HANDLING BOUNDARY CUTS WHILE REPROJECTING GIS VECTOR DATA

Kerkovits Krisztián

Affiliations:
Kerkovits Krisztián, PhD student;
Eötvös Loránd University, Department of Cartography and Geoinformatics;
1117 Budapest, Hungary, Pázmány Péter sétány 1/A;
kerkovits@map.elte.hu;

Abstract
When projecting the spherical Earth on to a plane, the map projection has distortions, but discontinuities turn up inevitably too. These so-called boundary cuts may cause illegible geometries after applying the projection. Most GIS-software suffer from this, therefore vector data needs to be pre-cut before reprojecting it. Using the following algorithm, these boundary cuts can be handled correctly producing good-looking geometries for cartographic purposes. This way a cartographer can quickly create base maps for his/her work in an arbitrary projection. A sample implementation of this method is shown as an OpenLayers plugin.

Keywords*: map projections, world maps, vector GIS, computer assisted cartography, software development*

DISCONTINUITIES IN GIS

When the whole surface of the Earth is projected onto a plane, it is unavoidable to have discontinuities at least at one point of the map. This is because the datum surface (in small-scale maps this is usually a sphere) is a closed and finite manifold, while the projected planar manifold does not possess these properties. Discontinuities may only be located at a singular point of a datum surface, but usually they follow lines. On the projected plane they always appear as lines following the boundary of the map. Any linear or polygonal map object crossing a discontinuity continues at another location of the map. In this article the term “boundary cut” refers to such discontinuities.

Most GIS programs do not handle boundary cuts. Line strings and polygons that cross a boundary cut are displayed by incorrectly connected nodes, which create a disturbing line network. Due to this problem most of the vector data is pre-cut at the antimeridian (180°). However, this still does not solve the issue (moreover, it might give worse results) if a midmeridian other than Greenwich is chosen for map-making. In addition, when an interrupted projection (e. g. the Goode homolosine) is used, additional discontinuities appear. Furthermore, for special reasons unusual aspects of projections may be used. The three best-known aspects are direct, oblique and transverse. In this case, the boundary cuts do not simply follow the graticule lines, but they rather cut the map along the metagraticule lines. (Terminology used as in Wray, 1974)

Recently as large quantity of GIS-data is becoming available, cartographers began to use the data more and more frequently to create a base map for their work. With the assistance of computers, reprojection of data takes only a few seconds, therefore an optimal, tailor-made projection can be chosen for each map. However, if the boundary cuts are disregarded, the result can be ineligible for cartographic purposes. Figure 1 shows a tailor-made projection optimized for oceans. On the upper map, the areas are jumbled, the lines cross the metapole-lines and flow over the map area. We may get such visualization if we use a common GIS for reprojecting data.

A better method is needed for not only geographic maps, and the boundary cuts also have to be handled in any global thematic map. As thematic data is often generated by GIS, it cannot be guaranteed that the shapes are pre-cut at the discontinuities of the chosen projection. While facing this problem, I was surprised that I could not find any routine answering this issue. The only exception is the D3 library of JavaScript (Bostock, 2016a), but it uses the internal methods of D3, and the result can be exported to the SVG vector format only, which contains no geographic information.

This problem rises only in those small-scale maps that display areas larger than a hemisphere. The boundary cuts can be neglected in regional maps, because they appear outside of the area of interest. Therefore, I suppose that the lower precision of small-scale maps is acceptable for the output.

Naturally, such a method is irreversible. Although I tried to preserve the topology as much as possible (for example by avoiding self-intersections and overlapping rings too), still the output can only be used for cartographic visualization.

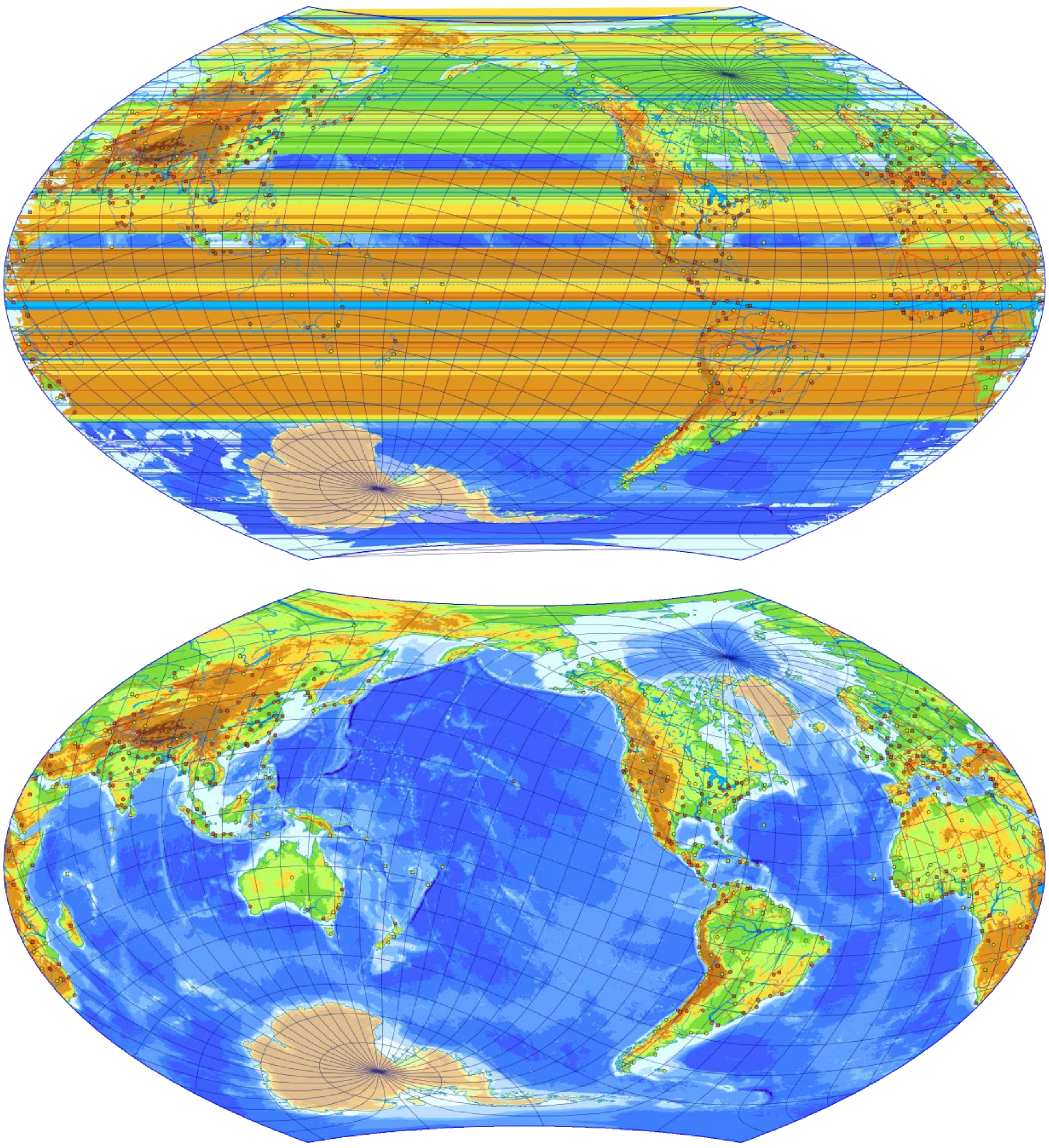

Figure 1. Boundary cuts handled wrongly (upper map) and correctly (lower map) on a plagal aspect of a Wagner-transformed Aitoff's projection

PROJECTIONS AND VECTOR DATA IN THE OPENLAYERS SYSTEM

The OpenLayers is an open source JavaScript web-GIS library, which supports any arbitrary projection. This feature makes it suitable for use in my research. It supports exporting reprojected features to various formats including GeoJSON, KML etc.; further, it can also display any vector data directly in the web browser. However, its raster reprojection method is rather rudimentary and are much better implementations using WebGL. (Jenny et al., 2015)

OpenLayers stores geometry in a format resembling GeoJSON and supports all geometry types of GeoJSON. There are six kinds of simple geometries (Butler et al., 2008):

- Point: array of coordinates
- MultiPoint: array of points
- LineString: array containing nodes in linking order
- MultiLineString: array of line strings
- Polygon: array of line strings, the first one is always the outer ring; others are holes (e.g., islands in a lake, local deviations in an isoband map). The first and last node of each ring is always the same.
- MultiPolygon: array of polygons.

Sometimes one object combines several types of simple geometries, which is called a GeometryCollection. No special routine is needed, as all parts of collections are of simple geometry. In OpenLayers there is a geometry type called Circle, which is not included in the GeoJSON standard. It is a Point geometry with a given radius, therefore it is regarded as a point.

To define a projection in OpenLayers, either the proj4 string or the direct and inverse transformation formulas are given (OpenLayers, n. d.). If the formulas between the projected coordinate systems and the geographic (EPSG:4326) coordinates are known, OpenLayers knows how to reproject the data between these coordinate systems. If the transformation formulas between the ellipsoid and the coordinate system A are given and between system A and system B too, OpenLayers has to be explicitly told to use both formulas for transforming the ellipsoid to system B.

Therefore, if a metacoordinate system is needed for a particular aspect, three transformations are needed (the difference between the WGS84 datum and the sphere is ignored due to small-scale mapping): the spherical rotation, the formulas between the metagraticule and the projected plane, and a function calling both transformations when projecting directly between the datum and the plane. Applying this model, computations can be made simpler using the metacoordinates of the shapes. The transformation formulas of the rotation are given in Snyder, 1987.

BOUNDARY CUTS IN VARIOUS PROJECTIONS

Most projections have a discontinuity at the 180th metameridian. In direct aspect, with a midmeridian λ_M other than Greenwich, this meridian will be $\lambda_M \pm 180°$. In the following the term antimeridian refers to this (meta)meridian. Since nearly all boundary cuts follow special metagraticule lines, the terms latitude (φ) and longitude (λ) are always understood as metalatitudes and metalongitudes. (The latter, in direct aspect with an arbitrary midmeridian, are $\lambda - \lambda_M$.)

Azimuthal and pseudoazimuthal projections

The use of an azimuthal projection to map the whole surface of the Earth is rather rare. However, it can be applied by a few special azimuthal projections (e.g. the azimuthal equal-area, the azimuthal equidistant and the Wiechel pseudoazimuthal). An example could be if oblique azimuthal equidistant projections are used for displaying distances from a particular point of the Earth (Reyes, 2012).

These projections have discontinuity only at one point: the antipode metapole. A line segment crosses a metapole, if the difference between the longitudes of two neighbouring points is approximately 180°. Taking latitudes into consideration it can easily be decided which metapole the line segment crosses. As the floating-point error of the computation of metacoordinates is high near the metapole, greater tolerance has to be given for vertices near the metapole. A heuristic formula may be used to determine this tolerance. It is possible to have a node exactly on the metapole. In this case, this point has to be removed, because the longitude value of the point may contain misleading information.

Although the longitude has no meaning at the pole, transformation formulas of projections with pole-lines map the points of the pole described by different longitudes to different points of the pole-line (see Figure 3). To get a good-looking output geometry, the endpoints should get the same longitude as the neighbours after splitting the lines at the pole. After cutting, the software must not transform the shapes back to the geographic coordinate system to avoid roundoff errors; instead, formulas of direct aspect should be used to reproject them from the metacoordinate system to planar coordinates.

If a polygon contains the antipode metapole, its boundary is not cut. In this case, not the inner, but the outer side of the ring should be filled. It is impossible to do so in GeoJSON, as most GIS software disregards winding order. My solution is to add the pole-line as an outer ring, and convert the original outer ring to a hole (Figure 2).

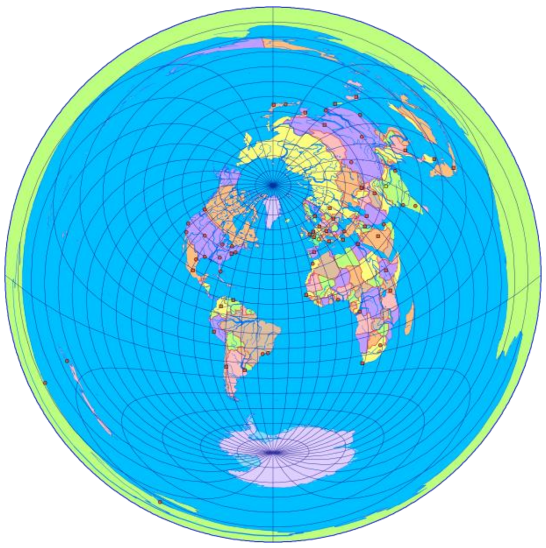

Figure 2. Filling Australia on an equidistant azimuthal projection centred on the North Atlantic Ocean

Far-side vertical perspective projections cannot display the whole surface of the Earth, but in a well-chosen oblique aspect the clipped part can be placed on a less important part considering the map thematics. This way almost the whole Earth can be displayed.

Knowing the exact place of the zenithal point, the boundary latitude of the map can be computed. Whilst areas metasouth from this latitude are not displayed at all, I got acceptable result only by clipping the geometries at the bounding metaparallel.

Other non-interrupted projections

Almost all other projections cut at the antimeridian. Cylindrical, conical, pseudocylindrical, polyconic and lenticular projections are not continuous along this metameridian.

Attention should be given to line segments that intersect the antimeridian while cutting geometries. This occurs when the longitude difference of two neighbouring nodes is greater than 180°. (If it is nearly 180°, it crosses the pole rather than the antimeridian.)

When substituting into the projection formulas, longitudes 180° and –180° are projected onto the opposite border of the map. It is essential to determine from the split line segment whether it lies on the metaeastern or metawestern hemisphere, and give the correct sign to the longitude of the intersection point. If this is missed, we will receive the same confused network as when the lines were not cut.

Numerous projections of this type have pole-lines instead of pole-points. The output may be ugly especially if the pole-line has some curvature. This occurs on some conical, polyconic and lenticular projection (Figure 3). Lines crossing the pole-line have to be cut; otherwise, they might flow over the map area. Using the same method as with azimuthal projections (now including both metapoles) produces satisfactory output. In this case, it is better to cut geometries first through the antimeridian and then at the poles. Here, it is easier to associate the holes with the correct outer ring.

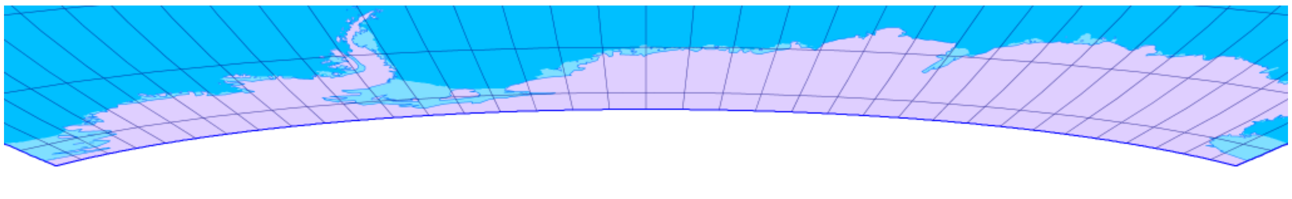

Figure 3. Pole-line of the Wagner-transformed Hammer's projection

Interrupted and polyhedral projections

To reduce distortions, projections may be applied in interrupted form. The most popular is the interrupted Goode homolosine projection. Most of these projections are not recommended in any aspect but direct. Boundary cuts follow meridians, but they usually appear between only one of the poles and an arbitrary parallel (Figure 4).

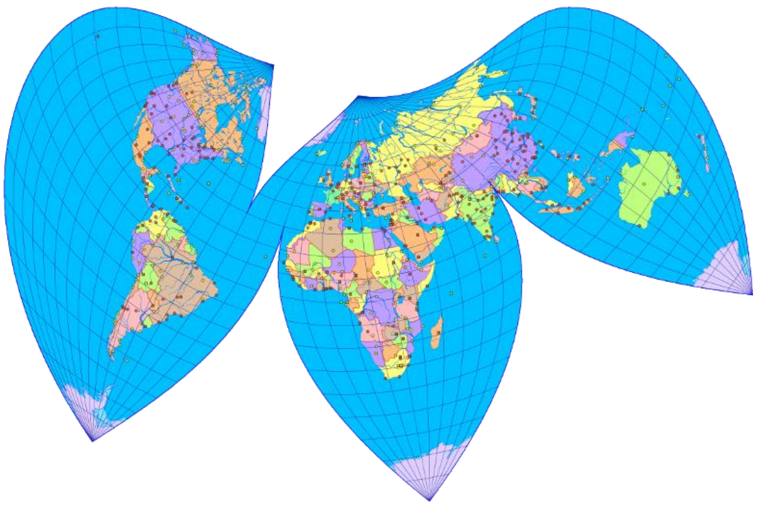

Figure 4. Interrupted Bonne's projection

The same solution is used as before, but with the following considerations:

- Some projections (e.g. the Berghaus or the William–Olsson) are star-shaped. (Figure 5) Cutting the antimeridian must not be performed in this case for the full meridian. Although it is generally a good idea to choose such midmeridian for computations, the antimeridian of which is followed by a boundary cut.
- Otherwise, the projection has at least one meridian that is cut through in its whole length. Then this meridian has to be the antimeridian, and the corresponding midmeridian has to be chosen for computations.
- We should not forget that a line segment is cut only if its intersection with the bounding meridian is in the latitude interval of the discontinuity.
- The antimeridian cut has to be performed first; otherwise false results are produced when looking for intersections with other meridians.
- This method is useful only for injective projections. Non-injective interrupted projections (like the Baranyi–Márton projection (Márton, 2006)) need more sophisticated approach that duplicates geometries and performs different cuts on each duplicate.

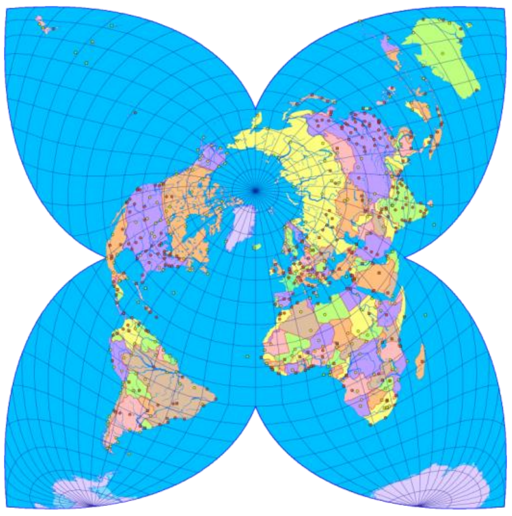

Figure 5. Simple oblique aspect of the William–Olsson projection

Polyhedral projections are still considered special in small-scale mapping because of their complexity. Distortions are significantly reduced if the polyhedron is close to the sphere, but several boundary cuts arise while unfolding the polyhedron to plane. Depending on the chosen polyhedron, these boundary cuts can follow any great circle, and the calculation of intersection points with line segments can be very complicated.

The octahedron is a special case. Its direct aspect can only have discontinuity following meridians and the Equator. Any other aspect can be reduced to this case using metacoordinates. These projections are also known as butterfly maps because of their appearance. These projections have to be handled as if they were simple interrupted projections. Some butterfly maps have cuts following the Equator too (Figure 6). Intersection of a line segment and the Equator can be computed, and if it falls on a boundary cut, it has to be split too.

This algorithm works only for the octahedron. If boundary cuts follow arbitrary great circles, more complicated calculations are needed (such projection is the Fuller Dymaxion). It might be easier to describe the cut lines in cubic projections by using geocentric Cartesian coordinates.

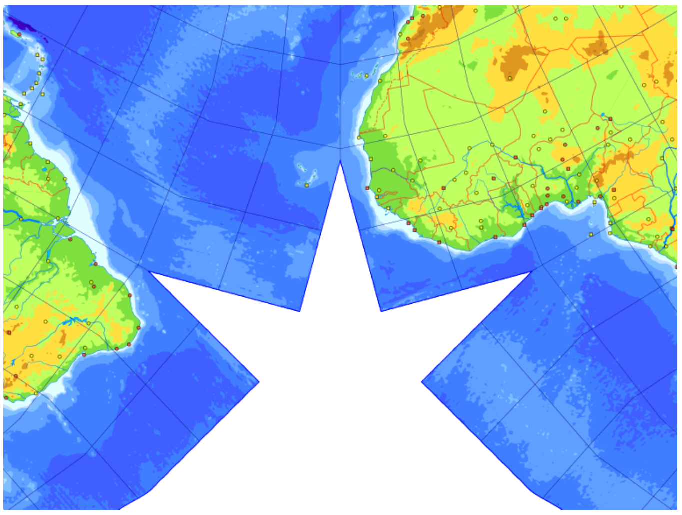

Figure 6. Boundary cuts of Waterman's projection

SPLITTING UP GEOMETRIES

After the exact locations of the boundary cuts have been determined, geometries have to be split. Points, line strings and polygons have to be dealt separately.

Points (Circles) and MultiPoints

If a point lies exactly on a boundary cut, it might cause errors during reprojection. For safety, it is advised to move the point a little bit away.

LineStrings and MultiLineStrings

Vertices of line strings are like points, which need to be moved away from boundary cuts. If a line segment between two neighbouring nodes is intersected by a boundary cut, the exact coordinates of the intersection point are to be computed. The following trigonometric equations are calculated very slowly on computers, therefore a filter must be established to calculate the equations for those line segments only, where the existence of a boundary cut is assumed.

The intersection of a great circle and a cutting meridian can be calculated using spherical trigonometry. Any angle in the formulas should be substituted in radians.

Let the coordinates of vertices be φ_A, λ_A and φ_B, λ_B. Then the spherical distance:

$$s = \operatorname{acos}(\sin\varphi_A \sin\varphi_B + \cos\varphi_A \cos\varphi_B \cos(\lambda_B - \lambda_A)) \quad (1)$$

The angle of the polar spherical triangle *ABN* at point *B* from the cosine rule:

$$\beta = \operatorname{acos}\frac{\sin\varphi_A - \sin\varphi_B \cos s}{\cos\varphi_B \sin s} \quad (2)$$

The latitude φ_S of the intersection point can be expressed from the cotangent four-part formula (λ_S is the longitude of the splitting meridian):

$$\tan\varphi_S \cos\varphi_B = \sin\varphi_B \cos(\lambda_S - \lambda_B) + |\sin(\lambda_S - \lambda_B)| \cot\beta \quad (3)$$

Rearranged:

$$\varphi_S = \operatorname{atan}\frac{\sin\varphi_B \cos(\lambda_S - \lambda_B) + |\sin(\lambda_S - \lambda_B)| \cot\beta}{\cos\varphi_B} \quad (4)$$

Naturally, these formulas may only be used if none of the vertices lie on the cutting meridian (because of the floating point error they also must not be too close to it). In this case, the intersection point is the same as the vertex lying on the meridian, so no calculation is needed. This formula must not be used if one of the vertices lies on the pole, because the intersection is in the pole.

If a parallel is cut, calculations become more complicated. Let

$$x = \left|\tan\frac{\lambda_S - \lambda_B}{2}\right| \quad (5)$$

Then equation (3) is arranged to zero using the identities of half angles:

$$\tan\varphi_S \cos\varphi_B - \sin\varphi_B \frac{1-x^2}{x^2+1} - \frac{2x}{x^2+1}\cot\beta = 0 \quad (6)$$

Bringing fractions to a common denominator, and arranging (6) as a quadratic equation of *x*:

$$(\tan\varphi_S \cos\varphi_B + \sin\varphi_B)x^2 - 2\cot\beta x + \tan\varphi_S \cos\varphi_B - \sin\varphi_B = 0 \quad (7)$$

Applying the quadratic formula:

$$x = \frac{2\cot\beta \pm \sqrt{4\cot^2\beta - 4(\cos\varphi_B \tan\varphi_s + \sin\varphi_B)(\cos\varphi_B \tan\varphi_s - \sin\varphi_B)}}{2(\cos\varphi_B \tan\varphi_s + \sin\varphi_B)} \quad (8)$$

Substituting *x* back, reducing the fraction by two and applying the identity $(a+b)(a-b) = a^2 - b^2$:

$$\left|\tan\frac{\lambda_S - \lambda_B}{2}\right| = \frac{\cot\beta \pm \sqrt{\cot^2\beta - \cos^2\varphi_B \tan^2\varphi_s + \sin^2\varphi_B}}{\cos\varphi_B \tan\varphi_s + \sin\varphi_B} \quad (9)$$

Now λ_S can be expressed from latitude φ_S and the angle β computed from equation (2).

$$\lambda_S = \lambda_B \pm 2\operatorname{atan}\frac{\cot\beta \pm \sqrt{\cot^2\beta - \cos^2\varphi_B \tan^2\varphi_s + \sin^2\varphi_B}}{\cos\varphi_B \tan\varphi_s + \sin\varphi_B} \quad (10)$$

We have four solutions. Usually, only one is correct, the other three are extraneous. (Of course, it is possible to choose two vertices on the sphere, which are on the same side of the parallel, but the geodetic between them intersects it twice. Let us see this case as if there were no intersection, because it not only makes the algorithm really complicated, but would also create negligible small areas.) The ± sign before the square root is –, if $\varphi_A > \varphi_B$, and + if $\varphi_A < \varphi_B$. The ± sign before the atan takes the sign of $\sin(\lambda_A - \lambda_B)$.

If $\exists n \in \mathrm{Z} : \lambda_A - \lambda_B \approx n\pi$, then the formulas above are numerically unstable. The intersection point in this case lies approximately on the meridian of either point *A* or *B*. The formula is also unstable if $\varphi_B \approx -\varphi_S$. If this occurs, a good approximation is given by limit calculus:

$$\lambda_s \approx \lambda_B \mp 2\,\mathrm{atan}(\tan\beta \sin\varphi_B) \quad (11)$$

Here ± takes the opposite sign of $\sin(\lambda_A - \lambda_B)$.

Now the line string can be cut at the intersection, adding the intersection point to both sections. It is important that these endpoints always lie on a boundary cut, therefore it is advised to move them a little towards their neighbouring vertex. If there already has been a vertex on the intersection, it must be deleted to avoid duplicate nodes.

The split up sections must be kept together as a MultiLineString. So a LineString containing n intersections with boundary cuts is converted to a MultiLineString of n+1 lines. No such conversion is needed for geometries that were MultiLineStrings originally.

Polygons and MultiPolygons

Polygons need a more sophisticated method. Although they are described with linear rings similar to line strings, if they are cut in the same way, their borders will not close, and so they cannot be filled. It must be taken into consideration that if a hole is split by a boundary cut, its border will be added to the outer ring as a depression. (See the top hole in Figure 7)

The winding order of polygons is not defined in GeoJSON. OpenLayers can return coordinates in consistent order; however, the result is usually wrong, hence unreliable near the antimeridian. As it is not trivial which side of a border is filled on a sphere, a routine is needed to determine it. I assume that there is no polygon larger than a hemisphere, so I always fill the smaller area.

First, the signed area of the ring is computed. Calculating the exact value is computationally expensive, but an approximation is achieved with a simple sum. (Chamberlain–Duquette, 2007)

$$A = \frac{1}{2}\sum_i \Delta\lambda \sin\varphi_i \quad (12)$$

where φ_i is the latitude of the vertex, and $\Delta\lambda$ is the longitude difference of node *i*+1 and *i*–1 in radians. Usually $\Delta\lambda = \lambda_{i+1} - \lambda_{i-1}$, but if its absolute value is greater than π, then the antimeridian is crossed: $\Delta\lambda = \pm 2\pi - (\lambda_{i+1} - \lambda_{i-1})$ Choose the sign that fulfils $|\Delta\lambda| < \pi$. If the absolute value of the signed sum of $\Delta\lambda$ is greater than π (area contains a pole), add 2π to or subtract 2π from the area. $|A|$ always must be less than or equal to 2π (a hemisphere). Add or subtract 4π if needed (it happens in the unlikely case when area contains both poles).

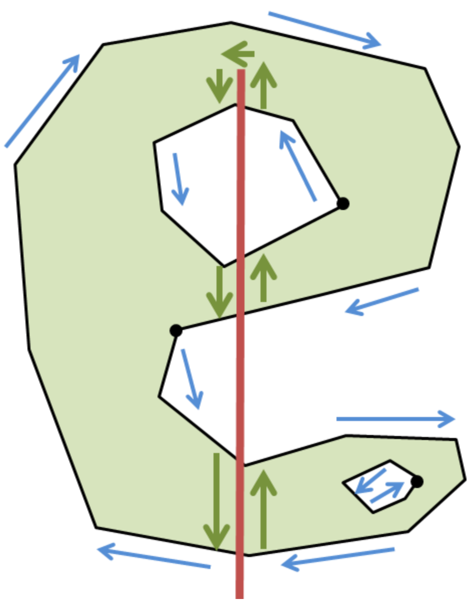

Figure 7. Splitting a polygon geometry

The signed area calculated in this way is positive if the winding order is clockwise, and negative if counter-clockwise. The ring that has the biggest absolute value of area is the outer ring, while all the others are holes. Rings are reversed if needed, so from now on, all outer rings are stored clockwise, and all holes are stored counter-clockwise. (Blue lines in Figure 7)

After making the winding order consistent, rings are split as line strings. If a boundary cut was intersected, the first and last segment can always be connected, as they always have a common point (the first and the last ones) in this format. (Marked by black circles in Figure 7) Holes need investigations too, if the outer ring has been cut. If a hole is not split, it can be added as a hole to the new geometry (like the lower hole in Figure 7). If it is cut, the first and last segments are connected, and open segments are stored in an array together with the segments of the outer ring.

In the next step, not closed segments are ordered separately by their start and endpoints counter-clockwise around the boundary cut. (Here, the counter-clockwise is understood on the sphere, not on the map! It might be different in the case of the antimeridian.) Line segments are connected from the endpoints to the next start point respectively. One must walk around the discontinuity counter-clockwise. (The boundary cut is at the left hand, just like the green lines in Figure 7.) This guarantees that the interior side of the original polygon is kept intact.

The boundary cut may not appear as a straight line on the map. Additional vertices need to be added to follow its curvature. I experienced that it is usually enough to add a vertex at every half degree to produce a good-looking small-scale map, so in my approach, the nodes are placed at a fixed distance. A scale-adaptive method can also be used to produce more accurate results (Sakshuwong–Angeli, 2012.; Bostock, 2016b), but when implementing it, attention must be paid that if a parallel is cut, the connecting lines must not follow a geodetic line, rather a small circle on the sphere.

It is possible that two segments at two different sides of a discontinuity have to be connected. (See at the top of Figure 7.) Connection must be executed by adding vertices and getting around the boundary cut counter-clockwise until reaching its end. Then the line is continued by placing vertices on the other side of the discontinuity. On general perspective azimuthal projections (where the discontinuity is a complete small circle) this must not happen; it must be avoided by ordering segments correctly.

If we are lucky, a part of the polygon is now already closed (e.g. at the lower right corner of the figure). If the segment was connected to another one, it must be stored as a partial result. The array of segments must be synchronized, as now the number of free endpoints decreases by two. If the procedure is continued, eventually all segments are closed.

After closing the outer rings, multiple outer rings may evolve. The interior rings kept intact must be examined to associate them with the corresponding outer ring. A bounding box test (the bounding box of the outer ring must fully contain the bounding box of the interior ring) is usually sufficient.

If a simple Polygon was cut into multiple parts, its type must be set to MultiPolygon.

IMPLEMENTING THE METHODS

The methods above were implemented as an OpenLayers Plugin. The API has two public methods. The ol.proj.rotateProjection takes a projection defined in direct aspect, and registers the transformation formulas from geographic to metacoordinates. It also defines the given aspect of the input projection. Non-direct aspects of projections must be defined in this way so as to make calculations with metagraticules possible without transforming them back to geographic coordinates after cutting.

The other method ([ol.format.Feature].readCutFeatures) is meant to replace the readFeatures method of OpenLayers. It reads the input file with the original parser of OpenLayers, and then the cuts are performed. For complicated projections, the metagraticule lines where the boundary cuts lie must be given manually as an array. Output feature collection can be drawn directly in the web browser as a vector layer, but can also be exported to a georeferenced file using OpenLayers' writeFeatures method. (This outputs only a string representation of the file, so the implementation of a file-writing algorithm is needed.)

A demonstration site was created (http://mercator.elte.hu/~kerkovits/reproj/), where a dataset of a small-scale world map can be viewed in any aspect of several projections. It is recommended to try the vertical perspective and the interrupted projections, where locations of boundary cuts depend on the projection parameters. You can even choose a hypsometric map to see the robustness of the algorithm.

REFERENCES

Bostock, Mike (2016a): Antimeridian Cutting. http://bl.ocks.org/mbostock/3788999 Last accessed: 28th April 2016.

Bostock, Mike (2016b): Adaptive Resampling. http://bl.ocks.org/mbostock/3795544 Last accessed: 28th April 2016.

Butler, Howard et al. (2008): The GeoJSON format specification. Tech. rep. Internet Engineering Task Force

Chamberlain, Robert G. – William H. Duquette (2007): Some algorithms for polygons on a sphere. Pasadena, CA: Jet Propulsion Laboratory, National Aeronautics and Space Administration

Jenny, Bernhard – Bojan Šavrič – Johannes Liem (2015): Real-time raster projection for web maps. International Journal of Digital Earth, pp. 1–15

Márton Mátyás (2006): Die Kartogaphische Darstellung der Ozeane in der geänderten Projektion IV. von Baranyi. Kartographische Nachrichten 56/3 pp. 145–148

OpenLayers 3 API Reference. http://openlayers.org/en/master/apidoc/ Last accessed: 20th April 2016.

Reyes Nuñez, José Jesús (2012): The first National Atlas of Cuba: Rediscovering the early 20th century country. Symposium on Discovery, Exploration, Cartography

Sakshuwong, Sukolsak – Gabor Angeli (2012): Adaptive Composite Map Projections in D3. http://web.stanford.edu/~sukolsak/projects/cs448b_final_paper.pdf Last accessed: 23th Nov. 2015.

Snyder, John Parr (1987): Map projections—A working manual. US Government Printing Office. No. 1395.

Wray, Thomas (1974): The seven aspects of a general map projection. Cartographica: The International Journal for Geographic Information and Geovisualization, 11/2 pp. 1–72.

BIOGRAPHY

I studied earth sciences at BSc level and cartography at MSc level at Eötvös Loránd University in Budapest. I obtained master degree in 2014. Since 2015, I have been a PhD student at the Doctorate School of Earth Sciences, Eötvös Loránd University, where I have been doing research in the topic of optimizing map projections.