BUFFER GENERATION USING SPHERICAL GEOMETRY

Abhinav Dayal

IDV Solutions Inc. 5913 Executive Drive, Suite 320, Lansing, MI 48911

1. INTRODUCTION

Buffering [2] is an important area of spatial analysis in GIS applications. The goal of buffering is to determine a neighborhood around a certain point, line or polygon feature based on some constant or variable distance measure. Major challenges for buffer computation are accuracy and computational complexity. Popular buffer generation algorithms [1][3][4] use parallel lines for emulating fixed distance. While this may be true for small distances on surface of earth, but for large distances the lines curve along the spherical surface of the earth.

There exists a non-linear relationship between distance and variations in latitude and longitude. Therefore, use of parallel lines to represent buffer along a feature would cause inaccuracies with increase in buffer size. We generate buffer with aid of spherical geometric computations. We reduce the set of line segments for intersection by using a conjoining line segment buffer approach.

We derive the final buffer by a linear point traversal over all the resulting points. We also use extended concavity tests to address the problem where large buffer size would cause buffer lines corresponding to successive concave input line segments to not intersect.

2. METHODOLOGY

As shown in Figure 1**Error! Reference source not found.**, we generate buffer in following steps:

Simplification: Given the input curve as a series of line segments, we first simplify it with a simple point elimination test. Given a start point, we eliminate all successive points till we reach a point more than a threshold distance from the start point. The new point becomes the start point.

Point Insertion for Accuracy: For lines spanning large distances, we add intermediate points so as to closely approximate the variations of distance.

IBB Generation: We then compute the Initial Buffer Boundary (IBB). Instead of parallel lines and circle approach mostly used [1][3][4], we use spherical geometry computations. For each end point on a line segment we find two points at bearings ±90° from the initial bearing of the line segment. These points give us lines corresponding to the parallel lines in the lines and circle approach. Similar to [1] we then compute intersections and finally compute the final buffer. However in the method used by [1] they find concavity at the middle point for any three consecutive points and check for intersection of the buffer line segments. Moreover, for certain buffer sizes, the buffer lines for two consecutive concave line segments may not intersect and the underlying procedure may fail. We address this by extending the concavity test to further line segments until the concavity holds true (Figure 2).

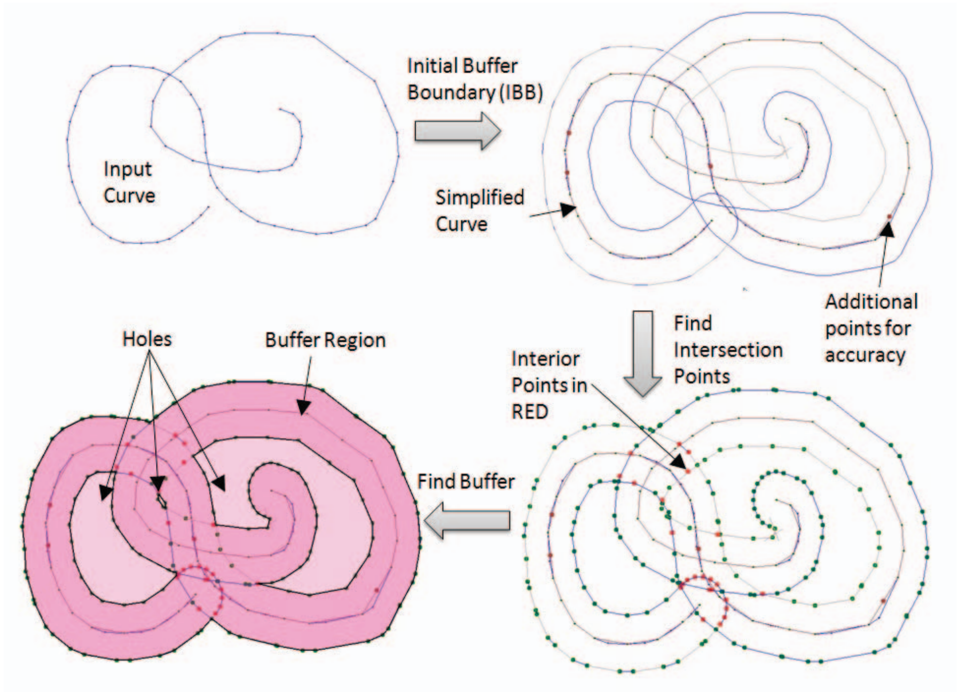

Figure 1. Buffer Generation Methodology.

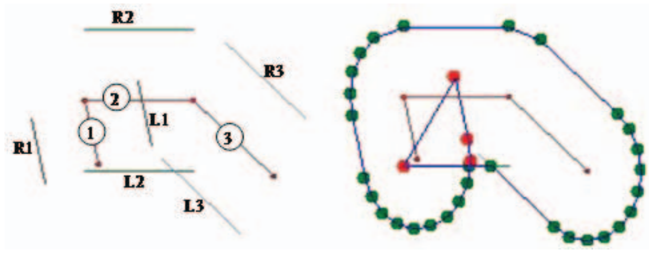

Figure 2. Inner Buffer lines corresponding to successive concave input line segments do not necessarily intersect. [Top] inner segments L1 and L2 for input line segments 1 and 2 do not intersect. [Bottom] Initial Buffer Boundary for this input.

Finally, for convex curves at certain end point, instead of using points on a circle centered at the end point of line segment with radius of buffer distance, we find points at a given distance and bearing from the end point using standard spherical geometry computations.

Finding Intersections: Once IBB is generated for each line segment, we compute the intersections between line segments in the IBB. We store separately all the intersection points that are not interior points. For each line segment we maintain a list of points (including the end points) on which that line segment intersects with any other line segment. The points in this list are stored in a sorted order from their squared Euclidean distance from the start of that line segment. We also maintain for each intersection/end point a list of line segments that containing that point in their list of intersection/end points.

Buffer Traversal: Starting with an intersection point on any IBB with min longitude value, we select a next point based on a simple cross product test. For two line segments $l1$ and $l2$, that intersect at the given point, if $l1 \times l2 > 0$, we choose next point on $l1$, else we choose next point on $l2$. We repeat this till we reach the start point completing the outer boundary of the buffer. For all visited points we remove them from list of intersection points. Since a buffer can have holes we repeat same procedure with the remaining points in the list. Here successful loop detection is indicated if during traversal we do not encounter an interior point.

3. RESULTS AND CONCLUSION

Results show that, use of IBB reduces almost by 70-90% line segments in a brute force approach for buffer generation. Intersection tests are further reduced by intersecting line segments of two IBBs only if the bounding boxes on the respective IBB's intersect. With extended concavity tests, we eliminate scenarios where inner buffer lines of concave consecutive line segments do not intersect. We use the world rivers data set that consists of 161 rivers of the world with a total of 29518 line segments. There was a 77.6% reduction in the line segments of generated IBB, as compared to those generated by brute force methods. Memory usage shows a clear trend towards decrease in memory with increase in buffer size. Even with 100K lines in IBB, it took about 6 seconds to compute the final buffer.

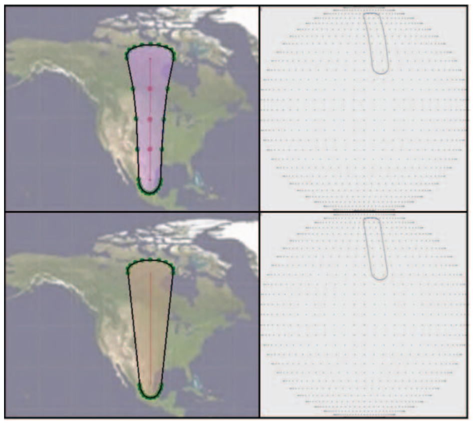

Figure 3. Improving correctness by additing additional points.

Finally, just to reemphasize, using spherical geometry ensures that "parallel" lines are parallel on spherical surface of the earth and not a rectangular plane as shown in Figure 3. It becomes even more important with increase in the span/size of the buffer. For example in Figure 3 (both top and bottom), the buffer starts thin at bottom and widens towards the top, in the rectangular projection (*Platte Carre*) for the actual spherical buffer that has parallel lines as shown in Figure 3 (top right).

Also notice how the algorithm introduces intermediate points in order to maintain accuracy. Figure 3 (bottom left) shows the resulting buffer in absence of intermediate points, which is not correct as evident from Figure 3 (bottom right), which shows how the buffer does not even fall on top of the surface of earth, rather it appears to be a tunnel from one point to another.

11. REFERENCES

[1] Peng DONG, Chongjun YANG, Xiaoping RUI, Liqiang ZHANG, Qimin CHENG. An Effective Buffer Generation Method. GIS [J]. Proc. IGARSS, 2003.

[2] Huang Xingyuan,Tang Qin. Geographical Information Systems [M]. Beijing: Higher Education Press, 1989.

[3] Wu Hehai. Problem of Buffer Zone Construction in GIS[J]. Journal of Wuhan Technical University of Surveying and Mapping, Vol..22 No 4, pp.358~365,1997.

[4] Yuan Wen, Cheng Chengqi. A New Buffer Generation Algorithm Based on Primitive[J]. Geomatics World, No.2, pp.13~17, 2002.