Offsetting Spherical Curves in Vector and Raster Form

Article *in* The Visual Computer · June 2018

DOI: 10.1007/s00371-018-1525-7

CITATIONS
7

READS
187

3 authors:

Troy Alderson
The University of Calgary
13 PUBLICATIONS **138** CITATIONS
SEE PROFILE

Ali Mahdavi-Amiri
The University of Calgary
38 PUBLICATIONS **347** CITATIONS
SEE PROFILE

Faramarz F. Samavati
The University of Calgary
158 PUBLICATIONS **2,018** CITATIONS
SEE PROFILE

CGI2018 manuscript No.
(will be inserted by the editor)

Offsetting Spherical Curves in Vector and Raster Form

Authors Withheld

Abstract In this paper, we present techniques for offsetting spherical curves represented in vector or raster form. Such techniques allow us to efficiently determine and visualize the region within a given distance of a spherical curve. Our methods additionally support multiresolution representations of the underlying data, allowing initial coarse offsets to be provided quickly, which may then be iteratively refined to the correct result. An example application of offsetting is also specifically explored in the form of improving the performance of inside/outside tests in the vector case.

Keywords Offset · Multiresolution · Spherical curves

1 Introduction

Our home — the Earth — is vast and ever-changing. Petabytes worth of geospatial data already fill servers across the Earth, and more continues to be generated every day and at increasingly high resolution. In order to manage and integrate all these data, a form of GIS called the Digital Earth framework has emerged, first proposed by former Vice President Al Gore [7]. Digital Earths use a 3D globe as a common reference model for geospatial data, and are often constructed atop a multilevel rasterization of the Earth known as a Discrete Global Grid System (DGGS) [12].

One of the core forms of data in a Digital Earth is geospatial vector data — sequences of connected points that define paths (e.g. rivers and roads) and boundaries (e.g. nation, lake, and ocean boundaries). Often these data are rasterized into a DGGS or map image (raster form), but can also be represented in terms of their original point sequences (vector form) to retain accuracy.

These data, in combination with robust geospatial queries, can be used to obtain important visual and numerical information about the Earth. For example, we can perform environmental assessments, such as in determining if a prospective oil well or waste site lies within a given distance of a body of water, or in determining the number of homes within a given distance of a wildfire. This type of assessment requires two types of queries: offsetting, also known as buffering, which determines the region that lies within a given distance of some geospatial vector; and inside/outside tests, which determine if a point lies within the region represented by a vector.

In the context of offsetting, the problem of big data reduces the utility of approaches that are based on precomputing and caching results. The space of all regions from which one may wish to offset is prohibitively large and not always known a priori, and the regions themselves can change over time. Combined with the arbitrariness of the distance used to define an offset, it becomes all but impossible to store a cached result for each one. This issue is compounded if we wish to precompute offsets at multiple levels of resolution.

Hence, we wish to develop algorithms to determine offset curves on demand and without the use of cached offset results, for the purposes of visualization or geospatial querying. The primary obstacles to this approach include managing the incredible size of many geospatial vectors (potentially consisting of millions of points) and in managing the spherical/non-Euclidean nature of geospatial data (which can distort the results of map-based and other Euclidean-based approaches). Ideally, such algorithms should return a result rapidly despite the vectors' size and minimize cartographic distortions.

One technique that can be used to manage vector size is to use a multiresolution data representation, such as a mip-map, wavelet transform, or DGGS. Such a representation allows a vector to be visualized or queried at a coarse resolution (for quick initial results) or at a high resolution on demand. In fact, a number of multiresolution representations, such as the wavelet transform for geospatial vectors from [1], can be calculated and stored without increasing the memory footprint of the original data.

In this paper, we present two algorithms for offsetting from geospatial curves on the sphere — one for curves represented in vector form, and another for curves represented in raster form. Each algorithm allows an approximation of the final result to be produced quickly, then refined as needed, by supporting multiresolution representations of the underlying data. An example application of spherical multiresolution and offsetting is explored in the form of an efficient algorithm for inside/outside tests in the vector case.

The rest of the paper is organized as follows. We review works related to Digital Earth and curve offsetting in Section 2, and present the terminologies used throughout this paper in Section 3. Then, we provide an overview of our algorithms for performing spherical curve offsetting in the vector case (Section 4) and the raster case (Section 5). This is followed by results in Section 6 and a conclusion to the paper in Section 7.

2 Related Work

Curves on the surface of the Earth (aka geospatial vectors) play a prominent role in GIS applications, including Digital Earth frameworks. These frameworks, which use a three-dimensional model of the Earth as a common reference for geospatial data, are underpinned by a data structure known as the Discrete Global Grid System [6] (see [12] for a survey). A Discrete Global Grid System (or DGGS) discretizes the Earth into a hierarchical set of cells, in which each cell represents a region of the Earth at a particular resolution.

DGGSs are in part characterized by the shape of their cells. These cells are typically formed by refining the faces of an initial polyhedron, which are then projected to the surface of a sphere or ellipsoid (e.g. the WGS84 ellipsoid [15]) using one of the many projections studied in cartography (e.g. Snyder's projection [22]). Conversions between different cell types, as described in [13], allow for interoperability between different DGGSs.

There are three primary means of representing and rendering a vector in a DGGS [24]: via rasterization into the cells of the DGGS, as in [9]; via representation in vector form, as in [18]; or via shadow volumes, as in [19]. For vectors remaining in vector form, spherical multiresolution representations may be used to reduce and losslessly restore the resolution of the vectors on demand in order to handle their large size. In [8], Grohs and Wallner present a framework for multiresolution that interpolates the original vector points, based on the use of the exponential map [4]. A non-interpolatory scheme based on simple geometric operations was presented in [1].

Curves on the surface of an object also play an important role in computer-aided design, where offsetting is a well-studied and important operation. As surveyed in [17] and [11], offsetting for CAD applications can be categorized into five main topics: calculating offsets for Bezier and B-Spline curves, approximating complex offsets, resolving self-intersections in the offsets, determining offsets for curves on surfaces (geodesic offsets), and calculating general offsets.

Examples of works that calculate geodesic offsets include the works of [16], which finds offsets for curves on a NURBS surface, and [23], which produces offsets for curves on a triangle mesh. In contrast, our work calculates offsets on a sphere that is represented implicitly rather than explicitly, and for which geodesics can be computed exactly with a simple closed form equation.

Our method for offsets in the vector case is closely related to the works of [10] and [5], as these works calculate offsets for polyline curves. In [10], Liu et al. calculate offsets for each line segment of the polyline, then resolve self-intersections in the result. Choi and Park [5] avoid artifacts in the final offset by eliminating problematic sections of the original curve before offsetting using a robust *local interference* test. The raw offset for the resulting (discontinuous) curve has no local self-intersections, and a similar process is executed on the raw offset to eliminate global self-intersections. Unlike these methods, our polylines are embedded on the surface of a sphere at multiple resolutions rather than in Euclidean space at a single resolution.

Algorithms that draw lines of a given thickness, such as Murphy's modification [14] of Bresenham's algorithm [3], bear some relation to offsetting on a raster grid. However, in general such algorithms do not translate well to arbitrary cell tilings/grids on a spherical manifold, particularly as these grids necessarily contain some irregularities. Furthermore, they are typically defined for a single resolution, and are therefore not designed to handle the complex hierarchies that can exist between non-square raster cells of different resolutions.

3 Terminology

Consider a sphere $S = (\mathbf{c}, r)$ of radius r centered at point $\mathbf{c}$. A great circle is given by the intersection of S with a plane passing through $\mathbf{c}$. A small circle is given by the intersection of S with a plane whose distance to $\mathbf{c}$ is greater than zero but less than r. The distance between two points p_0 and p_1 on S is defined as the arc length between them — $dist(p_0, p_1) = r\theta$, where θ is the angle between p_0 and p_1. Spherical linear interpolation (or SLERP) can be used to traverse the great circle arc between two points p_0 and p_1 [21], and is defined as

$$\mathrm{SLERP}(p_0, p_1, u) = \frac{\sin[(1-u)\theta]}{\sin(\theta)} p_0 + \frac{\sin(u\theta)}{\sin(\theta)} p_1.$$

A Discrete Global Grid System (DGGS) defined on S is a hierarchical set of grids $\{G_0, G_1, \cdots, G_{max}\}$ of increasingly fine resolution (i.e. $|G_i| < |G_{i+1}|$) up to an artbitrary maximum resolution, max. Each grid G_i is composed of a set of cells $g_j^{[i]}$ that discretize the Earth, and to which parent-child relationships are assigned.

A spherical curve (or geospatial vector) $C(u)$ on S is given by connecting a set of $n \geq 2$ points $P = \{\mathbf{p}_0, \mathbf{p}_1, \cdots, \mathbf{p}_{n-1}\}$ located on S with great circle arcs. For each value of $0 \leq u \leq 1$, $C(u)$ returns a point on the spherical curve. A spherical offset curve $O(v)$ that offsets from $C(u)$ by a distance of d is given by a set of points R located on S and connected with small circle arcs. For each value of u (resp. v), there exists a value of v (resp. u) such that $dist(C(u), O(v)) = d$.

For simplicity, throughout this paper we will assume that the sphere S is centered at the origin ($\mathbf{c} = \mathbf{0}$). The algorithms in this paper can be used with other spheres $S = (\mathbf{c}, r)$ by translating the sphere (by $-\mathbf{c}$) to lie at the origin, applying our algorithms, and then restoring the sphere to its original location (i.e. translating by $\mathbf{c}$).

4 Offsetting in the Vector Case

In this section, we present our method for determining and representing offset curves for geospatial data in vector form (see Section 4.1). In order to handle the potentially large size of the data, we can apply multiresolution techniques to the spherical offset curves (see Section 4.2) to be used for efficient visualization or inside/outside tests (see Section 4.3). In Section 4.4, we describe how to fix self-intersections in the results of our method.

4.1 Single Resolution (Vector Case)

Whereas a spherical curve $C(u)$ is defined by a set of points P connected by great circle arcs, a spherical offset curve $O(v)$ is defined by a set of points R connected by *small* circle arcs (see Figure 1). We refer to the points in R as "junction points", as they mark the point at which two small circle arcs meet. Determining the spherical offset curve $O(v)$ largely depends on determining the set of junction points R and the centers of the small circles, which we will call Q.

Two types of small circle arcs exist in a spherical offset curve: small circle arcs that correspond to offsetting from a single point $\mathbf{p}_i \in P$, and small circle arcs that correspond to offsetting from a great circle arc between consecutive points $\mathbf{p}_i$ and $\mathbf{p}_{i+1} \in P$. Given R and Q, all points in $O(v)$ can be found by spherically interpolating between the junction points. That is, given $\mathbf{r}_i$ and $\mathbf{r}_{i+1} \in R$ and the center $\mathbf{q}_i \in Q$ of the small circle arc that connects them, we can determine the points on the small circle arc as

$$SLERP(\mathbf{r}_i - \mathbf{q}_i, \mathbf{r}_{i+1} - \mathbf{q}_i, t) + \mathbf{q}_i \tag{1}$$

by varying $0 \leq t \leq 1$.

Surprisingly, and despite the non-linear nature of the spherical offset curve $O(v)$, the junction points R and the small circle centers Q can be found using Euclidean vector operations.

Consider, firstly, the small circles that offset from a single point $\mathbf{p}_i \in P$ (Figure 1c). The points on this type of small circle are equidistant to the circle's center $\mathbf{q}_{2i}$, but are also equidistant to $\mathbf{p}_i$. Consequently, for a sphere centered at the origin ($\mathbf{c} = \mathbf{0}$), the small circle center $\mathbf{q}_{2i}$ is a scaled version of $\mathbf{p}_i$ (refer to Figure 2a). In fact,

$$\begin{aligned} \mathbf{q}_{2i} &= r \cdot cos\left(\tfrac{d}{r}\right) \cdot \tfrac{\mathbf{p}_i}{||\mathbf{p}_i||} \\ &= cos\left(\tfrac{d}{r}\right) \cdot \mathbf{p}_i. \end{aligned} \tag{2}$$

Now consider the small circles that offset from a great circle arc between $\mathbf{p}_i$ and $\mathbf{p}_{i+1} \in P$ (Figure 1d). The points on this type of small circle are, similarly, equidistant to both the circle's center $\mathbf{q}_{2i+1}$ and to all points on the great circle. As a result, when $\mathbf{c} = \mathbf{0}$, the small circle center $\mathbf{q}_{2i+1}$ can be found by scaling the normal of the great circle's plane, say $\mathbf{n}_i = \frac{\mathbf{p}_{i+1} \times \mathbf{p}_i}{||\mathbf{p}_{i+1} \times \mathbf{p}_i||}$ (refer to Figure 2b). That is,

$$\mathbf{q}_{2i+1} = r \cdot sin\left(\frac{d}{r}\right) \cdot \mathbf{n}_i. \tag{3}$$

Finally, and most interestingly, consider the junction points $\mathbf{r}_i$. Each junction point lies at the intersection of two small circles, say those centered at $\mathbf{q}_i$ and $\mathbf{q}_{i+1}$. Since $\mathbf{q}_i$ and $\mathbf{q}_{i+1}$ are orthogonal to each other and have lengths of $\sin(\frac{d}{r})$ and $\cos(\frac{d}{r})$, it so happens that when $\mathbf{c} = \mathbf{0}$ the junction point can be found by adding $\mathbf{q}_i$ and $\mathbf{q}_{i+1}$ together (see Figure 2c), i.e.

$$\mathbf{r}_i = \mathbf{q}_i + \mathbf{q}_{i+1}. \tag{4}$$

In summary, we can determine a spherical offset curve at a distance of d from $C(u)$ using the following process:

1. Scale all the points in P by $cos(\frac{d}{r})$ to obtain the $\mathbf{q}_{2i}$ (Equation 2).
2. Find the plane normals for each of the great circle arcs in $C(u)$ and scale them by $r \cdot sin(\frac{d}{r})$ to obtain the $\mathbf{q}_{2i+1}$ (Equation 3).
3. Sum each consecutive pair of points in Q to obtain the $\mathbf{r}_i$ (Equation 4).
4. Apply $SLERP$ to the $\mathbf{r}_i$ in order to determine all points on the offset curve (Equation 1).

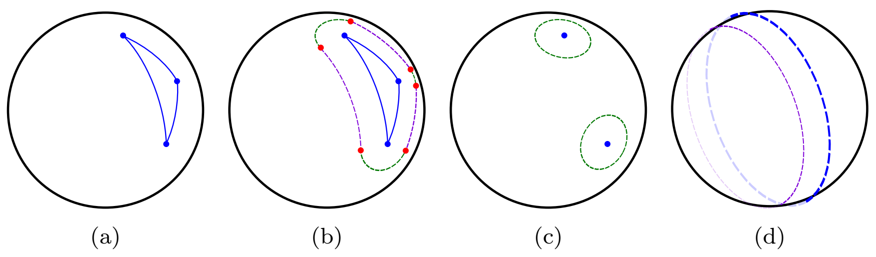

Fig. 1: (a) A spherical curve consists of a set of points connected by great circle arcs. (b) A spherical offset curve consists of a set of junction points (shown as red dots) connected by small circle arcs. (c) Some small circle arcs (shown in green) correspond to offsetting from a point. (d) Some small circle arcs (shown in purple) correspond to offsetting from great circle arcs.

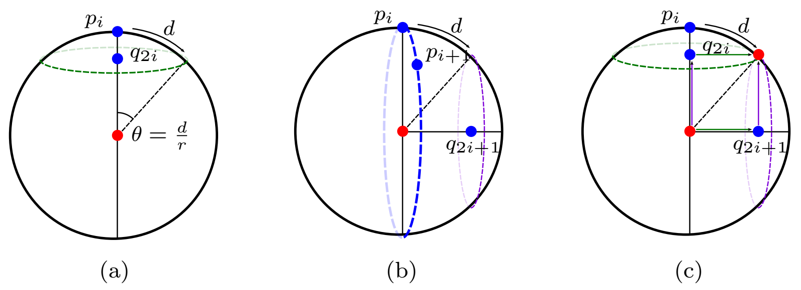

Fig. 2: (a) The center $\mathbf{q}_{2i}$ of the small circle that offsets from a point $\mathbf{p}_i$ is a scaled version of $\mathbf{p}_i$. (b) The center $\mathbf{q}_{2i+1}$ of the small circle that offsets from the great circle arc between $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ is a scaled version of the great circle's plane normal. (c) The junction points can be found by adding the small circle centers together.

Much as a spherical curve $C(u)$ can be sufficiently described using the set of points P, a consequence of Equation 4 is that the spherical offset curve $O(v)$ can be efficiently described using only the small circle centers Q. Furthermore, a given offset curve $O(v)$ can be altered to lie at any distance $d \geq 0$ simply be re-scaling the small circle centers in Q.[1]

4.2 Multiresolution (Vector Case)

Our method for finding spherical offset curves is linear in terms of the number of points used to define the spherical curve (i.e. $|P|$), however it can become slow if that number is exceedingly large. In order to combat this, we can use spherical multiresolution techniques in order to reduce and restore the resolution of the spherical curves on demand.

A spherical multiresolution technique — such as that of [8] or [1] — allows the set of points P that define $C(u)$ to be decomposed into a hierarchy of progressively smaller point sets $P = P^{[0]}, P^{[1]}, \cdots, P^{[k]}$ and associated details $D^{[1]}, \cdots, D^{[k]}$. The number of points in $P^{[i+1]}$ is typically half that of $P^{[i]}$, and the storage requirements for $P^{[i]}$ and its details $D^{[i]}$ are typically exactly the same as the storage requirements for the original set P (i.e. $|P^{[i]}| + |D^{[i]}| = |P|$). Given a set of points $P^{[i]}$ and its details $D^{[i]}$, any other level of the hierarchy can be reproduced, including the original set $P = P^{[0]}$.

In this section, we define a spherical multiresolution framework for offset curves $O(v)$ as well, such that the resolution of the offset curves can be reduced and restored on demand. Such a framework allows us to approximate the offset curve to obtain a quick initial result, which may then be refined for greater accuracy. Knowledge of the point sets $P^{[i]}$ is not required in order to use the framework, except to create the initial offset.

Suppose we are given a decomposed point set $P^{[k]}$ and details $D^{[k]}$ (see Figure 3b). Using these coarse points, we can apply the process in Section 4 to determine a coarse offset curve with small circle centers $Q^{[k]}$ (Figure 3c), which partially consists of scaled versions of the points in $P^{[k]}$. As the scaling factor is constant for a given distance d, those scaled points (the $\mathbf{q}_{2i}$) lie on a common sphere of radius $r \cdot cos(\frac{d}{r})$.

Hence, we can apply the spherical multiresolution technique to the $\mathbf{q}_{2i}$ (using appropriately scaled versions of the details $D^{[k]}$) in order to obtain scaled ver-

[1] At $d = 0$, $O(v)$ is equivalent to $C(u)$. In such a case, the set of small circle centers Q becomes the point set P interspersed with the zero vector $\mathbf{0}$ (representing the centers of the great circle arcs).

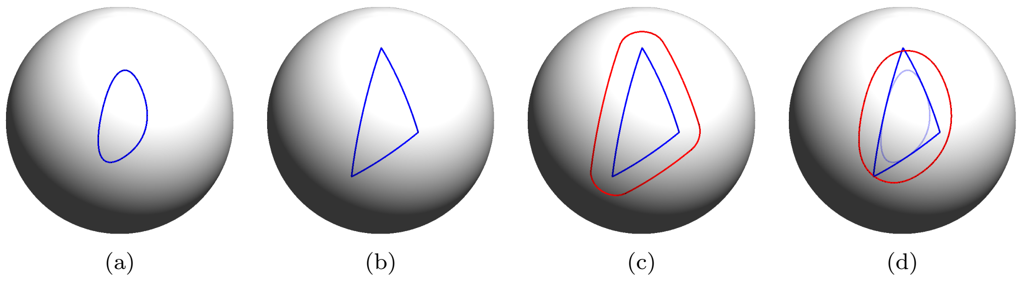

Fig. 3: (a) High resolution spherical curve. (b) The curve in (a) after reducing the resolution. (c) Coarse offset calculated using the coarse curve in (b). (d) The offset can be refined to produce the accurate high resolution offset, without restoring the resolution of the coarse curve.

sions of the points in $P^{[k-1]}$. By recalculating the $\mathbf{q}_{2i+1}$, we then obtain the small circle centers $Q^{[k-1]}$ for a higher resolution version of the offset curve. This method can be repeated as needed to obtain any desired resolution for the offset curve (shown in Figure 3d).

In summary, spherical multiresolution may be applied to an offset curve (given $Q^{[k]}$ and the set of details $D^{[1]}, \cdots, D^{[k]}$) through the following process:

1. Extract the $\mathbf{q}_{2i}$ (every second point in $Q^{[k]}$) into a set of points Q'.
2. Scale the details in $D^{[k]}$ by $cos(\frac{d}{r})$.
3. Apply spherical multiresolution to Q' using $D^{[k]}$ to obtain a new set of $\mathbf{q}_{2i}$.
4. Repeat steps 2 and 3 using $D^{[k-1]}, \cdots, D^{[1]}$ until the desired resolution is reached.
5. Recalculate the $\mathbf{q}_{2i+1}$ using Equation 3.
6. Using Equations 4 and 1, determine the points on the offset curve.

In addition to helping manage the incredible size of geospatial vectors, multiresolution can also be used to speed up inside/outside tests.

4.3 Inside/Outside Tests

Inside/outside tests examine whether a given point $\mathbf{p}$ lies inside or outside of a particular shape, and are essential to the environmental assessment tasks mentioned in Section 1. In the case of a Euclidean polygon, the traditional test is to cast a ray from the point and count the number of times the ray intersects the polygon. If the number of intersections is even, then $\mathbf{p}$ is outside; otherwise, it is inside (see [20]).

Generalizing this algorithm to spherical curves $C(u)$ and offset curves $O(v)$ presents some challenges, however. Firstly, a ray on the sphere forms a great circle, which will intersect $C(u)$ and $O(v)$ an infinite number of times. Secondly, calculating intersections with the small circle arcs of $O(v)$ is expected to be costly. This is exacerbated by the third issue, which is the significant number of intersection tests that must be performed given a high resolution spherical curve.

The first problem can be solved by selecting an appropriate stopping point for the ray, converting it into a finite great circle arc. In [2], this stopping point is selected as an arbitrary point $\mathbf{x}$ known to be inside $C(u)$ (resp. outside $C(u)$) that is not antipodal to $\mathbf{p}$. If the great circle arc between $\mathbf{p}$ and $\mathbf{x}$ intersects $C(u)$ an odd number of times, then $\mathbf{p}$ is outside (resp. inside); otherwise it is inside (resp. outside).

The second problem can be addressed using an alternative approach that entirely avoids the computation of small circle intersections. Consider that $\mathbf{p}$ will be inside $O(v)$ if and only if it is inside $C(u)$ or within the offset distance d of $C(u)$. Given the small circle centers $\mathbf{q}_i$ of the offset curve $O(v)$, a simple set of tests can be employed to determine whether $\mathbf{p}$ is within distance d of $C(u)$.

Consider a small circle formed by offsetting from $\mathbf{p}_i \in C(u)$, centered at $\mathbf{q}_{2i}$. As $\mathbf{p}$ is located on the sphere, it is within distance d of $\mathbf{p}_i$ if it lies above the plane of the small circle (see Figure 4a). Hence, $\mathbf{p}$ is in the "offset region" of $\mathbf{q}_{2i}$ if

$$(\mathbf{p} - \mathbf{q}_{2i}) \cdot \mathbf{q}_{2i} \geq 0 \tag{5}$$

or, equivalently

$$\mathbf{p} \cdot \mathbf{p}_i \geq r^2 \cdot \cos\left(\frac{d}{r}\right). \tag{6}$$

For small circles produced by offsetting from the great circle arc between $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ (centered at $\mathbf{q}_{2i+1}$), we can test if $\mathbf{p}$ is within the wedge shape formed by $\mathbf{p}_i$, $\mathbf{p}_{i+1}$, the plane of the small circle, and its reflection (see Figure 4b). That is, $\mathbf{p}$ is in the offset region of $\mathbf{q}_{2i+1}$ if

$$\begin{aligned} &|\mathbf{p} \cdot \mathbf{q}_{2i+1}| \leq \mathbf{q}_{2i+1} \cdot \mathbf{q}_{2i+1} \text{ AND} \\ &\mathbf{p} \cdot (\mathbf{p}_i \times \mathbf{q}_{2i+1}) \geq 0 \text{ AND } \mathbf{p} \cdot (\mathbf{q}_{2i+1} \times \mathbf{p}_{i+1}) \geq 0 \end{aligned} \tag{7}$$

or, equivalently

$$\begin{aligned} &|\mathbf{p} \cdot \mathbf{n}_i| \leq r \cdot \sin\left(\tfrac{d}{r}\right) \text{ AND} \\ &\mathbf{p} \cdot (\mathbf{p}_i \times \mathbf{n}_i) \geq 0 \text{ AND } \mathbf{p} \cdot (\mathbf{n}_i \times \mathbf{p}_{i+1}) \geq 0. \end{aligned} \tag{8}$$

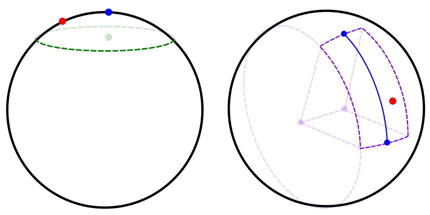

Fig. 4: (a) A point is within the offset region of another point if it lies above the associated small circle. (b) A point is within the offset region of an arc if it lies in the wedge shape formed by the arc's endpoints and two small circle planes.

The third and final problem can be solved using multiresolution. The core idea is to find a simplified version of $C(u)$ that can be queried as above yet still provide a correct answer. To do this, we keep track of the error between each simplified arc and the original arcs it approximates. For a given point $\mathbf{p}$ and stopping point $\mathbf{x}$, we reconstruct those arcs whose distance to $\mathbf{p}$ or $\mathbf{x}$ is less than its error.[2]

For the sake of simplicity, we illustrate this method using the linear/Faber multiresolution scheme from [1]. Refer to Figure 5. The reverse subdivision step consists of a subsampling (i.e. $\mathbf{p}_i^{[k]} = \mathbf{p}_{2i}^{[k-1]}$, for any $k > 0$). The difference between each discarded point, $\mathbf{p}_{2i+1}^{[k-1]}$, and the midpoint of its neighbours, $\mathrm{SLERP}(\mathbf{p}_{2i}^{[k-1]}, \mathbf{p}_{2i+2}^{[k-1]}, \frac{1}{2})$, is encoded into a detail $\mathbf{d}_i^{[k]}$. Hence, the resolution of the arc between $\mathbf{p}_i^{[k]}$ and $\mathbf{p}_{i+1}^{[k]}$ can be increased by reconstructing $\mathbf{p}_{2i+1}^{[k-1]}$ (using $\mathbf{d}_i^{[k]}$) and inserting it between $\mathbf{p}_i^{[k]}$ and $\mathbf{p}_{i+1}^{[k]}$.

Let $e_i^{[k]}$ denote the error that the simplified arc between $\mathbf{p}_i^{[k]}$ and $\mathbf{p}_{i+1}^{[k]}$ makes with the arcs it approximates. In this case, that is

$$\begin{aligned} e_i^{[k]} &= \max(e_{2i}^{[k-1]}, e_{2i+1}^{[k-1]}) \\ &\quad + \mathrm{dist}\left(\mathbf{p}_{2i+1}^{[k-1]},\ \mathrm{SLERP}(\mathbf{p}_{2i}^{[k-1]}, \mathbf{p}_{2i+2}^{[k-1]}, \tfrac{1}{2})\right), \end{aligned}$$

where $e_i^{[0]} = 0$ for all i. We can determine if $\mathbf{p}$ is within a distance $e_i^{[k]}$ of the arc between $\mathbf{p}_i^{[k]}$ and $\mathbf{p}_{i+1}^{[k]}$ (with normal $\mathbf{n}_i^{[k]}$) using a combination of Equations 6 and 8

$$\begin{cases} \mathbf{p} \cdot \mathbf{p}_i^{[k]} \geq r^2 \cdot \cos\left(\frac{e_i^{[k]}}{r}\right) & \text{if } \mathbf{p} \cdot (\mathbf{p}_i^{[k]} \times \mathbf{n}_i^{[k]}) \leq 0 \\ \mathbf{p} \cdot \mathbf{p}_{i+1}^{[k]} \geq r^2 \cdot \cos\left(\frac{e_i^{[k]}}{r}\right) & \text{if } \mathbf{p} \cdot (\mathbf{n}_i^{[k]} \times \mathbf{p}_{i+1}^{[k]}) \leq 0 \\ \mathbf{p} \cdot \mathbf{n}_i^{[k]} \leq r \cdot \sin\left(\frac{e_i^{[k]}}{r}\right) & \text{otherwise.} \end{cases} \tag{9}$$

Note that $e_i^{[k]}$ should be clamped to the range $[0, \pi]$ to avoid erroneous results due to the periodicity of sine and cosine.

We can additionally augment the algorithm in order to work on offset curves at the same time. If the distance between $\mathbf{p}$ and the simplified arc between $\mathbf{p}_i^{[k]}$ and $\mathbf{p}_{i+1}^{[k]}$ is less than $d - e_i^{[k]}$ (clamped to $[0, \pi]$), then $\mathbf{p}$ is in the offset curve $O(v)$. The simplified arc is reconstructed if the distance is less than $d + e_i^{[k]}$ (clamped to $[0, \pi]$).

4.4 Resolving Self-Intersections

In order to resolve self-intersections in the offset curves, we use a simple quadratic time (i.e. $O(n^2)$) algorithm that checks each small circle arc for intersections with any other small circle arc.

While generalizations of more efficient algorithms from Euclidean space could be used (such as that of [5]), even with double-precision computations, numerical instability can cause noticeable errors in the result. Furthermore, the impact to performance of the quadratic time algorithm can be mitigated by using multiresolution to display a quickly computed and cleaned-up coarse offset whilst a higher resolution version is determined in the background.

The process of fixing self-intersections in an offset curve works as follows. For each small circle arc of the offset (centered at $\mathbf{q}_i$ for some i), we attempt to intersect the arc with every other arc in the offset, excepting direct neighbours (i.e. those centered at $\mathbf{q}_j$, for all $j \neq i-1, i, i+1$). Among these, we identify the value of j that is furthest from i, i.e. j satisfying

$$\max_j (j - i) \bmod \frac{|Q|}{2},$$

and the corresponding intersection point, $\hat{r}$, made by the arcs centered at $\mathbf{q}_i$ and $\mathbf{q}_j$. Then, all arcs and junction points between i and j are removed, and $\hat{r}$ is set

[2] Reconstructing arcs close to $\mathbf{x}$ can be skipped in most practical scenarios through an appropriate choice of $\mathbf{x}$. For instance, if $C(u)$ is limited to a single hemisphere, then $\mathbf{x}$ can be chosen as the antipode of the centroid of $C(u)$, which is known to lie outside of $C(u)$ and its simplifications.

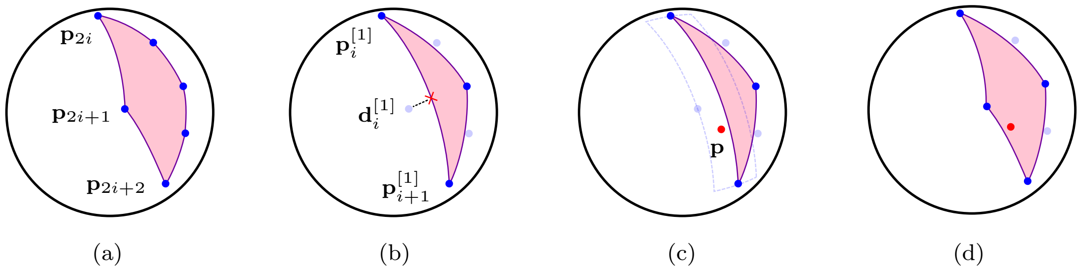

Fig. 5: (a) A spherical curve consisting of 6 points. (b) The curve after one application of reverse subdivision (subsampling). Details are found as the difference between a discarded point and its neighbours' midpoint. (c) Given point $\mathbf{p}$, we determine if $\mathbf{p}$ lies within the error region of each simplified arc. (d) Arcs with $\mathbf{p}$ in their error region are reconstructed.

as an endpoint for the arcs centered at $\mathbf{q}_i$ and $\mathbf{q}_j$. Refer to Algorithm 1 for pseudocode.

Algorithm 1 Algorithm for removing self-intersections from an offset curve.

```
Input: Small circle centers $Q = \{\mathbf{q}_0, \mathbf{q}_1 \cdots \mathbf{q}_{2n-1}\}$, junction
points $R = \{\mathbf{r}_0, \mathbf{r}_1 \cdots \mathbf{r}_{2n-1}\}$

for $i = 0$ to $|Q| - 3$ do
    $m = \frac{|Q|}{2}$, $j$ = null, $\hat{r}$ = null
    for $k = i + 2$ to $|Q| - 1$ do
        $\hat{temp}$ = intersection of the arcs centered at $\mathbf{q}_i$ and $\mathbf{q}_k$
        if $\hat{temp}$ exists and either $\hat{r}$ is null or $(k - i)$ mod $m >$
        $(j - i)$ mod $m$ then
            $j = k$, $\hat{r} = \hat{temp}$
        end if
    end for
    if $j$ is not null then
        if $(j - i) < m$ then
            Remove $\mathbf{q}_{i+1}, \cdots, \mathbf{q}_{j-1}$ from $Q$
            Remove $\mathbf{r}_{i+2}, \cdots, \mathbf{r}_{j-1}$ from $R$
            Replace $\mathbf{r}_{i+1}$ with $\hat{r}$ in $R$
        else
            Remove $\mathbf{q}_0, \cdots, \mathbf{q}_{i-1}$ and $\mathbf{q}_{j+1}, \cdots, \mathbf{q}_{2m-1}$ from $Q$
            Remove $\mathbf{r}_0, \cdots, \mathbf{r}_{i-1}$ and $\mathbf{r}_{j+2}, \cdots, \mathbf{r}_{2m-1}$ from $R$
            Replace $\mathbf{r}_{j+1}$ with $\hat{r}$ in $R$
            Reset $i$ to 0
        end if
    end if
end for
return $Q$, $R$
```

To calculate if an intersection exists between two small circle arcs and determine their intersection point, $\hat{r}$, one can intersect the planes of the two small circles to find a line, and then intersect the line with the sphere S. Any intersection point(s) between the line and the sphere must be verified to lie on both small circle arcs by, e.g., comparing the angles between the intersection point(s) and the endpoints of the arc. In the event that two intersection points lie on both arcs, the one that has the smallest arc distance to the first endpoint should be selected as $\hat{r}$.

Note that this method assumes that no more than $\frac{|Q|}{2}$ arcs must be removed between any given pair of $\mathbf{q}_i$ and $\mathbf{q}_j$, which is the case for most spherical curves, including the examples in this paper. Additionally, we do not produce new curves to represent holes in the offset region. The implementation of a general method that handles all scenarios is left for future work.

5 Offsetting in the Raster Case

In order to facilitate rendering and some efficient queries (e.g. inside/outside tests), geospatial vectors are often rasterized onto an image or onto the cells of a DGGS (see Figure 6). When an offset curve is requested, rather than re-request and re-rasterize the original vector data, it would be useful to apply the offset directly on the rasterized vector.

In this section, we present an algorithm that produces offsets for spherical curves rasterized into the cells of a raster grid or DGGS, without reference to the original vector data. We first present an algorithm that applies this offset in a single grid (Section 5.1), which we then use to quickly and efficiently produce offsets throughout the multiresolution hierarchy of a mip-map or DGGS (Section 5.2).

5.1 Single Resolution (Raster Case)

An algorithm that offsets from a rasterized spherical curve must calculate the distance between raster cells and the rasterized curve: if the distance is less than the offsetting distance d, then the cell is included in the offset, else it is outside. In order to produce an accurate and efficient offsetting method, two questions must be addressed. The first concerns how the distance

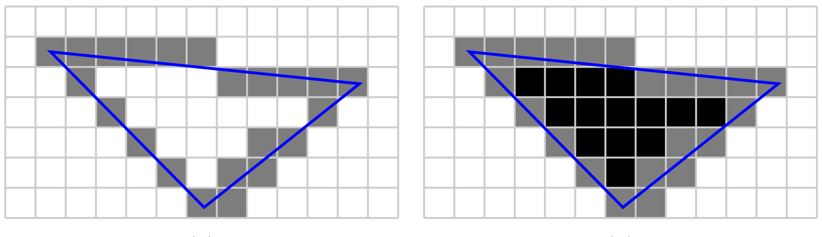

Fig. 6: (a) Like Euclidean curves, a spherical curve may be rasterized into an image or into the cells of a DGGS. (b) Cells contained in the vector may be marked as inside for efficient inside/outside queries.

between the curve and a raster cell can be accurately calculated if we do not possess the original vector data. The second concerns which raster cells must be checked for inclusion, as we wish to determine all the cells in the offset without checking every cell on the entire Earth.

In order to handle both of these questions, our method uses region-growing (which restricts the checked cells to only those cells in the offset and their immediate neighbours) and two distance checks — a minimum distance and a maximum distance — to determine which cells are definitely included in the offset and which cells are only possibly included (see Figure 7).

Consider the ordered sequence of cells $\{g_0, g_1, \cdots, g_k\}$ that are intersected by the spherical curve $C(u)$. Let d_g denote the length of the longest diagonal in any cell (or the smallest diameter such that any cell is contained in a circle of diameter d_g).

We begin by calculating an initial offset region around the first intersected cell g_0 by region-growing (see Figure 8). A cell g_i is contained in the offset region of g_0, and called a “contained cell”, if the distance between their centroids is less than $d_{min} = d - \frac{d_g}{2}$. A cell g_i is possibly in the offset region of g_0, and called a “frontier cell”, if the distance between their centroids is greater than d_{min} and less than $d_{max} = d + d_g$. The neighbours of frontier cells and contained cells are the only cells that need to be checked for inclusion in the offset of g_0.

We then move to the next intersected cell g_1 and calculate its offset region by evaluating frontier cells and their neighbours for inclusion using d_{min} and d_{max} (see Figure 9). If a frontier cell does not lie in the offset region of g_1 (i.e. its distance is greater than d_{max}), then the cell is marked as an “inactive” frontier cell, and it and its neighbours are no longer evaluated for inclusion in offset regions. We continue moving to the next intersected cell and evaluating active frontier cells for inclusion until the final cell g_k has been reached and the complete offset has been calculated.

Note that self-intersections in the offset will be resolved automatically, but the algorithm will break down should the set of active frontier cells become empty (which may occur if, e.g., the spherical curve $C(u)$ has self-intersections).

5.2 Multiresolution (Raster Case)

The method described in Section 5.1 will calculate the offset for $C(u)$ on a single grid G, however DGGSs are composed of several grids G_i arranged in a hierarchy. In each grid G_i, the curve $C(u)$ intersects a different sequence of cells $\{g_0^{[i]}, g_1^{[i]}, \cdots, g_{k_i}^{[i]}\}$. We require that if a cell $g_j^{[i]}$ is considered to be intersected by $C(u)$, then its parent $g_l^{[i-1]}$ must also be considered to be intersected by $C(u)$. Additionally, each grid G_i has it own individual cell size $d_g^{[i]}$.

Rather calculate the offset from scratch on each grid G_i, we can use the hierarchy between cells to achieve an efficient algorithm that operates hierarchically. To do so, we first redefine $d_g^{[i]}$ to be the smallest diameter such that any cell in G_i, *and all of its descendants*, can be contained in a circle of diameter $d_g^{[i]}$. While $d_g^{[i]}$ remains trivially unchanged for cells that are congruent throughout the hierarchy (i.e. their descendants cover the exact same area), $d_g^{[i]}$ will be different for incongruent cell shapes, such as hexagons.

Given $d_g^{[i]}$ defined thusly, notice that if a cell is marked as contained by the offset, then all of its descendants are contained by the offset as well. Similarly, if a cell is known to be completely outside of the offset, then all of its descendants are as well. Hence, the only cells in grid G_i $(i > 0)$ that need to be evaluated for inclusion in the offset are those descended from the frontier cells (active and inactive) found in grid G_{i-1}.

Therefore, we can efficiently determine offsets hierarchically in the raster case using the following process. First, we determine the offset in the coarsest grid G_0 using the method in Section 5.1. This provides an initial offset result that can be used for visualization/query purposes until finer offsets are found. Then, we iteratively determine the offset in the next grid G_i by using the descendants of the active frontier cells from G_{i-1},

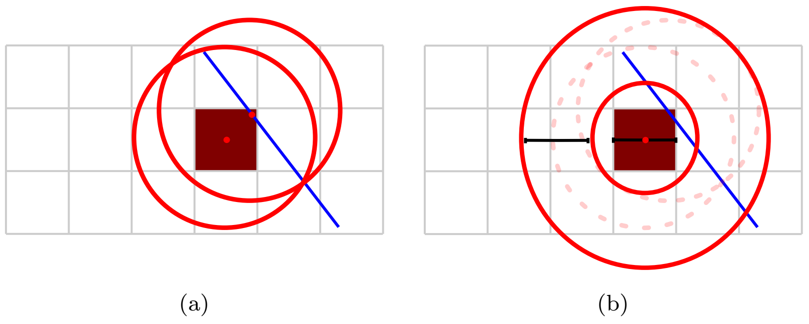

Fig. 7: (a) Without the original vector (shown in blue), discrepancies can result from using the centroids of the intersected cells to evaluate distances. (b) Two distance checks — a minimum and a maximum distance — can be used to evaluate which cells are definitely in the offset and which are possibly in the offset.

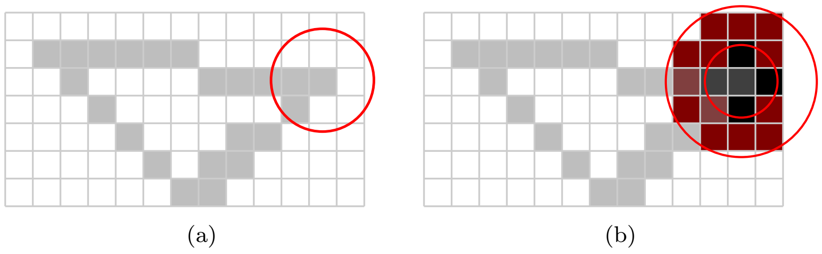

Fig. 8: (a) An initial offset region is calculated for a single cell. (b) Cells are marked as contained by the offset (shown in black) or as possibly contained by the offset (shown in red), based on two distance checks. Red cells are also known as frontier cells.

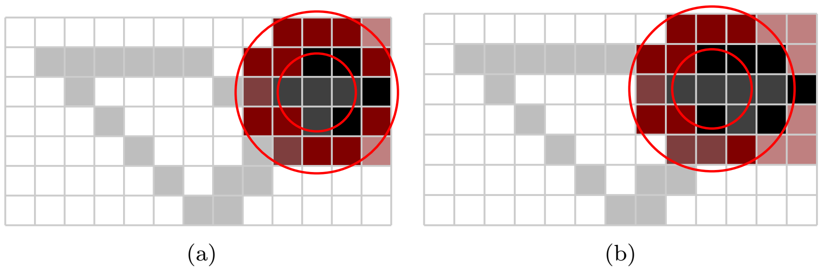

Fig. 9: (a), (b) The offset region for the next intersected cell is calculated. Frontier cells that are not in the offset region of the new cell become inactive (shown in light red).

rather than $g_0^{[i]}$, as an initial seed for the offset region, and by limiting our search space to the descendants of the active and inactive frontier cells from G_{i-1}.

6 Results

In this section, we present several results produced by applying our offset and inside/outside algorithms to spherical vectors.

Figures 10 through 12 illustrate the results of our algorithm in the vector case. In Figure 10, offsets are shown for spherical curves at a single resolution. Self-intersection artifacts in concave areas can be resolved using the methods from Section 4.4 (see Figure 10d). Figures 11 and 12 display instances of offsetting from real geospatial data — here, vectors representing the mainland borders of Brazil and China at different resolutions.

Figure 13 shows some results from the raster case. Offsets for a spherical curve rasterized into a hierarchical latitude/longitude grid system are shown at multiple resolutions. Using our methods, one can find a coarse offset at the coarsest resolution and then rapidly determine the offset at finer resolutions by limiting the search space to the frontier cells.

In Table 1, we present runtime performance results from our multiresolution approach to inside/outside tests. All tests were executed in the Unity game development

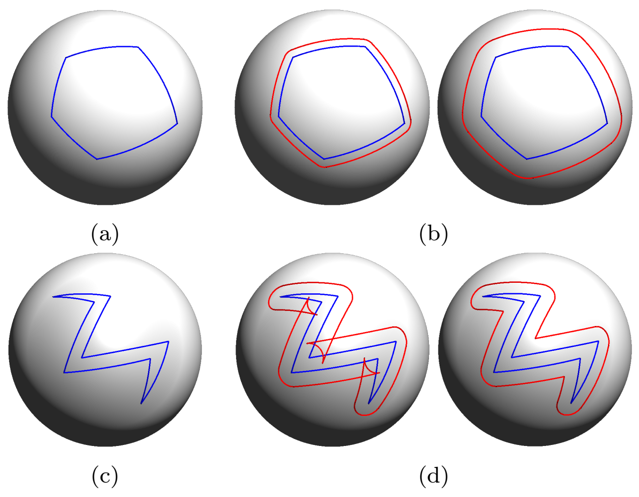

Fig. 10: (a), (c) Spherical curves each defined as a set of points connected with great circle arcs. (b) Offsets for the curve in (a), at different distances. (d) Offsets for the curve in (c), with and without self-intersections.

environment on a 64-bit Windows 10 machine with an Intel Core i7-6700K CPU and 16 GB of RAM. In each test run, a set of 10000 points were distributed uniformly (in the latitude/longitude grid) in and around the border of Mexico. A spherical curve representing this border (consisting of nearly 3400 vertices) and a corresponding offset curve (at a distance $d = 0.02$) were decomposed using multiresolution techniques to various levels of simplification. The runtime of the inside/outside tests on these curves, averaged over the 10000 points, appears to be nearly halved with each level of decomposition applied to the curve. Tests remained accurate to within half a percentage point (inaccuracies likely due to floating point errors).

Table 1: Average runtime performance of inside/outside tests.

Number of Decompositions	Spherical Curve Time	Offset Curve Time
0	1.84 ms (100 %)	1.42 ms (100 %)
1	1.10 ms (60.0 %)	0.91 ms (64.1 %)
2	0.56 ms (30.4 %)	0.46 ms (32.4 %)
3	0.28 ms (15.2 %)	0.23 ms (16.2 %)
4	0.14 ms (7.8 %)	0.12 ms (8.5 %)
5	0.07 ms (3.8 %)	0.07 ms (4.9 %)

7 Conclusion and Future Work

We have presented two algorithms that calculate offset curves from geospatial curves represented in two different data formats. These algorithms support multiresolution in the underlying data and can determine offsets for a desired resolution. An efficient algorithm for inside/outside tests in the vector case was additionally introduced, which makes use of both spherical multiresolution and offsetting to achieve significant speed gains.

For future work, one could focus on generalizing existing, robust offset methods from Euclidean space in such a way as to resolve self-intersections with minimal numerical instability. Another interesting direction for future work could be to examine methods by which to determine and represent offset curves on an ellipsoid.

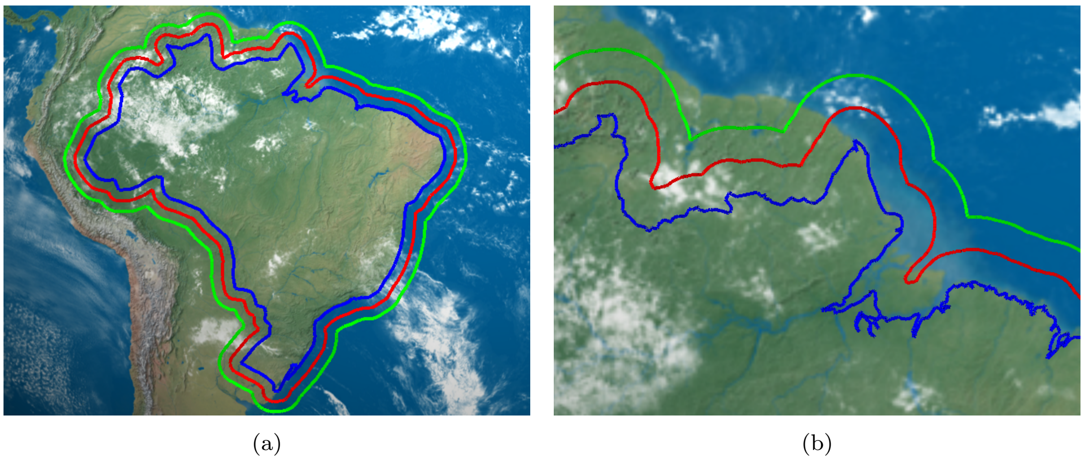

Fig. 11: Spherical curve representing the mainland border of Brazil with two offsets, one at $d = 0.02$ (shown in red) and one at $d = 0.04$ (shown in green). (a) Brazil and offsets at a low resolution. (b) Brazil and offsets at a high resolution.

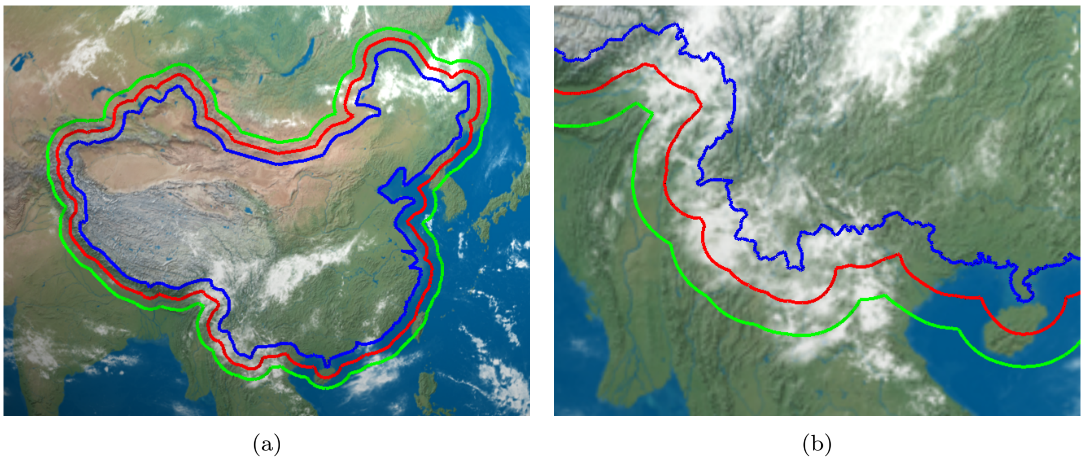

Fig. 12: Spherical curve representing the mainland border of China with two offsets, one at $d = 0.02$ (shown in red) and one at $d = 0.04$ (shown in green). (a) China and offsets at a low resolution. (b) China and offsets at a high resolution.

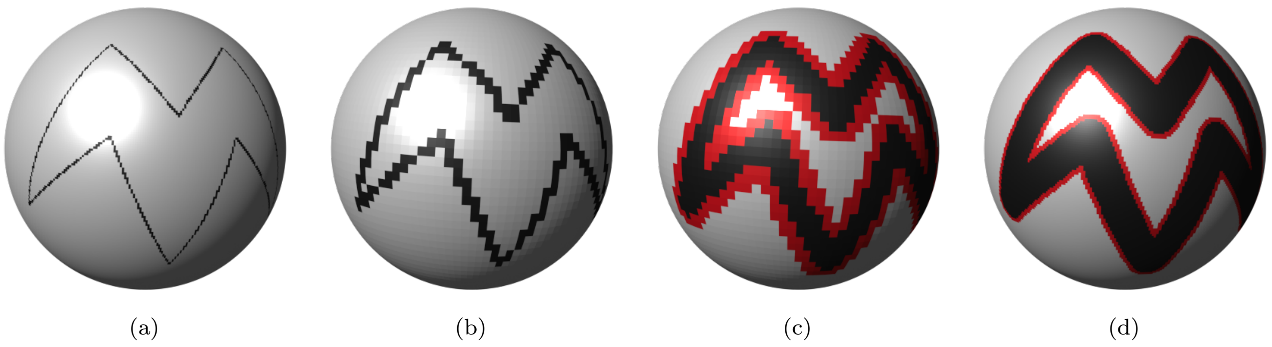

Fig. 13: (a) Spherical curve rasterized into a high resolution latitude/longitude grid. (b) The curve from (a) in a coarser grid. (c) Offset of the curve in the coarse grid. Contained cells are shown in black, frontier cells in red. (d) Offset of the curve in the high resolution grid. Cells descended from the cells in (c) that were marked as contained or outside are also marked as contained or outside.

References

1. Alderson, T.F., Mahdavi-Amiri, A., Samavati, F.F.: Multiresolution on spherical curves. Graphical Models **86**, 13–24 (2016)
2. Bevis, M., Chatelain, J.L.: Locating a point on a spherical surface relative to a spherical polygon of arbitrary shape. Mathematical Geology **21**(8), 811–828 (1989)
3. Bresenham, J.E.: Algorithm for computer control of a digital plotter. IBM Systems Journal **4**(1), 25–30 (1965)
4. do Carmo, M.P.: Differential Geometry of Curves and Surfaces, chap. 4-6. Prentice-Hall, Inc. (1976)
5. Choi, B.K., Park, S.C.: A pair-wise offset algorithm for 2D point-sequence curve. Computer-Aided Design **31**, 735–745 (1999)
6. Goodchild, M.F.: Discrete global grids for digital Earth. In: Proceedings of the 1st International Conference on Discrete Global Grids (2000)
7. Gore, A.: The Digital Earth: Understanding our planet in the 21st century. Speech (1998). California Science Center. Los Angeles, CA, USA
8. Grohs, P., Wallner, J.: Definability and stability of multiscale decompositions for manifold-valued data. Journal of the Franklin Institute **349**(5), 1648–1664 (2012)
9. Kersting, O., Döllner, J.: Interactive 3D visualization of vector data in GIS. In: Proceedings of the 10th ACM International Symposium on Advances in Geographic Information Systems, GIS '02, pp. 107–112 (2002)
10. Liu, X.Z., Yong, J.H., Zheng, G.Q., Sun, J.G.: An offset algorithm for polyline curves. Computers in Industry **58**(3), 240–254 (2007)
11. Maekawa, T.: An overview of offset curves and surfaces. Computer-Aided Design **31**(3), 165–173 (1999)
12. Mahdavi-Amiri, A., Alderson, T., Samavati, F.: A survey of Digital Earth. Computers & Graphics **53, Part B**, 95–117 (2015)
13. Mahdavi-Amiri, A., Harrison, E., Samavati, F.: Hierarchical grid conversion. Computer-Aided Design **79**, 12–26 (2016)
14. Murphy, A.S.: Line thickening by modification to Bresenham's algorithm. IBM Technical Disclosure Bulletin **20**(12), 5358–5366 (1978)
15. National Geospatial-Intelligence Agency: World Geodetic System. `https://www.nga.mil/ProductsServices/GeodesyandGeophysics/Pages/WorldGeodeticSystem.aspx`. [Online; accessed 30-March-2017]
16. Patrikalakis, N.M., Bardis, L.: Offsets of curves on rational B-spline surfaces. Engineering with Computers **5**(1), 39–46 (1989)
17. Pham, B.: Offset curves and surfaces: A brief survey. Computer-Aided Design **24**(4), 223–229 (1992)
18. Schneider, M., Guthe, M., Klein, R.: Real-time rendering of complex vector data on 3D terrain models. In: Proceedings of the 11th International Conference on Virtual Systems and Multimedia, VSMM '05, pp. 573–582 (2005)
19. Schneider, M., Klein, R.: Efficient and accurate rendering of vector data on virtual landscapes. In: Proceedings of the 15th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, WSCG '07, pp. 59–64 (2007)
20. Shimrat, M.: Algorithm 112: Position of point relative to polygon. Communications of the ACM **5**(8), 434 (1962)
21. Shoemake, K.: Animating rotation with quaternion curves. In: Proceedings of SIGGRAPH 1985, pp. 245–254 (1985)
22. Snyder, J.P.: An equal-area map projection for polyhedral globes. Cartographica **29**(1), 10–21 (1992)
23. Xin, S.Q., Ying, X., He, Y.: Efficiently computing geodesic offsets on triangle meshes by the extended Xin-Wang algorithm. Computer-Aided Design **43**(11), 1468–1476 (2011)
24. Zhou, M., Chen, J., Gong, J.: A virtual globe-based vector data model: Quaternary quadrangle vector tile model. International Journal of Digital Earth **9**(3), 230–251 (2016)