Kinetic Data Structures: Animating Proofs Through Time

Julien Basch* João Comba† Leonidas J. Guibas‡ John Hershberger§
Craig D. Silverstein¶ Li Zhang||

Kinetic Data Structures (KDS for short) are a new class of data structures aimed at keeping track of attributes of interest in systems of moving objects [1, 2]. In this video we illustrate the principles of KDS design in the context of some classical geometric problems, such as the calculation of the convex hull or closest pair of moving points in the plane. In an ordinary simulation the positions of the points are advanced by a fixed time step, and then the attribute of interest in incrementally recomputed. Since the combinatorial structure of the attribute changes irregularly, it is quite difficult to select a time-step size that avoids both oversampling and undersampling the system. Unlike such fixed step methods, a kinetic data structure performs an event-driven simulation where only events relevant to the attribute of interest are generated and processed. The result is a simulation that is always accurate and with a computation cost close to the minimum possible. This video presents animations for and contains the description of a few two-dimensional kinetic geometric algorithms:

- Convex Hull ([1]):

  We compute a static convex hull for a set of points as follows: we divide the points randomly into two groups (red and blue), compute the convex hull of each group recursively, and then merge the two hulls. During this process, we create certificates for the hull's correctness. At the top level each certificate is an oriented triangle connecting the red and blue sub-hulls — a certificate triangle verifies that one of its vertices is not on the convex hull. Altogether, the certificates from all levels of the recursion prove the correctness of the convex hull. When motion begins, each certificate remains valid until the triangle it represents degenerates into a line; this certificate failure constitutes a kinetic event. In order to process this event the kinetic structure modifies the certificate structure so that it once again is a valid proof of the current hull — and adjusts the convex hull as necessary in the process.

- Closest Pair of Points ([1]):

  In order to compute the closest pair of a set of points, we first locate a linear number of candidate edges that are guaranteed to contain the closest pair. All these candidate pairs are found by creating certain cone structures around each point. One of these cone families consists of cones of aperture 60 degrees symmetric around a horizontal line through each of the points. A candidate edge is created between the apex of each cone and the leftmost point in this cone. We repeat this construction by rotating the cone direction by $+60$ and -60 degrees around the origin. We will then be guaranteed that one of our candidate edges will actually be the closest pair. A set of certificates is devised to guarantee the correctness of these cone structures, and hence to maintain them as the points move. The closest pair is known exactly at all times, with a small computational cost.

- Voronoi Diagram, Delaunay Triangulation, and Minimum Spanning Tree:

  Kinetic data structures can be combined like classical data structures. The Voronoi diagram of a set of points, for instance, can be maintained as the points move via its dual, the Delaunay triangulation, whose kinetization is straightforward. The minimum spanning tree (MST) of the points is a subgraph of the Delaunay triangulation. We combine the kinetic structure that maintains the Delaunay triangulation

*Duke University; jbasch@cs.duke.edu.
†Stanford University; comba@cs.stanford.edu.
‡Stanford University; guibas@cs.stanford.edu.
§Mentor Graphics; john_hershberger@mentorg.com.
¶Stanford University; csilvers@cs.stanford.edu.
||Stanford University; lizhang@cs.stanford.edu.

with other kinetic structures based on the lengths of the Delaunay edges and obtain a way to track the MST of moving points.

References

[1] J. Basch, Leonidas J. Guibas, and J. Hershberger. Data structures for mobile data. In *Proc. 8th ACM-SIAM Sympos. Discrete Algorithms*, pages 747–756, 1997.

[2] Leonidas J. Guibas. Kinetic data structures: A state of the art report. In *Proc. 1998 Workshop Algorithmic Found. Robot.*, pages 191–209, Wellesley, MA, 1998. A. K. Peters.