Vertex Deletion for 3D Delaunay Triangulations

Kevin Buchin, Olivier Devillers, Wolfgang Mulzer, Okke Schrijvers, Jonathan Shewchuk

Vertex Deletion for 3D Delaunay Triangulations

Kevin Buchin TU Eindhoven k.a.buchin@tue.nl

Olivier Devillers INRIA Sophia Antipolis olivier.devillers@inria.fr

Wolfgang Mulzer Freie Universität Berlin mulzer@inf.fu-berlin.de

Okke Schrijvers Stanford University okkes@cs.stanford.edu

Jonathan Shewchuk UC Berkeley jrs@cs.berkeley.edu

DELAUNAY TRIANGULATIONS

The Delaunay triangulation (DT) of a point set is a triangulation of the convex hull such that the circumcircle of each triangle contains no other points (Fig. 1).

It is a classic structure in Computational Geometry and is used for instance for interpolation in Graphics and Scientific Computing.

We focus on deletions: given $DT(S)$ and a point $q \in S$, find $DT(S \setminus q)$. In 2D, there exist both theoretically and practically fast algorithms.

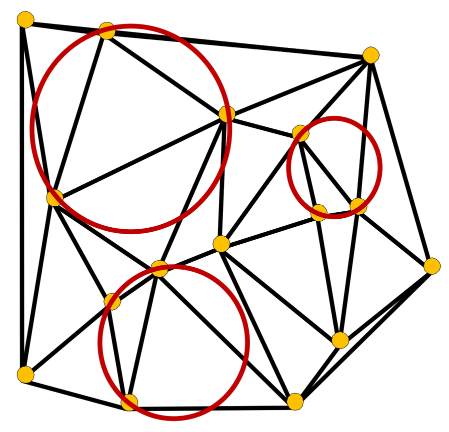

Fig. 1. The Delaunay triangulation $DT(S)$ of a point set S. The circumcircle of each triangle is empty.

The best known 3D algorithm runs in $O(d \log d + C(P))$ with $d = \deg(q)$, P the set of incident vertices and $C(P)$ is the structural cost of construction with a RIC. We reduce this to $O(d + C^{\otimes}(P))$ with $O(C^{\otimes}(P)) \leq O(C(P))$.

TRIANGULATE AND SEW

In previous work, "Triangulate and Sew" retriangulates the vertices incident to q and sews this result into the original triangulation. This process is shown in Fig. 2.

We reduce the **point location** time for the retriangulation by using information of the connectivity in the original triangulation.

We reduce the **structural complexity** by identifying and preventing the creation of simplices that would be discarded when sewed back into the triangulation.

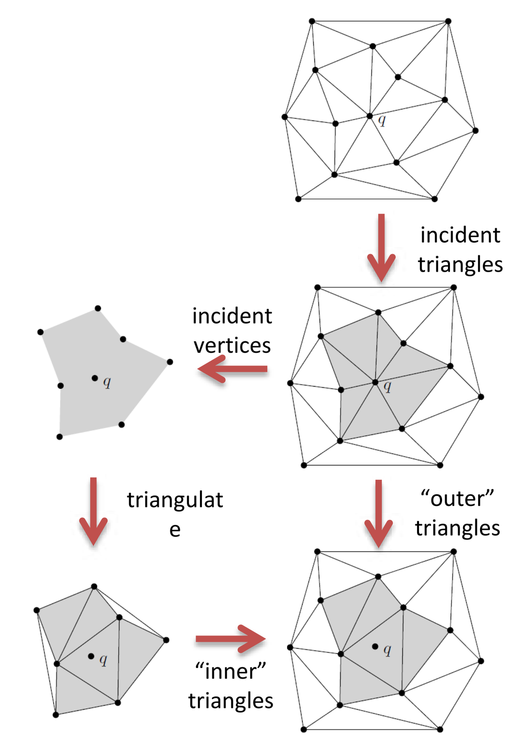

Fig. 2. Approach for deletions.

ALGORITHM

We apply the *"reverse deletion trick"* to delete point q.

On top, we remove points one-by-one in the lower-dimensional Link DT: $DT^{\ell}(Q)$, storing guides in the process.

We then reconstruct using the guides and the Conflict DT: $DT_q^{\otimes}(P)$, preventing unnecessary simplices from being created.

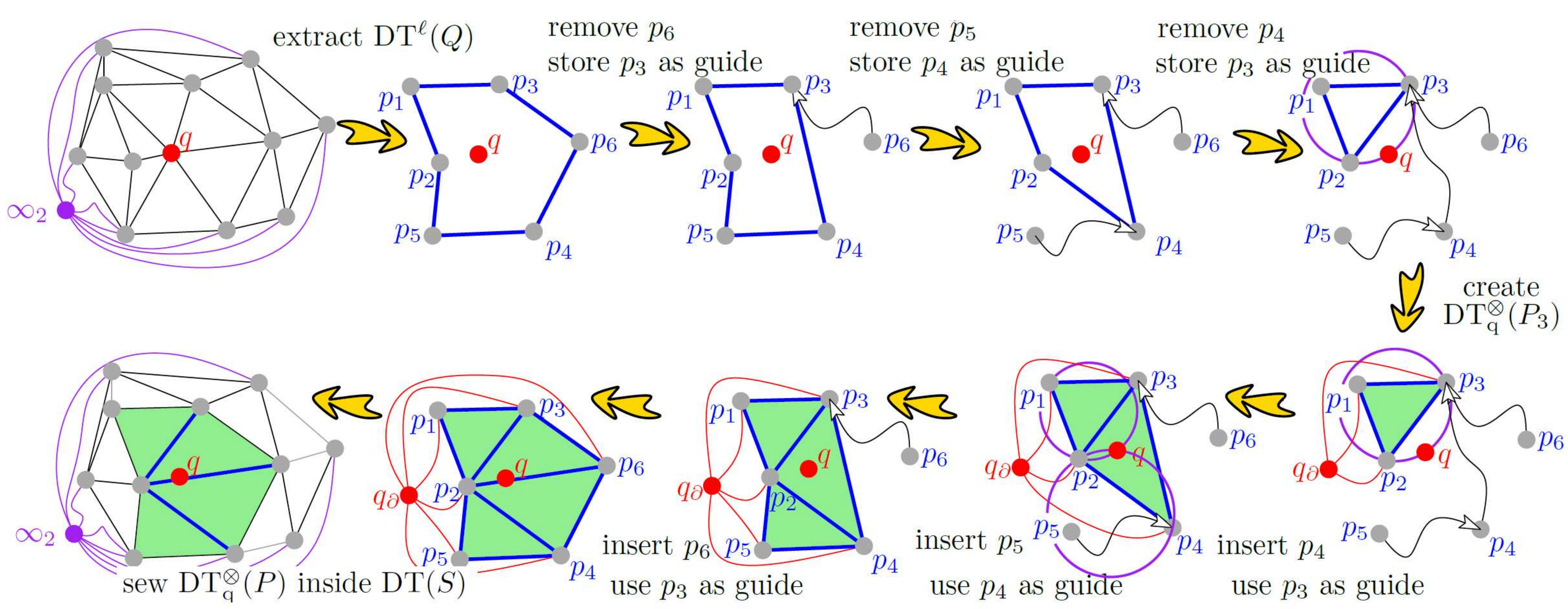

Fig. 3. The overview of how we delete points from a Delaunay triangulation.

SAMPLING AND GUIDES

By using a **guide** when we insert point p_i, we reduce the time needed for point location. By picking a point that will be a neighbor in the new triangulation, we can charge walking to the structural complexity. If the guide has constant degree, then finding a simplex in conflict with p_i takes constant time.

We use several **sampling schemes** that find low degree guides:

- Random low-degree edge
- Random triangle created during removal
- BRIO + Bounded Degree Spanning Tree

Using $C^{\otimes}(P)$ to denote the structural change induced by a randomized incremental construction of P, we have the following theorem:

Thm 1. Our algorithm runs in $O(C^{\otimes}(P))$ expected time using

i. Uniform sampling using triangle guides
ii. BRIO sampling using vertex guides

MANAGING BOUNDARIES

During the deletions in the Link DT and insertions in the Conflict DT we use alternative **geometric predicates**.

When deleting points from a 3D DT, the Link DT is a DT on a 2-dimensional topological sphere. We include the point q in each in-circle predicate -turning it into an in-sphere predicate- to ensure correct behavior.

When inserting points in the Conflict DT, we want to prevent the creation of simplices that are not in conflict with q. We apply an alternative in-sphere predicate for boundary simplices (containing the special q-boundary vertex: q_∂). We define the geometric predicate $insphere^{\otimes}(a, b, c, q_\partial, w)$ as $insphere^{\otimes}(b, a, c, d, w)$, where d is the vertex on the other side of triangle a, b, c. We can see the effect in Fig. 3 when p_5 is inserted: the boundary triangle p_2, p_4, q_∂ is found to be in conflict with p_5 and is replaced while boundary triangle p_1, p_2, q_∂ is not.

RESULTS

We implemented the different sampling schemes for finding a low degree guide and compared our implementation with CGAL.

All of our implementations compare favorably (see Fig. 4). When deleting low-degree points we require similar time while high degree points are a lot faster.

Both using guides, and using the Conflict DT reduce running time, with the largest benefit coming from the Conflict DT.

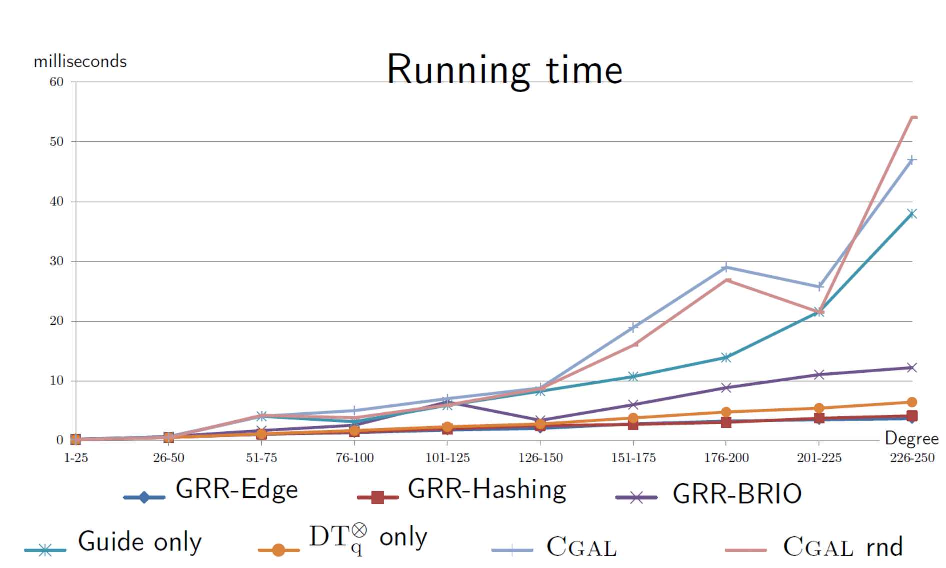

Fig. 4. The running time of our algorithm with different sampling schemes, compared to CGAL.