by the file name. The configuration file is needed for each step minus the manual cleaning of masks and the migration evaluation from a specified axis folder.

5.2 Draw black mask

PyRIS let you draw areas to exclude from the mask extraction for Landsat data and GEE raw masks using `--select-black-mask=<path-to-landsat-dir-or-to-gee-mask`. We need to specify the path to the Landsat folder for one image or for one GEE mask (with the file extension) that is representative, since the mask will be drawn one and for all. The command opens a display of the chosen data where the black mask can be drawn interactively, drawing the rectangles to exclude holding the left click while moving the pointers and for every area draw the local coordinates will be printed on the terminal; when you finish the selection simply close the display and the mask will be saved. The command does not modify the files, but it saves in the configuration files (that must be specified) the coordinates of the excluded areas under the `black_mask` entry of the `[Segmentation]` section, that can also be modified from the text editor.

5.3 Merge

The merge option uses the GDAL command `merge` to join more than a Landsat image for each year. It is used specifying `--merge=<path-to-landsat-to-merge>`. The output folder needs to be specified with `--output=<path-to-output-folder>`. This features requires GDAL merge and as for now only works on Linux, but the author is working on extending it also to Windows.

5.4 Segmenting Landsat data and GEE mask importing

Since the output data of this processes are the same, but the procedures are a little different in three cases, we will first present the three commands and then we will speak about the output files. If a mask is already been extracted from a specific year the software will skip the image, unless the mask and geotransf file are deleted.

5.4.1 Landsat data

With the flag `-S` or `--segmentation`, with the configuration file, PyRIS will begin the segmentation of the Landsat data and the mask extraction, with the data and the option specified in the configuration file, as seen in subsection 4.1. The mask are extracted using the index or indexes specified in the `method` entry of the configuration file and it uses a global or local thresholding using the specified option in the `thresholding` The software also applies filters to clean holes in the planform and to remove small objects using the width specified in the `channel_width` entry as base measure for the various filters and area removal used.
After this process the user need to select the mask features to analyse. With the `--label=<option>` the user is able to specify the optional automatic mask choice methods. The available options are:

- `auto`, to use the `max`, option when using local thresholding (specified in the configuration file), and `all`, when using global thresholding;
- `all`, to choose every feature present in the mask;
- `max`, to choose the largest feature present in the mask.

If the label option is not specified, a display prompt will appear for each Landsat image and the user will see all the features available labelled with a number and after closing the prompt PyRIS will ask to enter in the terminal the feature number to select, separated with a space if the user wants multiple features. The choice of multiple features still has some issues, but the author is working to solve them. After choosing the feature the software will fill hole in the mask and remove small objects, to clean the

mask.
The segmentation of the Landsat data is the PyRIS command that has the higher resources load, especially on the RAM, and we started using GEE also to avoid this issue.

5.4.2 Raw gee masks

Using the `-G` or `--gee-mask-import` flags PyRIS will import the gee mask and extract the features that can be chosen as for the Landsat case: the `--label=<option>` flag works also for this case and, if no automatic mask choice is used, the user will choose the features to analyse for in the same way as per the Landsat case. As in the Landsat case the mask will then be cleaned removing small objects and filling holes.

5.4.3 Clean gee masks

Using the `-C` or `--clean-gee-mask-import` flags PyRIS will import the clean gee mask, but, since the non needed features, have already been removed, the user will not be able to choose the features or use the `--label=<option>` flag. As in the Landsat case the mask will then be cleaned removing small objects and filling holes.

5.4.4 Output files

Whichever road you choose, all this commands will create a geotransf file and a mask file in their respective folders (folders structure, file naming and file contents will be described in section 6.

5.5 Manual mask cleaning

After the user creates the mask file, the `--clean-mask` allow for manual cleaning of each mask file. With `--clean-mask=<path-to-mask-to-clean,path-to-mask-to-clean>` the user can choose which masks to clean specifying their path or they can choose to cleaning all the mask in the `mask` folder using just the `--clean-mask` flag without any path. This command will open a display prompt where each mask is shown and the user can drag, while left clicking, selecting the areas to remove from the mask. After closing the prompt, the coordinates of the removed areas will appear in the terminal, followed by the message `overwrite mask file?[y/n]`, that lets the user choose whether to apply this changes or not. This is particularly useful to speed up the axis extraction in rivers with high number of branches that needs to be removed. Be aware that this process modifies the mask file directly and to revert it you will need to extract the mask from the Landsat data or from the GEE masks.

5.6 Skeletonization

The river skeleton can be extracted from the mask using the `-K` or `--skeletonization` flag and saves it in the `skeleton` folder. The skeletonization processes prunes spurs from the river and depending by the number of this spurs the pruning process can take a while. Each pruning iteration is printing on the terminal for each of the features found in the mask. Be careful when you manually clean the mask, since if you delete completely a feature the software will give an error in this stage. This command will create a skeleton file, described in section 6. If the software detects more than 15 junction it will prompt the following warning:

```
’’’
’Warning!’
’Expected at least <Number-of-junctions> recursion level.’
```

```
'Consider increasing the pruning iteration or calling pyris with the
--clean-mask flag'
'''
```

since it can lead to slow axis extraction or even non-correct paths choice when close to the domain edges. This warning will appear even if the pruning had plenty of iteration and stopped before the maximum number of pruning iterations inputted by the user in the `prune_iter` entry of the configuration files.
The user can also specify the mask folder with `--mask-dir=<path-to-mask-dir>`. If a particular mask has already been skeletonized previously the software will skip it, unless the skeleton file is deleted.

5.7 Axis extraction

From the skeleton and the mask PyRIS extracts the river axis and is inflection angle, curvature and width using the `-A` or `--axis` flag. For each feature selected in the segmentation process the software will search for a primitive where to put the beginning of the axis and it will extract the axis choosing for each junction which branch to keep based on the `reconstruction_method` specified in the configuration file with one of the three options described in section 4.1. The software iterates until it reaches the end of the axis or the `maxiter` parameter specified in the configuration file. The axis is then added to the previous feature axis and when all the feature of the mask are analysed, the axis is interpolated with a cubic spline calculating also the inflection angle, the curvature and the width on the new interpolated axis. With multiple features the software does not work properly, as said earlier, because it often does not does not start from the correct side for each feature, but the author is working to fix the issue.
With the `--mask-dir=<path-to-mask-dir>` the user can specify the position of the mask folder and with the `--no-geo` the mask will not be georeferenced. If the mask are not georeferenced every output data is in pixel (the same units as the one in the mask files). The command will save an axis file for each year in the `axis` folder, that will be described in section 6. If a particular axis has already been extracted the software will skip it, unless the axis file is deleted.

5.8 Migration evaluation

Using the `-M` or `--migration` flag the software correlates the bends between two consecutive planimetries and evaluates the migration rate for each point for the bend that it manages to correlate. The migration evaluation is the only step where the software overrides already existing data, but it is usually not a problem since the process is really fast compared to the other steps. The user can specifies the individual files (in the `txt` or `npy` format) to use to perform the calculation indicating the ordered sequence of filenames with `--axis-files=<axis-file-1,axis-file-2>` and the folder where to find the axis files with `--axis-dir=<path-to-axis-dir>`, that will be analysed ordered by name, if no axis file is specified. When specifying files it is possible also to specify which columns are the x and y ones to use for the computation with `--axis-columns=<x-column,y-column>`.
The user can also specify the output directory with `--output-dir=<path-to-output-dir>` and the smoothing coefficient for the channel with `--pfreq=<pfreq>`. The value needs to be 1 or greater, with 1 being the default option, that means no smoothing applied. The smoothing is performed with a Inverse Continuous Wavelet Transform (ICWT hereinafter).
The software will save a file for each year containing the correlated bend indices and the migration rates, the actual structure of the file and the data will be described in section 6. The migration is always saved in the first year of the two, so the last file will have just `NaN` as migration rates.
The software is also able to show the results of the migration evaluation with the option `--show`.

Keep in mind that the correlation process and the migration evaluation is not always successful, especially with fast varying meandering rivers or with a low number of planimetries over long time intervals.

5.9 Bar extraction

With the `-B` or `--bars` the software will extract the position of the sediment bars and track their migration. This function computes the free bars evolution in the river, but in order to do so needs to calculate the satellite image indexes once again and so it needs the presence of the Landsat data in the folder specified in the configuration file and it can not be used on the data extracted with GEE. The feature is still experimental and it has not been tested yet by the author of this guide.
With the `--show` flag the software will show the output results of the bars extraction.

6 Output

PyRIS produced various output files for each command. The file are all named with the name of the planimetry they represent using the year and the day of the year (`<year>_<day-of-year>.<ext>`) and they are each saved in a folder named with the type of the file. All the files are Python and Numpy[15] specific files (`.npy` and `.p`) to allow for faster calculation, but they can easily be converted to readable formats using simple Python commands.
The file produced by each command are:

- a `mask` file and a `geotransf` file;
- a `skeleton` file;
- a `axis` file;
- a `migration` file, produced by the migration evaluation;
- a `bendsep`, an `interpolation`, a `values` file and a `barcorr`, in the `bars` folder, that are a single file for each river, and not one per planimetry.

The description of the content of each file can be found in the next subsections, divided by the command that produced it.

6.1 Mask and geotransf files

A `mask` and a `geotransf` for each planimetry file are produced for each planimetry by the segmentation of the Landsat data or the extraction of the features from the GEE masks.
The `geotransf` file is a pickle (`.p`) file which contains the information needed to switch from the internal reference system of the image in pixel to the UTM reference system that the original data was in and allows for conversion in metric units of the outputs. It contains a dictionary with the following structure.

```
{'PixelSize': <PixelSize-value>, 'X': <X-value>, 'Y': <Y-value>,
'Lx': <Lx-value>, 'Ly': <Ly-value>}
```

[15] https://numpy.org/

The 'PixelSize' field contains the pixel width, the X and Y are the x and y coordinates of the the upper-left corner of the upper-left pixel, and the Lx and Ly are the number of pixels in the x and y directions.
The mask file is a .npy file that contains a matrix with the shape of the original data (Lx x Ly pixels) populated with 1 for each pixel of chosen feature and 0 for each pixel that does not contains it.

6.2 Skeleton file

A skeleton file for each planimetry is produced by the skeletonization process and is also a .npy file containing a matrix with the same shape as the original data. The matrix contains the skeleton of the mask, so the centreline pixel of each feature, numbered with a integer different for each separate feature, while the background has value 0.

6.3 Axis file

An axis file for each planimetry is produced by the axis extraction process and is a .npy file containing a table with one row for each planimetry point. Each planimetry points has 8 entries that are, in order:

- x, the geo-referenced x coordinate;
- y, the geo-referenced y coordinate;
- s, the along the axis coordinate, in meters;
- θ, the inflection angle of the centreline;
- C_s, the curvature of the centreline;
- B, the channel half width;
- x_p, the local x coordinate, using the pixel as unit of measure;
- y_p, the local y coordinate, using the pixel as unit of measure.

If the file are generated using the --no-geo option all the variables will be in the local coordinate system, using the pixel as unit of measure.

6.4 Migration file

A migration file for each planimetry is produced by the migration command and is a .npy file containing a table with one row for each planimetry point. For each point we have 7 entries:

- dx, the x component of the migration vector between this planimetry and the next one;
- dy, the y component of the migration vector between this planimetry and the next one;
- dx, the module of the migration vector between this planimetry and the next one;
- $ICWTC$, the curvature filtered using the ICWT filter;
- BI, a progressive index for each bend of this planimetry, each point of a bend will have the same value, greater or equal than 0, and the point before the first inflection point and after the last inflection point will be equal to -1;
- $B12$, the index of the correlated bend in the next planimetry;

- *BUD*, the index of the position of the point in the bend, this can have values:
  - 0 = the point is the bend apex;
  - -1 = the point is upstream of the bend apex;
  - +1 = the point is downstream of the bend apex;
  - 2 = the point is an inflection point.

Since the migration is evaluated between one planimetry and the next one, the last one will have `NaN` values in all the migration vector components and -1 in all the $B12$ values. The point before the first inflection point or after the last will also have `NaN` in all the migration vector components. The migration vector are measured in the same units as the axis, so they are in meters if the data is georeferenced, in pixel if the `--no-geo` options was used or in the axis unit of measure for user inputted axis files. Since they are vector, to evaluate the rate the user will need to divide the vector components by the time between two planimetries.

6.5 Bendsep, interpolation and values files

The bar detection command produces a total of four `.npy` files, the `bendsep` file, the `interpolation` file, the `values` file and the `barcorr`. Since the author of this guide did not test this function the content of the files are not yet described here.

7 Final remarks

References

Monegaglia, F., G. Zolezzi, I. Güneralp, A. J. Henshaw, and M. Tubino (2018). "Automated extraction of meandering river morphodynamics from multitemporal remotely sensed data". In: *Environmental Modelling & Software* 105, pp. 171–186. DOI: 10.1016/j.envsoft.2018.03.028.