Spectral Watertight Surface Reconstruction

Ravi Krishna Kolluri
University of California at Berkeley
rkolluri@cs.berkeley.edu

Jonathan Richard Shewchuk
University of California at Berkeley
jrs@cs.berkeley.edu

James F. O'Brien
University of California at Berkeley
job@cs.berkeley.edu

Introduction

We use spectral partitioning to reconstruct a watertight surface from point cloud data. This method is particularly effective for noisy and undersampled point sets with outliers, because decisions about the reconstructed surface are based on a global view of the model.

Algorithm

To reconstruct a surface from an unorganized point set S, we create a point set S^+ that adds the vertices of a cubical bounding box, then compute the Delaunay triangulation T and Voronoi diagram Q of S^+. We form a graph G whose nodes are vertices of Q, and use a spectral graph partitioning algorithm to cut this graph into two pieces, the inside and outside subgraphs. Because every Voronoi vertex in Q represents a tetrahedron in T, these labels are affixed to the tetrahedra too. If the points in S are sampled densely enough from a simple closed surface, then the surface is approximated reasonably well by the faces of T that separate the inside tetrahedra from the outside tetrahedra.

Our algorithm first identifies the set V of Voronoi vertices called poles [Amenta et al. 2001], which are likely to lie near the medial axis of the surface being recovered. The algorithm then constructs a sparse *pole graph* $G = (V, E)$. The set E of edges is defined as follows. For each sample point s with poles u and v, (u, v) is an edge in E. For each edge (s, s'), of the Delaunay tetrahedralization T, the edges (u, u'), (u, v'), (v, u'), and (v, v') are all edges of E, where u and v are the poles of s, and u' and v' are the poles of s'.

The edge weights are based on observations of Amenta et al. [2001]. If a sample s has a long, thin Voronoi cell, the likelihood is high that its poles u and v are on opposite sides of the surface. We assign a negative weight to edge (u, v). Let t_u and t_v be the tetrahedra in T whose duals are u and v. The circumscribing spheres of t_u and t_v intersect at an angle ϕ. We assign (u, v) a weight of $w_{u,v} = -e^{4+4\cos\phi}$. Next, let (u, v) be an edge of E that is not assigned a negative weight. We assign (u, v) a weight of $w_{u,v} = e^{4-4\cos\phi}$. If ϕ is close to $180°$, u and v are likely to lie on the same side of the surface, so we use a large, positive edge weight.

We know *a priori* that tetrahedra with vertices on the bounding box must be labeled outside. So, we fix their labels prior to the partitioning step by collapsing the poles dual to such tetrahedra into a single *supernode* z, yielding a modified graph G'.

From the modified pole graph G', we construct a *pole matrix* L. (L is often called the *Laplacian matrix*.) L is sparse and symmetric and has one row and one column for each node of the graph G'. For each edge (u, v) of G' with weight $w_{u,v}$, the pole matrix L has the entries $L_{ij} = -w_{u,v}$ and $L_{ji} = -w_{u,v}$. The diagonal entries of L are the row sums $L_{ii} = \sum_{j \neq i} |L_{ij}|$.

We partition G' by finding the eigenvector x associated with the smallest eigenvalue λ of the generalized eigensystem $Lx = \lambda Dx$, where D is a diagonal matrix whose diagonal is identical to that of L. Because L is a sparse matrix, we compute the eigenvector x using TRLAN, an implementation of the Lanczos algorithm [Pothen et al. 1990]. When this method is applied to smooth, well-sampled surfaces, we find that the eigenvector x is relatively polarized: most of its entries are clearly negative or clearly positive, with few entries near zero. Noisy models are more ambiguous; see Figure 1. Each entry of the eigenvector x corresponds to one tetrahedron of T. Suppose the entry corresponding to the supernode z is positive; then the nodes of G' whose entries are positive are labeled outside, and the nodes whose entries are negative are labeled inside.

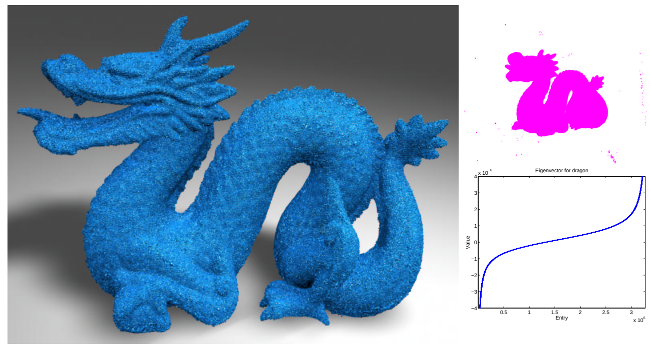

Figure 1: Reconstruction of the Stanford dragon from noisy raw data that includes outliers (upper right). At lower right are the sorted entries of the eigenvector used to reconstruct the model. 1,770,421 input points, 3,031,078 triangles, 229 minutes.

Spectral partitioning labels each tetrahedron whose dual Voronoi vertex is a pole. To label a tetrahedron t whose dual vertex v is not a pole, we examine the poles of its four vertices. If t has a vertex u that has a pole p that is labeled inside, and $\angle vup < 90°$, we label t inside. Otherwise, we label t outside.

After all the tetrahedra are labeled, a final step searches for locations where the surface is non-manifold, and attempts to make the surface manifold by relabeling selected tetrahedra from inside to outside.

We output every triangle at which an inside tetrahedron meets an outside tetrahedron. This is the reconstructed surface.

An alternative is to use power cells (all of which dualize to poles) instead of Delaunay tetrahedra, like Amenta et al.'s power crust algorithm. Power cells offer better reconstruction of sharp corners, but pay for it by generating many extra vertices in the surface.

The goal of our algorithm is to produce the same labeling as the provably good power crust algorithm when the surface is well-sampled and the samples are noiseless, but to behave more robustly otherwise. Because the spectral partitioner has a global view of the samples, it can fill large holes and correct for noise and undersampling in circumstances where the local labeling algorithm that Amenta et al. use fails. Outliers are usually removed: if every tetrahedron adjoining an outlier is labeled outside (or every tetrahedron is labeled inside), the outlier does not appear in the final surface.

References

AMENTA, N., CHOI, S., AND KOLLURI, R. 2001. The Power Crust, Unions of Balls, and the Medial Axis Transform. *Computational Geometry: Theory and Applications 19*, 2–3 (July), 127–153.

POTHEN, A., SIMON, H. D., AND LIOU, K.-P. 1990. Partitioning Sparse Matrices with Eigenvectors of Graphs. *SIAM Journal on Matrix Analysis and Applications 11*, 3 (July), 430–452.