Triangulating Point Clouds with Spherical Topology

Kai Hormann and Martin Reimers

Abstract. Triangulating a set of scattered 3D points is a common approach to surface reconstruction from unorganized point clouds. Besides several Voronoi-based and incremental algorithms, a recently presented technique parameterizes the sample points and uses a Delaunay triangulation of the parameter points in the parameter domain. This method is limited to reconstructing surfaces that are homeomorphic to a disc, and we show how it can be extended to handle spherical topology as well.

§1. Introduction

The problem we consider can be stated as follows: given a set $V = \{v_i\}$ of points $v_i \in \mathbb{R}^3$, $i = 1, \ldots, n$, find a triangulation $\mathcal{T}$ with these points as vertices, $V(\mathcal{T}) = V$. Some methods for solving this problem are based on the Voronoi diagram and the Delaunay tetrahedrization of the points [2–4]. Other incremental algorithms start with a single edge or triangle connecting two or three of the given points and successively add more points by generating additional edges and triangles [5,13,16].

In this paper we extend a method that was proposed by Floater and Reimers [12]. They observed that the usual linear parameterization methods for triangulations do not require the vertices to be organized in a globally consistent triangulation. The only information needed for setting up the linear system in the case of harmonic maps [8,17], shape preserving [9], or mean value parameterizations [10] is the triangle fan around each vertex, in other words an *ordered* set of neighbours $N(v)$ for each $v \in V$. And computing the distance weights for the spring model parameterizations [14] does not even require this ordering but an *unordered* neighbourhood $N(v)$ only. Several ways of defining neighbourhoods of unorganized points will be explained in Section 2.

Curve and Surface Design: Saint-Malo 2002
Tom Lyche, Marie-Laurence Mazure, and Larry L. Schumaker (eds.), pp. 215–224.

ISBN 0-9728482-0-7.

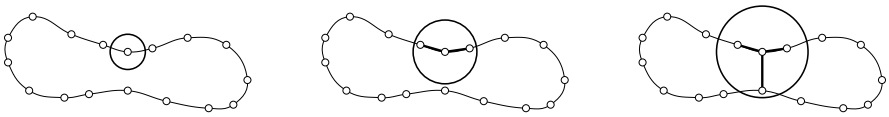

Fig. 1. Choosing different radii for the ball neighbourhood.

Once these neighbourhoods are specified, it is possible to apply one of the linear parameterization methods to determine parameter points $\psi(v)$, one for each $v \in V$, and use one of the standard methods for triangulating points in two dimensions, *e.g.* the Delaunay triangulation, to find a triangulation $\mathcal{S}$ of the parameter points $\psi(v)$. Finally, a triangulation $\mathcal{T}$ of the given point cloud can be obtained by collecting all triangles $[u, v, w]$ for which $[\psi(u), \psi(v), \psi(w)]$ is a triangle in $\mathcal{S}$.

The drawback of this method is that it can only handle point sets that are assumed to be sampled from a single surface patch, *i.e.*, a surface that is homeomorphic to a disc. However, more complex point sets can be split into several patches and each patch triangulated separately. In Section 3 we show how this can be done for points sampled from a surface homeomorphic to a sphere. For a better understanding of our algorithm, we explain its details by means of an example in Section 4.

§2. Local Neighbourhoods

The simplest way of defining a set of unordered neighbours $N(v)$ for a point v from a point cloud V is motivated by discretizing the concept of neighbourhood on the surface from which the data was sampled. Suppose the underlying surface is a two-manifold M, and let $d(v, w)$ be the geodesic distance, *i.e.*, the Euclidean length of the shortest path in M connecting v and w. Then we can define for each $v \in M$ the (open) r-neighbourhood

$$N_{r,M}(v) = \{w \in M : 0 < d(v, w) < r\}$$

for some radius $r > 0$ and the discrete version

$$\hat{N}_r(v) = N_{r,M}(v) \cap V$$

for the data set $V \subset M$. Of course, M is unknown and we cannot determine $\hat{N}_r(v)$ this way, but the ball neighbourhood

$$N_r(v) = \{w \in V : 0 < \|v - w\| < r\}$$

is a good approximation as long as the ball radius r is adequately chosen. If r is too large, then it may happen that the neighbourhood of a vertex contains points that were sampled from a different part of the surface and if r is too small then the set of neighbours can be empty for some points (see Figure 1).

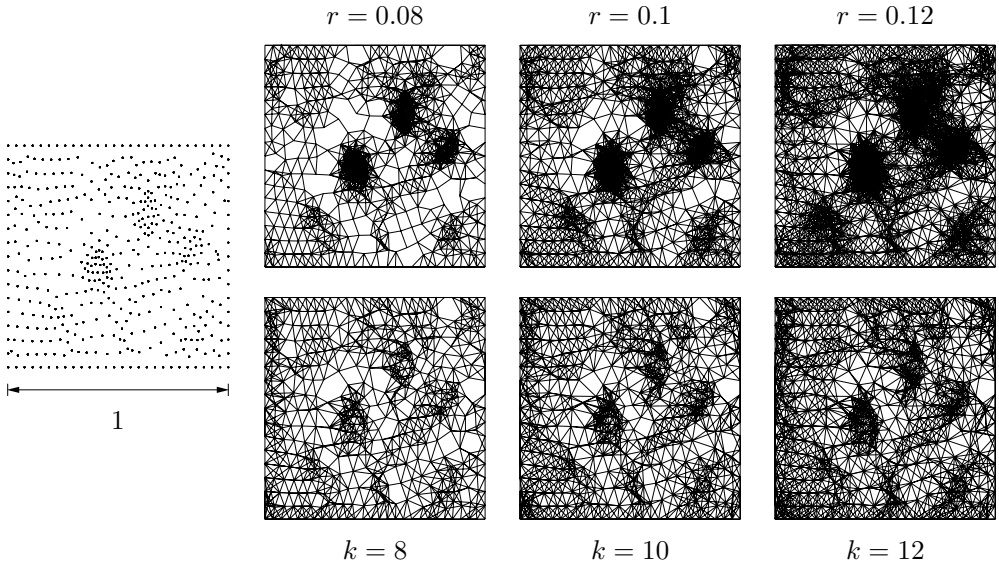

Fig. 2. Connectivity graph of the data set to the left for different ball neighbourhoods (top) and different numbers of nearest neighbours (bottom).

A data set $V \subset M$ is said to be ρ-dense [15] if ρ is the smallest value such that any sphere with radius ρ and center in M contains at least one point in V. For a ρ-dense data set we can assure $N_r(v) \neq \emptyset$ for all $v \in V$ if $r \geq 2\rho$ and if the distribution of the points is reasonably uniform in addition, $r = 2\rho$ usually is a good choice. However, determining ρ remains a rather computationally expensive task, and it is simpler to choose r interactively. More details can be found in [1].

If the density of the point set V varies, it might be preferable to adapt r to the local density of V in order to avoid big differences in the size of the neighbourhoods. Another strategy is to fix the size of $N(v)$ and let each neighbourhood consist of the k nearest points to v. In practice, $k \approx 10$ has proven to give good results.

Figure 2 illustrates the effect of choosing different values for r and k by showing the connectivity graph $G = G(V, E)$ of the data set which consists of the vertices V and all edges

$$E = \{[v, w] : w \in N(v) \text{ or } v \in N(w)\}$$

that are defined by the local neighbourhoods. Note that the neighbourhood $N_G(v)$ of $v \in V$ defined by this graph is symmetric and may be larger than $N(v)$ as in general $w \in N(v)$ implies $v \in N(w)$ only for the ball neighbourhood.

The ball neighbourhood as well as the neighbourhood defined by the k nearest points are unordered and allow to use parameterization methods based on distance weights. But computing harmonic, shape preserving, or mean value weights further requires the neighbours of each point to

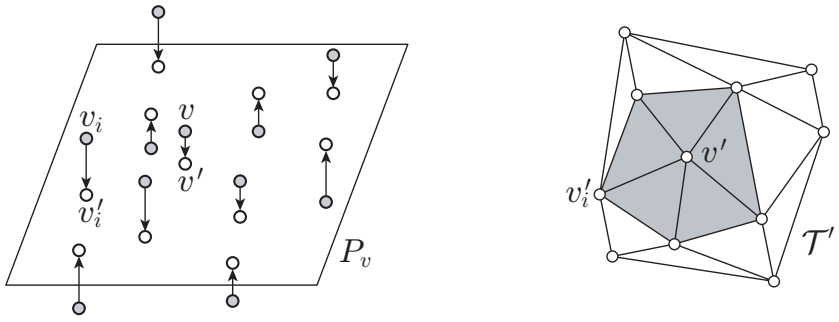

Fig. 3. Projecting the unordered neighbours into the least squares plane (left) and computing the Delaunay neighbourhood (right).

be ordered. This can be done as follows. Let $N(v) = \{v_1, \ldots, v_n\}$ be the unordered neighbours of a point $v \in V$. Then we first compute the least squares plane P_v of $\{v\} \cup N(v)$ and project v and $N(v)$ orthogonally into P_v, yielding new vertices v' and $v_1', \ldots, v_n'$. A slightly different approach is to rotate the vertices in $N(v)$ into the least squares plane of $N(v)$ through v [13]. However, in both cases we proceed by computing the Delaunay triangulation $\mathcal{T}'$ of the new vertices, defining the Delaunay neighbourhood

$$N'(v) = \{v_i \in N(v) : [v', v_i'] \in E(\mathcal{T}')\},$$

and order the vertices w in $N'(v)$ in the same way as the corresponding vertices w' around v' in $\mathcal{T}'$. The process is illustrated in Figure 3.

§3. The Algorithm in General

For triangulating a point cloud V that is sampled from a two-manifold M that is topologically more complicated than a disc, we can adapt the idea of Floater *et al.* [11] for partitioning triangulations with general topology. That is, we start with a coarse triangulation $\mathcal{D}$ with m triangles whose vertices $V(\mathcal{D})$ are a subset of V and which has the same topology as M. Then we partition the point cloud V into m subsets $V^1, \ldots, V^m$ such that each V^i corresponds to one of the triangles $T_i \in \mathcal{D}$.

For each edge $[v, w]$ in $E(\mathcal{D})$ we find the shortest path in the connectivity graph $G(V, E)$ connecting v and w, say

$$\Gamma(v, w) = [u_1, u_2] \cup [u_2, u_3] \cup \cdots \cup [u_{r-1}, u_r],$$

where $u_1 = v$, $u_r = w$, and $[u_i, u_{i+1}] \in E$. A common algorithm for determining shortest paths in a graph is Dijkstra's algorithm [6].

We further let $P_{vw} = \{u_1, \ldots, u_r\}$ be the set of vertices traversed by this path and define the boundary vertices as their union

$$V_B = \bigcup_{[v,w] \in E(\mathcal{D})} P_{vw}.$$

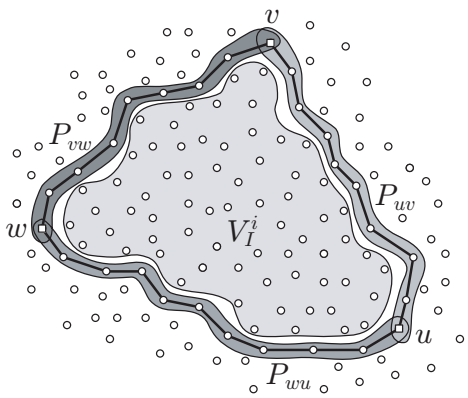

Fig. 4. The vertices traversed by the shortest paths between u, v, and w are collected in the sets P_{uv}, P_{vw}, and P_{wu}. The vertices that are surrounded by these boundary vertices are defined as the interior vertices V_I^i.

The remaining interior vertices $V_I = V \setminus V_B$ are then split into disjoint subsets $V_I^1, \ldots, V_I^m$ such that for each triangle $T_i = [u, v, w]$ in $\mathcal{D}$, the corresponding set of interior vertices V_I^i is bounded by the shortest paths connecting u, v, and w (see Figure 4). We collect all vertices traversed by these three paths in the set of *boundary* vertices $V_B^i = P_{uv} \cup P_{vw} \cup P_{wu}$ and finally let $V^i = V_B^i \cup V_I^i$ for all $i = 1, \ldots, m$.

We can now apply the method of Floater and Reimers [12] to each V^i, yielding triangulations $\mathcal{T}^i$ whose union $\mathcal{T} = \mathcal{T}^1 \cup \cdots \cup \mathcal{T}^m$ gives a triangulation of the whole point cloud V. Note that the order of the boundary vertices V_B^i as required by the meshless parameterization method [12] is given by the sequence of vertices in the shortest paths.

The final triangulation $\mathcal{T}$ is topologically equivalent to $\mathcal{D}$ because the boundaries of the reconstructed patches $\mathcal{T}^i$ align in the same way as the corresponding triangles T_i. Consider two neighbouring triangles T_i, T_j in $\mathcal{D}$ and their common edge $[v, w]$. Then the shortest path between v and w in G is part of both the boundaries of $\mathcal{T}^i$ and $\mathcal{T}^j$. In fact, $\Gamma(v, w) = \mathcal{T}^i \cap \mathcal{T}^j$.

§4. The Algorithm in Detail

For a better understanding of this algorithm, the following example may be helpful. The point cloud V in Figure 5 was sampled from an object homeomorphic to a sphere and we therefore chose the simplest triangulation representing this topology, namely a tetrahedron $\mathcal{D}$. As vertices we took those $v_i \in V$ that maximize $\langle v - s | w_i \rangle$, where s is the centroid of V and w_i are the vertices of a regular tetrahedron, centred in the origin. In the next step of the algorithm we split the interior vertices into four subsets $V_I^1, \ldots, V_I^4$, each corresponding to one of the four triangles in $\mathcal{D}$. Figures 6 and 7 illustrate how this is done.

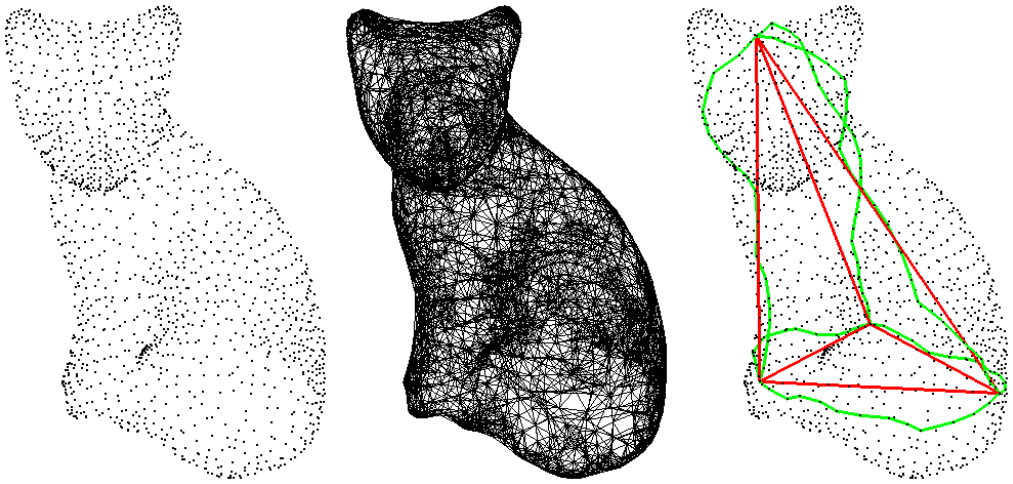

Fig. 5. Point cloud of a cat with 1,458 points (left), connectivity graph using the 12 nearest neighbours (middle), tetrahedron $\mathcal{D}$ and shortest paths corresponding to the edges of $\mathcal{D}$ (right).

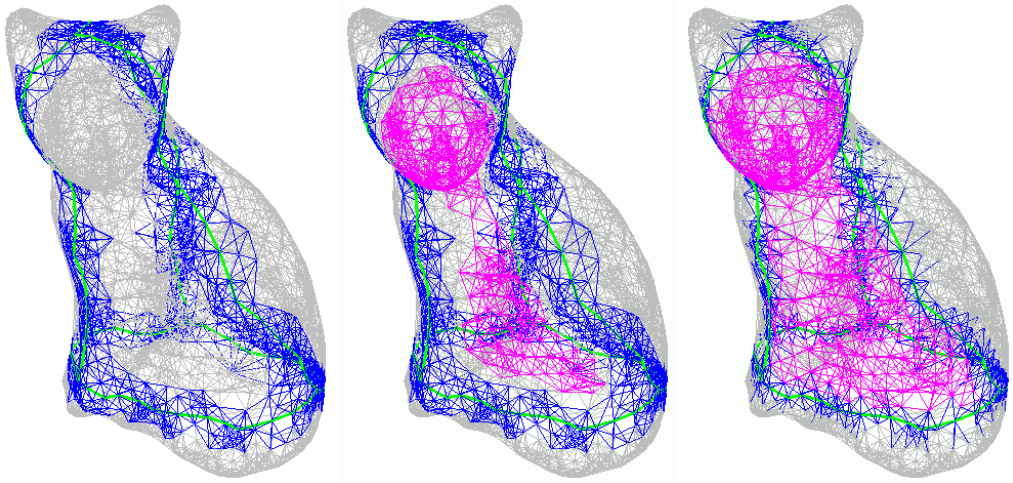

Fig. 6. Detecting interior vertices by first growing the shortest paths (left), then finding a connected component in the connectivity graph (middle), and finally growing that region (right).

Firstly, we temporarily enlarge the set of boundary vertices by adding all neighbouring vertices

$$V_B' = V_B \cup \bigcup_{v \in V_B} N_G(v),$$

and keep repeating this process until the number of connected components G^i in the connectivity graph of the remaining vertices $V_I' = V \setminus V_B'$ equals the number of triangles in $\mathcal{D}$. Then we let V_I^i be the vertices of the subgraphs G^i.

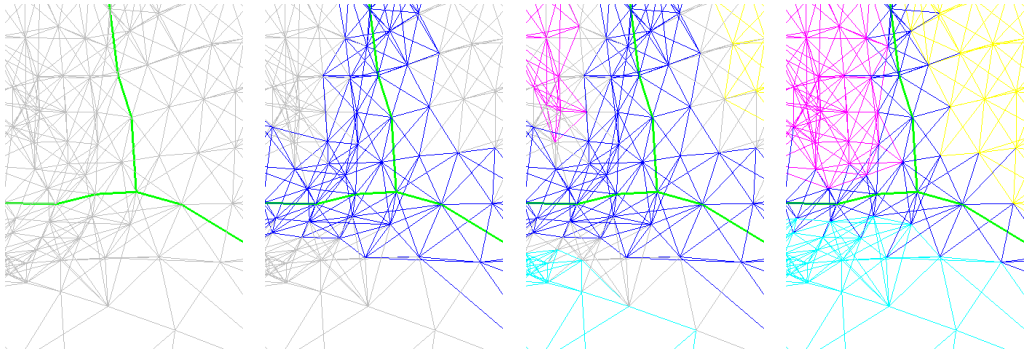

Fig. 7. Close-up view to the connectivity graph with shortest path (left). Interior vertices are found by growing the shortest paths (middle left), finding connected components in the connectivity graph (middle right), and growing these regions (right).

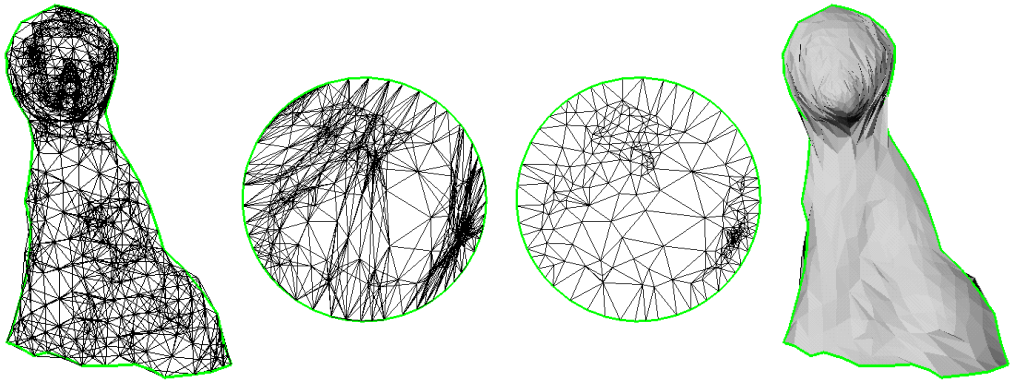

Fig. 8. Reconstructing one of the subsets V^i. Connectivity graph (left), parameterization (middle left), Delaunay triangulation (middle right), and triangulation $\mathcal{T}^i$.

Secondly, the subsets V_I^i are enlarged by adding the "missing" interior vertices. We assign each vertex in $V_B' \setminus V_B$ to the closest patch V_I^i, where distance is measured in the connectivity graph. For that purpose we used the Dijkstra algorithm on G with all $v \in V_I'$ as source points and with a mechanism to retrieve for each vertex $v \in V$ the closest source point.

Thirdly we identify each subset V_I^i with a triangle $[u, v, w] \in \mathcal{D}$ and add the boundary vertices in sequence,

$$V^i = V_I^i \cup P_{uv} \cup P_{vw} \cup P_{wu}.$$

Finally, we reconstruct each subset V^i with the meshless parameterization method, yielding triangulations $\mathcal{T}^i$ (see Figure 8) which are finally merged to the triangulation $\mathcal{T}$ of V shown in Figure 9. Although it looks a bit crinkly, it is a topologically correct triangulation which can be further optimized, *e.g.* with the method of Dyn *et al.* [7]. Figure 10 shows another example.

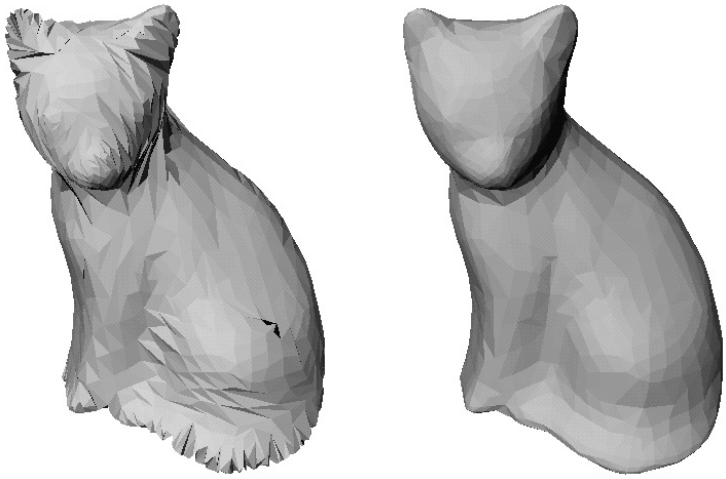

Fig. 9. Final reconstruction $\mathcal{T}$ with 2,912 triangles before (left) and after optimization (right).

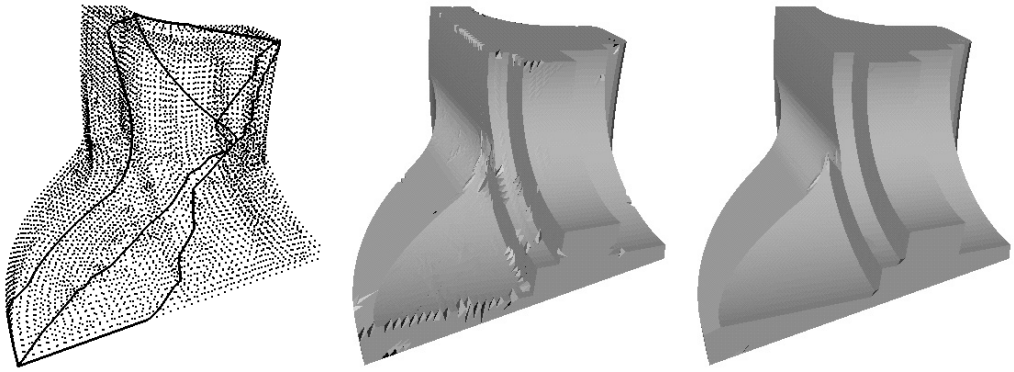

Fig. 10. Point cloud of the fandisk with 6,475 points and shortest paths (left), reconstructed triangulation with 12,946 triangles before (middle) and after optimization (right).

§5. Conclusion

We have presented a method for triangulating point clouds that reduces the computational complexity by splitting the problem into subproblems, each of which can be solved with the method of Floater and Reimers [12]. We have also shown how to automatically generate a suitable base mesh $\mathcal{D}$ in the spherical case. In order to handle general topology we need a base mesh that meets the following requirements.

As the algorithm creates triangulations that are topologically equivalent to the base mesh, $\mathcal{D}$ must provide the correct topology. Furthermore, the shortest paths $\Gamma(v, w)$ that correspond to the edges $[v, w]$ of $\mathcal{D}$ must not intersect and meet at the vertices of $\mathcal{D}$ only. Otherwise, degeneracies may be present in the reconstructed triangulation, and our method for

splitting the interior vertices into disjoint subsets can even fail. In fact, if one of the regions that are bounded by the boundary vertices V_B^i contains only few points, the growing procedure of the boundary vertices may "eat up" all interior vertices of that region or split it into disjoint components. It is in other words better to avoid long thin triangles in $\mathcal{D}$.

We also realize that $\mathcal{D}$ does not necessarily have to consist of triangles but can be a more general polyhedral object. The user could therefore specify some vertices of V by hand and connect them either by shortest or by interactively defined paths such that the so defined $\mathcal{D}$ meets the requirements. We are currently investigating methods to automate the construction of suitable meshes for general topology and hope to report on the results elsewhere soon.

Acknowledgments. This work was supported in part by the German Research Foundation (DFG) under grant HO-2457/1-1.

References

1. Amenta, N. and M. Bern, Surface reconstruction by Voronoi filtering, Discrete and Computational Geometry **22** (1999), 481–504.
2. Amenta, N., M. Bern, and M. Kamvysselis, A new Voronoi-based surface reconstruction algorithm, Comp. Graphics (SIGGRAPH Proc.) **32** (1998), 415–421.
3. Amenta, N., S. Choi, T. K. Dey, and N. Leekha, A simple algorithm for homeomorphic surface reconstruction, J. Comput. Geom. and Appl. **12** (2002), 125–141.
4. Amenta, N., S. Choi, and R. Kolluri, The power crust, unions of balls, and the medial axis transform, Comput. Geom. **19** (2001), 127–153.
5. Bernardini, F., J. Mittleman, H. Rushmeier, C. T. Silva, and G. Taubin, The ball-pivoting algorithm for surface reconstruction, IEEE Trans. Visualiz. and Comp. Graphics **5** (1999), 349–359.
6. Dijkstra, E. W., A note on two problems in connexion with graphs, Numer. Math. **1** (1959), 269–271.
7. Dyn N., K. Hormann, S.-J. Kim, and D. Levin, Optimizing 3D triangulations using discrete curvature analysis, in *Mathematical Methods for Curves and Surfaces: Oslo 2000*, T. Lyche and L. L. Schumaker (eds.), Vanderbilt University Press, Nashville, 2001, 135–146.
8. Eck, M., T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle, Multiresolution analysis of arbitrary meshes, Comp. Graphics (SIGGRAPH Proc.) **29** (1995), 173–182.
9. Floater, M. S., Parameterization and smooth approximation of surface triangulations, Comput. Aided Geom. Design **14** (1997), 231–250.

10. Floater, M. S., Mean value coordinates, Comput. Aided Geom. Design **20** (2003), 19–27.
11. Floater, M. S., K. Hormann, and M. Reimers, Parameterization of manifold triangulations, in *Approximation Theory X: Abstract and Classical Analysis*, C. K. Chui, L. L. Schumaker, and J. Stöckler (eds.), Vanderbilt University Press, Nashville, 2002, 197–209.
12. Floater M. S. and M. Reimers, Meshless parameterization and surface reconstruction, Comput. Aided Geom. Design **18** (2001), 77–92.
13. Gopi, M., S. Krishnan, and C. T. Silva, Surface reconstruction based on lower dimensional localized Delaunay triangulation, Comp. Graphics Forum (Eurographics Proc.) **19** (2000), C467–C478.
14. Greiner, G., and K. Hormann, Interpolating and approximating scattered 3D data with hierarchical tensor product B-splines, in *Surface Fitting and Multiresolution Methods*, A. Le Méhauté, C. Rabut, and L. L. Schumaker (eds.), Vanderbilt University Press, Nashville, 1997, 163–172.
15. Hoppe, H., T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, Surface reconstruction from unorganized points, Comp. Graphics (SIGGRAPH Proc.) **26** (1992), 71–78.
16. Mencl, R. and H. Müller, Graph-based surface reconstruction using structures in scattered point sets, in *Proc. CGI '98*, 1998, 298–311.
17. Pinkall, U., and K. Polthier, Computing discrete minimal surfaces and their conjugates, Experimental Mathematics **2** (1993), 15–36.

Kai Hormann
Caltech, MS 256-80
1200 E. California Blvd.
Pasadena, CA 91125, USA
`hormann@cs.caltech.edu`

Martin Reimers
Department of Informatics
Postbox 1080 Blindern
N-0316 Oslo, Norway
`martinre@ifi.uio.no`