September 12, 2008

# Towards Surface Reconstruction in Linear Time

Benoît Hudson
Toyota Technological Institute at Chicago
bhudson@tti-c.org

**Abstract**

In a popular statement of the surface reconstruction problem in low dimensions, we have an input set of points drawn from a manifold (which is unknown), and are asked to compute an approximation to the manifold. Under some assumptions on the point cloud, the Restricted Delaunay Triangulation (RDT) achieves this. To find the RDT, traditionally we first compute the Voronoi diagram, filter out parts of the diagram, and dualize what remains. The first step can take quadratic space and time in 3D or in higher dimension. Importing two recent results from mesh refinement, I show how to compute a smaller starting point, the Clipped Voronoi Diagram, that takes only linear space and can be found as quickly as sorting the points. If the inputs are in integer or floating point format, this takes only linear time. Filtering the clipped Voronoi diagram should only take linear additional time, but I leave the details for future work.

## 1 Sampling Assumptions

The input is a set of *samples* $S$, all of them lying on an unknown manifold $M$. In order for it to be possible to find $M$, we need to impose conditions on $S$. Consider the medial axis of the manifold: all points in space equidistant to at least two points of $M$. At any point $x$ on $M$, we can define a *local feature size* as the distance from $x$ to the medial axis. Define the *reach* of $M$ to be the minimum local feature size of any point; for convenience, let us rescale so that the reach is 1. The requirements are **(A)** no point $x$ on $M$ is more than $\epsilon$ from at least one of the samples (the sample is $\epsilon$-dense), and **(B)** no two samples are closer than $\lambda$ from each other (the sample is $\lambda$-sparse). This is somewhat more strict than should be necessary: I conjecture that my algorithm is both correct and fast if we allow the sampling density to vary with the local feature size.

## 2 Restricted and Clipped, Delaunay and Voronoi

The Voronoi cell of a sample $p$, which I denote $V(p)$, is the set of all points $x \in \mathbb{R}^d$ closer or equidistant to $p$ than to any other sample. The **restricted Voronoi** cell $V_{|M}(p)$ is the Voronoi cell intersected with the manifold $M$. Edelsbrunner and Shah [ES97] noted that the dual of the restricted Voronoi diagram—the restricted Delaunay triangulation (RDT)—forms a triangulation that is homeomorphic to the manifold; later authors showed that it is also pointwise close to the manifold, and the normals to the RDT are close to the normals on the manifold. Computing the restricted Voronoi is tricky since $M$ is unknown; however, it is a subset of the unrestricted Voronoi diagram. Algorithms exist that compute the entire Voronoi diagram, filter it, and dualize what remains so that we are left with the RDT of a manifold similar to $M$ (see [Dey07]).

Points in the restricted Voronoi cell of a sample $p$ are close to $p$. Indeed, consider $x \in V_{|M}(p)$. By the $\epsilon$-dense sampling condition, we know that there is some sample at distance $\epsilon$ from $x$. Since $p$ is the closest sample, $|px| \leq \epsilon$. This motivates computing the **$\beta$-clipped Voronoi** cell of a sample, which is cheaper than computing the entire Voronoi cell. Let $\mathrm{NN}(p)$ be the distance from $p$ to its nearest neighbour. The $\beta$-clipped Voronoi cell is that portion of the Voronoi cell is within distance $\beta\,\mathrm{NN}(p)$ of $p$. Given the $\lambda$-sparse assumption, we know that $\mathrm{NN}(p) \geq \lambda$, which leads to the following conclusion:

**Lemma 1** *The $(\epsilon/\lambda)$-clipped Voronoi cell of a sample $p$ is a superset of the restricted Voronoi cell of $p$.*

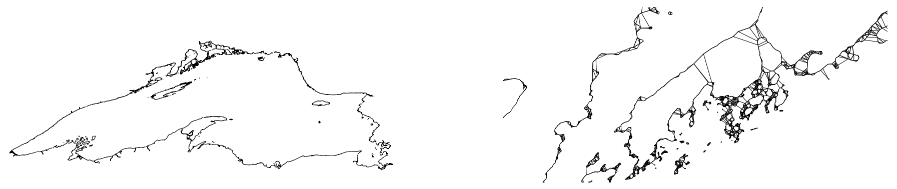

Figure 1: The dual of the clipped Voronoi diagram of the nodes of the Lake Superior data set. It is qualitatively close to our goal, but a closeup (**left**) reveals it still needs some filtering.

## 3 Computing clipped Voronoi cells

Under certain conditions, Hudson and Türkoğlu [HT08] showed how to compute the clipped Voronoi cell in constant time after building a quadtree. The conditions were tightly focused on the mesh refinement problem, but they apply here:

**Lemma 2** *We can compute the $(\epsilon/\lambda)$-clipped Voronoi cell of a sample in constant time after preprocessing.*
**Proof:** Theorem 7 from Hudson and Türkoğlu [HT08] proves we can compute the clipped Voronoi cell of $p$ in constant time assuming no other sample $q$ simultaneously suffers two conditions: that it has a very nearby neighbour, $\mathrm{NN}(q) \in o(\mathrm{NN}(p))$, and that the Voronoi cell of $q$ has bad aspect ratio. The second clause is irrelevant here since the first clause never occurs in our case. $\mathrm{NN}(p) \leq 2\epsilon$ from the $\epsilon$-dense assumption, and $\mathrm{NN}(q) \geq \lambda$ from the $\lambda$-sparse assumption: $\mathrm{NN}(q) \geq (\lambda/2\epsilon)\,\mathrm{NN}(p)$. ■

The technique requires constructing a quadtree with certain properties on the size of the quadtree cells. Said quadtree has $O(m)$ cells, where $m$ is the number of vertices in the smallest possible good-quality mesh of the input. *A priori*, $m$ could be much larger than $n$. However, very recent work bounds $m$ in some cases: in particular, the mesh of an $\epsilon$-net (such as $S$) contains only $m \in O(n)$ vertices [HMPS09, Cor. 4.2]. Computing the quadtree can be done in $O(n \log \Delta + m)$ time in a real RAM model, where $\Delta$ is the geometric spread of the input. Our sampling assumptions allow the spread to be at most linear, so this collapses to $O(n \log n)$. Even better, if the inputs are in integer format, the quadtree can be computed by first sorting the samples according to their Morton numbers (interleaving the bits of the coordinates into one integer $d$ words wide), then applying some post-processing that takes time $O(m) = O(n)$ [BET99]. Because floating-point numbers sort properly when considered as if they were integers, this technique works equally well on floating point inputs. Sorting integers takes only linear time.

**Theorem 3** *Computing the clipped Voronoi diagram takes $O(n)$ time in a word RAM model if inputs are in floating point, or $O(n \log n)$ time in a real RAM model, assuming $\epsilon$, $\lambda$, and the dimension $d$ are constants.*

## References

[BET99] Marshall W. Bern, David Eppstein, and Shang-Hua Teng. Parallel construction of quadtrees and quality triangulations. *IJCGA*, 9(6):517–532, 1999.

[Dey07] Tamal K. Dey. *Curve and Surface Reconstruction*. Cambridge University Press, 2007.

[ES97] Herbert Edelsbrunner and Nimish R. Shah. Triangulating topological spaces. *IJCGA*, 7(4):365–378, 1997.

[HMPS09] Benoît Hudson, Gary L. Miller, Todd Phillips, and Don Sheehy. Size complexity of volume meshes *vs.* surface meshes. In *SODA*, 2009. Accepted to appear, available from `www.cs.cmu.edu/~bhudson`.

[HT08] Benoît Hudson and Duru Türkoğlu. An efficient query structure for mesh refinement. In *CCCG*, 2008.