Smooth Surface Reconstruction from Noisy Clouds

Boris Mederos & Luiz Velho & Luiz Henrique de Figueiredo

IMPA–Instituto de Matemática Pura e Aplicada,
Estrada Dona Castorina 110, 22461-320 Rio de Janeiro, RJ, Brazil
boris@impa.br, lvelho@impa.br, lhf@impa.br

ABSTRACT

We describe a new method for surface reconstruction and smoothing based on unorganized noisy point clouds without normals. The output of the method is a refined triangular mesh that approximates the original point cloud while preserving the salient features of the underlying surface. The method has five steps: noise removal, clustering, data reduction, initial reconstruction, and mesh refinement. We also present theoretical justifications for the heuristics used in the reconstruction step.

1 Introduction

We consider the problem of surface reconstruction and approximation from noisy scattered data points. Several algorithms are known for this important problem when the data are noise-free [1, 2, 3, 4, 6, 7, 5, 14, 11, 12, 8], including a number of recent algorithms with theoretical guarantees [1, 2, 3, 4, 6]. Those algorithms use a 3D Delaunay triangulation of the original point cloud to compute a triangular surface mesh. Computing the Delaunay triangulation can be slow for large point clouds and susceptible to numerical errors.

Gopi et al. [12] proposed an algorithm based on differential geometry that projects the neighborhood of each sample point on a tangent plane, computes the 2D Delaunay triangulation of this projected neighborhood, and lifts it to 3D.

Hoppe et al. [14] estimate a tangent plane at each sample point using its k-nearest neighbors and use the distance to the plane of the nearest sample as an estimative of the signed distance function to the surface. The zero set of this distance function is extracted using the marching cubes algorithm. This approach behaves well with noisy data.

Other algorithms [6, 5, 8] use a greedy approach. The ball pivoting algorithm [5] rolls a ball over the sample points; it requires the normal to the surface at each sample point, but it can create artifacts. Boissonnat [7] starts by finding an initial edge pq in the triangulated reconstruction; then he computes a tangent plane around the edge, projects the k-nearest neighbors of both vertices onto this plane, and determines the point r that maximizes the angle $\angle \bar{p}\bar{r}\bar{q}$ (where the bar represents projected points on the tangent plane). This point r determines a surface triangle prq. The procedure is repeated for each border edge, resulting in a triangulated surface. Boyer [8] proposes an incremental algorithm over the 3D Delaunay triangulation of the samples and relies on regular interpolants.

Our algorithm uses a different approach: it starts by performing a robust smoothing of the given point cloud, followed by a data reduction step that extracts from the smoothed point cloud a set of representative points near the underlying surface. A rough surface approximation is reconstructed from these representative points using a new incremental algorithm based on k-nearest neighbors. This is step somewhat similar to what is done in other greedy algorithms [7, 5, 8, 9, 13]: For each border edge, the algorithm uses angle criteria to select a point to make a triangle with the edge. The initial triangulation is refined using a method based on a novel projection operator [15] that is able to approximate points to the point cloud in a robust way. Our algorithm does not need Delaunay triangulations and it can handle surfaces with borders and noisy point clouds.

A different approach to surface reconstruction is to use implicit formulations. Recent work along this line includes the work of Ohtake et al. [16] on partitions of unity with piecewise quadratics and the work of Xie et al. [18] on local implicit quadric regression, which also handles noisy data.

Section 2 describes in detail the five steps of our method: smoothing, clustering, reduction, triangulation, and refinement. Section 3 gives theoretical justifications for the heuristics used in the reconstruction step. Finally, Section 4 discusses several examples of the method in action.

2 Our Method

Starting from a possibly noisy point cloud $P \subset R^3$ that samples an unknown C^1 surface S, our method produces a refined mesh approximating S. The method has five steps:

1. Smoothing: Smooth the original point cloud based on a robust projection procedure.
2. Clustering: Split the new point cloud into a set of clusters.
3. Reduction: Compute a representative point of each cluster near the unknown surface S.
4. Triangulation: Build a triangulated surface over the set of representative points.
5. Refinement: Refine the initial mesh into a finer mesh that is adapted to the geometry of S.

We shall now describe each of these steps in detail.

2.1 Smoothing

Starting from a possibly noisy point cloud P near a surface S, the goal of this step is to eliminate the noise in the data while trying to preserve the salient features of the underlying surface. The new point cloud Q is obtained by applying a new robust smoothing operator Q to each point $p \in P$. We shall give a short description of the operator Q; for details, see [15].

Each sample point $p \in P$ is transformed into a new point $Q(p) = p + t^* n^*$, where n^* is an estimated normal vector for the surface S near p and t^* is a displacement along n^*. The optimal values for t^* and n^* are computed by the robust fitting of a hyperplane H, passing through the point p + tn and orthogonal to n, in a neighborhood N(p) of the point p. The optimal hyperplane is found by minimizing a cost functional with respect to t and n, subject to the restriction $\|n\| = 1$:

$$\sum \Phi(t,n) = \rho(h_q) w(\|q - p\|), \quad q \in N(p)$$

where $h_q = n^T(q - p - tn)$ is the height of the point q with respect to the hyperplane H, and

$$\rho(x) = 1 - e^{-x^2/\sigma^2} \text{ and } w(x) = e^{-x^2/\sigma^2}.$$

(For details about the numerical method used to solve this minimization problem and its convergence, see [15].)

The new position Q(p) can be seen as the projection of p onto a local linear approximation H of the surface S.

The estimation of this approximating hyperplane H is robust to gross deviation of the points q ∈ N(p): the influence of each point in N(p) is controlled by a robust error norm ρ, which penalizes points with large heights, and by a Gaussian weight function w, which gives less influence to points far away from p. The sensitivity of the cost functional to outliers is controlled by user-supplied parameters σ_ρ and σ_w.

The result of this step is a smoothed point cloud Q = {Q(p) : p ∈P} in which the salient features of the original data have been preserved. (Having no precise definition of what a feature is, we rely on visual inspection here.) See Figure 1.

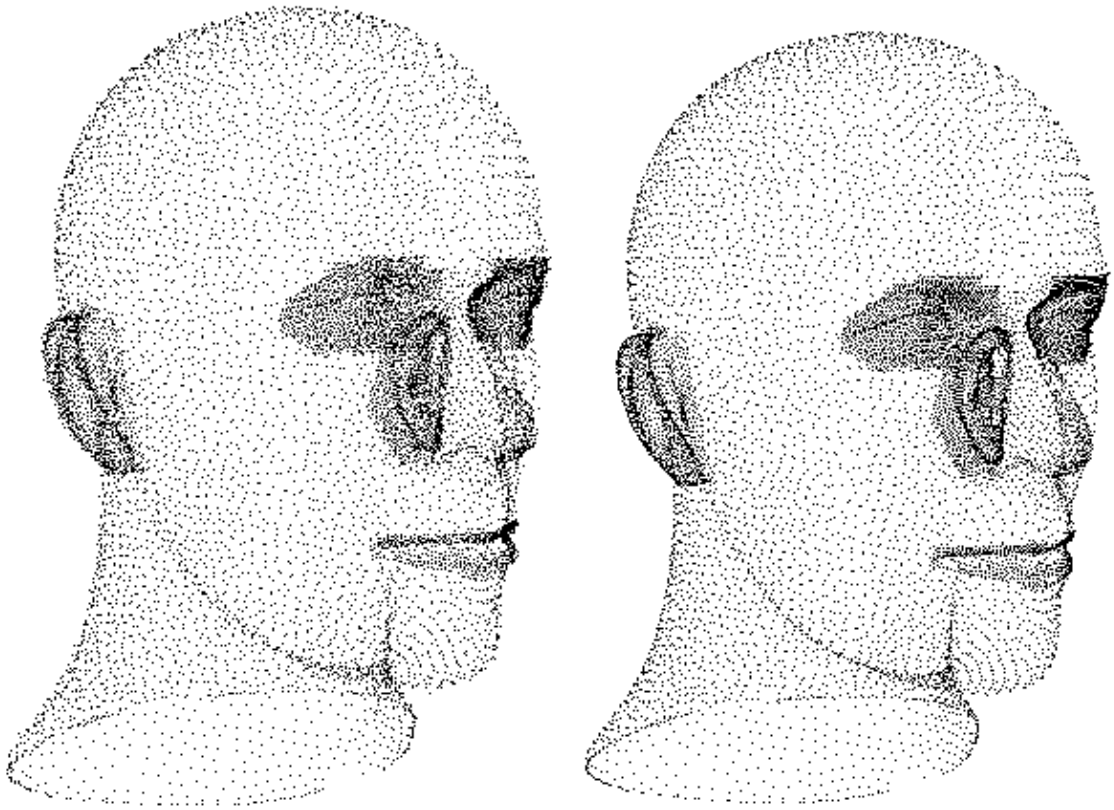


Figure 1: Point clouds: noisy (left) and smoothed (right).

2.2 Clustering

The goal of this first step is to partition the smoothed point cloud Q into a finite set of clusters, such that the curvature of S varies little within each cluster. Since S is not known, its curvature must be estimated from the sample points in Q.

We use a hierarchical clustering method based on a BSP tree. (This method has been used in several recent algorithms; see, for instance, [17].) Each node in the BSP tree contains a subset N = $\{p_1,\ldots,p_n\}$ of Q. To decide whether or not to subdivide the node, we use a covariance analysis of the points in N, that is, we look at the eigenvalues of the covariance matrix C of N:

$$C = \begin{vmatrix} p1 - \bar{p} \\ \vdots \\ pn - \bar{p} \end{vmatrix}^T \begin{vmatrix} p1 - \bar{p} \\ \vdots \\ pn - \bar{p} \end{vmatrix} = \sum_{i=1}^{n} (pi - \bar{p})(pi - \bar{p})^T, \quad \bar{p} = \frac{1}{n}\sum_{i=1}^{n} pi.$$

Since C is a symmetric, positive semi-definite, 3 × 3 matrix, its three eigenvalues are real and we can order them: $\lambda_1 \leq \lambda_2 \leq \lambda_3$. These eigenvalues measure the variation of the points in N along the directions of their corresponding eigenvectors v_1, v_2, v_3.

The eigenvectors v_2 and v_3 define the directions of highest variation and define a regression plane Π for N. The eigenvector v_1 is normal to Π and its eigenvalue λ_1 measures the variation of the points in N with respect to the plane Π. So, small values of λ_1 mean that all points in N are approximately on Π. Hence, the ratio

$$\sigma = \frac{\lambda_1}{\lambda_1 + \lambda_2 + \lambda_3}$$

is naturally a good measure of the flatness of the point set N and can be used as an estimate for the curvature of S around N. We use this flatness measure as one subdivision criterion for the BSP tree.

More precisely, a node is divided in two when both conditions below are satisfied:

1. The ratio σ is larger than a user-defined tolerance σ_{max};
2. The number n of points in N is larger than a user-defined value n_{min}.

The nodes are divided by classifying the points of N with respect to the regression plane Π. The points of N that are on the same side of Π will form a new node. The leaves of the BSP tree are the clusters we seek. See Figure 2.

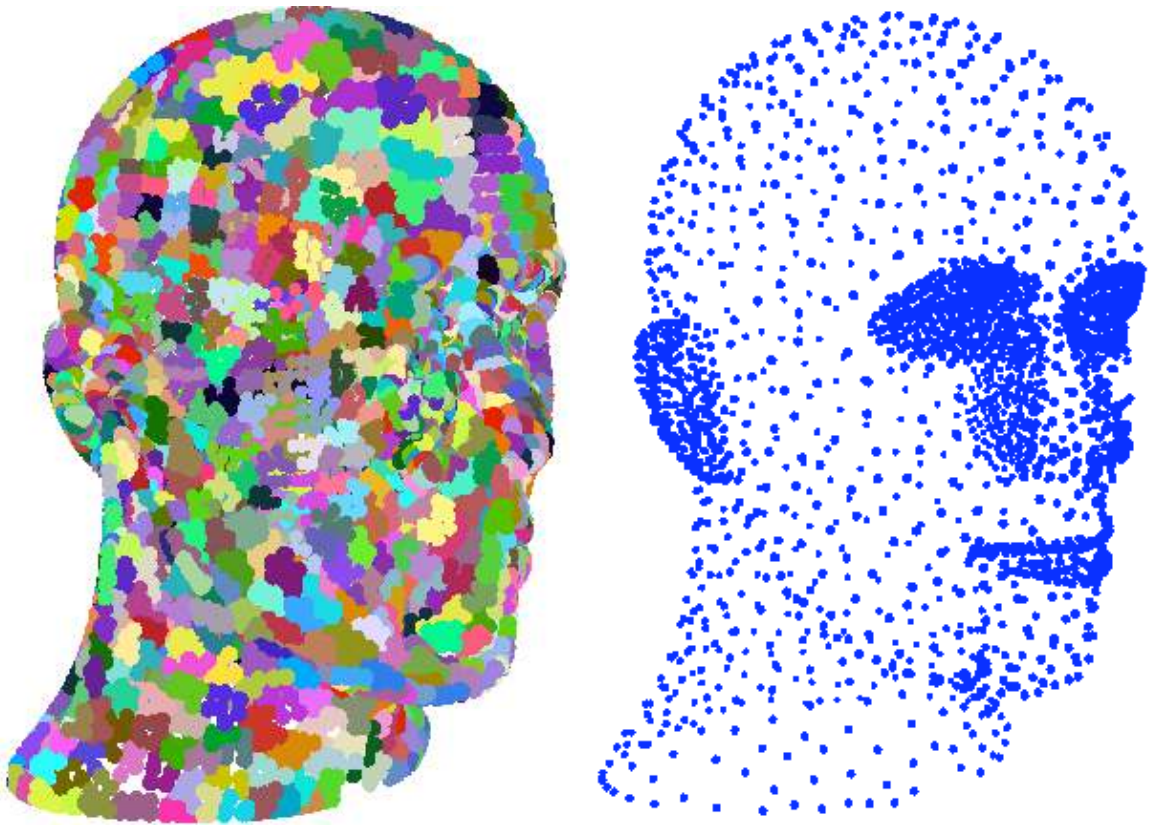


Figure 2: Clusters (left) and representatives (right).

2.3 Reduction

The goal of this step is to find a small set of points to be the vertices of a triangulated surface that will be a rough, initial approximation of the surface S. We do this by selecting a representative point for each cluster found in the previous step, thus effectively reducing the size of the original point cloud while capturing the overall geometry of the underlying surface.

To find a representative point for each cluster, we use the robust projection procedure Q, already used in Section 2.1. We start with the centroid of the points in a cluster N:

$$c = \frac{1}{|N|} \sum_{q \in N} q.$$

This point is not necessarily near S, due to the influence of small curvature in the cluster and to the presence of noise in the original data. In order to move c towards S, we project c onto the point cloud Q using the robust projector procedure Q, as described in Section 2.1. The point Q(c) is the representative of the cluster N. See Figure 2.

2.4 Triangulation

The next step is to build a triangulated surface over the set R of representative points found in the previous step. The goal is to find a rough approximation of the surface S, which will be refined later. We use an incremental triangulation algorithm that introduces new features in the basic framework of previous incremental algorithms [7, 5, 8].

The main idea behind the algorithm is to determine incrementally triangles in the restricted Delaunay triangulation of R (also called restricted Delaunay triangles), because these triangles form a piecewise linear manifold homeomorphic to the original surface [2].

The algorithm computes a sequence of triangulated surfaces with border. At each step, it chooses a border edge, finds a new triangle associated to this edge, and updates the current surface. The algorithm maintains a half-edge data structure H that represents the current surface and a list L of half-edges that represents the current boundary. (We used CGAL [19] for these data structures.) Algorithm 1 is a summary of the main steps:

Algorithm 1:

```
Build a 3d-tree on the set of representatives R.
Get an initial triangle and initialize H and L with it.
repeat
Remove a half edge uv from L.
usv ← the triangle adjacent to uv in H.
K ← set of k-nearest neighbors of u.
q ← GET_POINT(usv, u, K)
Insert uqv into H and its border edges into L.
until L is empty.
```

We start by building a 3d-tree on R. This tree is used to identify efficiently the set K(p) of the k-nearest neighbors of any point $p \in R$.

The initial surface contains a single triangle. This triangle is inserted into H and all its edges are inserted into L. To find the initial triangle, we select a point $p \in R$ and determine its nearest neighbor $q \in R$. These two points define the first border edge, pq. We then find the point $r \in K(p)$ that maximizes the angle $\angle prq$. The initial triangle is the triangle pqr, which is in the Delaunay triangulation of K(p).

Once the initial surface has been found, we continue by removing border edges from L until it is empty. For

each such edge uv, we select a point q ∈ K(u) to create a new triangle uqv, which is then added to H and L.

The number of edges of uqv inserted in L varies: it is zero when the triangle uqv fills a hole in H; it is one when uq or vq are consecutive edges of uv in the border of the current surface; otherwise, it is two. When q is already in H, we have to join two connected components of the border or to split one component in two new connected components. We determine the point q using Algorithm 2. Here, θ, ε_1, and ε_2 are user-defined tolerances.

Algorithm 2: GET_POINT (usv, u, K):
$d_u \leftarrow$ distance from u to its nearest neighbor.
$d_v \leftarrow$ distance from v to its nearest neighbor.
$d_{min} \leftarrow \min\{d_u, d_v\}$.
repeat
$C_\theta \leftarrow$ set of points t such that:
(i) The dihedral angle between the triangles usv
and utv is in $[\pi - \theta, \pi + \theta]$.
(ii) Triangle utv does not violate the topology of H.
(iii) $\max\{d(u,t), d(v,t)\} < \varepsilon_1 d_{min}$.
(iv) $\angle uv\ t < \varphi$ and $\angle v\ ut < \varphi$.
if $C_{\theta/=} \varnothing$ then
Determine the set C'_θ of points $q \in C_\theta$ with maximum angle $\angle uq\ v$.
else
$\theta \leftarrow \theta + \varepsilon_2$.
end if
until $C'_{\theta/=} \varnothing$ or $\theta > \pi 2 + \varepsilon_2$.

Condition (i) in Algorithm 2 is based on the following theorem, which we prove in Section 3:

Theorem 1 Let P be a β,r-sampling of a surface S with $r < \frac{1}{4}$. Then the angle between two adjacent triangles sharing an edge in the restricted Delaunay triangulation of P is at least $\pi - 2(\frac{2r}{1-4r} + \arcsin(\sqrt{3}r(1 - r)))$.

This theorem shows that the dihedral angle between two adjacent restricted Delaunay triangles converges to π as the sampling density increases (that is, as $r \to 0$). We start with an initial region $[\pi - \theta, \pi + \theta]$ with small θ to determine the set of points C_θ in this region that satisfy criteria (ii) and (iii). If C_θ is not empty, we find the point $q \in C_\theta$ with maximum angle $\angle uq\ v$; otherwise, we increase θ by ε_2 and repeat the procedure until we find such a point q or θ gets too large.

Condition (ii) is used to eliminate candidate points t that would violate the topology of the current reconstructed surface if they were selected. We proceed in the following way: For each point $q \in K$ that is on the boundary of the current surface we compute the tangent plane Π_q using the star of q. Then we check whether the projection of the triangle uvt and the projection of the star of q onto Π_q intersect at anything different from a vertex or an edge: in that case we reject the point t as producing a topological violation.

Condition (iii) and (iv) are used to determine border edges. Condition (iii) is based on the following theorem, which is proved in Section 3:

Theorem 2 Let P be a β,r-sampling of a surface S with $r < \frac{}{13}$. Let T_1 and T_2 be two adjacent triangles sharing an edge uv in the restricted Delaunay triangulation of P. Then:

- The length of the longest edge of T_2 is at most $\frac{2\beta}{1-3r}d$, where $d = \min\{d_u, d_v\}$ and d_u (respectively, d_v) is the distance of u (respectively, v) to its nearest neighbor.
- If s is a sample point not on the same connected component as v, then the distance between s and v is larger than $\frac{1}{}r d_u$.

This theorem shows that there is little variation between the length of the edges of two restricted Delaunay triangles adjacent to an interior edge. In our experiments we observed that the length of edges of the adjacent of the candidate triangles to a boundary edge are very different. The theorem also gives a lower bound on the length of a candidate edge joining two points on different connected components. Based on this bound, condition (iii) guarantees that, for a sufficiently dense sampling, the algorithm never joins points from separate components.

Condition (iv) gives a bound for the angles adjacent to the edge uv of a restricted Delaunay triangle and is based on the following theorem, which is proved in Section 3:

Theorem 3 Let P be a β,r-sampling of a surface S. Let T be a triangle in the restricted Delaunay triangulation of P. Then the angles in T are less than $2\arccos(\frac{}{1-r\ 2\beta})$.

In practice, we have no way to estimate β and r, and so we use a user-selected value ε_1 as the constant $\frac{2\beta}{1-3r}$ and a user-selected angle φ instead of $2\arccos(\frac{}{1-\beta r})$ (we got good results with $\varphi = 140^{\circ}$, for sufficiently dense samplings). A point t is eliminated if $\max\{d(u,t),d(v,t)\} \geq \varepsilon_1 d_{min}$ or if one of the angles $\angle uv\ t$ and $\angle uv\ t$ is at least φ. If C_θ is empty, then the edge uv is a candidate border edge. If C_θ is empty for all θ tested, then uv is a border edge.

If C_θ is not empty, we compute the subset C'_θ of points q with maximum angle $\angle uq\ v$. We justify this criterion as follows: In R^2, given a Delaunay triangle uvs, its adjacent Delaunay triangle uvq is the one with maximum angle $\angle uq\ v$. Although this is a criterion for R^2, we use it for surfaces because the surface normal varies little in the surface neighborhood of a point u containing the vertex of the restricted Delaunay triangulation (see [1, 2, 3, 4]); in other words, we may consider this neighborhood as flat.

This triangulation algorithm is fast (timings are given in Section 4). Figure 3 shows the triangulated surfaces over the set of representatives shown in Figure 2.

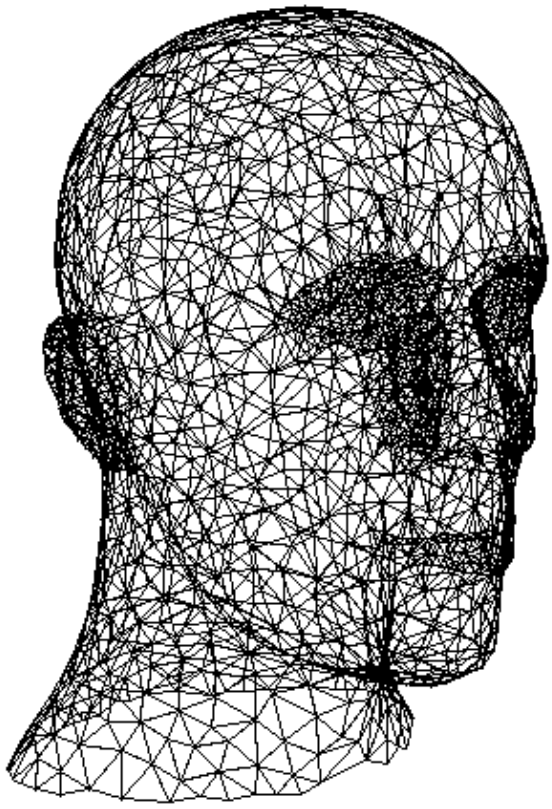


Figure 3: Triangulation.

2.5 Refinement

The goal of this step is to refine the initial coarse triangulation, built in the previous step, into a finer triangulation adapted to the geometry of the unknown surface S. The refinement is done by refining each edge in the initial triangulation, until there are no more edges to be refined.

We propose the following method to refine an edge uv: We start with the middle point of the edge. If this point is too far from the point cloud Q, then it is projected onto Q and a new edge is added to each adjacent triangle of the edge uv, dividing each triangle into two new triangles. We repeat this procedure for each edge in the original triangulation. The details are discussed below.

More precisely, for each edge uv we compute its midpoint m and its normal n_m:

$$m = u+-v,\ nm = -nu-+-nv-,\ 2\ //nu + nv//$$

where n_u and n_v are the normals of u and v respectively, computed using the local triangulation (star) of each vertex.

Using the set K = K(m) of k-nearest neighbors of m in the point cloud Q, we minimize the following functional with respect to t:

$$\sum \Phi(t) = //p- m + t\cdot nm\ //2.\ p\in K$$

This is interpreted as finding the point on the line through m in the direction of n_m that is closest to the original surface S. The minimum of Φ(t) occurs at

$$1\ \sum \alpha = ---\ (p- m)\cdot nm.\ k\ p\in K$$

We use α as a measure of the need for refinement. If α is smaller than a fixed, user-defined tolerance ε, then we do not refine the edge; otherwise, we project the point $m + \alpha \cdot n_m$ onto the smoothed point cloud Q using the robust projection operator Q described in Section 2.1, producing the new point $\bar{m} = Q(m + \alpha \cdot n_m)$. Then we replace the triangles adjacent to uv, say auv and buv, with the new triangles $u\bar{m}a$, $v\bar{m}a$ and $u\bar{m}b$, $vm\bar{b}$, respectively. If the ratio between the length of the minimum edge and the maximum edge of the triangles auv or buv is too small, we do not divide the triangles.

This procedure is applied to all the old edges for which α is larger than ε. In the resulting mesh, we flip the diagonals of each quadrilateral formed by the pairs of adjacent triangles if the length of one diagonal (the common edge) is larger than the length of the other diagonal (the segment joining the opposite vertex).

The final result is a refined triangulated surface near the original point cloud that captures its salient features. Figure 4 shows two refinement steps of the initial triangulation shown in Figure 3.

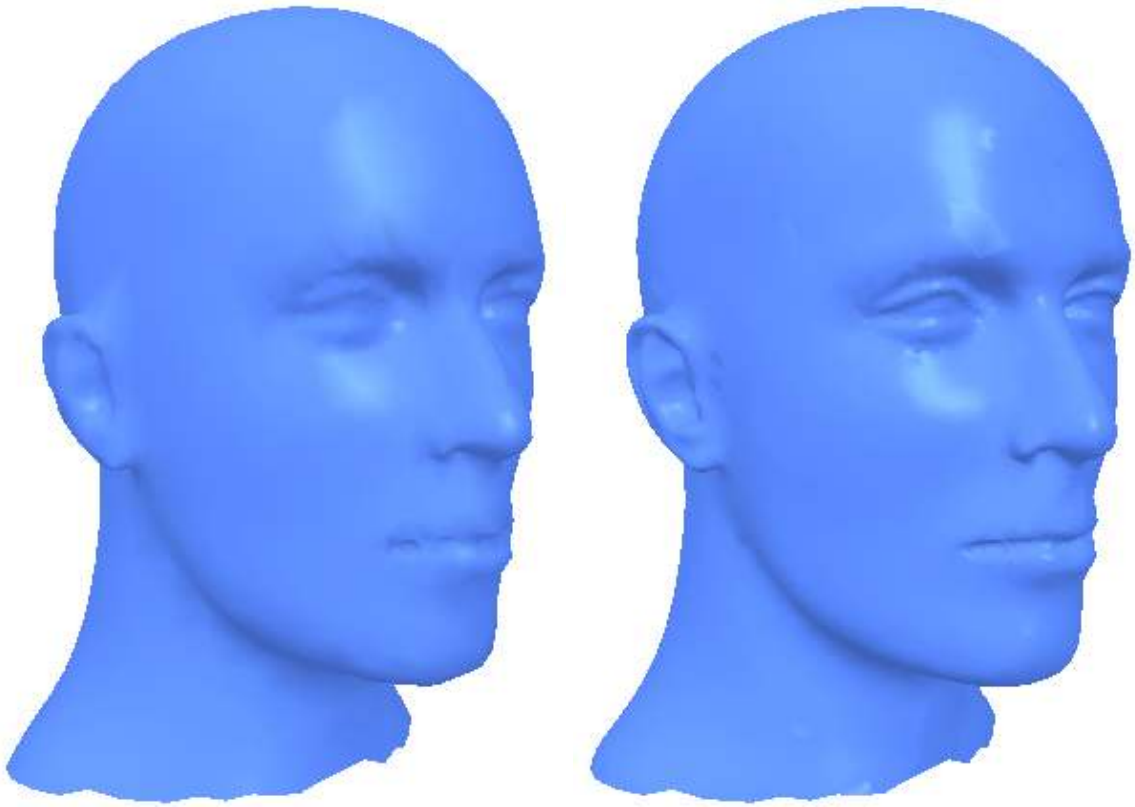


Figure 4: Refinement.

3 Theory

We now prove the theorems that justify the heuristics used in the reconstruction algorithm explained in Section 2.4. We start with a short introduction to the related theory.

Amenta et al. [2] defined the local feature size lfs(x) of a sample point x as the distance of x to the medial

axis of the surface S. They also defined the concept of r-sample:

Definition 1 A set $P \subseteq S$ is an r-sample if we have $B(p, r \cdot \mathrm{lfs}(p)) \cap P \neq \emptyset$ for all $p \in P$.

Here, and in the sequel, B(p,r) is the ball centered at p with radius r. We shall adopt a different sampling criterion, proposed by Dey et al. [10]:

Definition 2 A set $P \subseteq S$ is a β,r-sample, for $\beta > 1$, if the following two conditions are valid:

1. $B(x, r \cdot \mathrm{lfs}(x)) \cap P \neq \emptyset$ for all $x \in S$.
2. $B(p, r\beta \cdot \mathrm{lfs}(p)) \cap P = \{p\}$ for all $p \in P$.

The Voronoi diagram of a set of samples $P \subset S$ induces a decomposition on the surface S called the restricted Voronoi diagram. The restricted Voronoi cell associated to a point $p \in P$ is $V_p = V_p^{\circ} \cap S$, where V_p° is the Voronoi cell of p in the Voronoi diagram of P. The restricted Delaunay triangulation is the dual of the restricted Voronoi diagram, obtained in the following way: pq is an edge of the restricted Delaunay triangulation if $V_p \cap V_q \neq \emptyset$; pqr is a face of the restricted Delaunay triangulation if $V_p \cap V_q \cap V_r \neq \emptyset$. Amenta et al. [2] proved that the polyhedron defined by the restricted Delaunay triangulation is homeomorphic to the original surface for sufficiently dense r-samplings ($r \leq 0.1$).

The following two lemmas were proved by Amenta et al. [2]. The first lemma bounds the angle between the normals of points sufficiently close. The second lemma bounds the angle between the normal to a restricted Delaunay triangle and the surface normal at the vertex with angle at least $\pi/3$.

Lemma 1 If $r < 1/3$, then, for any two points $p,q \in P$ with $d(p,q) < r \cdot \min\{\mathrm{lfs}(p), \mathrm{lfs}(q)\}$, the angle between the normals n_p and n_q is at most $\frac{r}{1-r}3r$

Lemma 2 If T is a restricted Delaunay triangle and s is a vertex of T with angle at least $\pi/3$, then the angle between the normal of S at s and the normal of T is at most $\arcsin(\frac{\sqrt{3}r}{1-r})$.

These two lemmas are used to prove Theorem 1, which bounds the angle between two adjacent restricted Delaunay triangles:

Theorem 1 Let P be a β,r-sampling of a surface S with $r < 1/4$. Then the angle between two adjacent triangles sharing an edge in the restricted Delaunay triangulation of P is at least $\pi - 2(\frac{2r}{1-4r} + \arcsin(\frac{\sqrt{3}r}{1-r}))$.

Proof: Let $T_1 = p_1p_2p_3$ and $T_2 = p_1p_2q_3$ be two restricted Delaunay triangles sharing an edge p_1p_2 and let $s = V_{p_1} \cap V_{p_2} \cap V_{p_3}$. Because the points of P nearest to s are p_1, p_2, and p_3, we have $d(s,p_i) < r \cdot \mathrm{lfs}(s)$ (i = 1,2,3). Since $\mathrm{lfs}(s) < d(s,p_i) + \mathrm{lfs}(p_i)$ and $d(s,p_i) < r \cdot \mathrm{lfs}(s)$, we obtain $\mathrm{lfs}(s) < \frac{1}{1-r} \mathrm{lfs}(p_i)$, which implies

$$d(s,p_i) < \frac{r}{1-r} \min\{\mathrm{lfs}(s), \mathrm{lfs}(p_i)\}.$$

It follows from Lemma 1 that the angle between the normals n_s and n_{p_i} is less than $\frac{r}{1-4r}$ Hence, the angle between the two normals n_{p_i} and n_{p_j} is less than $\frac{2r}{1-4r}$ We assume without lost of generality that p_3 is a vertex with angle at least $\pi/3$. Applying Lemma 2, we conclude that the angle between the normal of the surface at p_3 and the normal of the triangle T_1 is less than $\arcsin(\frac{\sqrt{3}r}{1-r})$. Combining these inequalities, we get:

$$\angle(n_{p_1}, T_1) < \angle(n_{p_1}, n_{p_3}) + \angle(n_{p_3}, T_1) < \frac{2r}{1-4r} + \arcsin(\frac{\sqrt{3}r}{1-r}).$$

In particular, $\angle(n_{p_1},T_1) < \frac{2r}{1-r} + \arcsin(\frac{r}{1-r})$. Analogously, we get $\angle(n_{p_1},T_2) < \frac{2r}{1-r} + \arcsin(\frac{r}{1-r})$, Hence, the angle between T_1 and T_2 is at least $\pi - 2(\frac{2r}{1-r} + \arcsin(\frac{r}{1-r}))$, as stated.

Note that β was not used in the proof and so this theorem is valid in the more general case of an r-sampling.

The next two results are used in the proof of Theorem 2. They are also from Amenta et al. [2], but do not appear explicitly as lemmas. We include them here for completeness.

Lemma 3 Let P be an r-sampling of S with $r < 1$. Then the distance between two adjacent vertices uv in the restricted Delaunay triangulation of P is less than $\frac{2r}{1-r}\,\mathrm{lfs}(u)$.

Proof: Let T = uvw be a restricted Delaunay triangle and let s be its dual restricted Voronoy vertex. Then $d(u,v) < 2d(u,s) < 2r \cdot \mathrm{lfs}(s)$. On the other hand, since $d(u,s) < r \cdot \mathrm{lfs}(s)$ and $\mathrm{lfs}(s) < d(u,s) + \mathrm{lfs}(u)$, we obtain $\mathrm{lfs}(s) < \frac{1}{1-r}\,\mathrm{lfs}(u)$. Hence, $d(u,v) < \frac{2r}{1-r}\,\mathrm{lfs}(u)$, as claimed.

Lemma 4 The radius of the Voronoi ball circumscribing a restricted Delaunay triangle T is less than $\frac{r}{1-r}\,\mathrm{lfs}(x)$, where x is any of the vertices of T.

Proof: Let c be the restricted Voronoi vertex that is the center of the ball circumscribing T. Then the radius of the ball is d(c,x) and $d(c,x) < r \cdot \mathrm{lfs}(c)$. Since $\mathrm{lfs}(c) < d(c,x) + \mathrm{lfs}(x)$, we conclude that $\mathrm{lfs}(c) < \frac{1}{1-r}\,\mathrm{lfs}(x)$. Hence, $d(c,x) < \frac{r}{1-r}\,\mathrm{lfs}(x)$.

We can now prove Theorem 2, which defines the criterion used to determine border edges and guarantees that points in different connected components are never joined by an edge.

Theorem 2 Let P be a β,r-sampling of a surface S with $r < \frac{1}{3}$. Let T_1 and T_2 be two adjacent triangles sharing an edge uv in the restricted Delaunay triangulation of P. Then:

- The length of the longest edge of T_2 is at most $\frac{2}{1-\beta-3r}d$, where $d = \min\{d_u,d_v\}$ and d_u (respectively, d_v) is the distance of u (respectively, v) to its nearest neighbor.
- If s is a sample point not on the same connected component as v, then the distance between s and v is larger than $\frac{1}{1-r}d_u$.

Proof: a) Since $\mathrm{lfs}(v) < d(u,v) + \mathrm{lfs}(u)$ and $d(u,v) < \frac{2r}{1-r}\,\mathrm{lfs}(v)$ by Lemma 3, we have that

$$2r\,1-3r\,\mathrm{lfs}(u) > \mathrm{lfs}(v)\text{------}\mathrm{lfs}(v) =\text{------}\mathrm{lfs}(v).\ 1-r\ 1-r$$

By symmetry, and because $\frac{1-r}{1-3r} > 1$, we conclude

$$\mathrm{lfs}(y) > 1--3r\text{-}\mathrm{lfs}(x)\ 1-r$$

for $x,y \in \{u,v\}$. Now $d_u > r\beta\,\mathrm{lfs}(u)$ and $d_v > r\beta\,\mathrm{lfs}(v)$. Hence,

$$d = \min\{du,dv\} > r(1--3r)\mathrm{lfs}(x),\ \beta(1-r)$$

for x = u,v. On the other hand, any edge of the triangle T_2 is inside the Voronoi ball with center at $c = V_u \cap V_v \cap V_t$. By Lemma 4, the diameter of this Voronoi ball is less than $\frac{2r}{1-r}\,\mathrm{lfs}(x)$ for x = u,v. Thus, the length of the longest edge of T_2 is at most $\frac{2r}{1-r}\,\mathrm{lfs}(x) < \frac{2\beta}{1-3r}d$.

b) Because s is on a different connected component we have that $d(s,v) > \mathrm{lfs}(v)$ since the segment joining s and v has to intersect the medial axis. On the other hand, we have that $d_u < r \cdot \mathrm{lfs}(v)$ and the inequality follows.

Our last theorem bounds the internal angles of a restricted Delaunay triangle:

Theorem 3 Let P be a β,r-sampling of a surface S. Let T be a triangle in the restricted Delaunay triangulation of P. Then the angles in T are less than $2\arccos(\frac{}{12-\beta r})$.

Proof: Let T = uvt be a restricted Delaunay triangle and let s be its dual restricted Voronoi vertex. Let α_1 = /uv s and α_2 = /sv t. Then /uv s ≤ $\alpha_1 + \alpha_2$. Let us concentrate on α_1 for the moment. Because T_1 = uvr is an isosceles triangle, α_1 is acute and

$$d(u,v)\text{- }1\text{--r- }\cos(\alpha 1) = 2d(s,v) > 2\beta ,$$

because $d(u,v) > \frac{}{r\ \beta}$ lfs(v) and $d(s,v) < \frac{}{\text{-r- }1\text{-r}} \cdot$ lfs(v) by Lemma 4. Analogously, $\cos(\alpha_2) > \frac{}{1\text{-r }2\beta}$. Hence, /uv s ≤ $\alpha_1 + \alpha_2 \le 2\arccos(\frac{}{1-2r\beta})$, as stated.

4 Results

We tested our algorithm on several data sets available on the Internet [20, 21]. These data were contaminated with noise by disturbing each point a small fraction of the diagonal of the bounding box of the point cloud. The results appear in Figures 5–8, which show the noisy data, the initial triangulation and the final triangulation after two refinement steps. (The pictures in this paper are available in full size and color at http://www.impa.br/~boris/Research.html.)

Table 1 shows the times (in seconds) taken in each step of the algorithm and the sizes (points and triangles) of the result models. Note that the smoothing step dominates the total time. Table 2 shows the parameters used in each case; σ_ρ and σ_w are given as a fraction of h, the mean separation between sample points. The Knot model required two smoothing passes; the second pass used σ_ρ = 2.5h and σ_w = 1.5h.

M	S	C	T	R1	R2
Dino	56194 p	8613 p	17187 t	51561 t	154683 t
179.25 s	163.21 s	3.53 s	6.06 s	1.64 s	4.81 s
Dog	195586 p	14658 p	29308 t	87924 t	263772 t
1044.16 s	1004.96 s	12.16 s	15.22 s	2.99 s	8.83 s
Igea	134359 p	6512 p	13020 t	39060 t	117180 t
427.57 s	407.94 s	7.52 s	6.76 s	1.38 s	3.97 s
Dragon	100056 p	7448 p	14788 t	44364 t	133092 t
312.74 s	292.94 s	5.92 s	7.97 s	1.46 s	4.45 s
Knot	71255 p	4224 p	8406 t	25218 t	75654 t
418.7 s	407.71 s	3.66 s	4.27 s	0.7 s	2.36

Table 1: Results on a 1.8Ghz Pentium 4 with 512Mb of RAM running Linux (M=model, S=Smoothing, C=Clustering, T=Triangulation, R1=Refinement 1, R2=Refinement 2, s=seconds, p=points, t=triangles).

Table 2: Parameter values.

5 Conclusion

We have presented a complete framework for surface reconstruction in the presence of noise, giving as output a refined triangular mesh with points near the original point cloud. This framework combines a method for removing noise from the data [15] with a method for surface reconstruction that first triangulates a simplified point cloud and then refines it. The simplification step partitions the original point cloud into clusters of approximately coplanar points and then selects one representative point from each cluster to form the simplified point cloud. The triangulation algorithm does not need to compute 3D Delaunay triangulations, which in the worst case requires quadratic time and which is prone to numerical errors. The refinement method is fast and gives a fine triangular mesh adapted to the geometry of the underling surface.

Our method is controlled by several parameters that can be selected by the user to suit the data. Parameters in the range of those used in Section 4 seem to work well for data sets similar to those presented here. The selection of a smaller set of parameters and the automatic determination of those parameters are open issues left for future work.

We have provided some theoretical results that motivated the heuristics used in the reconstruction step. We say heuristics because in practice it is not possible to verify that the assumptions needed to prove those theorems hold, specially for noisy data. In particular, it is not possible to verify the sampling conditions used originally by Amenta et al. [2] and Dey et al. [10], because neither the underlying surface nor the sampling process are known. Nonetheless, based on several tests with noisy data, we believe that the smoothing step does preserve the salient features of the data and that the heuristics used in the reconstruction step, which are motivated by similar theoretical results, do preserve the topological and geometric information required for surface reconstruction.

Acknowledgments. The authors are partially supported by CNPq research grants. The authors are members of Visgraf, the computer graphics laboratory at IMPA, which is sponsored by CNPq, FAPERJ, FINEP, and IBM Brasil. This work is part of Boris Mederos's doctoral thesis at IMPA.

References

1. Amenta N., Bern M., Kanvisellis M.: A new Voronoi based surface algorithm. SIGGRAPH'98, pp. 415–421, 1998.

2. Amenta N., Bern M.: Surface reconstruction by Voronoi filtering, Discrete and Computational Geometry 22(4):481–504, 1999.

3. Amenta N., Choi S.: One-pass Delaunay filtering for homeomorphic 3D surface reconstruction. Technical Report TR99-08, University of Texas at Austin, 1999.

4. Amenta N., Choi S., Kolluri R.: The power crust. Proc. 6th ACM Symp. on Solid Modeling, pp. 249–260, 2001.

5. Bernardini F., Mittelman J., Rushmeier H., Silva C., Taubin G.: The ball-pivoting algorithm for surface reconstruction. IEEE Transactions on Visualization and Computer Graphics 5(4):349–359, 1999.

6. Boissonnat J.-D., Cazals F.: Smooth surface reconstruction via natural neighbor interpolation of distance

function. Computational Geometry 22(1–3):185–203, 2002.

7. Boissonnat J.-D.: Geometric structures for three-dimensional shape reconstruction. ACM Transactions on Graphics 3(4):266–289, 1984.

8. Boyer E., Petitjean S.: Curve and surface reconstruction from regular and non regular point sets. Computational Geometry 19(2–3):101–126, 2001.

9. Crossno P., Angel E.: Spiraling edge: fast surface reconstruction from partially organized sample points. Proc. Visualization'99, pp. 317–324, 1999.

10. Dey T. K., Giesen J., Goswami S., Zhao W.: Shape dimension and approximation from samples. Discrete and Computational Geometry 29(3):419–434, 2003.

11. Freedman D.: Efficient simplicial reconstruction of manifolds from their samples. IEEE Transaction on Pattern Analysis and Machine Intelligence 24(10):1349–1357, 2002.

12. Gopi M., Krishnan S., Silva C.: Surface reconstruction based on lower dimensional localized Delaunay triangulation. Computer Graphics Forum 19(3):C467–C478, 2000.

13. Gopi M., Krishnan S.: A fast and efficient projection-based approach for surface reconstruction. Proc. SIBGRAPI 2002, pp. 179–186.

14. Hoppe H., DeRose T., Duchamp T., McDonald J., Stuetzle W.: Surface reconstruction from unorganized points. Proc. SIGGRAPH'92, pp. 71–78, 1992.

15. Mederos B., Velho L., Figueiredo L. H.: Robust smoothing of noisy point clouds. To appear in Proc. SIAM Conference on Geometric Design and Computing 2003, 2004.

16. Ohtake Y., Belyaev A., Alexa M., Turk G.: Multi-level partition of unity implicits. Proc. SIGGRAPH'03, pp. 463–470, 2003.

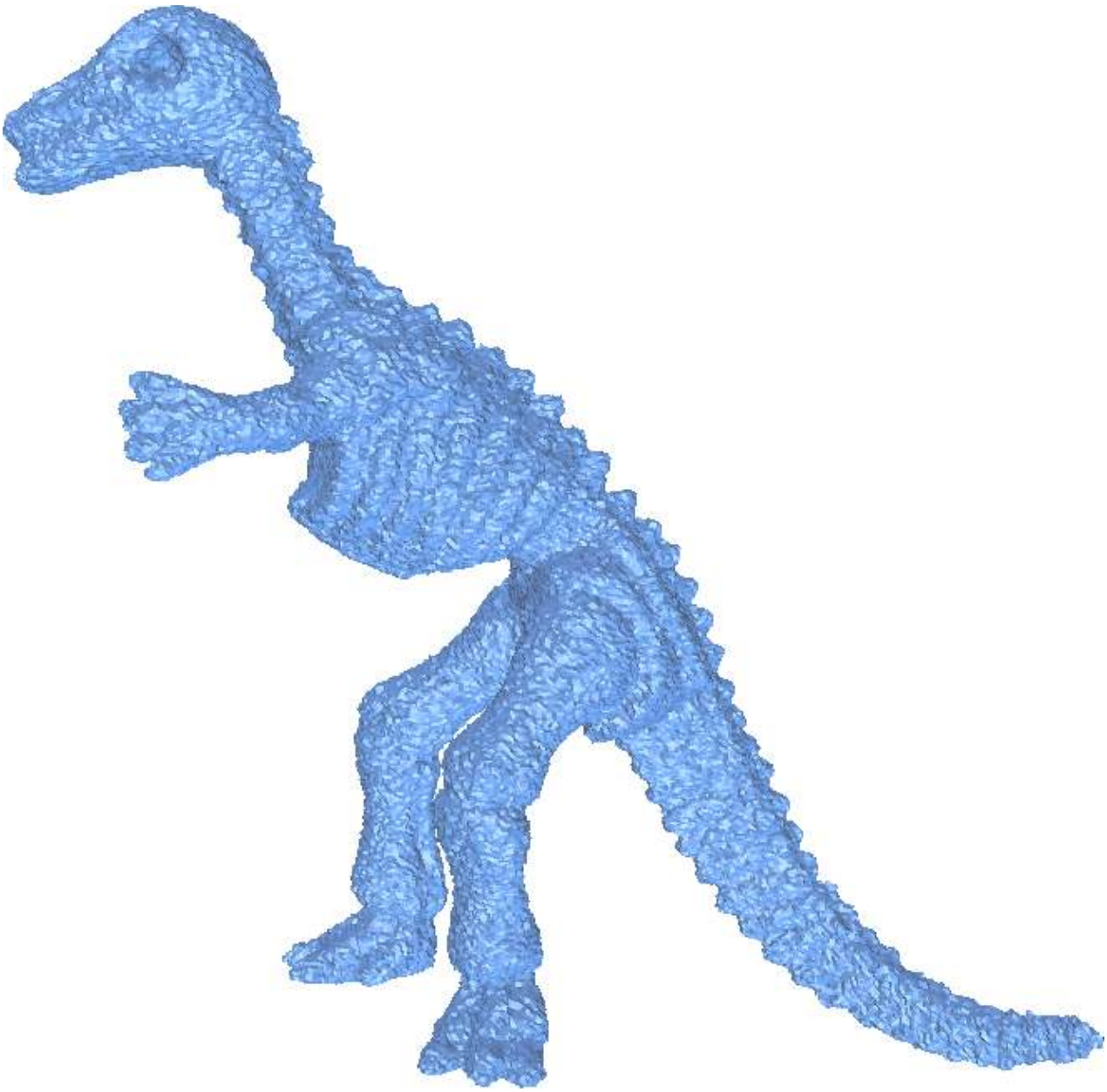
17. Pauly M., Gross M., Kobbelt L.: Efficient simplification of point-sampled surfaces. Proc. Visualization'02, pp. 163–170, 2002.

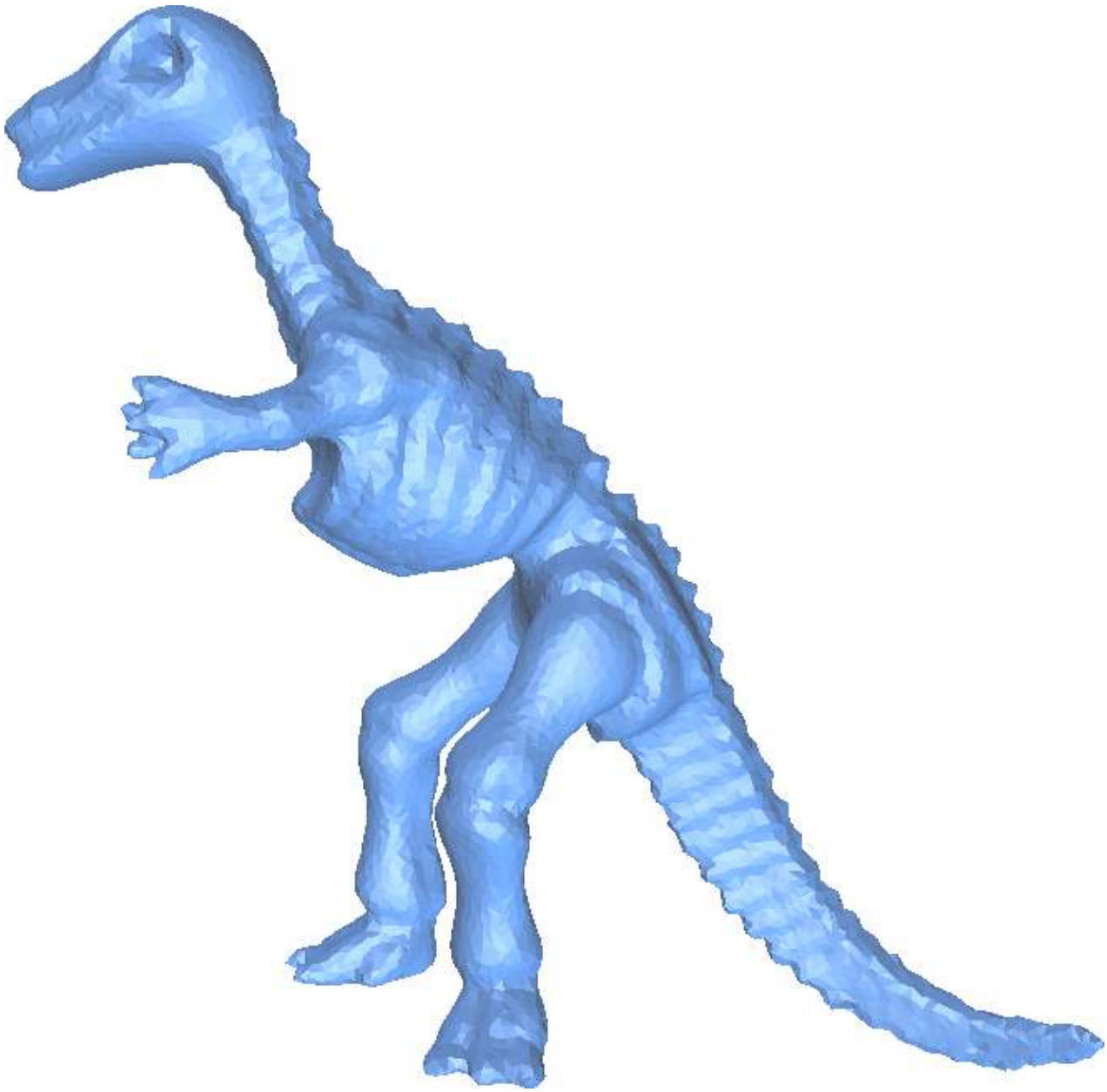
18. Xie H., Wang J., Hua J., Qin H., Kaufman A.: Piecewise C^1 continuous surface reconstruction of noisy point clouds via local implicit quadric regression. Proc. Visualization'03, pp. 91–98, 2003.

19. Computational Geometry Algorithms Library.
http://www.cgal.org

20. Large Geometric Model Archive.
http://www.cc.gatech.edu/projects/large_models

21. Hugues Hoppe's home page.
http://research.microsoft.com/~hoppe/





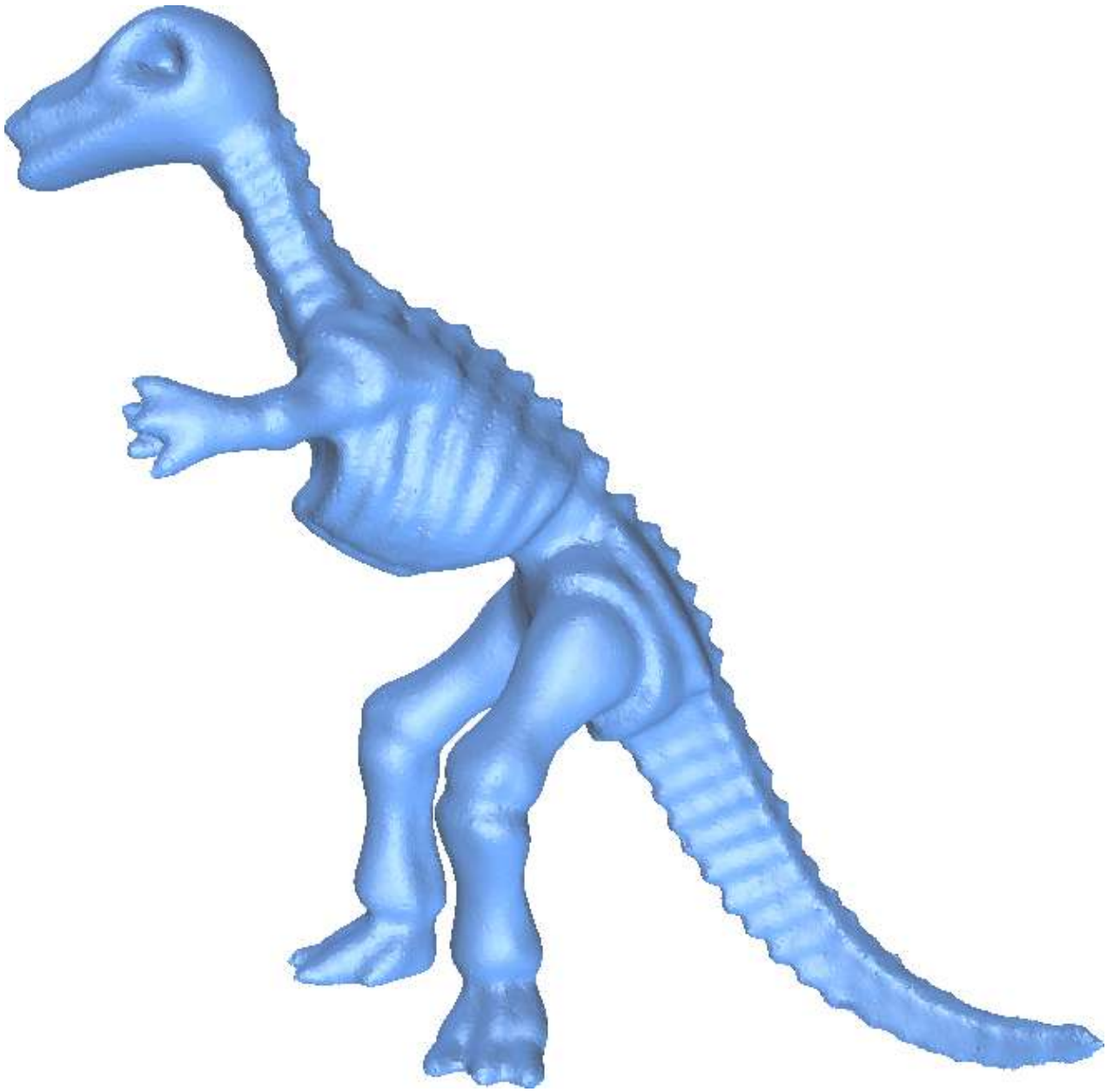
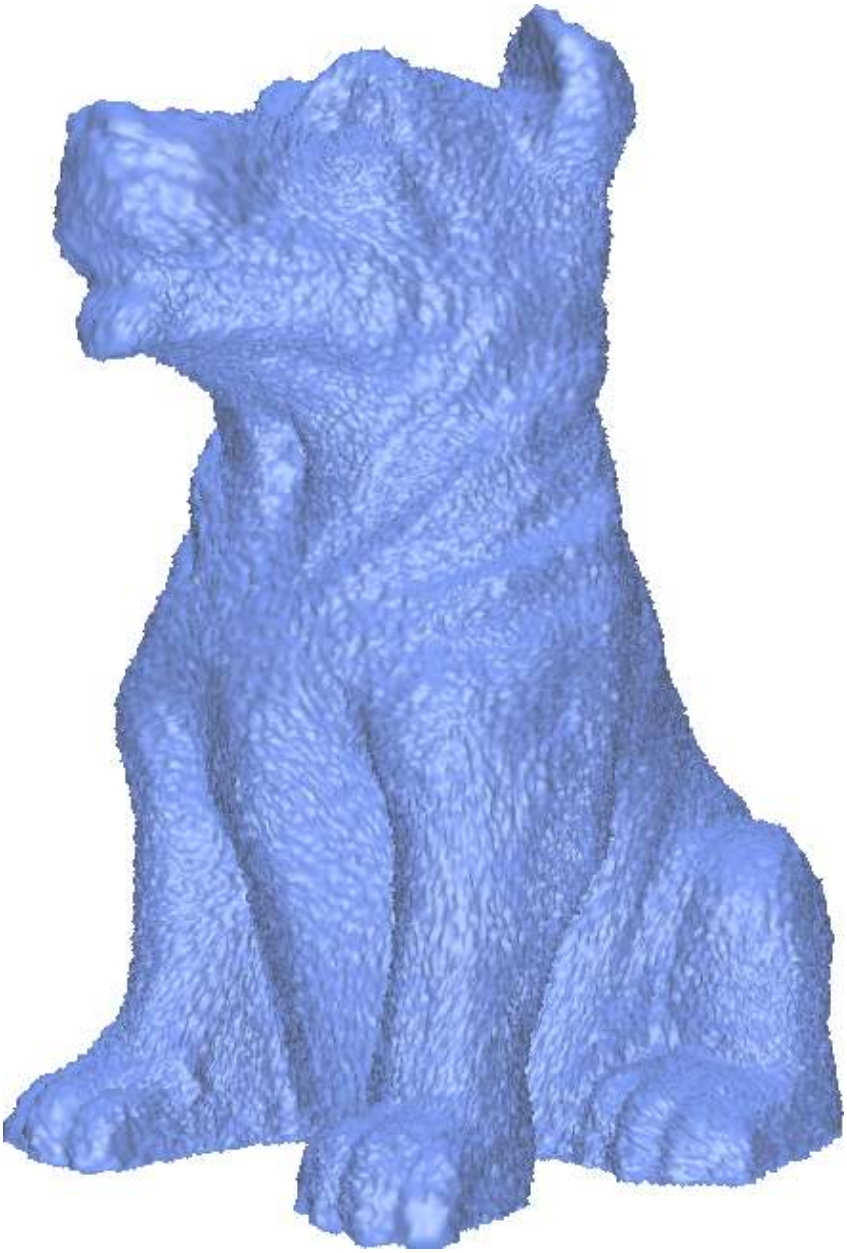
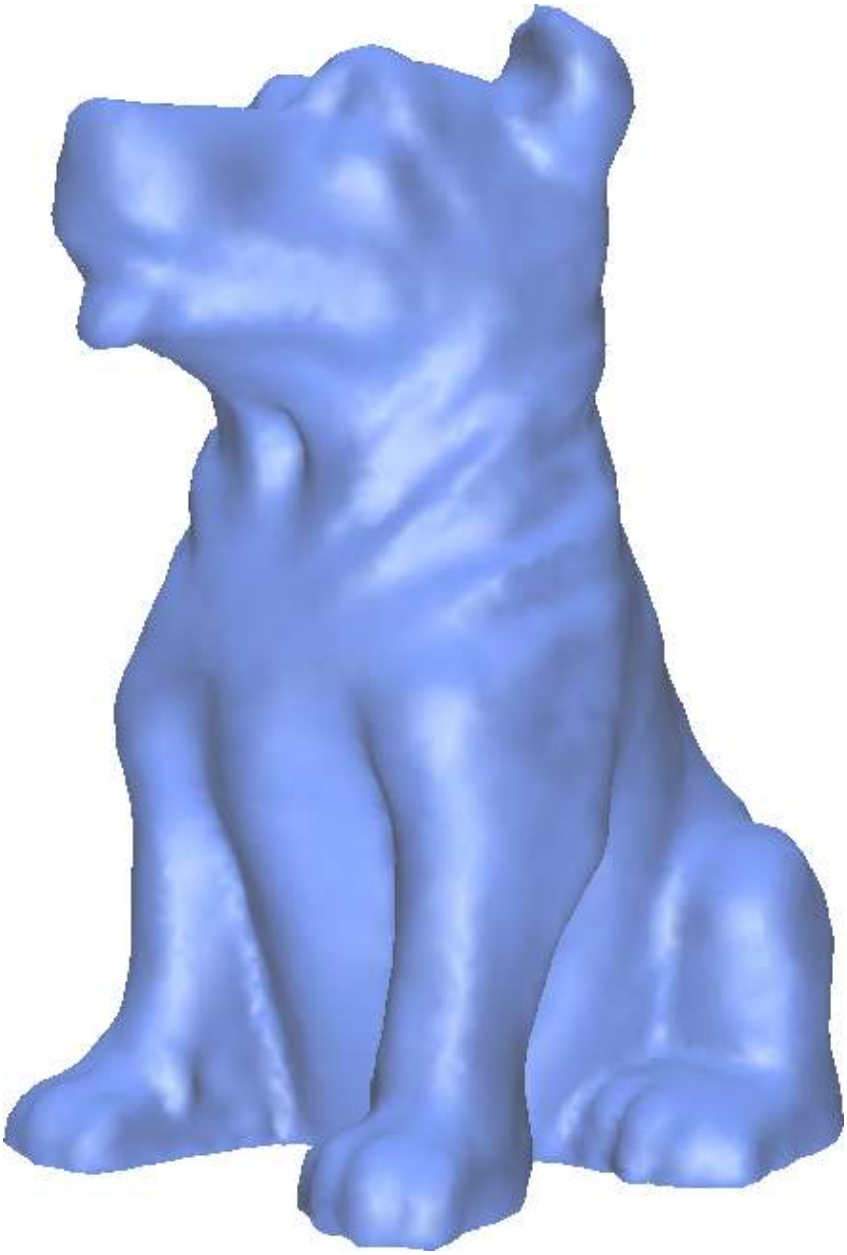


Figure 5: Smooth reconstruction of Dino: noisy data (top), initial triangulation (middle), and final model (bottom).





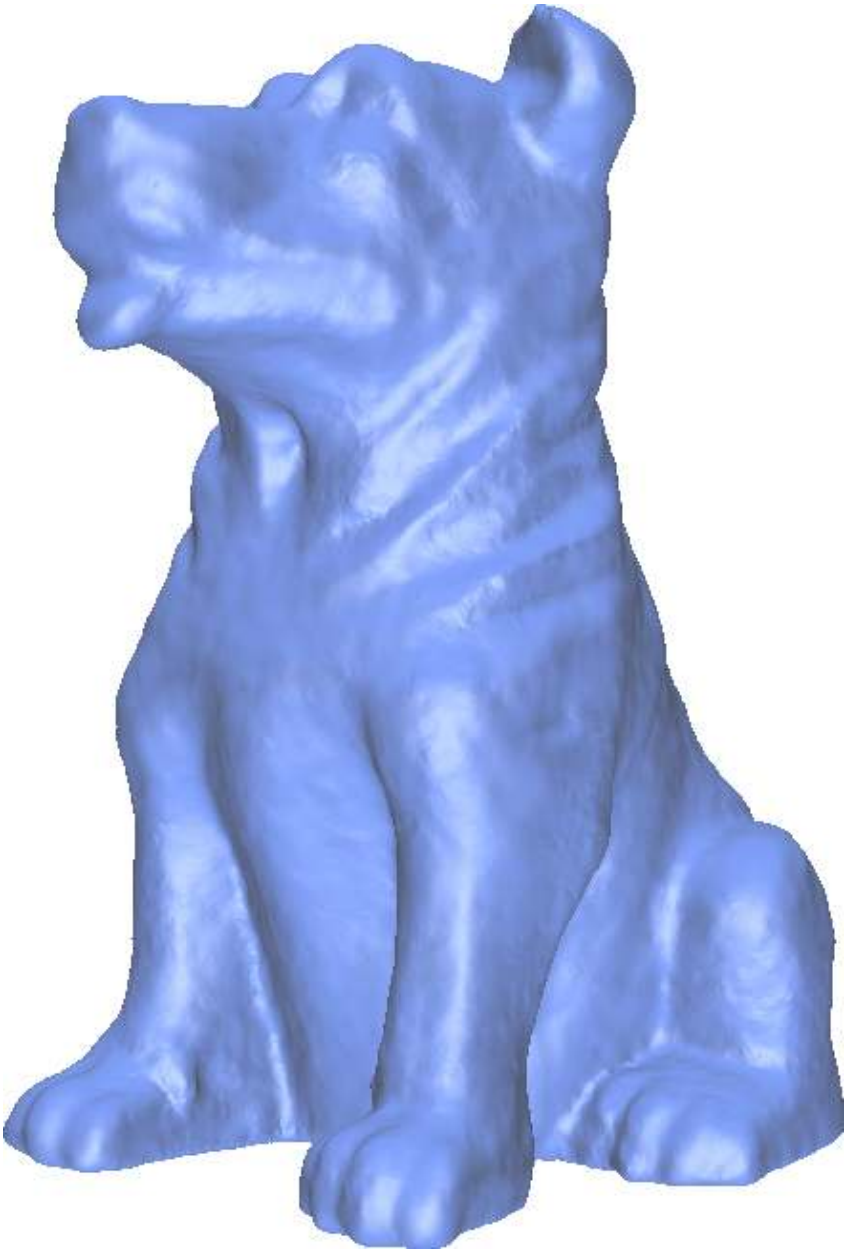
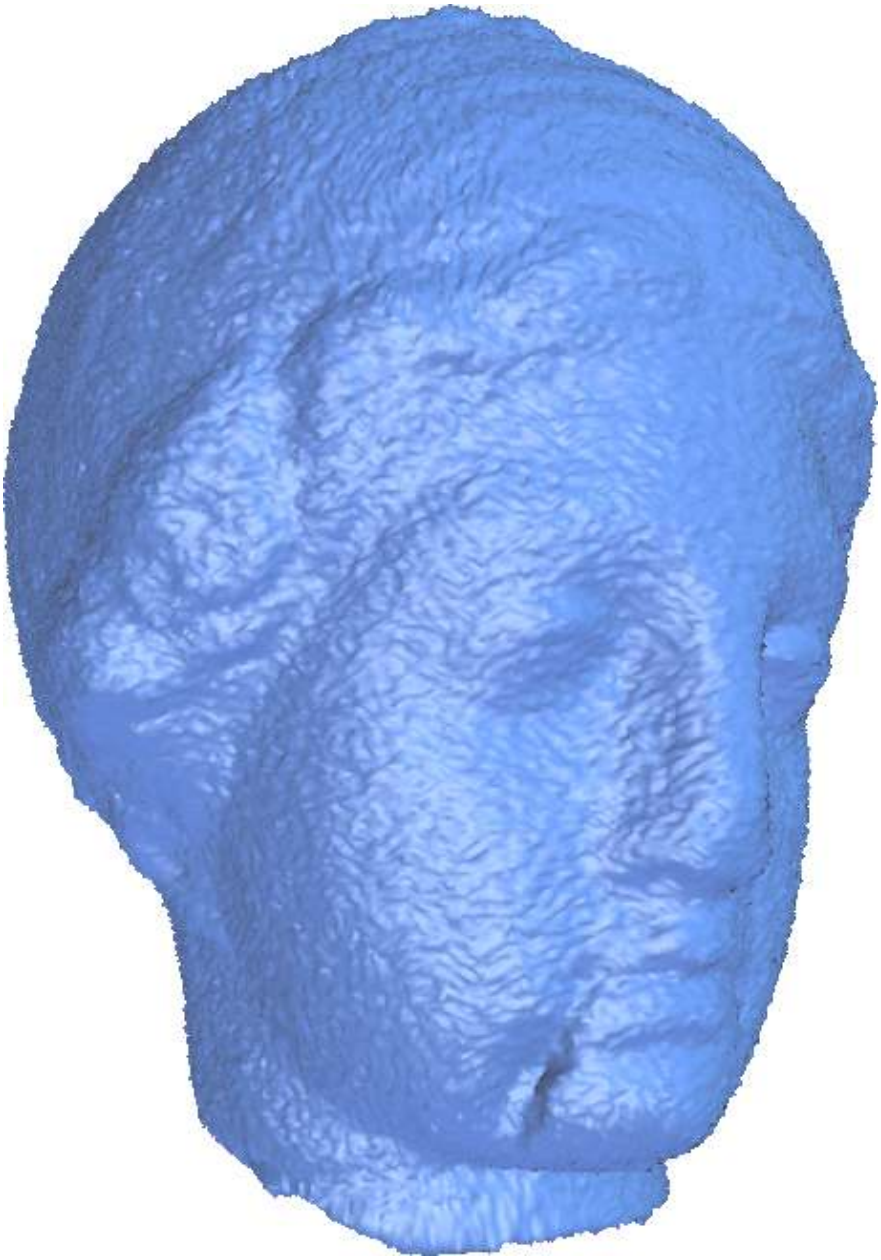


Figure 6: Smooth reconstruction of Dog: noisy data (left), initial triangulation (middle), and final model (right).



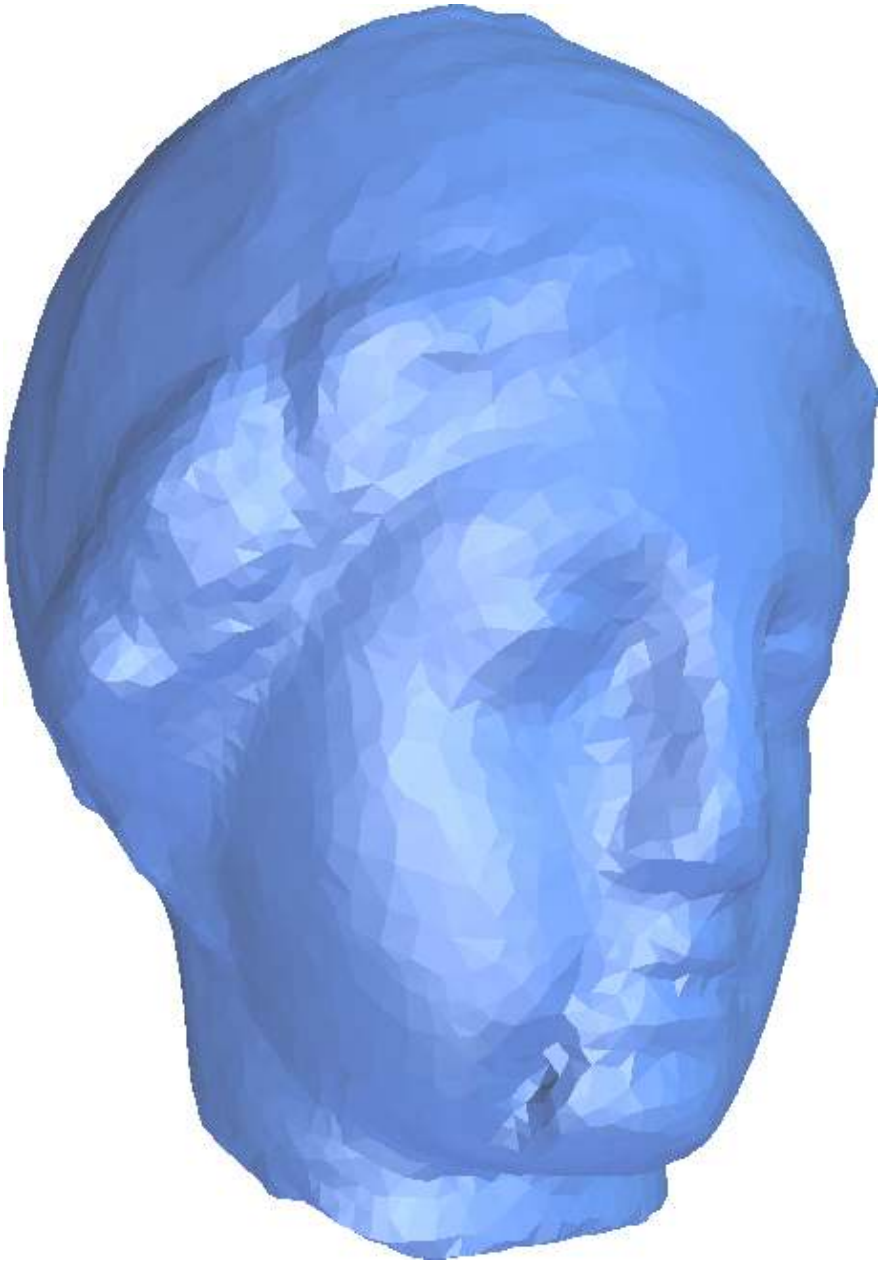
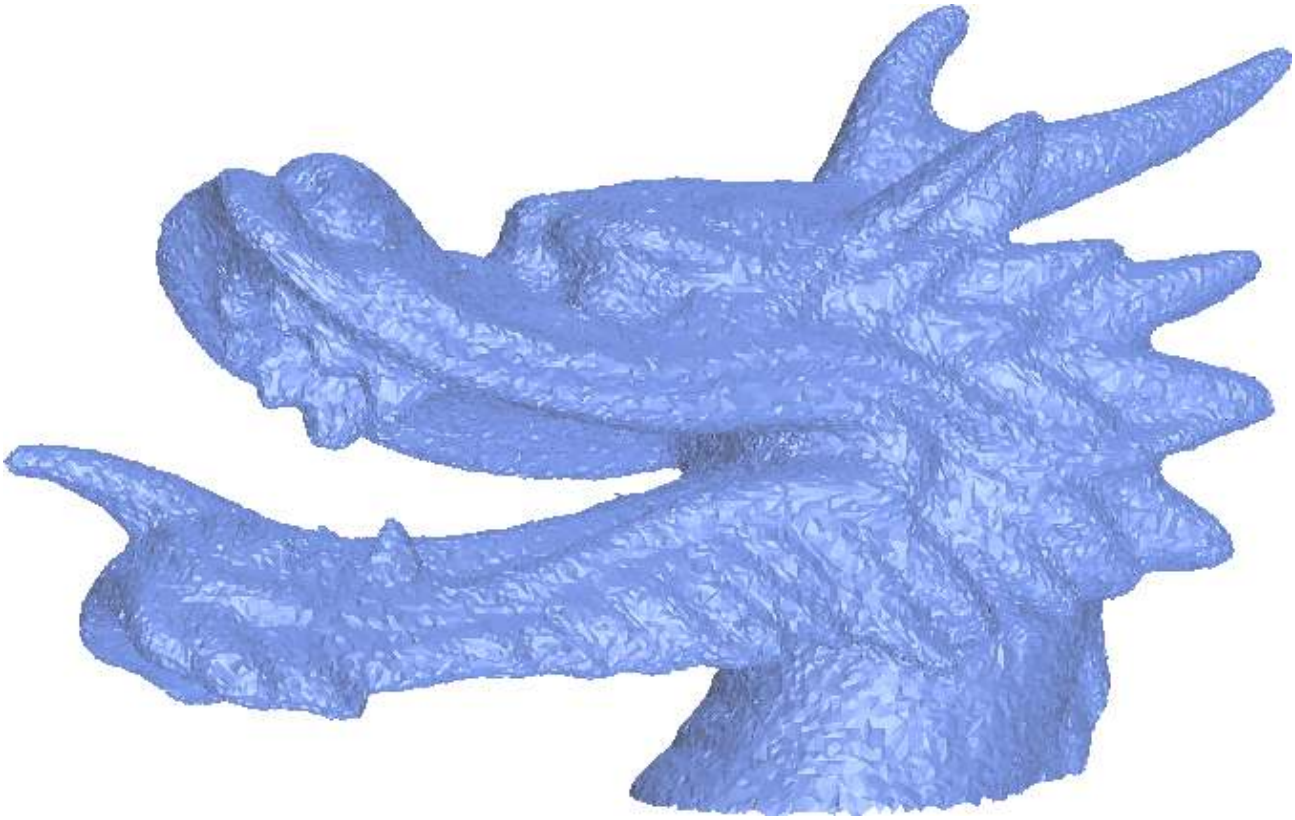
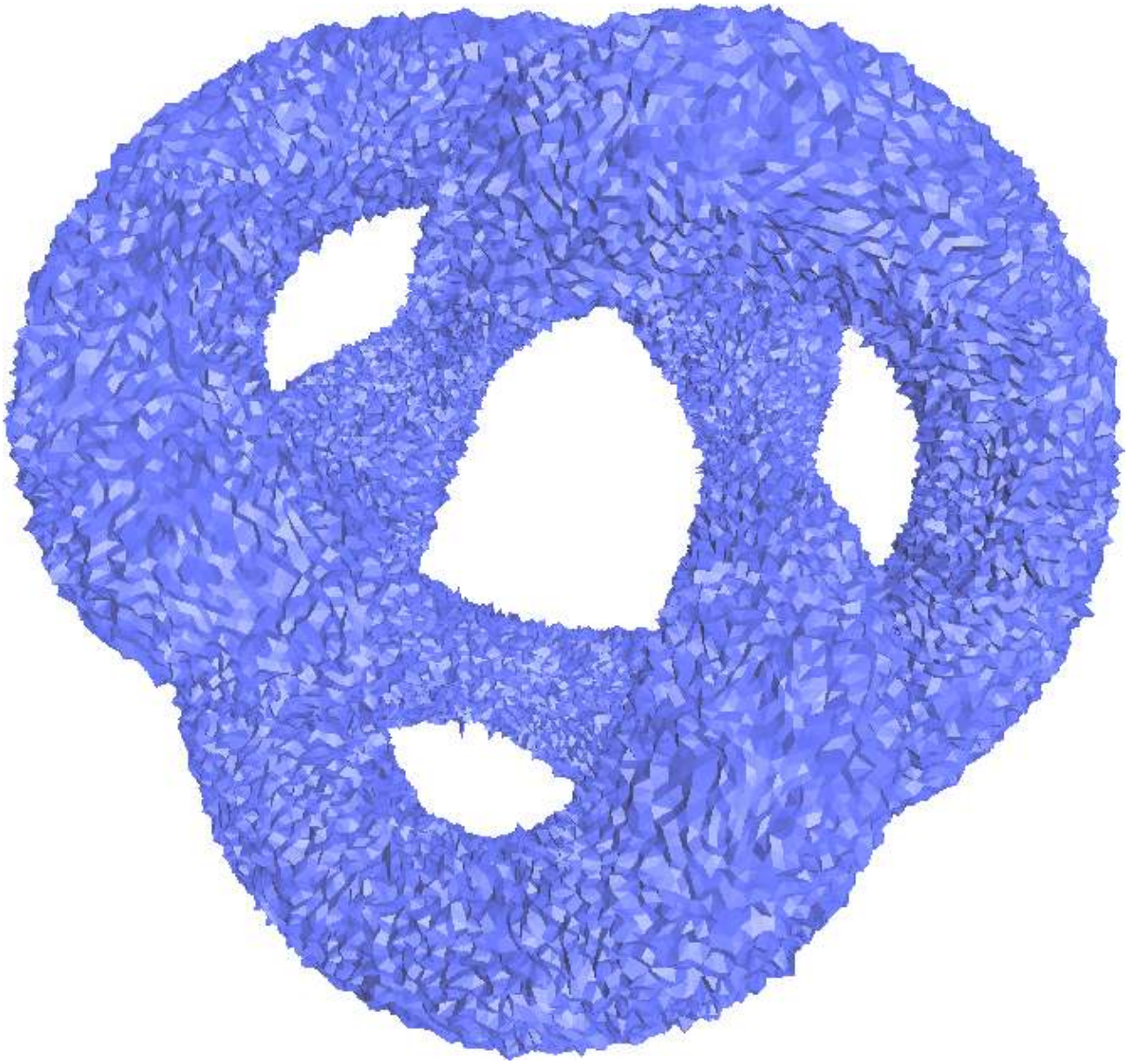
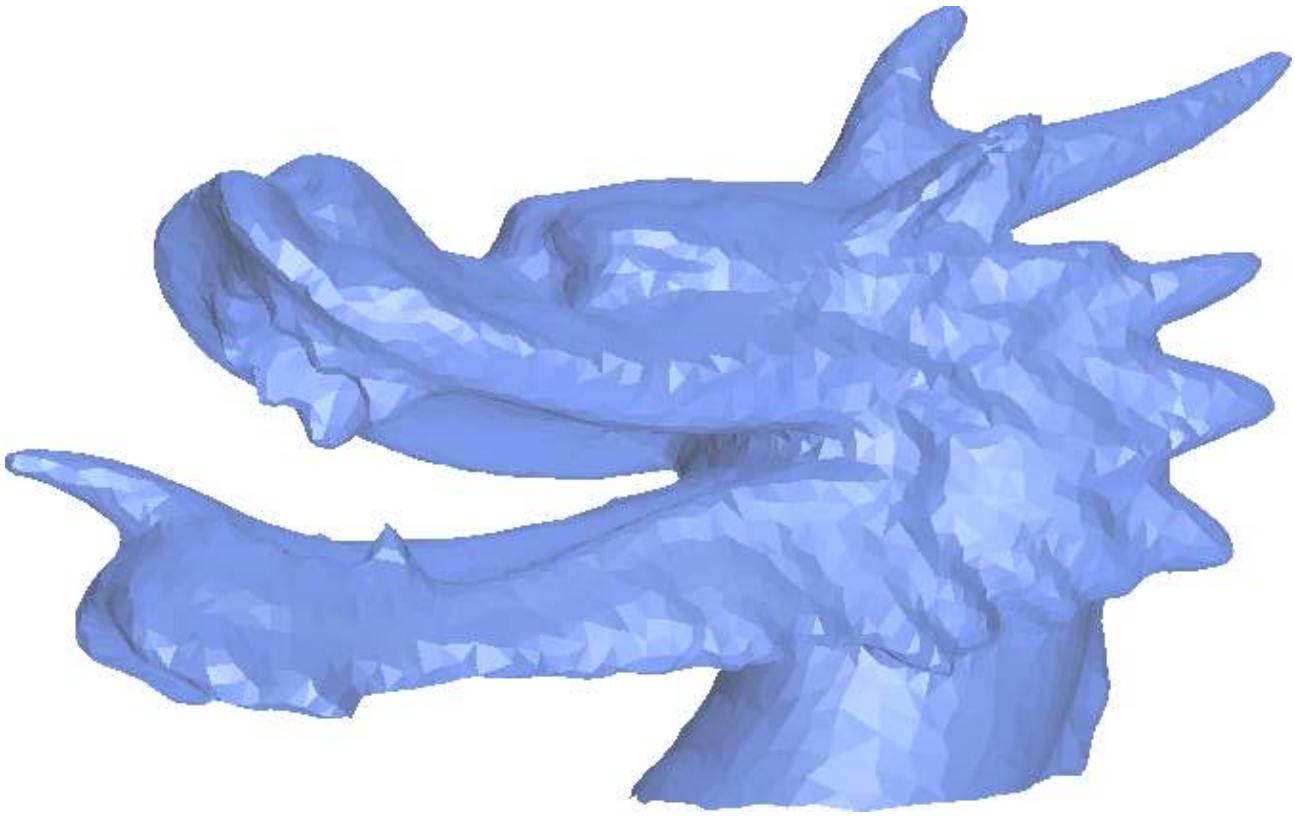
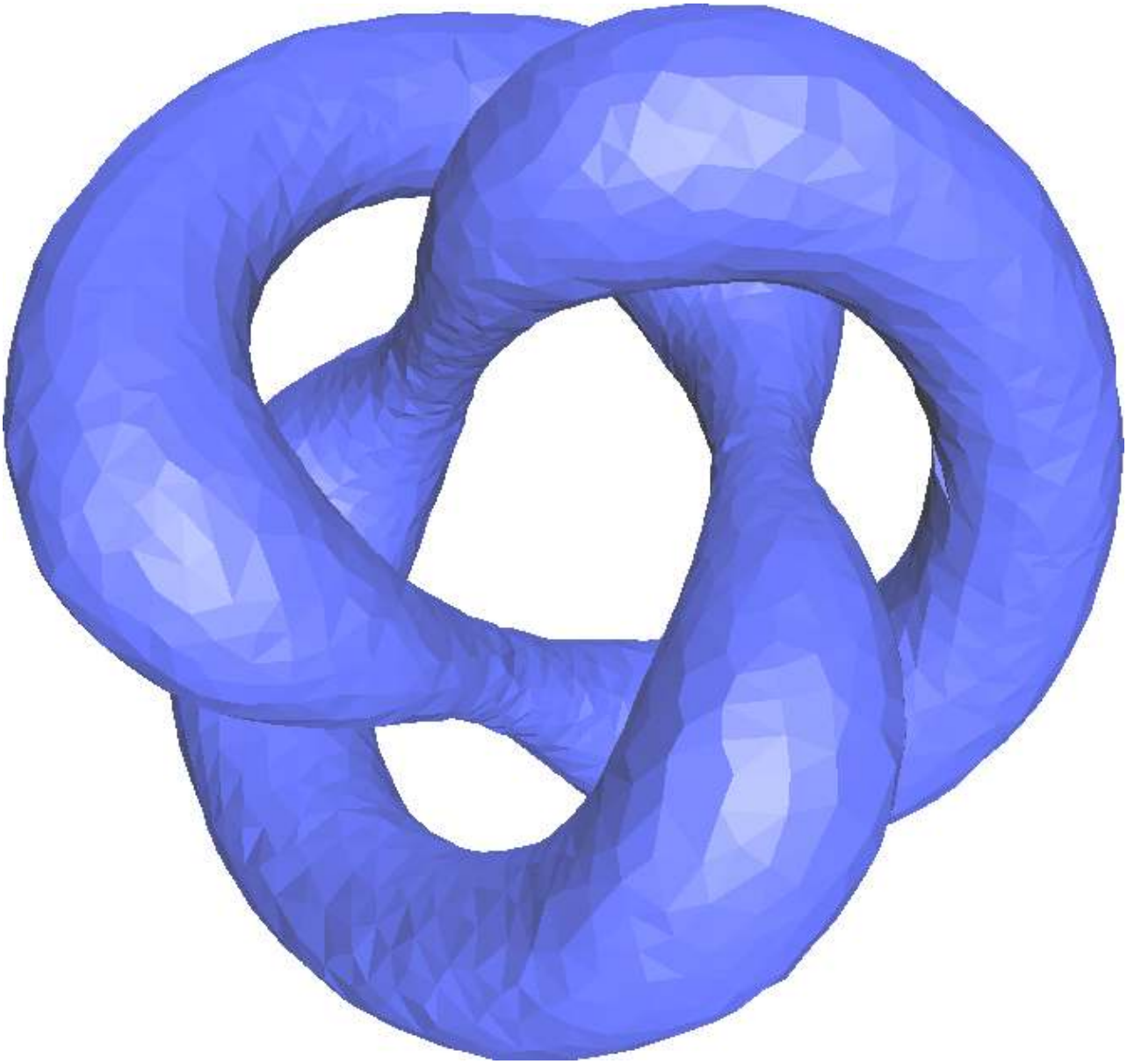


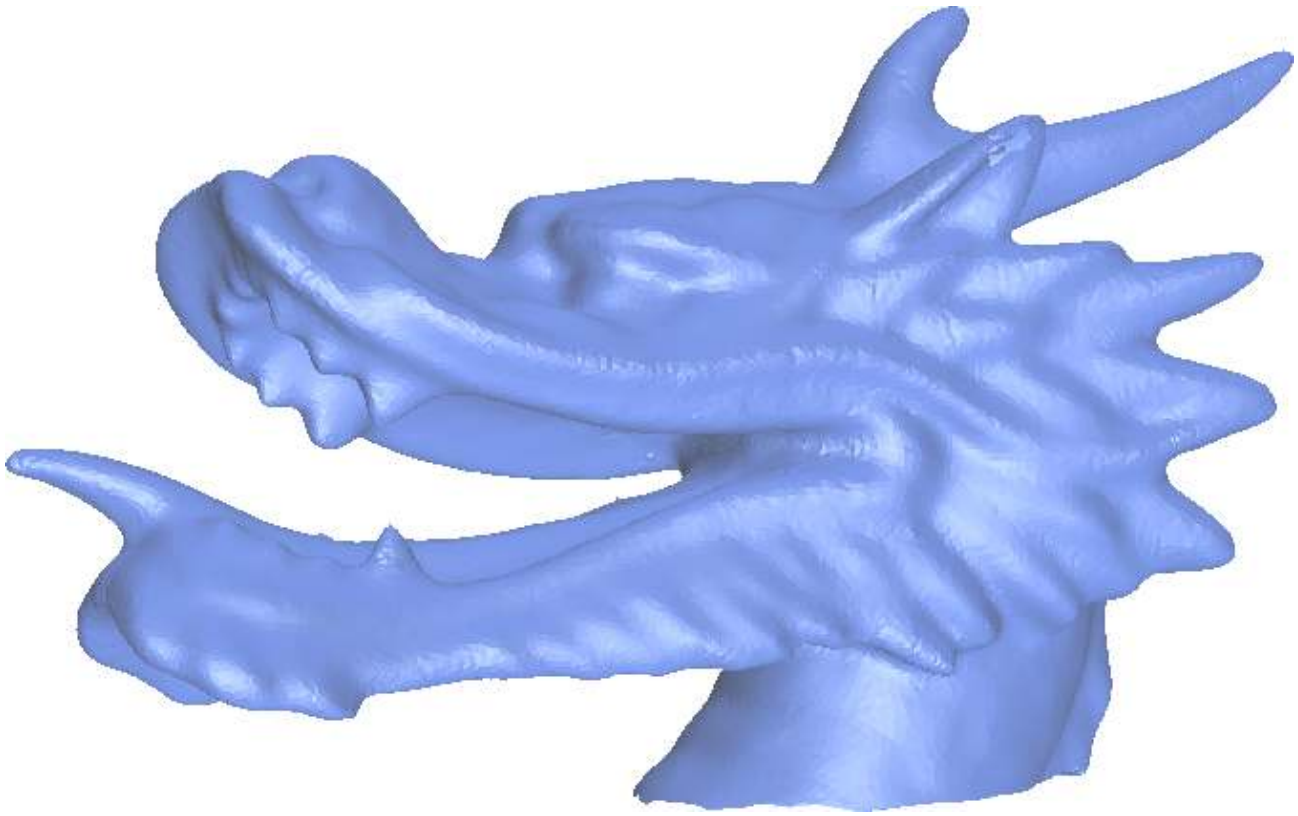
Figure 7: Smooth reconstruction of Igea: noisy data (left), initial triangulation (middle), and final model (right).











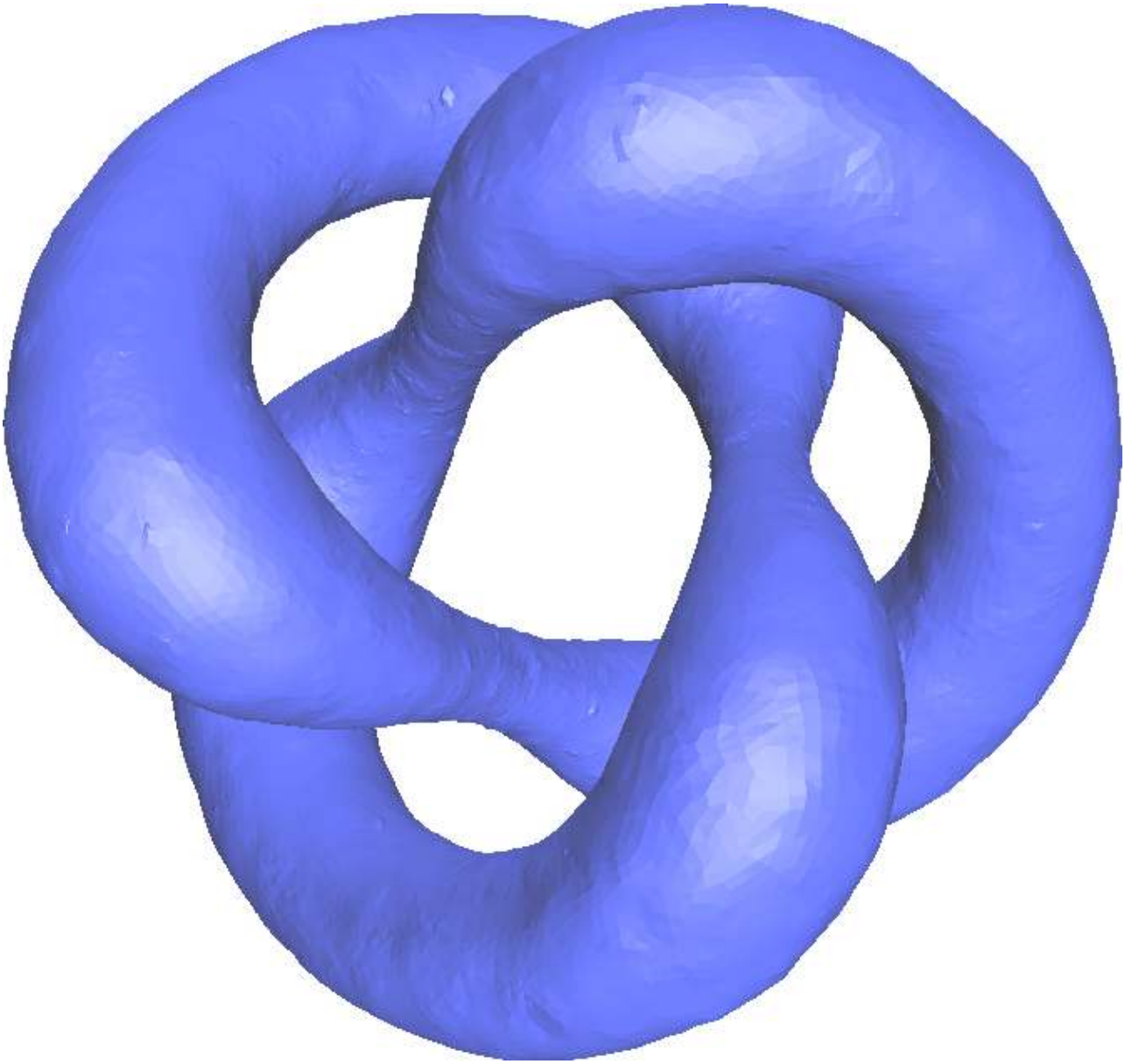


Figure 8: Smooth reconstruction of Dragon and Knot: noisy data (top), initial triangulation (middle), and final model (bottom).