centerline

Release 1.0.1

Filip Todić

Feb 11, 2023

CONTENTS:

Roads, rivers and similar linear structures are often represented by long and complex polygons. Since one of the most important attributes of a linear structure is its length, extracting that attribute from a polygon can prove to be more or less difficult.

This library tries to solve this problem by creating the the polygon's centerline using the Voronoi diagram.

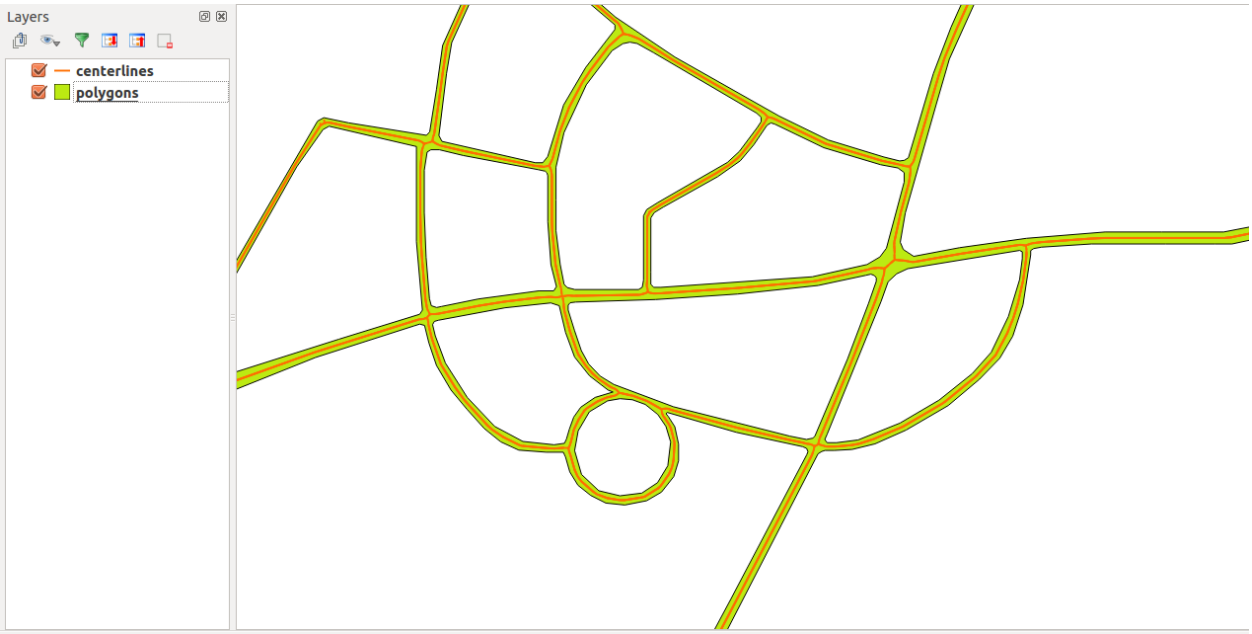

Fig. 1: The source and the output geometry visualized in QGIS.

CHAPTER

ONE

FEATURES

- A command-line script for creating centerlines from a vector source file and saving them into a destination vector file: `create_centerlines`

```
$ create_centerlines input.shp output.geojson
```

- The `Centerline` class that allows integration into your own workflow.

```
>>> from shapely.geometry import Polygon
>>> from centerline.geometry import Centerline

>>> polygon = Polygon([[0, 0], [0, 4], [4, 4], [4, 0]])
>>> attributes = {"id": 1, "name": "polygon", "valid": True}

>>> centerline = Centerline(polygon, **attributes)
>>> centerline.id == 1
True
>>> centerline.name
'polygon'
>>> centerline.geoms
<shapely.geometry.base.GeometrySequence object at 0x7f7d24116210>
```

1.1 Installation

If you want to use the `create_centerline` command-line tool, you need to install GDAL and the corresponding python-GDAL version. `GDAL` is a translator library for raster and vector geospatial data formats whose binary needs to be installed system-wide, whereas `python-GDAL` should be installed into a virtual environment.

1.1.1 Installing GDAL

GDAL binary

Linux

If you are using Linux, the GDAL library is probably already located in one of your distribution's repositories. If so, you can install it using your distribution's package manager, along with the other necessary dependencies.

Note: The names of packages may vary between distributions.

If you are using Fedora, run the following command:

```
$ sudo dnf install gdal gdal-devel gcc-c++ redhat-rpm-config
```

python-GDAL

Once installed, locate the GDAL's headers and set the *include* path to the `CPLUS_INCLUDE_PATH` and `C_INCLUDE_PATH` environment variables:

```
$ whereis gdal
gdal: /usr/include/gdal /usr/share/gdal

$ export CPLUS_INCLUDE_PATH=/usr/include/gdal/
$ export C_INCLUDE_PATH=/usr/include/gdal/
```

After that, you can proceed to installing GDAL in the virtual environment (i.e. `python-GDAL`). Please not that the version of python-GDAL installed in the virtual environment **should correspond as much as possible** to the version of the system-wide installation as much as possible. For instance, if the system-wide installation is 2.1.4, and there is no matching Python library, feel free to install the closest *minor* version (e.g. 2.1.3):

```
$ gdalinfo --version
GDAL 2.1.4, released 2017/06/23

# Activate your virtual environment
$ pip install GDAL==2.1.3
```

See also:

For more info, visit Stack Exchange.

1.1.2 Installing centerline

You can download and install the package from PyPI using pip:

```
$ pip install centerline
```

GDAL can be installed either directly (see *python-GDAL*) or by specifying the `gdal` extra dependency:

```
$ pip install centerline[gdal]
```

Warning: `pip install centerline[gdal]` can be error-prone because multiple GDAL versions are supported and `pip` will automatically try to retrieve the latest version which you may or may not have installed system-wide.

1.1.3 Development

If you want to contribute to this library, apart from installing GDAL, you have to:

1. fork and clone the repository:

```
$ git clone git@github.com:user/centerline.git
```

2. install the library in develop mode:

```
$ pip install -e .[dev,gdal,lint,test,docs]
```

3. run the test suite to make sure everything is in order:

```
$ tox
```

1.2 Usage

1.2.1 Command-line interface

This library provides the `create_centerlines` command-line script for creating centerlines from a vector source file and saving them into a destination vector file. The script supports all OGR's vector file formats which enables conversion between different formats:

```
$ create_centerlines input.shp output.geojson
```

1.2.2 Python

If you want to use the `Centerline` class directly, you can import it and instatiate it with geometric data (of type `shapely.geometry.Polygon` or `shapely.geometry.MultiPolygon`) and the object's attributes (optional):

```
>>> from shapely.geometry import Polygon
>>> from centerline.geometry import Centerline

>>> polygon = Polygon([[0, 0], [0, 4], [4, 4], [4, 0]])
>>> attributes = {"id": 1, "name": "polygon", "valid": True}

>>> centerline = Centerline(polygon, **attributes)
>>> centerline.id == 1
True
>>> centerline.name
'polygon'
>>> centerline.geometry.geoms
<shapely.geometry.base.GeometrySequence object at 0x7f7d24116210>
```

1.3 API

1.3.1 Module contents

1.3.2 Submodules

1.3.3 centerline.exceptions module

exception centerline.exceptions.**CenterlineError**(**args*, ***kwargs*)

Bases: Exception

default_message = 'An error has occured while constucting the centerline.'

exception centerline.exceptions.**InvalidInputTypeError**(**args*, ***kwargs*)

Bases: *CenterlineError*

default_message = 'Input geometry must be of type shapely.geometry.Polygon or shapely.geometry.MultiPolygon!'

exception centerline.exceptions.**TooFewRidgesError**(**args*, ***kwargs*)

Bases: *CenterlineError*

default_message = 'Number of produced ridges is too small. Please adjust your interpolation distance.'

exception centerline.exceptions.**UnsupportedVectorType**(**args*, ***kwargs*)

Bases: *CenterlineError*

default_message = 'No OGR driver was found for the provided file.'

1.3.4 centerline.converters module

centerline.converters.**get_ogr_driver**(*filepath*)

Get the OGR driver based on the file's extension.

Parameters
filepath (*str*) – file's path

Raises
UnsupportedVectorType – unsupported extension

Returns
OGR driver

Return type
osgeo.ogr.Driver

1.3.5 centerline.geometry module

class centerline.geometry.**Centerline**(*input_geometry*, *interpolation_distance=0.5*, ***attributes*)

Bases: object

Create a centerline object.

The `attributes` are copied and set as the centerline's attributes.

Parameters

- **input_geometry** (shapely.geometry.Polygon or shapely.geometry.MultiPolygon) – input geometry
- **interpolation_distance** (*float, optional*) – densify the input geometry's border by placing additional points at this distance, defaults to 0.5 [meter]

Raises

exceptions.InvalidInputTypeError – input geometry is not of type shapely.geometry.Polygon or shapely.geometry.MultiPolygon

assign_attributes_to_instance(*attributes*)

Assign the `attributes` to the *Centerline* object.

Parameters

attributes (*dict*) – polygon's attributes

input_geometry_is_valid()

Input geometry is of a shapely.geometry.Polygon or a shapely.geometry.MultiPolygon.

Returns

geometry is valid

Return type

bool

1.4 Frequently Asked Questions

1.4.1 QH6214 qhull input error: not enough points to construct initial simplex

This error occurs when there is an inconsistency in units. For example, the input data is in EPSG:4326 which uses degrees, whereas the `create_centerline` script's default border density is in meters.

There are two possible solutions to this problem:

1. Specify the border density in degrees

```
$ create_centerlines input.shp output.geojson 0.00001
```

2. Transform the input data into a metric system (e.g. EPSG:3857)

1.4.2 The level of detail is too high/The geometry is too complex

Adjust the `create_centerline`'s border density parameter. The script is ment to serve a wide range of applications, some of which require a higher level of detail.

Since the Voronoi diagram is used to calculate the centerline's geometry, that means that depending on the specified border density occasionaly the centerline will have a few branches sticking out. The branches can be removed manually using QGIS.

1.4.3 Number of produced ridges is too small

Depending on the polygon's structure and the border density parameter set by the user, the Voronoi algorithm might not be able to produce enough vertices that lie completely inside the polygon (i.e. do not intersect the polygon's boundary). In that case, the output centerline can be reduced to a single line or even a single point.

When the centerline consists of less than two lines, the `TooFewRidgesError` is raised, the `create_centerline` script skips the geometry in question and continues processing the other geometries.

1.5 Contributing

To contribute to this project, fork it, clone it and install it in development mode:

```
$ git clone git@github.com:fitodic/centerline.git
$ pip install -e .[dev,gdal,lint,test,docs]
```

The most important dependency for development is tox. It is used for running the test suite, building the documentation and changelog, validation (linting, manifest and PyPI description) and creating a new project release. To validate it is successfully installed, run:

```
$ pip show tox
Name: tox
Version: 3.12.1
Summary: tox is a generic virtualenv management and test command line tool
Home-page: http://tox.readthedocs.org
Author: Holger Krekel, Oliver Bestwalter, Bernát Gábor and others
Author-email: None
License: MIT
Location: /home/username/.virtualenvs/centerline/lib/python3.7/site-packages
Requires: py, filelock, virtualenv, setuptools, six, pluggy, toml
Required-by:
```

1.5.1 Tests

To run the test suite, run:

```
$ tox
```

or more specifically:

```
$ tox -e py37-gdal2.3.3
```

1.5.2 Changelog

This project uses towncrier for changelog management. You don't need to install it locally since you'll be using it through `tox`, but please adhere to the following rules:

1. For each pull request, create a new file in the *changelog.d* directory with a filename adhering to the *#pr.(feature|bugfix|doc|removal|misc).rst* schema. For example, *changelog.d/23.bugfix.rst* that is submitted in the pull request 23. `towncrier` will automatically add a link to the note when building the final changelog.
2. Wrap symbols like modules, functions, or classes into double backticks so they are rendered in a monospace font.
3. If you mention functions or other callables, add parentheses at the end of their names: `func()` or `Class.method()`. This makes the changelog a lot more readable.

If you have any doubts, you can always render the changelog to the terminal without changing it:

```
$ tox -e changelog -- --draft
```

1.5.3 Documentation

To build the documentation, run the following command:

```
$ tox -e docs
```

The same command is used for building the documentation on readthedocs.org.

1.5.4 Releasing a new version

The CI environment should build packages and upload them to the PyPI server when a tag is pushed to *origin*. Therefore, if you want to make a new release, all you have to do is run the *release* environment in *tox*:

```
$ tox -e release
```

This environment will merge the changelogs from the *changelog.d* directory into `CHANGELOG.rst`, bump the **minor** version (by default) using bumpversion, commit the changes, create a tag and push it all to *origin*.

If you want to make a **patch** release, run:

```
$ tox -e release -- patch
```

If Travis CI builds were successful, the new release should be automatically uploaded to PyPI.org.

1.6 Changelog

The format is based on Keep a Changelog and this project adheres to Semantic Versioning.

Changes for the upcoming release can be found in the *changelog.d* directory in this repository. Do **NOT** add changelog entries here! This changelog is managed by towncrier and is compiled at release time.

1.6.1 1.0.1 (2023-01-23)

Misc

- Restore Python 3.7+ support and update metadata. (#39)

1.6.2 1.0.0 (2023-01-07)

Features

- Add support for Shapely 2 (#30)

Deprecations and Removals

- – The *Centerline* class will no longer extend the Shapely's *MultiLineString* because *Centerline* sets custom attributes which will be prohibited starting from Shapely 2.0. You can use *centerline.geometry* instead of *centerline* wherever you need access to the object's geometry.
  – Drop Python 2.7 support (#30)

1.6.3 0.6.4 (2022-09-17)

Bugfixes

- Fix shapely 2 deprecation warning for iterating multipolygons (#34)

1.6.4 0.6.3 (2020-01-02)

No significant changes.

1.6.5 0.6.2 (2020-01-02)

Misc

- Update the CI deploy stage. (#26)

1.6.6 0.6.1 (2020-01-02)

Misc

- Run the test suite against Python 3.8. (#25)

1.6.7 0.6.0 (2019-06-25)

Deprecations and Removals

- Replace argparse with Click. This should improve Windows support of the `create_centerline` command-line script. (#23)

Misc

- Convert the package to the src/ layout and the tests to pytest. The modules have been renamed, and the `Centerline` class has been refactored to enable overrides. (#23)

1.6.8 0.5.2 (2019-01-27)

Fixed

- Package versioning that caused a broken upload

1.6.9 0.5.1 (2019-01-27)

Changed

- Set the minimum `GDAL` version to 2.3.3

Fixed

- Drop the `path` keyword argument from `fiona.open` calls #20.

Removed

- Python 3.5 support

1.6.10 0.5.0 (2018-09-09)

Added

- `MultiPolygon` support

1.6.11 0.4.2 (2018-08-22)

Added

- `GDAL` 2.3.1 to the CI configuration

Changed

- Moved the `coverage` configuration to `setup.cfg`
- Moved the package's metadata to `setup.cfg`

Fixed

- Error when `MultiLineString` degenerates into `LineString` (#14). Thanks mxwell!

Removed

- MANIFEST.in
- **`Centerline` from the `centerline` namespace. To import the `Centerline`** class, use `from centerline.main import Centerline`

1.6.12 0.4.1 (2018-01-07)

Fixed

- Ignore the `osgeo` package when building the documentation on readthedocs.org.

1.6.13 0.4.0 (2018-01-07)

Added

- Sphinx documentation

Fixed

- Add a comma to the list of development requirements

1.6.14 0.3.0 (2017-11-26)

Added

- `pylama` and `isort` configuration
- `pylama` and `isort` checks in the Travis build
- `utils` and `io` modules
- `create_centerlines` script and function for creating centerlines that is format agnotic. All OGR vector file formats should be supported.

Changed

- The `Centerline` class extends Shapely's `MultiLineString` class
- Replaced the `shp2centerline` script with `create_centerlines`

Removed

- Support for `GDAL<2.0`
- Support for `Fiona<1.7`
- `shp2centerline` script

1.6.15 0.2.1 (2017-06-18)

Fixed

- Read the `README.rst` from `setup.py`

1.6.16 0.2.0 (2017-06-18)

Added

- `CHANGELOG.md`
- `.coveragerc`
- Travis CI configuration
- Test and package configuration in `setup.cfg`
- Use `pytest` for test execution
- Test the import of the `Centerline` class

Changed

- `MANIFEST.in`
- `.gitignore`
- Reorganize the project's requirements (both in `*.txt` files and `setup.py`)
- Fix PEP8 errors in `setup.py`
- Convert README from MarkDown to ReStructuredText

1.6.17 0.1.0 (2016-01-15)

Added

- The `Centerline` class
- The logic for calculating the centerline of a polygon
- The `shp2centerline` command for converting polygons from a Shapefile into centerlines and saving them into another Shapefile

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

C

INDEX