Segment Voronoi diagrams in CGAL

Menelaos I. Karavelas
Computer Science and Engineering Department
University of Notre Dame, U.S.A.
mkaravel@cse.nd.edu

Abstract

Voronoi diagrams are one of the most fundamental and useful constructs in computational geometry. In this talk we focus on the Euclidean Voronoi diagram of segments on the plane and the corresponding CGAL implementation.

The Voronoi diagram of segments in CGAL is designed after the CGAL 2D triangulations and makes use of CGAL's Triangulation Data Structure. The algorithm implemented is incremental and allows for generic data, i.e., there is no restriction as to whether the input segments intersect or not. Upon insertion of a site (a point or a segment) the algorithm finds the region of conflict of the new site with the existing Voronoi diagram, and then repairs the Voronoi diagram accordingly. In order to find the conflict region the algorithm needs to find the nearest neighbor of the site to be inserted. This is either done by a simple walk on the Voronoi diagram or by making use of an additional data structure, the Voronoi diagram hierarchy. The Voronoi diagram hierarchy is a hierarchy of Voronoi diagrams with the property that the diagram at each level only contains a randomly chosen subset of the sites in the lower level diagram; the bottom-most level is the segment Voronoi diagram for the entire set of sites. As a result our implementation offers for free the capability of performing fast nearest neighbor queries on the Voronoi diagram.

The user has the ability to control several parameters of the segment Voronoi diagram CGAL package. Choices include:

1. whether or not to use the hierarchy of Voronoi diagrams,
2. what traits and data structure to use,
3. how the predicates are to be computed, and finally,
4. indicate whether or not intersecting sites are to be supported or not.

The latter choice is exploited by the algorithm to reduce the internal memory/storage requirements, as well as for a more efficient implementation of the predicates. The CGAL package offers various models for all of the above parameters. We have implemented a model for the required data structure, as well as various models for the traits class that provide additional flexibility to the user. All traits classes are parameterized by a CGAL kernel, allowing the user to choose the desired number type. Moreover, the user can indicate what type of arithmetic operations are to be used in conjuction with the chosen number type. Finally, we have implemented models of the traits concept that perform arithmetic filtering, in addition to the simpler traits that assume the existence of an exact CGAL kernel.