УДК 004.67

Rodriges Zalipynis R.A.

NEW METHODS FOR VISUALIZATION OF LARGE VOLUMES OF ECOLOGICAL AND CLIMATIC DATA

Effective visualization is crucial for data understanding. This paper describes a method for constructing georeferenced isolines from a global latitude-longitude regular grid. Unlike previous efforts, it represents isolines with polygons and guarantees that all of them are closed. This dramatically reduces the number of objects a visualization subsystem must deal with. The method is operational in Climate Wikience to provide interactive, 3D, real-time, terrain-following isolines for large volumes of ecological and climatic data.

Introduction

For over 15 years, the majority of climate research is based on climate reanalysis archives [1, 2]. They contain retrospective data for up to 80 atmospheric parameters with 6 hour interval on regular latitude-longitude grids for the previous 30 years. Plotting isolines is crucial preparation step before any further map interpretation [3].

Surprisingly, in spite of great impact on data understanding, there are no tools to plot isolines effectively on-demand which often required when exploring large volumes of georeferenced data. This paper presents isolines construction method consisting of 3 stages. First, it uses CONREC algorithm [4] to build segments on locally triangulated latitude-longitude grid. Second, it builds R-Tree to accelerate searching of neighbor segments. Lastly, it connects the segments into closed polygons both to speed up the visualization and enable GIS features.

The straightforward visualization of CONREC segments is prohibitively expensive since there are tens of thousands of segments for a single grid. A visualization subsystem is incapable to handle this number of objects effectively.

Also, segments connection is not a trivial task since global latitude-longitude coordinate system has peculiarities on the poles and near -180° longitude. In addition, for real data CONREC does not always generate necessary segments what results in gaps and unclosed isolines. Special heuristics were invented to handle these situations properly and effectively.

Background on Climate Wikience

Climate Wikience [5] consists of two main parts. Desktop GUI (Climate Wikience) responsible for interactive 3D visualization of georeferenced data and ChronosServer, enabling real-time data delivery from the cloud to thousands of concurrent GUIs.

For vast amounts of climate reanalysis data, it is impossible to create isolines for all available grids in advance and deliver them on demand. This is because the method for

isolines construction has many parameters and isolines, if built, will take several times more space than grids themselves.

Climate Wikience is highly interactive GUI which allows users to explore the Earth ecological and climatic data in 3D. It is impractical to store terabytes of those data on local hard drive. Climate Wikience queries ChronosServer seamlessly to the user to retrieve the required data for visualization. Usually, the data exchange is carried out per grid bases. For example, "SELECT DATA FROM r2.pressure.msl WHERE TIME = 01.01.2003 00:00" will return AMIP/DOE Reanalysis 2 regular 2.5°×2.5° latitude-longitude grid for mean sea level pressure for 2003 Jan 01, 00:00 UTC (fig. 1).

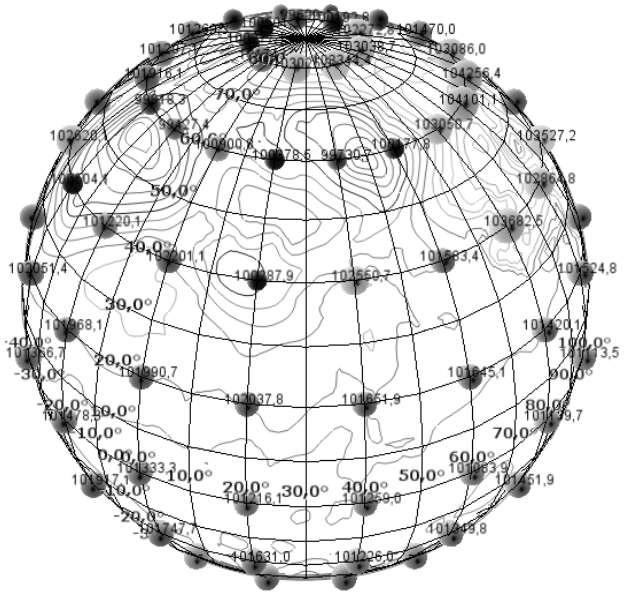

Figure 1 – Mean sea level pressure regular grid with isobars for 01/01/2003 00:00 UTC shown at 20° and 500 GPa intervals respectively. Circle sizes are proportional to pressure values. Built in 3D with Climate Wikience using AMIP/DOE Reanalysis 2

Isolines

For a given function of two variables, latitude, and longitude, defined over a geographic area, a contour line represents a curve along which the function value is constant [3].

For climatic data, isolines are vital for interpretation of an atmospheric state. For example, cyclones are identified visually as several closed isobars nested in each other. Cyclones are key atmospheric components responsible for heat transfer from the tropics to the pole, define local weather conditions, strongly influence severe weather events, droughts and floods. Some of them, like hurricane Katrina, result in lives loss and devastation. However, today's publications contain very poor isolines plots which vague the perception.

To plot isolines, one need to specify levels for which isolines will be built,

$$L=\{l_0, l_2, \ldots, l_m\} \qquad 1)$$

For example, for pressure levels this may be (in hPa): L = {1000, 925, 850, 700, 600, 500, 400, 300, 250, 200, 150, 100, 70, 50, 30, 20, 10}.

At least two ways exist to specify the levels. The first one is the enumeration of level values as shown above while second is specifying *min, max* and *step* parameters from which the levels will be generated. The l_i is calculated as

$$l_i = min + step \times I, \quad 2)$$

$$m = (max – min)/step, \quad 3)$$

where the division in (3) is integer. In either case, the algorithms in this paper operate with the set of contouring levels L, regardless of the isolines specification way.

CONREC Algorithm

The CONREC algorithm was introduced in 1987 by Paul Bourke [4]. To author's best knowledge it is the only description of a contouring routine available on the Web. Let each rectangle of a global regular latitude-longitude grid has coordinates

$$\begin{matrix} f(lat, lon), & f(lat, lon+\delta), \\ f(lat+\delta, lon), & f(lat+\delta, lon+\delta), \end{matrix} \quad 4)$$

where δ is the resolution of a grid, for example, 2.5 for 2.5°×2.5° grid. Each rectangle is divided onto 4 triangles by its diagonals. The center point is assigned the average value of its corresponding corners.

To build a contour line for level l_i, each triangle is intersected with the plane $p(lat, lon) = l_i$. The result of intersection (if it takes place) is a segment. A human eye perceives them as a continuous curve once drawn on a computer display but in fact it is a large number of independent elements. According to [4], there are 10 possible outcomes of the plane p and a triangle intersection (fig. 2).

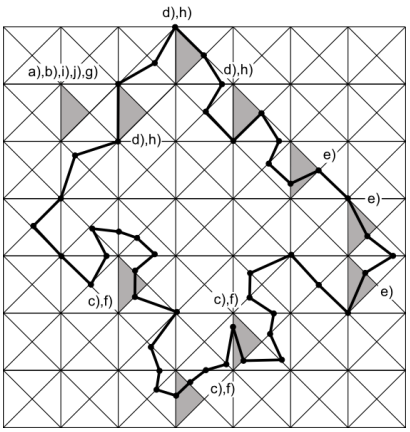

Figure 2 – Possible outcomes of plane and triangle intersection

For cases (a), (b), (i), (j), (g) CONREC does not generate segments. They occur when all triangle vertices lie below (a) or above (j) plane *p*, only a single vertice lies on a plane *p* and all the rest are below (b) or above (i) it. Lastly, all vertices may lie on the plane *p* (g).

Other cases result in a segment: two vertices lie below and one above plane *p* (c) or, vice versa, two vertices lie above and one below (f). Also, two vertices may lie on the plane *p* and one vertice below (d) or above it (h). The (e) case takes place when one of the vertices lie on the plane *p*, one above and one below it.

The input data to CONREC must be prepared in a special way to build segments for global latitude-longitude grid. There is only a single point at a pole because of longitude convergence. The value for a pole must be replicated for each longitude to form rectangles instead of triangles in order not to alter the CONREC algorithm. These rectangles will have two vertices with the same coordinates and values.

Also, longitude coordinates -180° and +180° must be considered equivalent. Any reference to -180° or +180° must retrieve data stored at +180°.

Segments glue algorithm

Let S_i denote the set of all segments for a single regular grid found by CONREC algorithm for level $l_i \in L$. The algorithm assumes that isolines do not intersect. For each segment in S_i, a bounding box is created to put it in R-Tree to accelerate search operations. Let *R-TREE*(*i*) be a constructed R-Tree for all segments from S_i.

While constructing R-Tree the following rule must be preserved. If one of the segment endpoints has longitude value equal to -180°, it must be changed to +180° if other endpoint has positive longitude value and must not be changed if its sign is also negative.

The algorithm *ISOLINES-GLUE* (fig. 4) takes as input the segments S_i and *R-TREE*(*i*). It yields the set of closed isolines I_i for level *i* and parts of isolines U_i that have gaps for the same level. Parts are merged into closed isolines with the algorithm *ISOLINES-GLUE-U* (fig. 6) presented later in this paper.

The algorithm represents an isoline *l* as closed polygon with a sequence of points $l = \langle p_0, p_1, \ldots, p_N \rangle$ where $p_0 = p_N$ and $p_i = (lat_i, lon_i)$ where lat_i and lon_i are latitude and longitude coordinates respectively for point p_i. Let *s*(1) and *s*(2) be the endpoints for the segment $s \in S_i$. Let also *l*[*i*] denote point p_i of isoline *l*.

The algorithm builds isolines sequentially, one at a time. It starts from an arbitrary segment that has not been marked yet as part of another isoline. New segments are attached only at one end of the isoline under construction (fig. 3).

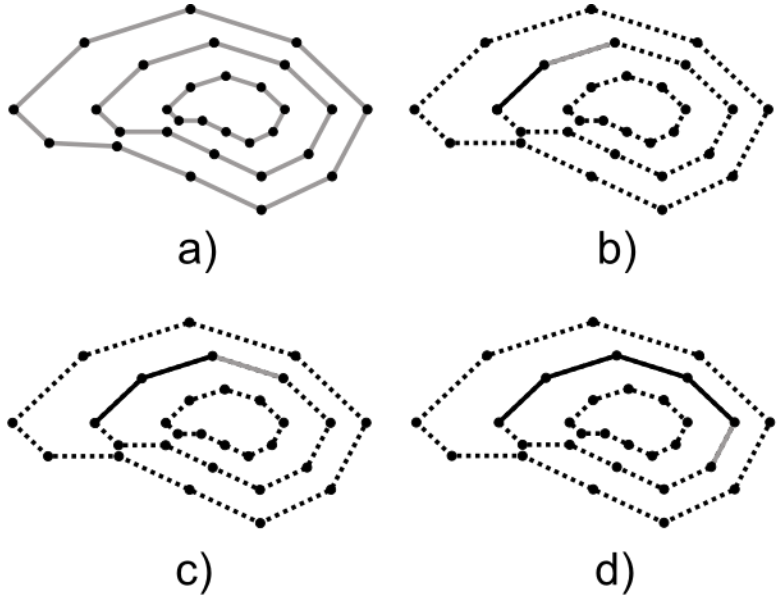

Figure 3 – Attachment of new segments to an isoline. Black – visited segments, grey – candidates for gluing, dotted – not visited segments

ISOLINE-GLUE: S_i, *R-TREE(i)* $\rightarrow$ I_i, U_i
$I_i \leftarrow \{\varnothing\}$, $U_i \leftarrow \{\varnothing\}$
while $S_i \neq \varnothing$
    $s \leftarrow S_i$ *// choose arbitrary segment from* S_i
    $S_i \leftarrow S_i \setminus \{s\}$
    $l \leftarrow \langle s(1), s(2) \rangle$ *// new isoline which is being built*

    $i \leftarrow 1$ *// last index in sequence l corresponding to point* $p_1 = s(2)$
    while $l[i] \neq l[0]$ *// while not closed*
        $C \leftarrow$ *R-TREE(i).neighbor-search(*$l[i]$*)* $\cap$ S_i
        $D \leftarrow \{ c \in C : dist(l[i], c(1)) \leq \varepsilon \vee dist(l[i], c(2)) \leq \varepsilon \}$

        if $|D| = 1$

0

        then $d \leftarrow D$
        else if $|D| \geq 1$
            then $d \leftarrow$ *heuristic-tie(D) (fig. 6)*
            else $U_i \leftarrow U_i \cup \{l\}$
                Break

        $l \leftarrow l + (dist(l[i], d(1)) \leq \varepsilon) \; ? \; d(1) : d(2)$
        $i \leftarrow i + 1$
    $I_i \leftarrow I_i \cup \{l\}$

Figure 4 – Segments glue algorithm

In line 3 the algorithm takes any segment which has not yet participated in gluing. In line 5 it creates new isoline with a single segment chosen previously at line 3. The loop in lines 6–15 seeks for segments to continue isoline l until it becomes closed or a special case (line 14) is encountered leaving it unclosed. Although it is only a part of an isoline it is nevertheless called isoline.

Fast neighbor search is performed using *R-TREE*(i) to determine segments located close to the point $l[i]$ and prune already used one (line 8). Only a segment with one of its endpoints equal to the point $l[i]$ may become a candidate for attachment to the isoline under construction. Function $dist(\cdot,\cdot)$ takes two points as its arguments and calculates Euclidian distance between them. A constant parameter ε is usually set to 0.01 and introduced to deal with inaccuracies taking place in floating point calculations.

Normally, one candidate segment must be found (line 10). Several candidates (line 12) are possible when two isolines touch each other in one point (fig. 5c). Note, that all of the segments shown on figure 6a may comprise a single isoline in reality. However, the grid resolution is insufficient in this case to determine the real situation. Thus, the most correct solution is to treat them as two separate isolines.

A heuristic rule is used to choose between the candidates. It selects a segment which endpoint is the farthest from the point $l[i–1]$. This endpoint must not be equal to $l[i]$ (determined similar as in line 16). On figure 5c, $l[i–1]$ = A, $l[i]$ = O and OD, OB, OC are candidate segments. Their endpoints not equal to $l[i]$ are D, B, and C respectively. The distance is measured between A and D, A and B, A and C. Finally, OD segment is chosen since point D is the farthest from point A = $l[i–1]$. This rule was devised after practical study of isolines touching each other. This case occurs frequently in real data, especially with slowly varying fields like mean sea level pressure.

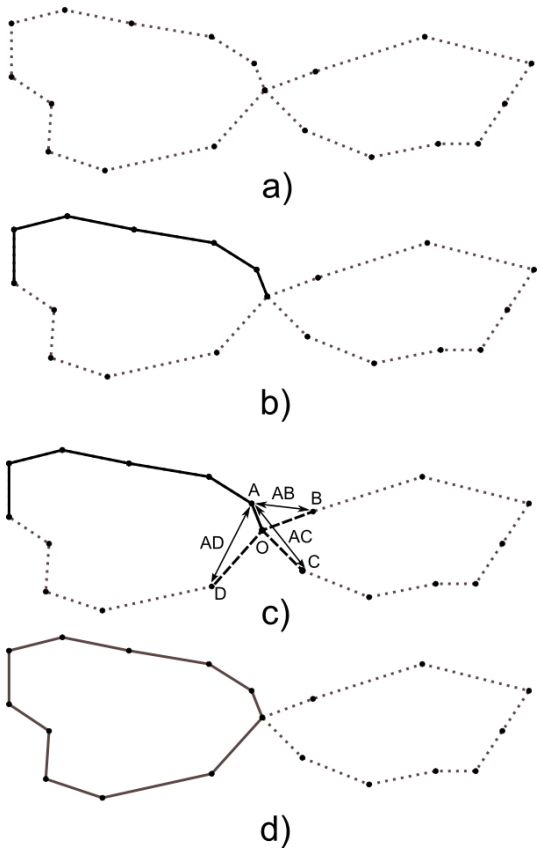

Figure 5 – Heuristic rule to break ties with several candidates

In case of $|D| = 0$ (no candidates for isoline continuation), it is added to the set of unclosed isolines (line 14). Ternary C-style operator *a*?*b*:*c* returns *b* if condition *a* is true or *c* otherwise (line 16).

Unclosed isolines occur due to absence of segments connecting isoline parts. Recall, that for certain cases CONREC generate only points (b), (i) or does not generate segments at all (g). While this is formally correct, for real data this causes gaps in an isoline curve (when several triangle vertices with distinct coordinates have equal values).

Consider the case when only one segment is absent for an isoline. Thus, U_i will contain at least two parts that will need to be glued with each other. This takes place since the algorithm stops once it does not find a segment to continue the isoline. However, the remaining segments belonging to the same isoline will be glued into a separate isoline since they are not allowed to attach to that part of the isoline that contains already used segments. The remaining part of the isoline may be also split into several parts depending on the segment endpoint from which the construction started.

However, if isoline construction starts from the segment having one of its endpoints equivalent to one of the endpoints of the missing segment, the first and the last isoline endpoints must be checked (line 13, fig. 6).

In line 7 the algorithm seeks for an isoline part to glue with isoline part *l*. It selects the closest isoline to *l* within constant distance μ. The *len*(*l*) function returns the number of elements in sequence *l* (number of points in isoline part). For an ERA-Interim [6] grid with 1.5°×1.5° resolution, μ is chosen to be $\sqrt{2}$ ×1.5. This is the maximum length of segment that could be added to an isoline by this algorithm. The maximum length of a segment built by

CONREC is 1.5. However, for real data none of the segments may be built for any of the triangles of a single rectangle when it has the same values in all of its endpoints.

```
ISOLINE-GLUE-U: U_i, I_i → I_i
while U_i ≠ ∅
    l ← U_i // get arbitrary unclosed isoline
    U_i ← U_i \ {l}

    i ← len(l) – 1
    while l[i] ≠ l[0]   // while not closed
        D ← { u ∈ U_i : dist(l[i], u[0]) ≤ μ ∨ dist(l[i], u[len(u) –1]) ≤ μ }

        if |D| ≥ 1
        then d ← D
        else if |l| ≤ 2
            then break
            else if |D| = 0
                then if dist(l[i], l[0]) ≤ μ
                    then d ← <l[0]>
                    else error

        l ← l + d
        i ← i + len(d) – 1
        U_i ← U_i \ {d}
    I_i ← I_i ∪ {l}
```

Figure 6 – Glue algorithm of unclosed isolines

If an isoline has only 2 points (line 10) it is totally removed. This happens for real data when small islands of a particular value exist. However, the grid resolution is insufficient to build an isoline with non zero area. These confluent isolines have no meaning to a person exploring data visually.

The algorithm reports an error when two or more successive segments are absent or when the data are incorrect (line 15). This case has not been observed.

Performance evaluation

The method was implemented on Java, embedded in Climate Wikience, and tested on the machine with characteristics shown in table 1.

Table 1. Machine characteristics

OS	RAM	Processor	Java ver.
Windows 7	2 GB	AMD Athlon II Dual-Core P320 (2.1 GHz)	1.6.0.26

Table 2 lists the time for each step of the algorithm, the number of segments generated by the CONREC and isolines number resulted from gluing the segments. Note, that the number of isolines is always significantly smaller (almost in 200 times) than the number of segments. Using polygons instead of separate segments considerably reduces load onto a visualization subsystem.

Mean sea level pressure ERA-Interim 1.5°×1.5° (240×120 points) grids were taken to evaluate the performance of the implemented method. The runtime is almost the same for each of these grids, thus, table 2 shows typical runtime values for a randomly taken grid.

Measurements were done for 4 different steps (the first row of table 2). The isolines levels were calculated using formulas (2) and (3). The min and max values were not fixed. For each grid minimum and maximum values that it contains were taken.

Table 2. Performance characteristics of the method implementation

	1000	500	250	100
CONREC, ms	13,63	19,27	74,28	356,25
R-Tree, ms	80,11	137,32	243,87	628,73
GLUE, ms	157,16	306,37	249,27	845,31
TOTAL, ms	250,90	462,96	567,42	1830,29
Segments, #	15358	30898	61332	154740
Isolines, #	86	170	332	859

The execution time for *ISOLINE-GLUE-U* algorithm is negligible and not shown. Typically, less than 1% of total isolines have missing segments.

With 100 Pa step isobars are very dense. In certain regions distance between two neighbor isobars reaches 20 km and less.

The execution time of each method stage reveals that a great deal of time is spent on the construction of R-Tree (fig. 7).

Two observations make possible to eliminate both the time required to construct the R-Tree and the time for neighbor search in *ISOLINE-GLUE* algorithm that uses it.

The first observation is about the nature of CONREC algorithm: only one segment per triangle may exist. The second is about nature of grid structure: all of the triangles have homogenous coordinates.

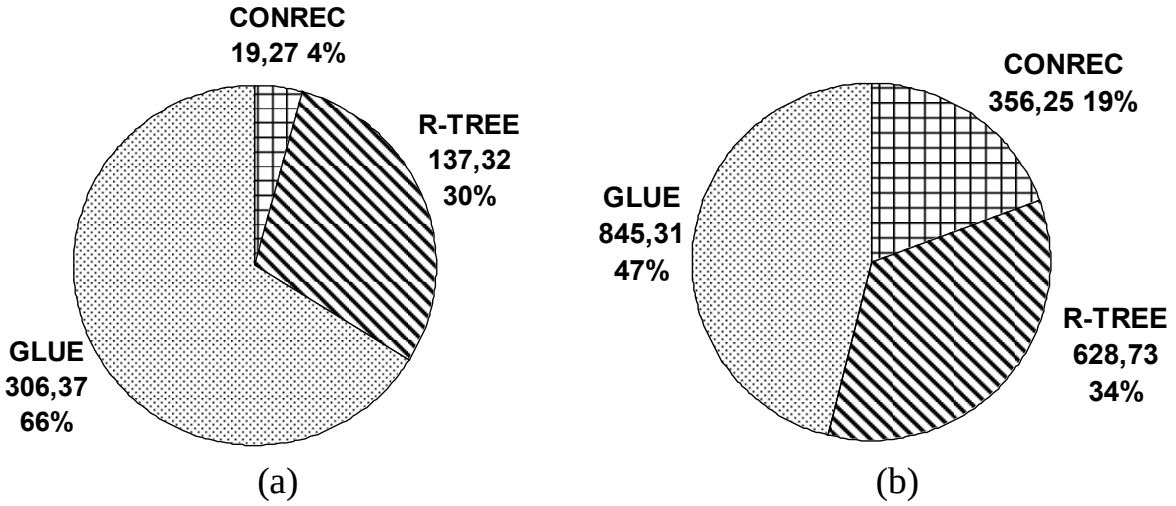

Figure 7 – Execution time of method stages for step 500 (a) and 100 (b)

Thus, a triangle containing point with given latitude and longitude coordinates may be easily located in O(1). A simple bucket data structure may be used with constant search time instead of R-Tree.

Conclusion

This paper presented isolines construction method and evaluated its performance on real data. Unlike other isoline plotting algorithms, the proposed method represents isolines by polygons instead of separate segments. The polygons are closed and their number is almost 200 times less than the number of segments for climate reanalysis grids.

This enables GIS operations on them as well as their efficient interactive exploration. Also, the method proposes heuristics to deal with cases that frequently occur in real data like guessing missing segments.

The proposed method is successfully used in Climate Wikience which constructs isolines on-the-fly in real-time in 3D for any grid in a climate reanalysis archive.

This paper does not contain colored 3D image of isolines due to grayscale printing of collected articles this paper belongs to. The reader is encouraged to try out the method in action himself. Climate Wikience is freely available at wikience.donntu.edu.ua.

Acknowledgements

This work was supported by Award No. UKM1-2973-DO-09 of the U.S. Civilian Research & Development Foundation (CRDF). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of CRDF.

Literature

1. Kalnay, E. et al., The NCEP/NCAR 40-Year Reanalysis Project. Bull. Amer. Meteor. Soc. (77), pp. 437–471, 1996.

2.Compo G.P. et al., The Twentieth Century Reanalysis Project. Part A., Q. J. R. Meteorol. Soc., pp. 137 - 128, January, 2011.
3.Contour line [Electronic resource] – Access method: http://en.wikipedia.org/wiki/Contour_line (09.04.2012).
4.CONREC: A Contouring Subroutine [Electronic resource] – Access method: http://paulbourke.net/papers/conrec/ (09.04.2012).
5. Rodriges Zalipynis R.A., Zapletin E.A., Averin G.V. The Wikience: Community Data Science. Concept and Implementation., Proc. of the 7th Intl. Scientific-Technical Conference "Informatics and Computer Technologies" (ICT–2011), Vol. 1, pp. 113–117, Donetsk, November 22–23, 2011.
6. Dee D. P. et al., The ERA-Interim reanalysis: configuration and performance of the data assimilation system, Q. J. R. Meteorol. Soc., Vol. 137 (656), pp. 553–597, 2011.