Coastline generator: A tool for generating topographic tessellations around polygons and lines

Bruno Lattanzio[1 [0000−0001−5380−0915]], Dalponte Ayastuy María[1,2 [0000−0002−1412−5694]] and Diego Torres[1,2 [0000−0001−7533−0133]]

[1] Depto CyT, Universidad Nacional de Quilmes
Roque Saenz Peña 352, Bernal, Buenos Aires, Argentina.
{brunoj.lattanzio,mdalponte}@unq.edu.ar
[2] LIFIA,CICPBA-Facultad de Informatica, Universidad Nacional de La Plata
50 y 120, La Plata, Buenos Aires, Argentina.
diego.torres@lifia.info.unlp.edu.ar

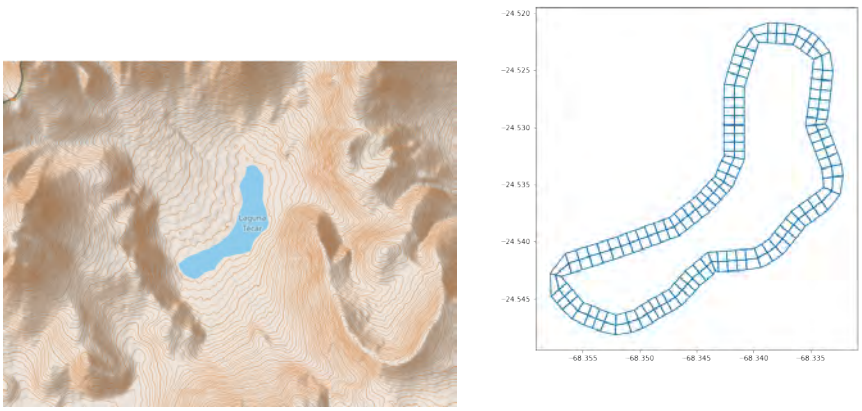
Abstract. Many citizen science projects that carry out survey tasks based on location require that the territory to be studied be fragmented into smaller areas with the objectives of, on the one hand, keeping a record of the level of coverage of the regions, and on the other hand, to present spatially bounded objectives to the volunteers of the project. In some cases the sampling areas are related to a terrain feature, such as when surveying the shores of rivers and lakes. Therefore, the aforementioned segmentation must respect the topographical shape of the geographical object to be studied (river or lake).In this work, this type of tessellation is defined as **topographic tessellation** (TT). Aiming at building the TT, indicating the distance it should have from the shore and the specific measurements of each smaller area is needed. This article presents a framework for the automatic generation of topographic tessellations, which are sets of disjoint and adjacent polygons that form a mosaic following the shape of a georeferenced geometry, and builds a new geographical layer. This tool is useful for spatial-task asignment decision-making.

1 Introduction

Collaboration between professional scientists and volunteers committed to research is not new. Actually, the recognition of the scientific task as a paid profession is a fact whose antiquity does not exceed two centuries and previously it was carried out by amateur scientists. However, these citizens who do science have not disappeared and their observation skills are very useful in astronomical, natural sciences or archaeology projects, among others. Today formal scientists work in teams with citizens scientists on projects that have been specifically designed to give amateurs a role for the benefit of the project's objectives[7].

Citizen science encompasses a range of methodologies that encourage and support the contributions of the public to the advancement of scientific and

Received April 2023 Accepted July 2023; Published August 2023



(a) (b)

Fig. 1. (a) Lake polygon and (b) topographic tessellation

engineering research and monitoring in ways that may include co-identifying research questions; co-designing/ conducting investigations; co-designing/ building/ testing low-cost sensors; co-collecting and analysing data; co-developing data applications; and collaboratively solving complex problems [9]

Usually, each CLCS sets its territory coverage objectives, such as, for example, guaranteeing that all the affected territory is surveyed [1]. This leads to the discussion about what level of granularity or precision is needed. A possible approach to specify the needed precision, as a first step requires to divide the territory into smaller areas, which allows indicating how many samples are needed in each of them. Aiming at this segmentation, specific cartography is needed to delimit the project's sampling areas and, in particular to the AppEar project, this set of areas are located around the shores of rivers, lakes, lagoons and estuaries.

In order to pursue its objectives, each project needs to dynamically define the tasks that it must present to volunteers based on the current situation of the project, and to assist in this sense (decision making) it is necessary to have a model of the space to be surveyed with a certain level of granularity where the activity of people can be registered.

Therefore it is necessary to generate a disjoint and adjacent set of areas -a tessellation- that are located following the shape of a geometry. Moreover, this areas must be organized in a regular shape grid where the number of rows and the dimension of its cells. In this work, this kind of tessellation is defined as Topographic Tessellation (see image 1 b). In the case of a river, the geometry that represents it can be a line or a polygon -depending on river's width-, and in the case of a lake or lagoon, the geometry is a polygon (see image 1 a).

Received April 2023 Accepted July 2023; Published August 2023

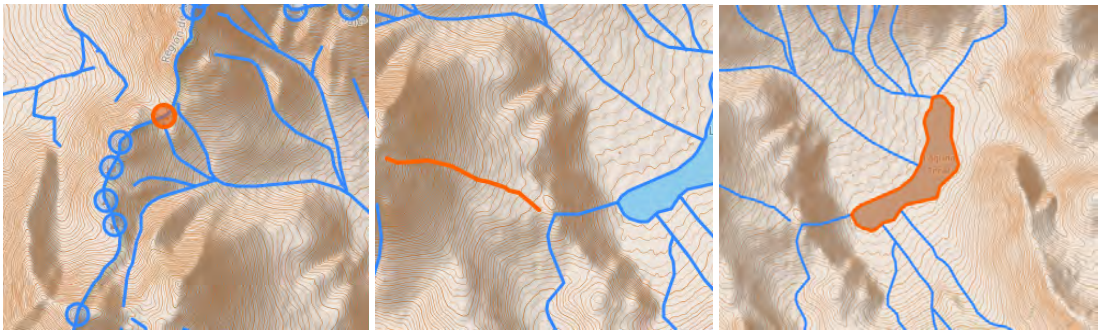
The discipline of geographic information systems (GIS) defines and uses different tessellated models to represent information about the earth[4], sometimes to build a hierarchical discretization of a high definition raster image, or to support a vector representation of terrain surface, such as the Voronoi regions [5] or the Delaunay triangulation [8].

No articles directly related to the problem presented in this work were found, but there are works that address similar problems by calculating the Voronoi regions from a set of centroids, as is the case of Fleischman et al. [3] and Du and Gunzberg [2]. These approaches are not directly applicable in this domain because it requires a set of centroids and a bounding area to be defined. Without denying that this problem could be thought of in those terms, the definition of the centroids would represent additional and artificial steps.

Therefore, this paper presents and details a framework capable of generating a *Topographic Tessellation (TT)* following the shape of a line or a polygon. Also, the presented approach of cartographic generation allows to customize the grid's dimensions (cells width and row count) to be adaptable to the project's needs.

This article is organized as follows. In section 2 the background and domain-specific concepts are defined. In section 3 the system architecture and topographic tessellation algorithm is detailed. Finally the section 4 shares discussions and future work.

2 Background: Topographic tessellation



(a) Point element (b) Multiline element (c) Polygon element

Fig. 2. GIS elements examples

In this work, since cartography of the Argentine territory is used, provided by the National Geographic Institute [3] (NGI), it is appropriate to use the WGS84 reference system, which is the same used by the NGI. The hydrographic data that were used were obtained from the GIS layer of the NGI, in the section of

[3] https://www.ign.gob.ar/NuestrasActividades/InformacionGeoespacial/CapasSIG

Received April 2023 Accepted July 2023; Published August 2023

hydrocography and oceanography, subindex of *continental waters*. These data are available in GeoJSON, an open standard format designed to represent geographical elements together with its non-spatial attributes, based on JSON. Some of these non-spatial attributes are described in Table 1.

Field	Type	Description
Entidad	Integer	Object ID
Objeto	String	Polygon subtype: Channel, waterflow, reservoir, perennial water mirror, rural reservoir, wetland, intermittent water mirror. Line type: Aqueduct, Canal, Perennial water currents, intermittent water currents, waterfall waterfall jump, reservoir wall, ditch.
FNA	String	Complete name of the geographical object
FDC	String	Id of the name and type of source used to capture the information. May include date and other additional data

Table 1. Input dataset metadata

Two methods are used to represent geographic phenomena in ways that can be encoded in spatial databases, called raster and vector methods. Although both can be used to encode both continuous fields and discrete objects, in practice there is a strong association between raster and fields, and between vector and discrete objects. One of the most common forms of raster data comes from remote-sensing satellites, which capture information as high definition images. In a vector representation, all lines are represented as points connected by a straight line, and areas are captured as a series of points or vertices connected by straight lines, called polygons. Lines are represented in the same way, and the term polyline (or multiline) has been coined to describe a curved line represented by a series of straight segments connecting vertices [6].

Definition 1 (Point). *Geographic coordinate defined by the tuple (latitude, longitude) of gegraphic coordinates. Negative values in latitude refer to points in the southern hemisphere, and negative values in longitude refer to points located west of the Greenwich meridian.*

As an example, see Figure 2 (a) where a point with latitude: -24,5073190, and longitude: -68,3958578-57,9380456 is shown.

Definition 2 (Multiline). *A multiline object is an sequence of more than two points.*

In Figure 2 (b) a multiline example is shown, made up with twenty nine points.

Definition 3 (Polygon). *A polygon is a multiline, where the first point is also the last one, making a closed geometry.*

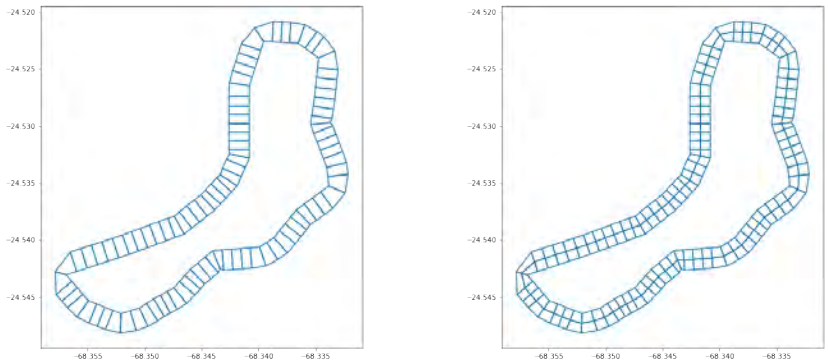
The last example in Figure 2 is a polygon with 63 points.

Received April 2023 Accepted July 2023; Published August 2023

Definition 4 (Topographical Tessellation). *Set of adjacent and disjoint cells that are organized following the shape of a given geometry. In other words, the upper limit of the mosaic is the geometry's limit, and the lines that separate the cells' rows are parallel to that limit.*

Two examples of Topographical Tessellations are shown in Figure 3.

3 Approach



(a) TT with one row (b) TT with two rows

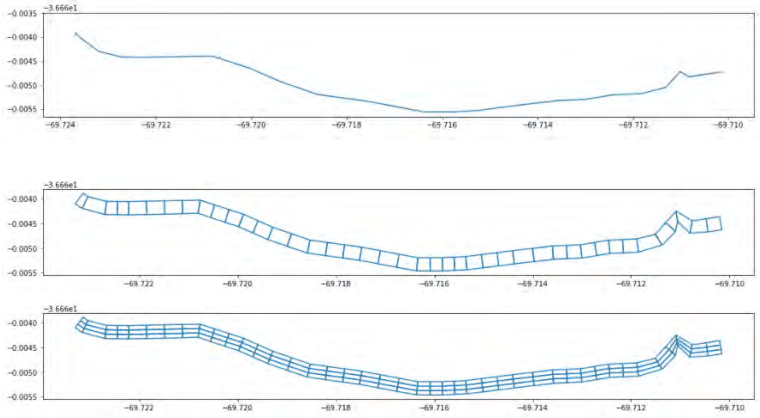
Fig. 3. Topographical tessellations for a lake

As presented in the introduction, the task of surveying the state of the coasts of rivers and lakes must be associated with a cell of the topographic tessellation (TT). As an example, consider a project that needs one sample every 50 meters around the lake. This requirement would need a TT with one row of 50 meters width cells, like the one depicted in Figure 3 (a). Another situation could need also a second sample of 70 meters from the shore, and in this case a different TT would be needed, with two rings or rows around the lake. This is graphically explained in Figure 3 (b). Also, the different sampling requirements can be applied to the geometry of a river, as is depicted in Figure 4.

3.1 General Architecture

This framework has three main components (see Figure 5). The *CoastlineGenerator* object receives an input dataset in geoJson format and returns a new dataset with the topographic mosaic (TT) of each of the geometries of the original dataset, complemented with the necessary metadata so that it can be exported in geojson format (see Figure 6) . The *GridConnector* object receives

Received April 2023 Accepted July 2023; Published August 2023



(a) Original river geometry

(b) TT with one row

(c) TT with three rows

Fig. 4. Topographical tessellations for a river

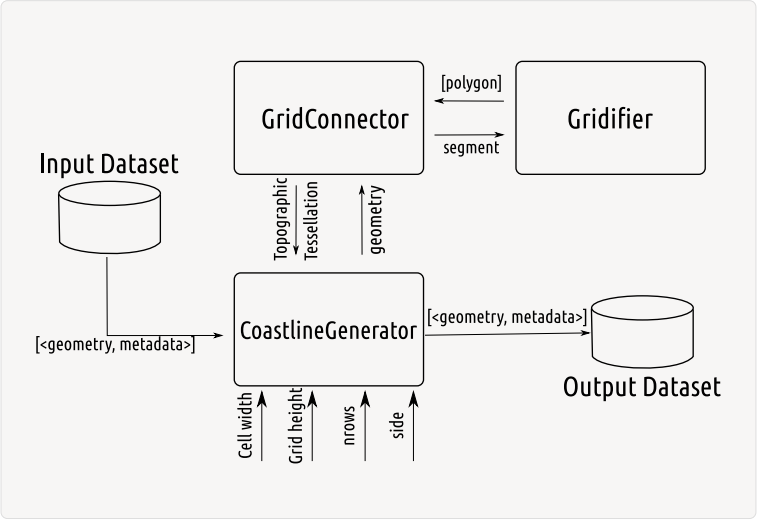


Fig. 5. Coastline generator architecture

Received April 2023 Accepted July 2023; Published August 2023

	geometry	cid	pos	gid	source_object	source_gna
0	POLYGON ((-69.72500 -36.65485, -69.72519 -36.6...	0	left	264890	Acueducto	Acueducto
1	POLYGON ((-69.72519 -36.65486, -69.72538 -36.6...	1	left	264890	Acueducto	Acueducto
2	POLYGON ((-69.72538 -36.65487, -69.72557 -36.6...	2	left	264890	Acueducto	Acueducto
3	POLYGON ((-69.72557 -36.65488, -69.72572 -36.6...	3	left	264890	Acueducto	Acueducto
4	POLYGON ((-69.72572 -36.65492, -69.72586 -36.6...	4	left	264890	Acueducto	Acueducto
..	...	...	...	...	...	...
21	POLYGON ((-69.72803 -36.65463, -69.72826 -36.6...	21	right	264890	Acueducto	Acueducto
22	POLYGON ((-69.72824 -36.65480, -69.72857 -36.6...	22	right	264890	Acueducto	Acueducto
23	POLYGON ((-69.72826 -36.65472, -69.72858 -36.6...	23	right	264890	Acueducto	Acueducto
24	POLYGON ((-69.72857 -36.65488, -69.72886 -36.6...	24	right	264890	Acueducto	Acueducto
25	POLYGON ((-69.72858 -36.65479, -69.72886 -36.6...	25	right	264890	Acueducto	Acueducto

Fig. 6. Output dataframe

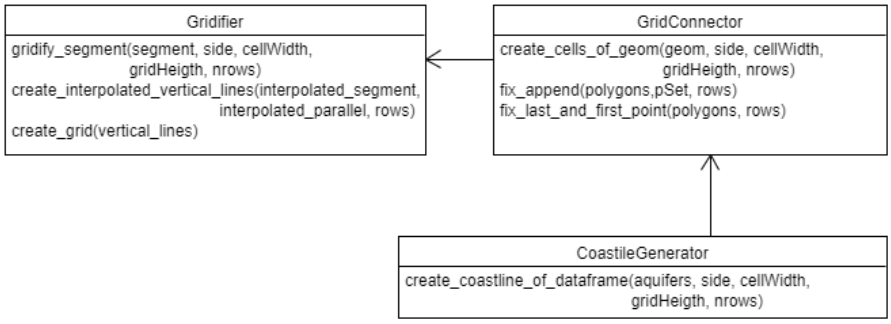


Fig. 7. UML Class diagram

Received April 2023 Accepted July 2023; Published August 2023

a complex geometry (composed of several segments) that can be multiline or polygon, collaborates with Gridifier to generate the grids of each segment and then joins each grid with the adjacent one. The *Gridifier* object is responsible for generating a grid of cells from a segment, with the parameters that define the grid: side, width of the cells, number of rows, total height of the grid.

3.2 Gridifier

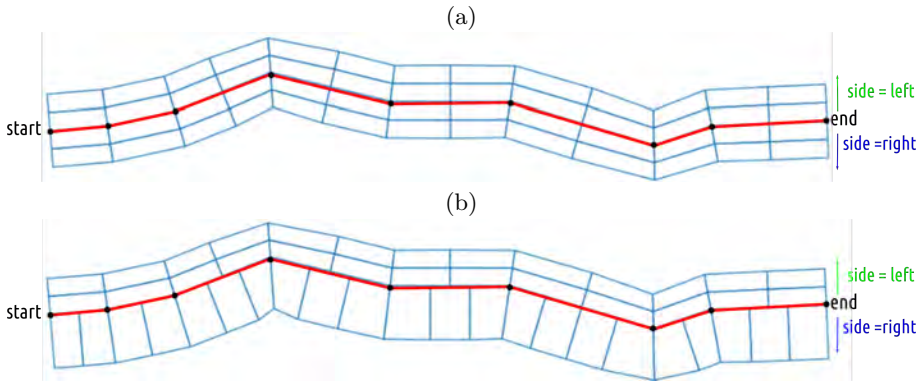


Fig. 8. TT of a river with same parameters (a) and different parametes (b) in both sides

As was said, this object generates a grid of cells (non overlaping adyacent cells) from a given segment that is received as a parameter. It also needs the position where the cells are generated (`side` parameter), the width of each cell (`cellWidth` parameter), the total transversal distance (`gridHeight` parameter)to the original segment, and the number of rows (`nrows` parameter) of cells of the grid.

The `side` parameter, which possible values are {`left, right`} indicates the orientation of the grid with respect to the sequence of points in the input segment. The generated grid is going to be limited by the input segment and a parallel line distanced from the first as indicated by the parameter `gridHeight`. An example is shown in Figure 8 (a), where the red line indicates de input geometry of a river, and travelling from the start point to the end point, left side is the upper grid of the figure, while right side is a the lower one, and both grids are built from the same input parameters. In figure 8 (b) left side grid has 2 rows, a cell width of **40** meters and a grid height of **20** meters, while right side grid has 1 row, a cell width of **20** meters and a grid height of **30** meters.

Generating the grid requires calculating the length of said segment to obtain the number of columns, of the required size, that this segment admits. On the other hand, a parallel line is generated using the method called `paralleloffset`

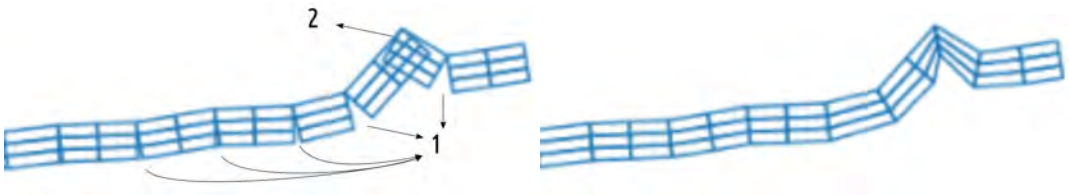
Received April 2023 Accepted July 2023; Published August 2023

provided by the Shapely[Tecnologia] library, located at the distance indicated by the parameter `gridHeigth` and the `side`. Both segments are subdivided to find the lateral vertices of each cell, that will define the vertical lines connecting the two parallel segments. If the parameter `nrows` is greater than one, these verticals are also partitioned to set the rows of the grid. This process generates a list of polygons that correspond to each of the grid cells, and is depicted in the algorithm if Listing 1.1. Following example of Figure 8 (a), each grid is a set of 26 polygons, unlike Figure 8 (b) that left side grid has 26 polygons and right side grid has 22 poligons.

```
gridify(segment, side, cellWidth, gridHeigth, nrows)
    n_columns = (segment.geometry_length() // cellWidth) + 1
  parallel = paralleloffset(segment, side, gridHeigth)
  interpolated_segment  = segment.interpolate(n_columns)
  interpolated_parallel = parallel.interpolate(n_columns)
  vertical_lines = create_interpolated_vertical_lines(
    interpolated_segment, interpolated_parallel, nrows)
  polygons = create_grid(vertical_lines)
  return polygons
```

Listing 1.1. Topographic Tessellation algorithm

3.3 GridConnector



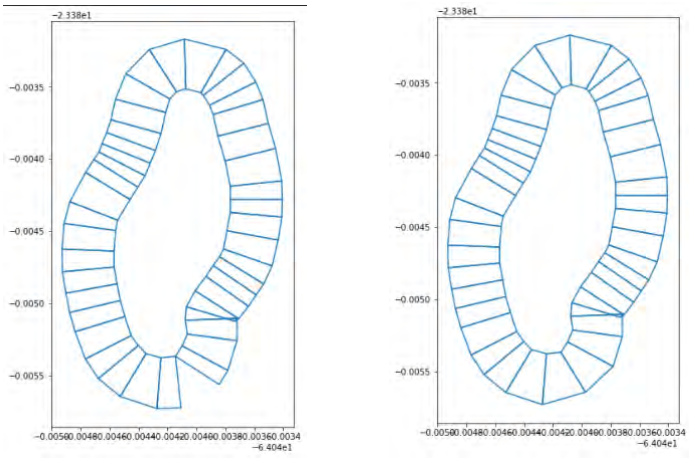
(a) Overlaping (marked as 2) and separation problems (marked as 1)

(b) Final TT with interpolated points

Fig. 9. GridConnector fixAppend issue solved

The main function of this object is to connect the grids that were generated for each segment - of a MultiLineString or Polygon - by the *Gridifier* object. With this objective, it takes the input geometry, divides it into segments (see Listing 1.2, line 5), i.e. taking two points at a time. Then, it collaborates with the Gridifier object to generate the grid of each segment, finally connects the vertices of the generated grids obtaining a topographic tessellation. If the original geometry has polygonal type, the cells corresponding to the space between the last and the first point are created.

Received April 2023 Accepted July 2023; Published August 2023



(a) TT over polygon without fixLastAndFirstPoint

(b) TT over polygon with fixLastAndFirstPoint

Fig. 10. Polygons TT completed

Since the grids generated by the *Gridifier* object are generated from simple segments, they have a rectangular structure. However, each vertex of the polyline generates an angle between the two consecutive segments, and therefore an angle between the respective grids, which leads to problems of overlapping or separation in the path of the TTs. This two problems are presented in an example of a TT from a multiline in Figure 9 (a): with number 1 a separation problem is indicated, and with number 2 an overlapping problem. To solve this, a mechanism was developed to replace conflicting vertices with new vertices resulting from an interpolation process, as shown in 9 (b). This mechanism, encapsulated in the `fixAppend` method of the *GridConnector* object, also allows the cells corresponding to the closure of a polygons object to be generated, as can be seen in Figure 10.

```
create_cells_of_geom(geom, side, cellWidth, gridHeigth, nrows
    ):
  polygons = []
  For i in geom.index:
      segment = LineString(geom[i:i+1])
      pSet = Gridifier.gridify_segment(segment, side,
    cellWidth, gridHeigth, nrows)
      polygons = fix_append(polygons,pSet, nrows)
  if ((geom.type == 'MultiPolygon') | (geom.type == 'Polygon'
    )):
      polygons = fix_last_and_first_point(polygons, rows)
  res = gpd.GeoDataFrame({"geometry" : polygons})
  res['cid'] = res.index
  res['side'] = side
```

Received April 2023 Accepted July 2023; Published August 2023

```
12   return res
```

Listing 1.2. GridConnector algorithm

The dataset generated by GridConnector consists of a row for each new polygon generated (representing a cell of the grid), whose identification is a data composed of a value `cid`, which is an integer value between 0 and the number of cells of the TT, and the `side` value. As an example, consider the multiline geometry of Figure 8 (b): the generated dataset has 26 polygons with `side=left` and `cid`$\in$ `[0..25]`, and 22 polygons with `side=right` and `cid`$\in$ `[0..21]`.

3.4 CoastlineGenerator

The purpose of this object is to generate a Topographical Tessellation for each element of an homogeneous geopraphic dataset, this means that all elements have the same geometry type (multiline, line or polygon). the output dataset contains a set of polygons that represent the cells of all the tessellations. These polygons that correspond to the same input geographic element are related by the **gid** data that identifies its geometry (river or lake) that was used to generate the TT.

Particularly, the CoaslineGenerator iterates over the dataset keeping certain alphanumeric data (metadata) that are of interest in this domain: the **GID**, the **object** and the **GNA** of the geometric object (see Listing 1.3, line 3), and then it generates the TTs based on the **side** parameter. In the case of rivers represented by a multiline, it is important to generate the TT of both sides (`side=both`).

After creating the TT on one element of the dataset, it adds the metadata to each generated cell. To identify each one, a composed identifier made up by the **GID**, the CID and the side parameter is used (see Listing 1.3, line 13).

```
1  create_coastline_of_dataframe(aquifer, side, cellWidth,
       gridHeigth, nrows)
2  for row in aquifer:
3      metadata = row.metadata
4      geom = row.geometry
5      if(side == "both"):
6          rside = gridConnector.create_cells_of_geom(geom, "
       right", cellWidth, gridHeigth, nrows)
7          lside = gridConnector.create_cells_of_geom(geom, "
       left", cellWidth, gridHeigth, nrows)
8          cells = pd.concat([rside,lside])
9      else:
10         cells = gridConnector.create_cells_of_geom(geom, side
       , cellWidth, gridHeigth, nrows)
11
12     cells[objeto],cells[gna] = metadata.objeto, metadata.gna
13     cells[gid] = metadata.gid()
```

Listing 1.3. Topographic Tessellation algorithm

Received April 2023 Accepted July 2023; Published August 2023

3.5 Tecnologìa

For the processing of geospatial information, GeoPandas[4] library was used. GeoPandas is an open source project that extends the data types used by Pandas to allow spatial operations on geometric types. In addition, GeoPandas uses shapely as a geometric data manipulation and analysis library.

To specify the reference system, PyProj was used, a Python interface for PROJ. PROJ is generic coordinate transformation software that transforms geospatial coordinates from one coordinate reference system (CRS) to another. This includes map projections and geodetic transformations.

The source code developed for this work is available as a python notebook on the Kaggle platform[5].

4 Discussion and Future work

This work presented an strategy to generate a topographic tessellation from a given multiline or polygon geometry. Despite it was an ad-hoc development for the AppEAR project, this can be usefull for the application in other domains such as urban planning, or any situation where to represent or analyze the area around a geometry is needed.

The approach that was developed in this work can be considered as a cold start for an adaptation strategy based on the behavior of the community. That is, an initial set of areas that can then be adjusted by calculating Voronoi regions taking as centroids the samples or from other hot spots that represent the busiest areas. This complemented approach allows having a dynamic set of sampling areas to propose a better distribution of the samples.

The segmentation of space within a polygon -which could represent an island- is scheduled for future work, as is the task of performing a topographic tessellation around a single point.

Unlike the polygons that delimit lakes, where the TTs are generated with the parameter `side=right` so that they correspond to the outer side of the lake, the polygons of the islands must be interpreted in the opposite way to generate the TTs inside the polygon. In other words, if the coast of a lake is normally generated from the right (external) side, instead in an island it should be generated from the left (internal) side.

On the other hand, as the used cartography used is about rivers and lakes, it doesn't have objects representing islands, meaning that there is no direct and automatic way to identify when an island is present. This means that these cases must be addressed individually, this is, find -by observation- the island and generate its TT coast on the internal side, which would be the left.

Un punto puede abordarse pensando como una circunferencia en torno al punto In this work, both linear and polygonal geometries are considered, but so far those that are only composed of a point, are not taken into account. In the

[4] https://geopandas.org/en/stable/
[5] https://www.kaggle.com/brunolattanzio/coastlinecellsgenerator/edit/run/72445421

Received April 2023 Accepted July 2023; Published August 2023

AppEar project case study these would correspond to, as an example, waterfalls, jumps or dams. The main reason why they have not been taken into account is that, despite being an individual geometric object, they are usually part of another object. That is to say, although the algorithm cannot generate a coast for a waterfall, it can do it for the river of which the waterfall is part of.

However, methods have been thought to address this problem in future works, such as defining a circle around the point and dividing it using a similar mechanism to that of the other geometries.

References

1. Dalponte Ayastuy, M., Torres, D.: Relevance of non-activity representation in traveling user behavior profiling for adaptive gamification. In: Proceedings of the XXI International Conference on Human Computer Interaction. Interacción '21, Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3471391.3471431, https://doi.org/10.1145/3471391.3471431
2. Du, Q., Gunzburger, M.: Grid generation and optimization based on centroidal voronoi tessellations. Applied Mathematics and Computation **133**(2), 591–607 (2002). https://doi.org/https://doi.org/10.1016/S0096-3003(01)00260-0, https://www.sciencedirect.com/science/article/pii/S0096300301002600
3. Fleischmann, M., Feliciotti, A., Romice, O., Porta, S.: Morphological tessellation as a way of partitioning space: Improving consistency in urban morphology at the plot scale. Computers, Environment and Urban Systems **80**, 101441 (2020). https://doi.org/https://doi.org/10.1016/j.compenvurbsys.2019.101441, https://www.sciencedirect.com/science/article/pii/S0198971519302856
4. Gold, C.: Tessellations in gis: Part i—putting it all together. Geo-spatial Information Science **19**(1), 9–25 (2016). https://doi.org/10.1080/10095020.2016.1146440, https://doi.org/10.1080/10095020.2016.1146440
5. Gold, C.M., Remmele, P.R., Roos, T.: Voronoi methods in GIS, pp. 21–35. Springer Berlin Heidelberg, Berlin, Heidelberg (1997). https://doi.org/10.1007/3-540-63818-0_2, https://doi.org/10.1007/3-540-63818-0_2
6. Longley, P.A., Goodchild, M.F., Maguire, D.J., Rhind, D.W.: Geographic information science and systems. John Wiley & Sons (2015)
7. Silvertown, J.: A new dawn for citizen science. Trends in Ecology & Evolution **24**(9), 467–471 (2009). https://doi.org/https://doi.org/10.1016/j.tree.2009.03.017, https://www.sciencedirect.com/science/article/pii/S016953470900175X
8. Tsai, V.J.D.: Delaunay triangulations in tin creation: an overview and a linear-time algorithm. International Journal of Geographical Information Systems **7**(6), 501–524 (1993). https://doi.org/10.1080/02693799308901979, https://doi.org/10.1080/02693799308901979
9. Vohland, K., Land-Zandstra, A., Ceccaroni, L., Lemmens, R., Perelló, J., Ponti, M., Samson, R., Wagenknecht, K.: The science of citizen science. Springer Nature (2021)

Received April 2023 Accepted July 2023; Published August 2023