Closed Loop Simplification

Release 0.00

David Doria

June 29, 2011

Army Research Laboratory, Aberdeen MD

Abstract

This document presents an implementation of an algorithm to find a low edge-count approximation of a complex, discrete, 2D closed contour. This implementation is based on the algorithm described in "Using Aerial Lidar Data to Segment And Model Buildings" and "A Bayesian Approach to Building Footprint Extraction from Aerial LIDAR Data."

The code is available here: https://github.com/daviddoria/ClosedLoopSimplification

Contents

1 Introduction

This document presents an implementation of an algorithm to find a low edge-count approximation of a complex, discrete, 2D closed contour. Our goal is to represent the outline of an object in a simple fashion. This type of algorithm is commonly found in applications involving building detection from aerial LiDAR data.

This implementation is based on the algorithm described in [1] and [2].

2 Algorithm

2.1 Input

The input to this algorithm is an ordered set of points describing a closed contour. In Figure 1 we show a synthetic example of such a data set. The contour is visualized by joining adjacent points on the contour with an edge (a line segment).

Figure 1: A closed 2D contour.

2.2 Contour Simplification

It is often desirable to reduce a contour to a simpler approximation of the same shape. For example, in the case of building detection, we often want to represent a building as a rectangle (4 line segments). Depending on the density of the input, the boundary at this point could be represented by hundreds of line segments, mostly describing the noise in the data. To reduce the number of line segments in the boundary, we use the following procedure:

- Create a directed graph of the contour using exactly the vertices of the rough boundary as the vertices and the ordered line segments of the rough boundary as directed edges.
- Attempt to add every other possible edge (from each vertex to every other vertex ahead of it in the contour ordering) only if the new edge passes a straightness test.
- The straightness test is described in [1] as the sum of the distances from every point between the two end points to the proposed edge. We have found it more convenient to set this threshold based on the average of these distances, as the length of the edges then is not a factor in determining the straightness criteria.

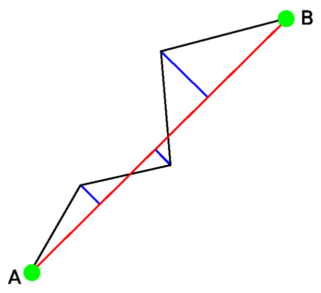

Figure 2: The straightness computation of a proposed edge between vertices A and B. Black: the rough boundary. Red: the proposed edge. Blue: the distances from all points on the boundary between A and B to the proposed edge.

- By finding any full loop shortest path on this graph, we will have significantly improved (i.e. reduced the line count of) our boundary. However, it is not a well posed graph theoretic problem to ask for the shortest path from a vertex to itself (a loop) and expect a non-zero answer (i.e. the shortest path from a vertex to itself is to not move at all!). To remedy this, we add the ordered vertices around the boundary twice, as well as duplicate the edges on this second loop of vertices. Now we can ask for the shortest path from vertex i to vertex $i+N$ where N is the original number of vertices in the rough boundary.
- If we compute this shortest path from any random vertex, we will have a solution to our original problem of finding a low-line segment count approximation to the boundary. However, the choice of starting vertex actually can change the solution, though often only slightly. Because of this, we find the shortest path starting at all vertices, and choose the shortest one as our solution.

[Demonstrate the case where the starting point is not on the best shortest path. This is why we must find all shortest paths from all points, not just one.]

3 Demonstration

4 Code Snippet

5 Future Work

In the polygonal approximation, there is a minimum straightness parameter which must be set which has a major impact on the resulting simplification. Removing the need to manually specify this parameters would make this algorithm more robust to different data types, as well as provide the best possible results on any particular data set.

References

[1] Wang, O., *Using Aerial Lidar Data to Segment And Model Buildings*. University Of California Santa Cruz Masters Thesis, 2006 1, 2.2

[2] Wang. O, Lodha. S, Helmbold, D., *A Bayesian Approach to Building Footprint Extraction from Aerial LIDAR Data*. 3D Data Processing, Visualization, and Transmission 2006 1