EUROGRAPHICS 2000 / M. Gross and F.R.A. Hopgood
(Guest Editors)

Volume 19 (2000), Number 3

Surface Reconstruction based on Lower Dimensional Localized Delaunay Triangulation

M. Gopi† S. Krishnan C.T. Silva

AT&T Labs - Research
180 Park Avenue
Florham Park, NJ 07932

Abstract

We present a fast, memory efficient algorithm that generates a manifold triangular mesh S passing through a set of unorganized points $P \subset \mathcal{R}^3$. Nothing is assumed about the geometry, topology or presence of boundaries in the data set except that P is sampled from a real manifold surface. The speed of our algorithm is derived from a projection-based approach we use to determine the incident faces on a point. We define our sampling criteria to sample the surface and guarantee a topologically correct mesh after surface reconstruction for such a sampled surface. We also present a new algorithm to find the normal at a vertex, when the surface is sampled according our given criteria. We also present results of our surface reconstruction using our algorithm on unorganized point clouds of various models.

1. Introduction

The problem of surface reconstruction from unorganized point clouds has been, and continues to be, an important topic of research. The problem can be loosely stated as follows: *Given a set of points P which are sampled from a surface in $\mathcal{R}^3$, construct a surface S so that the points of P lie on S*. A variation of this *interpolatory* definition is when *S approximates* the set of points *P*.

There are a wide range of applications for which surface reconstruction is important. For example, scanning complex 3D shapes like objects, rooms and landscapes with tactile, optical or ultrasonic sensors are a rich source of data for a number of analysis and exploratory problems. Surface representations are a natural choice because of their applicability in rendering applications and surface-based visualizations (like information-coded textures on surfaces). The challenge for surface reconstruction algorithms is to find methods which cover a wide variety of shapes. We briefly discuss some of the issues involved in surface reconstruction.

We assume in this paper that the inputs to the surface reconstruction algorithm are sampled from an actual surface (or groups of surfaces). A proper reconstruction of these surfaces is possible only if they are "sufficiently" sampled. However, sufficiency conditions like sampling theorems are fairly difficult to formulate and as a result, most of the existing reconstruction algorithms ignore this aspect of the problem. Exceptions include the works of [3, 5, 2].

If the surface is improperly sampled, the reconstruction algorithm can produce artifacts. A common artifact is the presence of spurious surface boundaries in the model. Manual intervention or additional information about the sampled surface (for instance, that the surface is manifold without boundaries) are possible ways to eliminate these artifacts. The other extreme in the sampling problem is that the surface is sampled unnecessarily dense. This case occurs when a uniformly sampled model with a few fine details can cause too many data points in areas of low curvature variation.

The choice of underlying mathematical and data structural representation of the derived surface is also important. The most common choice are triangular or polygonal mesh representations. Triangular meshes also allow us to express the topological properties of the surface, and it is the most popu-

† graduate student at Department of Computer Science, University of North Carolina at Chapel Hill, USA

lar model representation for visualization and rendering applications.

Currently, most of the surface reconstruction algorithms that guarantee a "good" quality triangulation and are theoretically sound typically produce higher dimensional simplicials like tetrahedra. A second stage of these algorithms remove interior facets to produce the final triangulation. Therefore, these algorithms usually take on the order of a few minutes to run on data sets of moderate sizes (about 20000 to 30000 points) and their applicability to very large data sets (order of millions of points) is not very clear. In this paper, we present an algorithm which guarantees a correct reconstruction (and a good quality triangulation) under some assumptions about the underlying object and runs more than an order of magnitude faster than the above mentioned algorithms.

1.1. Main Contributions

In this paper, we present a fast and efficient algorithm for surface reconstruction from unorganized point clouds based on localized two-dimensional Delaunay triangulation. Our algorithm incrementally develops an interpolatory surface using the surface oriented properties of the given data points. The main contributions of this paper include:

- **Sampling criteria:** We present a new local sampling criteria for the problem of surface reconstruction. The criteria is based on directional curvatures on the surface that is sampled. Based on this criteria and some well placed assumptions about the underlying surface, our algorithm produces the correct reconstruction.
- **Fast surface reconstruction algorithm:** Our algorithm is based on the *advancing front* techniques for surface reconstruction. Each iteration of our algorithm advances the reconstructed surface boundary by choosing one point on it and computing all the faces incident on it.
- **Normal estimation algorithm:** We also present a new and simple algorithm for robust estimation of normals for the points in an unorganized point set. It is very similar in approach to the method of Hoppe et. al. [16], but we believe that the formulation is different.
- **Fast Delaunay neighborhood computation:** We have developed a fast and simple algorithm to compute the Delaunay neighborhood on a plane around a point, given an angle ordered set of possible candidate Delaunay neighbors. This is supported by a fast algorithm for ordering of a set of points by angle around a reference point.
- **System implementation:** We have developed a system based on the above results and have applied it to a number of models of varying sizes. The empirical performance of our system is very encouraging and can generate surfaces from point clouds of sizes around 100,000 points in few tens of seconds.

2. Previous Work

The problem of surface reconstruction has received significant attention from researchers in computational geometry and computer graphics. In this section, we give a brief survey of existing reconstruction algorithms. We use a classification scheme by Mencl et. al. [21] to categorize the various methods. The main classes of reconstruction algorithms are based on *spatial subdivision*, *distance functions*, *surface warping* and *incremental surface growing*.

The common theme in spatial subdivision techniques is that a bounding volume around the input data set is subdivided into disjoint cells. The goal of these algorithms is to find cells related to the shape of the point set. The cell selection scheme can be surface-based or volume-based.

The surface-based scheme proceeds by decomposing the space into cells, finding the cells that are traversed by the surface and finding the surface from the selected cells. The approaches of [16, 11, 4, 3] fall under this category. The differences in their methods lie in the cell selection strategy. Hoppe et. al. [16, 17] use a signed distance function of the surface from any point to determine the selected cells. Bajaj et. al [4] construct an approximate surface using α-solids to determine the signed distance function. Edelsbrunner and Mucke [22, 11] introduce the notion of α-shapes, a parameterized construction that associates a polyhedral shape with a set of points. The choice of α has to be determined experimentally. More recently, Guo et. al. [13] use visibility algorithms and Teichmann et. al. [27] use density scaling and anisotropic shaping to improve the results of reconstruction using α-shapes. For the two-dimensional case, Attali [3] introduces *normalized meshes* to give bounds on the sampling density within which the topology of the original curve is preserved.

The volume-based scheme decomposes the space into cells, removes those cells that are not in the volume bounded by the sampled surface and creates the surface from the selected cells. Most algorithms in this category are based on Delaunay triangulation of the input points. The earliest of these approaches is Boissonat's [8] "Delaunay sculpting" algorithm that successively removes tetrahedra based on their circumspheres. Veltkamp [29] uses a parameter called γ-indicator to determine the sequence of tetrahedra to be removed. The advantage of this algorithm is that the γ-indicator value adapts to variable point density. However, both the approaches of Boissonat and Veltkamp cannot handle objects with holes and surface boundaries. Amenta et. al. [2, 1] use a Voronoi filtering approach based on three-dimensional Voronoi diagram and Delaunay triangulation to construct the *crust* of the sample points. They provide theoretical guarantees on the topology of their reconstructed mesh given "good" sampling.

The distance function of a surface gives the shortest distance from any point to the surface. The surface passes through the zeroes of this distance function. This approach leads to approximating instead of interpolatory surfaces

[16, 7, 10]. Hoppe et. al. [16] use a Reimannian graph to compute consistent normal throughout the surface to determine the signed distance function. The approach of Curless and Levoy [10] is fine-tuned for laser range data. Their algorithm is well suited for handling very large data sets.

Warping-based reconstruction methods deform an initial surface to give a good approximation of the input point set. This method is particularly suited if a rough approximation of the desired shape is already known. Terzopoulos et. al. [28] use *deformable superquadrics* to fit the input data points. A different approach to warping was suggested by Szeliski et. al. [25] with *oriented particles*. By modeling the interaction between the particles, they construct the surface using forces and repulsion.

The basic idea behind incremental surface construction is to build-up the surface using surface-oriented properties of the input data points. The approach of Mencl and Muller [19, 20] is to start with a global wireframe of the surface generated using Euclidean minimum spanning tree construction, and to fill it iteratively to complete the surface. Boissonnat's surface contouring algorithm [8] starts with an edge and iteratively attaches further triangles at boundary edges of the emerging surface using a projection-based approach. This algorithm is similar in vein to our approach. A crucial difference between our methods is that Boissonnat's algorithm is edge-based, while ours is vertex-based. We also provide guarantees on quality triangulation. Further, his algorithm can only generate manifolds without boundaries.

The Spiraling-Edge triangulation technique proposed by Crossno and Angel [9] is also related to ours. Major differences include the fact that they make several limiting assumptions about the data, including normal information for each point, and also an estimate of each point's neighbors. Their algorithm works by creating a star-shaped triangulation between a point and its neighbors. But the paper provides no theoretical foundation for the actual triangulation computed, including no estimates for the sampling necessary to produce correct triangulations.

Another recent advancing-front triangulation scheme is the Ball-Pivoting Algorithm (BPA) of Bernardini et al [6]. Given a point cloud and a radius ρ, BPA finds an interpolatory surface where each of its triangles are characterized by the fact that the ball of radius ρ that sits on its vertices has no internal point (i.e., it is an ρ-exposed triangle). The algorithm works by finding a "seed" ρ-exposed triangle, then extending the surface as far as it can by "pivoting" a ball of radius ρ along each boundary edge of the current surface (which is continuously updated). Under some sampling conditions, BPA is guaranteed to finish with a correct triangulation. One shortcoming of BPA is the fact that it does not allow for reconstructing surfaces out of variable-sampled points without multiple passes and they assume that the normal information is available for the input point set.

3. Algorithm Overview

Our surface reconstruction algorithm takes a set of unorganized 3D points S as input with no other additional information like normals. The output of the algorithm is a set of triangles, which defines a manifold surface with or without boundary, passing through the input set of points. This algorithm uses a progressive triangulation technique where the triangulation incrementally progresses over the surface. The neighbors of a vertex in the final triangulation is computed on its tangent plane. Hence it is a local triangulation technique.

Our surface reconstruction algorithm goes through four major steps: normal computation, candidate point selection, Delaunay neighbor computation, and finally the triangulation step. This section gives a brief description of all the above steps.

Normal Computation: The first step in our algorithm is to compute the normal at all sample points. This step is performed only if we the normal information is not part of the input. This step also consistently orients the normals of the sample points to get an orientable manifold.

Candidate points selection: This step chooses those points which might be possible neighbors to a vertex in the final triangulation. Using our sampling criteria described in Section 4, we compute this candidate point set (P_p) for every sample point p.

Delaunay Neighbor Computation: We map each of the candidate points in the set P_p on the tangent plane at p by a simple rotation about a well defined axis on the tangent plane. The set of these mapped candidate points are referred as P_p^T. Then we compute the local Delaunay neighborhood from the set P_p^T around p in its tangent plane. This computation is repeated for all the points in S, and the final surface triangulation is determined from this neighborhood relationship.

The candidate point set P_p plays a crucial role in determining the final triangulation. In order to obtain the correct surface, we must impose a certain sampling criteria. In the next section we formulate this criteria mathematically. We also justify the above algorithm using the sampling criteria.

4. Sampling Criteria

In this section, we present a sampling criteria to guarantee a triangulation homeomorphic to the surface $\mathbf{F}$. The sampling density at a point along a particular direction (in the tangent plane) is *inversely proportional* to the *directional curvature* at that point. The geometric intuition behind this criteria is that the positioning of the normals of the set of point samples on the Gaussian sphere is uniform provided the product of the directional curvature and the arc length on the surface is constant (see [24] for an explanation of this fact). In the

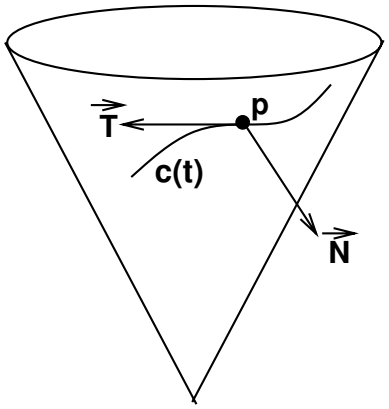


Figure 1: *The Darboux Frame*

rest of this section, we shall quantitatively explain what the above criteria means using concepts from differential geometry. We shall start by giving a few definitions and notations which will be used by us. More detailed explanations can be found in the appendix.

- Consider a point p on a surface $\mathbf{F}$ as shown in Figure 1. Let the normal vector to $\mathbf{F}$ at p be $\vec{N}$. Given a unit vector $\vec{v}$ on the tangent plane at p, define a curve $c(t) : [-\varepsilon, \varepsilon] \rightarrow \mathbf{F}$ such that $c(0) = p$, $c'(0) = v$. The **Darboux frame** at p is defined as the orthonormal differential frame $\vec{T} = \vec{v}, \vec{B} = \vec{N} \times \vec{T}, \vec{N}$. Figure 1 shows the Darboux frame on the surface of a cone. It is easy to see that for surfaces with a well defined tangent plane everywhere, every point on the surface has a unique Darboux frame associated with it in a given direction v in the tangent plane.
- **Surface curvature:** Associating a local differential frame at every point on the surface allows us to measure some geometric invariants on the surface. If we walk infinitesimally along a direction $\vec{v}$, the change of the surface normal in the direction $\vec{v}$ is called the *normal curvature*. As we move along different directions in the tangent plane, the *normal curvature* varies. The directions with minimum and maximum normal curvatures are called principal directions and the corresponding curvatures in these directions are called principle curvatures. These principal directions are orthogonal to each other. In the rest of this paper, we will refer to the principal curvatures as k_1 and k_2 (or k_{min} and k_{max}).
- **Local surface as a height function:** We make use of the well known implicit function theorem [23] to express the surface in the neighborhood of a point as a height function in terms of the principal curvatures. We will represent the height function in the local neighborhood as

$$h(x,y) = \frac{1}{2}(k_1 x^2 + k_2 y^2) + \text{higher order terms}$$
$$h(r,\theta) \approx \frac{r^2}{2}(k_1 \cos^2\theta + k_2 \sin^2\theta),$$

where $r = \sqrt{x^2+y^2}$ and θ is the angle the vector (x,y) makes with the x-axis. The derivation of the above expression is given in detail in the appendix.
- The *Euler equation* relates the normal curvature, k_v, at some point p on the surface along a direction $\vec{v}$ in the tangent plane to the principal curvatures, k_1 and k_2. Let the principal directions at p be $\vec{v}_1$ and $\vec{v}_2$. Then

$$k_v = k_1 \cos^2\theta + k_2 \sin^2\theta,$$

where θ represents the angle $\vec{v}$ makes with $\vec{v}_1$.
Using this result on the expression for $h(r,\theta)$ above, we get

$$h(r,\theta) = \frac{k_v r^2}{2} \tag{1}$$

- It is possible to describe the behavior of the normal vector along space curves on a surface $\mathbf{F}$ passing through some point p. The equation below can be derived from the Cartan's equations for differential frames [18, 23].

$$\partial\vec{N} = -k_v \partial s \vec{T} - t\partial s \vec{B}, \text{or} \tag{2}$$
$$|\partial\vec{N}| = \sqrt{k_v^2 + t^2}\ |\partial s| \tag{3}$$

Here k_v is the normal curvature and t (also known as *geodesic torsion*) intuitively measures the twist along the $\vec{N} - \vec{B}$ plane.
The quantity $\sqrt{k_v^2 + t^2}$ is called the *total curvature* of the space curve through p. We simplify the above equation for the special case of planar curves through p for which t is always zero. Then the total curvature becomes k_v, the normal curvature. Our sampling criterion now can be formulated mathematically as

$$k_v \partial s = constant \tag{4}$$

Intuitively, this sampling implies that the dot product between the normals at a given point on the surface and the nearby point samples is constant. Higher the curvature in a particular direction, closer the point samples should be and vice-versa.
- We call a point p on a surface $\mathbf{F}$ a *regular* point if the surface in the neighborhood of p is homeomorphic to an open disk. We replace the constant term in equation 4 by δ and the arc length ∂s by the edge length. Given a *regular* point p on $\mathbf{F}$ with unit normal N_p, let C_p^δ be the (closed) contour on $\mathbf{F}$ around p such that

$$k_v |pq| = \delta, \forall q \in C_p^\delta, v = (\vec{pq} - (\vec{pq} \cdot N_p) N_p),$$

where $|pq|$ denote the Euclidean arc length along the

surface. It is clear that C_p^δ partitions surface **F** into two (or more) parts: one that contains p and others that do not. We define the former partition as the *immediate* δ-*neighborhood* (B_p^δ) of the point p.

- We call a point set S a δ-*sampling* of a surface **F** if every point $p \in \mathbf{F}$ has a closest point q in the sample set S such that $q \in C_p^\delta$. In the above definition, δ is a parameter that can be changed to obtain different samples of the surface.

The definition of a δ-sampling is clearly a local sampling criterion and makes an intrinsic assumption about the tubular neighborhood of the underlying object. This has the disadvantage of not being sensitive to global features of the object like two layers of the same object coming very close to each other. We make some assumptions on the tubular neighborhood of the object. Given a point $q \in \mathbf{F}$ not in the *immediate* δ-*neighborhood* of p, we place a bound (function of δ and principal curvatures) on the dot product of the vector $\vec{pq}$ with the unit normal at p. For the rest of this discussion, we will assume that

- The surface under consideration is smooth and that the ratio of the maximum to minimum principal curvature at any point is bounded above by the constant ρ.
- The arc pq_v along the surface can be replaced by the edge pq_v. This assumption is reasonable for small δ.

In this section, we shall state without proof a couple of properties about δ-sampled surfaces. The proofs are given in the appendix.

Lemma 1 Given a δ-sample S of a surface **F** and two points $p, q \in S$ such that $q \in C_p^\delta$. Then $p \in C_q^\delta$.

The above lemma will later be used to claim the symmetry in the choice of neighborhood around a point. We now proceed to show that ratio of the distances between any two points $q, r \in C_p^\delta$ to p is bounded.

Let $p \in \mathbf{F}$. Define the contour C_p^δ around p as before.

Theorem 1 (a) Consider any two points $q, r \in C_p^\delta$. Then the maximum ratio of edge distances $\frac{|pq|}{|pr|}$ is bounded above by a function of principal curvatures.
(b) Let N_p be the unit normal to **F** at p and let $\hat{pq}$ denote the unit vector from p to q. Define the height function $H(p,q) = |N_p \cdot \vec{pq}|$. Then $H(p,q)$ is bounded above by the quantity $\frac{\sqrt{1+\delta^2}-1}{k_{min}}$. k_{min} is the smaller of the principal curvatures.
(c) Define the angle function $D(p,q) = |N_p \cdot \hat{pq}|$. Then $D(p,q)$ is a constant, $\frac{\sqrt{1+\delta^2}-1}{\sqrt{1+\delta^2}+1}$, for all $q \in C_p^\delta$.

The result about the height function can be used to precisely quantify how close two different parts of the model can come so that a δ sampling is sufficient for the reconstruction algorithm. Since the maximum height value for any point in C_p^δ is bounded by $\frac{\sqrt{1+\delta^2}-1}{k_{min}}$, we bound the distance between two different layers of the model to be greater than twice this value. Observe that this condition is similar to putting a bound on the distance of any point from its closest medial axis feature [2, 1]. On the converse side, if we know that two different layers of an object are within distance d of each other, then we can impose a δ-sampling on the surface such that $\delta \leq \frac{\sqrt{d^2 k_{min}^2 + 4dk_{min}}}{2}$.

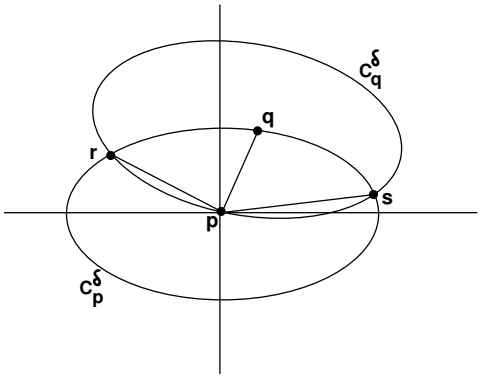


Figure 2: *Angle between adjacent samples on* δ*-sampled surface*

The above results show that the points in the neighborhood of a point p on a δ-sampled surface satisfy strict bounds on the ratio of the distances and the angles from the normal. Further, the assumption about different layers not coming too close together justifies our local sampling criterion. We shall now argue that instead of computing the three dimensional Voronoi diagram of the sample points, it is sufficient to compute the local two-dimensional Voronoi cell of each sample point in its local tangent plane.

We will first try to bound the maximum angle deviation between adjacent sample points on the contour C_p^δ. Intuitively, given a δ-sampled surface, adjacent points on the contour cannot lie arbitrarily far away since the sampling will not be preserved. For the smooth curves which define C_p^δ, the angle bound is 90°. Figure 2 illustrates one such case. Consider the point q in C_p^δ. The contour C_q^δ can be thought of as a similar curve with a small rotation of the principal directions at p because of proximity to p and the fact that we are associating a differential reference frame at every point on the smooth surface. Adjacent point samples on C_p^δ are constrained by the fact that they have to lie in the intersection points of C_p^δ and C_q^δ. Let the intersection points be r and s. It is easy to see that the angles $\angle rpq$ and $\angle qps$ are less than the corresponding angles if the points r, q and s are flattened into the tangent plane at p. The extreme values of angles occur when q is along one of the principal directions of p. In either of these cases, the sum of the angles $\angle rpq + \angle qps < 180^\circ$ with symmetric positioning of r and s. This shows that the maximum deviation between adjacent samples is less than 90°.

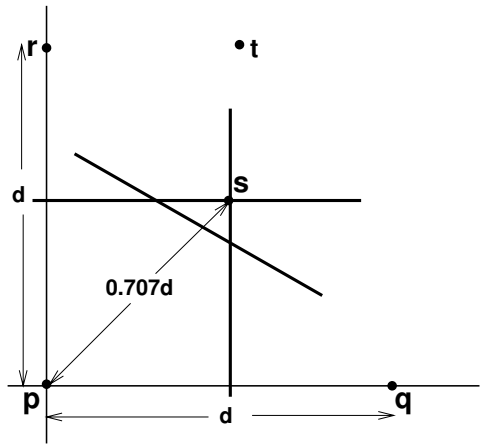


Figure 3: *Voronoi sites in the plane*

Theorem 1(c) showed that the expression for the angle function $D(p,q)$ (cosine of the actual angle) is constant for all points on C_p^δ whose value is small for small values of δ. Consider the plane Π_q formed by the normal vector N_p and the vector $\vec{pq}$. This plane intersects the tangent plane at p along a particular direction $\vec{v}$. The angle between $\vec{v}$ and $\vec{pq}$ which is just $|\pi/2 - \cos^{-1} D(p,q)|$ is thus also very small. This implies that we can map the points on the sampled contour C_p^δ to points on the tangent plane by a rotation of $\vec{pq}$ in their corresponding plane Π_q without affecting the neighborhood around p and little change in the relative distances between adjacent sample points.

The above result suggests a simple scheme for completing the triangulation around the point p. Take the sample points in the δ-neighborhood of p. Map them onto the tangent plane at p using the scheme above. Compute the two-dimensional Voronoi cell of p with the neighborhood points. Its dual determines the triangulation around p. Unfortunately, the above scheme has a small problem. Consider the situation in Figure 3. Let q and r be two adjacent samples on the contour C_p^δ 90° apart at a distance d from p. Considering p, q and r in isolation produces a Voronoi vertex s at a distance $d/\sqrt{2}$ away. It is now possible that some other sample point t which is outside C_p^δ but within distance $l = \sqrt{2}d$ from p can alter the Voronoi cell at p. However, points which are further than l from p cannot affect its Voronoi cell. Therefore, we modify our earlier triangulation scheme by considering sample points in the 2δ-neighborhood of p.

There is an interesting connection between the *restricted Voronoi diagram* in Amenta et. al. [1] and our Voronoi cell computation. In their paper, [1] define the *restricted Voronoi diagram* as the cell decomposition induced on the surface **F**: the boundaries of the cells on **F** are simply the intersections of **F** with the three-dimensional Voronoi cell boundaries. They also define a *good triangle* with vertices from the sample set S if it is dual to a vertex of the *restricted Voronoi diagram* and go on to show that for a good enough sampling of the surface, the *good triangles* form a polyhedron homeomorphic to **F**. Figure 4 shows the similarity between the *restricted Voronoi diagram* computation of [1] and the three-dimensional lifting of our 2D Voronoi cell computation on the local tangent plane.

5. Algorithm

In this section, we will describe in more detail, the steps we briefed in Section 3.

5.1. Computation of Vertex Normal

The first step in our algorithm is to find the normal, and thus the tangent plane of the surface, at every sample point p. This is computed using the closest neighbor information. We choose $k-$*nearest* neighbors of p for this purpose. We need to find a vector which is a good representative of the normal to that surface at p. We propose that the normal vector $\vec{n_p}$ is the vector that minimizes the variance of the dot product between itself and the vectors from p to its $k-$*nearest* neighbors. If the $k-$*nearest* neighbors are q_1 to q_k, then the vectors from p to its $k-$*nearest* neighbors are $\vec{V_i} = q_i - p$, $1 \le i \le k$. We want to find $\vec{n_p}$ such that it minimizes

$$\frac{\sum_{i=1}^{k} (D_i - \frac{\sum_{i=1}^{k} D_i}{k})^2}{k} \tag{5}$$

where $D_i = \vec{n_p} \cdot \vec{V_i}$.

The vectors $\vec{V_i}$ can be viewed as the coordinates of the k-*nearest* neighbors with p as the origin. Removing the scale factor $\frac{1}{k}$ from the above equation, we get

$$\min(\sum_{i=1}^{k} (\vec{n_p} \cdot \vec{V_i} - \frac{\sum_{i=1}^{k} \vec{n_p} \cdot \vec{V_i}}{k})^2), \text{ or} \tag{6}$$

$$\min(\sum_{i=1}^{k} ((\vec{V_i} - \frac{\sum_{i=1}^{k} \vec{V_i}}{k}) \cdot \vec{n_p})^2) \tag{7}$$

If p is at the origin, the centroid of k nearest candidate points is $C = \frac{\sum_{i=1}^{k} \vec{V_i}}{k}$. Thus, the above equation can be rewritten as,

$$\min(\sum_{i=1}^{k} ((\vec{V_i} - C) \cdot \vec{n_p})^2) \tag{8}$$

If A is a $k \times 3$ matrix where $\vec{V_i} - C$ defines the row vectors, then the above expression reduces to

$$\min(\|A\vec{n_p}\|_2) \tag{9}$$

This minimization problem can be posed as a standard singular value decomposition problem [12]. The eigenvector

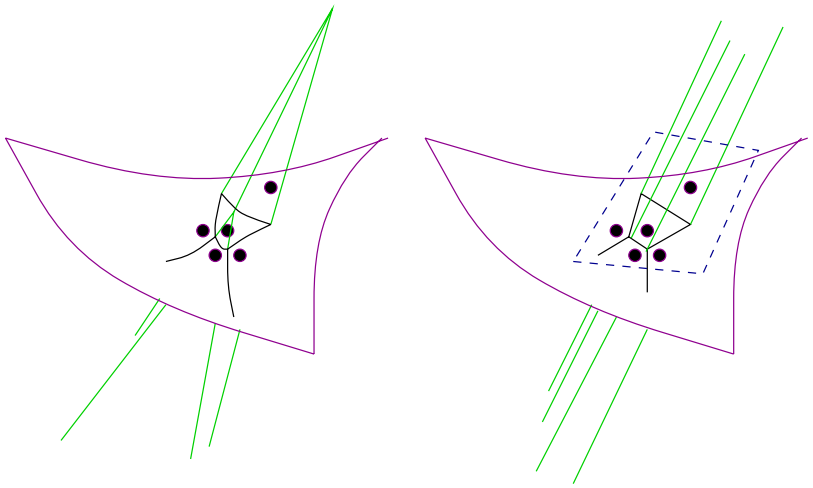


Figure 4: *Left: Restricted Voronoi diagram of [AB98], Right: Three-dimensional lifting of our 2D Voronoi cell computation*

which corresponds to the smallest eigenvalue of A is the normal vector which minimizes the above equation. Hoppe et. al. [16] proposed the use of principal component analysis of a covariance matrix to determine the normals. Even though our formulation is very different, it turns out that the resulting normal vectors computed by both methods are the same.

5.1.1. Propagation of Normal Direction

The normal vector found by the above process is correct only upto sign. To find a consistent orientation of the surface, we fix the orientation of one of the normals and propagate this information to rest of the points. We use the technique proposed by Hoppe et. al. [16] to do the propagation. Hoppe et. al. pose this problem as a minimum spanning tree problem, where the vertices of the model are the vertices of the graph, and the edges of the model are the edges in the graph. The weight of the edge between the vertex i and j is assigned to be $(1 - |\vec{n}_i \cdot \vec{n}_j|)$, where $\vec{n}_i$ and $\vec{n}_j$ are the normals at the vertices i and j computed using the method given in the previous section. The minimum spanning tree of the thus constructed graph would give the propagation sequence of normals for the consistent orientability of the model. An arbitrary vertex of the graph is assumed to be the root and the normal is propagated to its children recursively. When the normal direction is propagated from vertex i to vertex j, if $n_i \cdot n_j$ is negative, then the direction of n_j is reversed; otherwise it is left unchanged.

5.2. Computation of Principal Curvatures

We use an adaptation of the method described by Taubin [26] to compute the curvature tensor at every point. Taubin's method was described for a surface mesh. Since we have only a discrete set of points, we choose the $k-$*nearest* neighbors of p instead of the explicit neighborhood defined by the mesh. Given p and one of the nearest neighbors q_i, the normal curvature along the direction $\vec{v}_i$ $(= \vec{pq_i} - (\vec{n_p} \cdot \vec{pq_i})\vec{n_p})$ in the tangent plane at p is defined as $k_p^{v_i} \approx \frac{2(\vec{n_p} - \vec{n_{q_i}}) \cdot \vec{pq_i}}{\|\vec{pq_i}\|^2}$.

Taubin [26] shows that the principal curvatures and directions correspond to linear combinations of the (non-zero) eigenvalues and eigenvectors of the rank deficient 3×3 matrix $M_p = \frac{1}{2\pi} \int_{-\pi}^{\pi} k_p^v \vec{v}\vec{v}^T d\theta$, where $\vec{v}$ is represented in terms of the principal directions ($\vec{v}_1^p$ and $\vec{v}_2^p$) as $\vec{v} = \cos\theta \vec{v}_1^p + \sin\theta \vec{v}_2^p$, where θ is the angle $\vec{v}$ makes with $\vec{v}_1^p$.

The integral is discretized into a weighted sum of the $k-$*nearest* neighbors normal curvatures. We first compute the angles that $\vec{v}_i$ makes with an arbitrary frame in the tangent plane. Then we sort the angles in counterclockwise order $(\{\alpha_1, \alpha_2, \ldots, \alpha_k\})$. The discrete form is

$$M_p = \Sigma_{i=1}^{k} w_p^i k_p^{v_i} \vec{v}\vec{v}^T,$$

where $w_p^i = (\frac{\alpha_{i+1} - \alpha_{i-1}}{4\pi})$.

5.3. Candidate point selection

To complete the triangulation around a point p, we consider all the points in its proximity determined by the sampling criteria. Given a δ-*sampling* S of a surface F, we consider all the sample points in the 2δ neighborhood of $p \in F$. By theorem 1, we know that the maximum ratio of distances of two points in the contour C_p^{δ} is the ratio of the principal curvatures at the point p. When we consider all points inside the 2δ-neighborhood, this ratio simply scales up by a factor of two. This gives us a value of $m_p = \frac{2k_{max}}{k_{min}}$. This constant m_p is used by our candidate point selection algorithm described here.

This step is similar to clustering algorithms used by other triangulation schemes [16, 14]. We first apply a *distance criterion* to prune down our search for candidate adjacent points in the spatial proximity of p. It is executed in two stages. In the first stage, the simpler L_∞ metric is used to define the proximity around p. Our algorithm takes an axis-aligned box of appropriate dimensions centered at p and returns all the data points inside it. The second stage of pruning uses a Euclidean metric, which further rejects the points that lie outside a *sphere of influence* centered at p. Our data structure for this stage of pruning is a depth pixel array similar to the *dexel* structure proposed in [15]. We maintain a 2D pixel array into which all data points are orthographically projected. The points mapped on to the same pixel are sorted by their depth (z) values. This data structure makes the search for candidate points very easy.

By using this data structure, this search is limited to the pixels around the pixel where p is projected. The size of the bounding box for choosing the candidate points using L_∞ metric, and the radius of the *sphere of influence* used in the second stage of pruning is determined by the value m_p. Assuming that the distance of p from its closest neighbor is s, the bounding box and the sphere should enclose all points, which are at the distance less than $m_p s$ from p.

The set of points obtained by these two pruning stages is further pruned by computing the height values of these points in p's local tangent plane. If the height value is greater than $\frac{\sqrt{1+4\delta^2}-1}{k_{min}}$ (refer to Theorem 1b), we remove those points from consideration because they lie outside the tubular neighborhood of the surface near p. The points that remain are the possible Delaunay neighbors of p.

5.4. Triangulation

The next step is the find the neighbors of each vertex to find the final triangulation of the surface. Once we have a consistent orientation of the normals for all the vertices, we map the candidate points as found in Section 5.3 onto the tangent plane at the reference point p. This mapping is the rotation of the vector V_i, from p to the candidate point, on the plane defined by the normal and V_i, to the tangent plane at p. These projected candidate points on the tangent plane are ordered by angle around p.

5.4.1. Fast Ordering by Angle

The candidate points are transformed to the local coordinate system and are mapped as explained above to the tangent plane at p. These points have 2D coordinates, with an implicit definition of coordinate axes. The projected candidate point set is partitioned on the basis of the quadrant where the points lie on this tangent plane in the local coordinate system. The points within each quadrant are ordered by angle using a simple method as explained below.

The square of the sine of the angle from the implicit $x-$ *axis* can be easily computed without any use of square-root or trignometric functions. We find the square of sine as it is a close-to-linear function. We discretize the range between 0 to $\pi/2$ degrees and store its sine-square values in a look-up table. Any value of sine-square can be looked up in the table to give the angle between 0 to $\pi/2$. The values in-between the discretized values represented in the table are linearly interpolated and the angle is found. We use the sine-square value to order the points within the same quadrant, and we use the angle computed in this section to identify surface boundaries and to fill up holes in the triangulation.

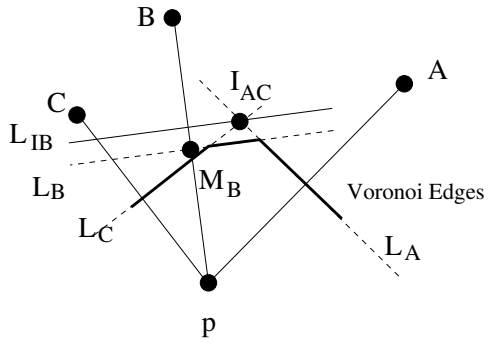


Figure 5: *Finding Delaunay neighbors. Given points A, B and C, check whether B is a Delaunay neighbor of p. The thick edges are local Voronoi edges around p.*

5.4.2. Delaunay Neighbor Computation

This section describes the details of finding the neighbors for the reference point p from the candidate point set. We use 2D local Delaunay triangulation around p to find the neighbors of p. We compute the Delaunay neighbors of p using the following algorithm. Given the ordering of the candidate vertices around p, the goal is to choose the subset which are its Delaunay neighbors. The basic function of the algorithm takes the reference point and three consecutive points in angle order A, B, and C, from the ordered candidate point set to check whether the middle point (B) could be a Delaunay neighbor to p in the presence of A and B. Figure 5 explains this algorithm pictorially.

```
bool CheckDelaunay(p, A, B, C)
{
    L_A = Perpendicular bisector of the line segment pA.
    L_B = Perpendicular bisector of the line segment pB.
    L_C = Perpendicular bisector of the line segment pC.
    I_AC = Intersection point of L_A and L_C.
    L_IB = Line parallel to L_B, and passing through I_AC.
    M_B = Mid point of the line segment pB.
    If both p and M_B lie on the same side of L_IB, then
        B is a local Delaunay neighbor to p
            when compared with A and B.
        return TRUE.
```

Model	No. of points	No. of Triangles	Normal Comp.Time (in secs)	Triangulation Time (in secs)	Total Time (in secs)
Oil Pump	30937	61772	8.18	10.63	20.99
Club	16864	33643	4.30	3.32	7.90
Bunny	34834	69630	9.25	8.31	18.64
Foot	20021	39862	5.23	4.77	10.53
Skidoo	37974	75364	9.73	6.42	16.66
Mannequin	12837	25438	3.37	3.59	7.62
Phone	83034	165981	22.40	19.42	44.41

Table 1: *Performance of our algorithm: See Figure 7* (color plate)*

else
    B is not a Delaunay neighbor to p.
    return *FALSE*.
}

If $q_1 \ldots q_n$ are the ordered projected candidate points of p, then the above function is called for every triplet q_{i-1}, q_i, q_{i+1}. If the test passes, then the next triplet q_i, q_{i+1}, q_{i+2} is tested. But, if the test fails, then q_i is rejected, and the algorithm backtracks with the call $q_{i-2}, q_{i-1}, q_{i+1}$ to re-evaluate q_{i-1} for its validity. As the ordering of the vertices is by angle, and hence is cyclic, any point can be chosen as the first point q_1. We choose the closest point to p as q_1, as it is always a Delaunay neighbor, and further, it is used as a terminating condition for the backtracking algorithm. The implementation of the above algorithm is optimized for speed, and each call to the above function takes less than 35 mathematical operations.

The Delaunay neighbors of each vertex is found and is stored in a list ordered by angle. Vertices A, B, and C form a triangle if and only if $\{B\ C\}$, $\{C, A\}$, and $\{A, B\}$ are consecutive voronoi neighbors in ordered voronoi neighbor lists of A, B, and C respectively. There are a few degenerate cases and problems arising out of improper sampling. These cases are described in the next section.

6. Implementation and Performance

We have implemented the algorithm described in this paper. This section describes a few of the implementation issues and give the performance of our method on various models.

Most of the sample point clouds that we obtain in practice do not necessarily satisfy our sampling criteria. Further, the positions of these points might be in some kind of geometric degeneracies. In the triangulation stage of our algorithm, assume that A, B, C, and D form a quadrilateral in their Delaunay neighborhood relationship. There might be cases where both BD and AC are Delaunay edges. The other case is the hole formation by the above quadrilateral, where neither BD nor AC is a Delaunay edge. In the first case, a simple contention detection and removal method is used to unambiguate the triangulation. In the second case where the hole is left due to lack of Delaunay edges, we cannot determine if is an actual hole in the model or a hole formed due to sampling artifacts.

We use a simple thresholding strategy based on number of edges in the hole to decide whether to fill the hole or not. This in combination with the hole size forms a good operating rule of thumb to decide the hole filling operation. The average of the normals of the vertices forming the hole is taken as the projection plane normal, where the vertices and the edges forming the hole is projected. A simple 2D triangulation algorithm is used to fill up the 2D hole. This is projected back in 3D to achieve 3D hole filling. This simple heuristic for finding the projection plane works reasonably well for all the practical models we have used in this paper.

We ran the implementation of our algorithm on various models, and the results have been documented in Table 1. All timings are measured on an SGI-Onyx with an R10000 processor running at 194 MHz. Figure 7* (color plate) shows the result of our algorithm on a few models. Our algorithm and implementation is extremely modular and is suitable for parallelization without any change.

7. Conclusion

We have presented a new, simple, and fast surface reconstruction algorithm for unorganized point clouds. We have introduced a new sampling criteria that a given point cloud has to respect so that our algorithm generates the correct reconstructed surface. Our algorithm falls under the classification of advancing front paradigm for surface reconstruction. We proceed by computing a local neighborhood around each sample point and computing the triangulation in its local tangent plane. We have implemented our algorithm and shown its performance on a number of models with varying sample densities and curvature.

References

1. N. Amenta and M. Bern. Surface reconstruction by voronoi filtering. In *ACM Symposium on Computationl Geometry*, 1998.

2. N. Amenta, M. Bern, and M. Kamvysselis. A new voronoi-based surface reconstruction algorithm. In *Proceedings of ACM Siggraph*, pages 415–421, 1998.
3. D. Attali. r-regular shape reconstruction from unorganized points. In *ACM Symposium on Computationl Geometry*, pages 248–253, 1997.
4. C. Bajaj, F. Bernardini, and G. Xu. Automatic reconstruction of surfaces and scalar fields from 3d scans. In *Proceedings of ACM Siggraph*, pages 109–118, 1995.
5. F. Bernardini and C. Bajaj. Sampling and reconstructing manifolds using alpha-shapes. In *Proc. of Ninth Canadian Conference on Computational Geometry*, pages 193–198, 1997.
6. F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 1999.
7. E. Bittar, N. Tsingos, and M. P. Gascuel. Automatic reconstruction from unstructured data: Combining a medial axis and implicit surfaces. *Computer Graphics Forum, Proceedings of Eurographics*, 14(3):457–468, 1995.
8. J. D. Boissonnat. Geometric structures for three-dimensional shape representation. *ACM Transactions on Graphics*, 3(4):266–286, 1984.
9. P. Crossno and E. Angel. Spiraling edge: Fast surface reconstruction from partially organized sample points. *IEEE Visualization '99*, pages 317–324, October 1999.
10. B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proceedings of ACM Siggraph*, pages 303–312, 1996.
11. H. Edelsbrunner and E. Mucke. Three dimensional alpha shapes. *ACM Transactions on Graphics*, 13(1):43–72, 1994.
12. G.H. Golub and C.F. Van Loan. *Matrix Computations*. John Hopkins Press, Baltimore, 1989.
13. B. Guo, J. Menon, and B. Willette. Surface reconstruction from alpha shapes. *Computer Graphics Forum*, 16(4):177–190, 1997.
14. B. Heckel, A. C. Uva, and B. Hamann. Clustering-based generation of hierarchical surface models. In C. M. Wittenbrink and A. Varshney, editors, *IEEE Visualization '98: Late Breaking Hot Topics Proceedings*, pages 41–44, 1998.
15. T. Van Hook. Real-time shaded NC milling display. In *Proceedings of ACM Siggraph*, pages 15–20, 1986.
16. H. Hoppe, T. Derose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized point clouds. In *Proceedings of ACM Siggraph*, pages 71–78, 1992.
17. H. Hoppe, T. Derose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh optimization. In *Proceedings of ACM Siggraph*, pages 21–26, 1993.
18. J. Koenderink. *Solid Shape*. MIT Press, 1989.
19. R. Mencl. A graph-based approach to surface reconstruction. *Computer Graphics Forum, Proceedings of Eurographics*, 14(3):445–456, 1995.
20. R. Mencl and H. Muller. Graph-based surface reconstruction using structures in scattered point sets. *Proceedings of CGI '98*, 1998.
21. R. Mencl and H. Muller. Interpolation and approximation of surfaces from three-dimensional scattered data points. *State of the Art Reports, Eurographics '98*, pages 51–67, 1998.
22. E. P. Mucke. *Shapes and Implementations in Three-Dimensional Geometry*. PhD thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, 1993.
23. B. O'Neill. *Elementary Differential Geometry*. Academic Press, London, UK, 1966.
24. C. Silva and G. Taubin. Curvature-based estimation of surface sampling. In *SIAM Conference on Geometric Design*, 1999.
25. R. Szeliski and D. Tonnesen. Surface modeling with oriented particle systems. In *Proceedings of ACM Siggraph*, pages 185–194, 1992.
26. G. Taubin. Estimating the tensor of curvature of a surface from a polyhedral approximation. In *Proc. of ICCV*, pages 902–907, 1995.
27. M. Teichmann and M. Capps. Surface reconstruction with anisotropic density-scaled alpha shapes. In *Proceedings of IEEE Visualization*, pages 67–72, 1998.
28. D. Terzopoulos, A. Witkin, and M. Kass. Constraints on deformable models: Recovering 3d shape and nonrigid motion. *Artificial Intelligence*, 36:91–123, 1988.
29. R. C. Veltkamp. Boundaries through scattered points of unknown density. *Graphical Models and Image Processing*, 57(6):441–452, 1995.

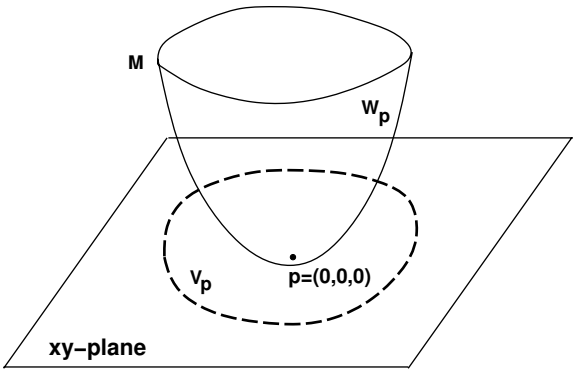


Figure 6: *Quadratic approximation of a surface*

Appendix

In this section, we shall prove some of the theorems from differential geometry which we used in the section on sampling.

Surface Approximations using Power Series

Consider a 2-manifold $\mathbf{F} \subset \mathcal{R}^3$ and a point $p \in \mathbf{F}$ as shown in Figure 6. Without loss of generality, we make the following assumptions about p and $\mathbf{F}$.

- p is the origin,
- the tangent plane of $\mathbf{F}$ at p ($T_p(\mathbf{F})$) is the $z = 0$ plane, and
- the two principal directions of $\mathbf{F}$ at p are the coordinate axes $e_1 = (1,0,0)$ and $e_2 = (0,1,0)$.

It is easily seen that these conditions can be achieved by a simple rigid transformation of $\mathbf{F}$. To proceed further, we make use of a result from classical differential geometry which we state here without proof.

Theorem 2 [23] There exists a small neighborhood W_p of $p \in \mathbf{F}$ such that the map $\pi : (x,y,z) \Rightarrow (x,y)$ is a one-to-one map with its image being an open set $V_p \subset \mathcal{R}^2$. Moreover the map π is a diffeomorphism.

The point $(x,y,z) \in \mathbf{F}$ is a point in the local neighborhood of p and is defined in a local coordinate system at p.

The fact that π is a diffeomorphism implies that π^{-1} exists and that π and π^{-1} are smooth mappings. Therefore, we can approximate the surface in the neighborhood of p (W_p) as

$$W_p = \{(x,y,h(x,y)) : (x,y) \in V_p\}$$

Here, $h(x,y)$ intuitively represents the surface as a height function. Further, the tangent plane of $W_p \subset \mathbf{F}$ is given by the basis vectors $\frac{\partial W_p}{\partial x} = (1,0,\frac{\partial h}{\partial x}(0,0))$ and $\frac{\partial W_p}{\partial y} = (0,1,\frac{\partial h}{\partial y}(0,0))$. Since, we assumed that the tangent plane at p is the $z = 0$ plane, $\frac{\partial h}{\partial x}(0,0) = \frac{\partial h}{\partial y}(0,0) = 0$.

Shape operator is a familiar concept in differential geometry. Essentially, the shape operator at a point $p \in \mathbf{F}$ (denoted by $S_p(\mathbf{F})$) is a linear operator that maps an element of $T_p(\mathbf{F})$ to another element in $T_p(\mathbf{F})$. If v_{p1} and v_{p2} are a set of basis vectors for $T_p(\mathbf{F})$, $S_p(av_{p1} + bv_{p2}) = cv_{p1} + dv_{p2}$ (a,b,c and d are real valued scalars). For the special case of a vector v being a principal direction, $S_p v = Kv$, where K is the principal curvature. In our particular case, the tangent plane is spanned by e_1 and e_2. The shape operator applied to the vectors e_1 and e_2 are given by

$$\begin{aligned} S_p e_1 &= \frac{\partial^2 h}{\partial x^2}(p)e_1 + \frac{\partial^2 h}{\partial x \partial y}(p)e_2 \\ S_p e_2 &= \frac{\partial^2 h}{\partial x \partial y}(p)e_1 + \frac{\partial^2 h}{\partial y^2}(p)e_2 \end{aligned} \tag{10}$$

Since e_1 and e_2 are the principal directions, we can conclude that $\frac{\partial^2 h}{\partial x \partial y}(0,0) = 0$ and that $\frac{\partial^2 h}{\partial x^2}(0,0)$ and $\frac{\partial^2 h}{\partial y^2}(0,0)$ are the principal curvatures (denoted by k_1 and k_2 respectively).

We now use Taylor's formula to expand $h(x,y)$ around the origin $(0,0)$. Thus,

$$\begin{aligned} h(x,y) &= h(0,0) + x\frac{\partial h}{\partial x}(0,0) + y\frac{\partial h}{\partial y}(0,0) \\ &\quad + \frac{1}{2}(x^2\frac{\partial^2 h}{\partial x^2} + 2xy\frac{\partial^2 h}{\partial x \partial y} + y^2\frac{\partial^2 h}{\partial y^2}) \\ &\quad + \text{higher order terms} \\ &= \frac{1}{2}(k_1 x^2 + k_2 y^2) + \text{higher order terms} \\ &\approx \frac{r^2}{2}(k_1 \cos^2\theta + k_2 \sin^2\theta), \end{aligned}$$

where $r = \sqrt{x^2+y^2}$ and θ is the angle the vector (x,y) makes with the x-axis.

Euler Equation

The *normal curvature* k_v at a point p on the surface in a given direction $\vec{v}$ on the tangent plane is defined as the curvature of the intersection curve of the surface with the plane formed by the vectors $\vec{v}$ and the surface normal at p. Using the *shape operator*, it can be written as

$$k_v = S_p(\vec{v}) \cdot \vec{v}$$

We can represent $\vec{v}$ in terms of the principal directions ($\vec{v}_1$ and $\vec{v}_2$) as $\vec{v} = \cos\theta\vec{v}_1 + \sin\theta\vec{v}_2$, where θ is the angle $\vec{v}$ makes with $\vec{v}_1$. Therefore

$$\begin{aligned} k_v &= S_p(\cos\theta\vec{v}_1 + \sin\theta\vec{v}_2)\cdot(\cos\theta\vec{v}_1 + \sin\theta\vec{v}_2) \\ &= (k_1\cos\theta\vec{v}_1 + k_2\sin\theta\vec{v}_2)\cdot(\cos\theta\vec{v}_1 + \sin\theta\vec{v}_2) \\ &= k_1\cos^2\theta + k_2\sin^2\theta \end{aligned}$$

The above equation is also known as the *Euler equation*. It expresses the normal curvature at a point on the surface in terms of the principal curvatures there.

Frenet-Serret equations for Darboux Frame

We will now present the equations which govern the behavior of the Darboux frame for space curves on a surface $\mathbf{F}$ passing through p. These equations are a modified form of the well-known Frenet-Serret equations [18, 23].

$$\begin{pmatrix} \frac{\partial\vec{T}}{\partial s} \\ \frac{\partial\vec{B}}{\partial s} \\ \frac{\partial\vec{N}}{\partial s} \end{pmatrix} = \begin{pmatrix} 0 & g & k_v \\ -g & 0 & t \\ -k_v & -t & 0 \end{pmatrix} \begin{pmatrix} \vec{T} \\ \vec{B} \\ \vec{N} \end{pmatrix}$$

The entries in the above matrix define the geometrical invariants at the point in consideration. k_v is the component of the acceleration along the surface normal (*normal curvature*). g (or *geodesic curvature*) is the component of the acceleration in the tangent plane. t (also known as *geodesic torsion*) intuitively measures the twist along the $\vec{N}-\vec{B}$ plane.

Of this, the third equation governing the change in normal is of most interest to us.

$$\begin{aligned} \partial\vec{N} &= -k_v\partial s\vec{T} - t\partial s\vec{B}, or \\ |\partial\vec{N}| &= \sqrt{k_v^2 + t^2}|\partial s| \end{aligned}$$

Proofs of various theorems

We shall prove some of the theorems claimed in section 4.

Lemma 1 Given a δ-sample S of a surface $\mathbf{F}$ and two points $p, q \in S$ such that $q \in C_p^\delta$. Then $p \in C_q^\delta$.

Proof: We will first show that the angle between the normal at p and the normals at all the points on C_p^δ is a constant, and is related to δ. Specializing equation (2) by zeroing out the torsion factor, we know that

$$\vec{N} + \partial\vec{N} = \vec{N} - k_v\partial s\vec{T} \tag{11}$$

Therefore the cosine of the angle between $\vec{N}$ and $\vec{N}+\partial\vec{N}$ is simply $1/\sqrt{1+(k_v\partial s)^2}$ or $1/\sqrt{1+\delta^2}$ which is a contant. Since the dot product is just a function of δ which is constant throughout the surface, and since the deviation (or dot product) is symmetric, $p \in C_q^\delta$.

Let $p \in \mathbf{F}$. Define the contour C_p^δ around p as before.

Theorem 1 (a) Consider any two points $q, r \in C_p^\delta$. Then the maximum ratio of edge distances $\frac{|pq|}{|pr|}$ is bounded above by a function of principal curvatures.

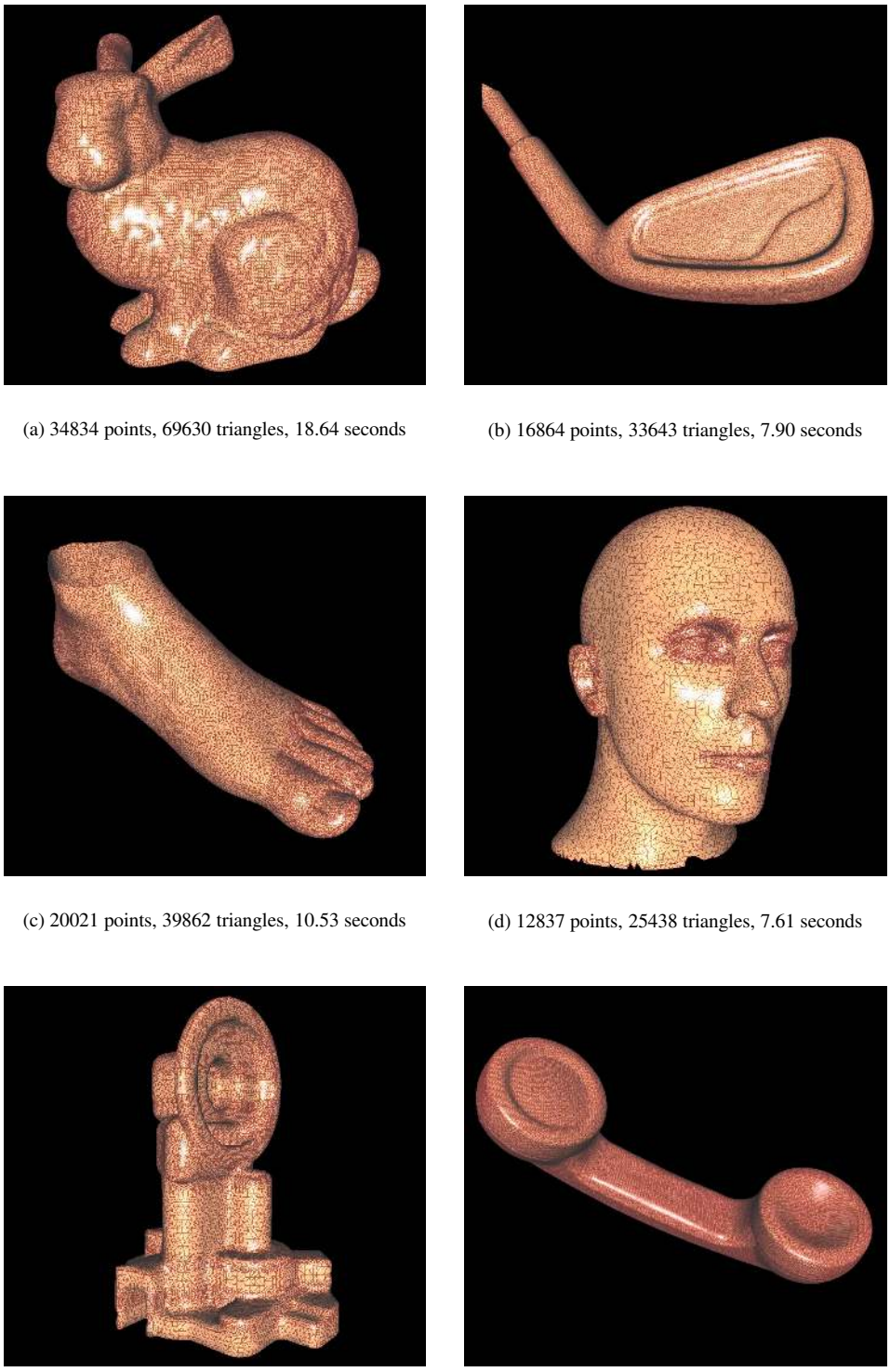
(b) Let N_p be the unit normal to $\mathbf{F}$ at p and let $\hat{pq}$ denote the unit vector from p to q. Define the height function $H(p,q) = |N_p \cdot \vec{pq}|$. Then $H(p,q)$ is bounded above by the quantity $\frac{\sqrt{1+\delta^2}-1}{k_{min}}$. k_{min} is the smaller of the principal curvatures.

(c) Define the angle function $D(p,q) = |N_p \cdot \hat{pq}|$. Then $D(p,q)$ is a constant, $\frac{\sqrt{1+\delta^2}-1}{\sqrt{1+\delta^2}+1}$, for all $q \in C_p^\delta$.

Proof: (a) From equation (4), $k_{v_q}|pq| = k_{v_r}|pr| = \delta$. Therefore, $\frac{|pq|}{|pr|} = \frac{k_{v_r}}{k_{v_q}}$. This ratio reaches maximum when k_{v_r} is maximum and k_{v_q} is minimum. They are attained when the two directions v_r and v_q are the principal directions v_1 and v_2. Therefore, the maximum ratio at every point is the ratio of the principal curvatures at that point. But this is bounded by our assumption.

(b) In order to prove the height function bound, let us assume without loss of generality that p is the origin and the normal vector at p is $(0,0,1)$ and q is given by the point $(x,y,h(x,y))$. Then $H(p,q)$ is simply $h(x,y)$. From equation (1) we know that $h(x,y)$ can be rewritten as $\frac{k_v r^2}{2}$. Also our δ-sampling criterion gives us the equation $k_v\sqrt{r^2 + \frac{k_v^2 r^4}{4}} = \delta$. After some simple algebraic manipulation and replacing $\frac{k_v r^2}{2}$ by h, we get the quadratic equation $k_v^2 h^2 + 2k_v h - \delta^2 = 0$. The positive solution of this equation is $\frac{\sqrt{1+\delta^2}-1}{k_v}$. It reaches a maximum when the normal curvature is minimum. This value is attained at the smaller of the two principal curvatures.

(c) The angle function $D(p,q) = \frac{h(x,y)}{\sqrt{x^2+y^2+h^2(x,y)}}$. From equation (1) and simplifying after substituting $r = \sqrt{x^2+y^2}$, we get $D^2(p,q) = \frac{k_v^2 r^2}{(4+k_v^2 r^2)}$. Using a similar manipulation as was done for proving the height function bound but substituting $k_v^2 r^2$ by d, we get the quadratic equation $d^2 + 4d - 4\delta^2 = 0$. The positive root of this quadratic equation is the constant $2(\sqrt{1+\delta^2} - 1)$. Plugging this value into the expression for $D^(p,q)$, we get $\frac{\sqrt{1+\delta^2}-1}{\sqrt{1+\delta^2}+1}$. Therefore, all the points in the curve C_p^δ make the same angle with the normal at p.



(a) 34834 points, 69630 triangles, 18.64 seconds

(b) 16864 points, 33643 triangles, 7.90 seconds

(c) 20021 points, 39862 triangles, 10.53 seconds

(d) 12837 points, 25438 triangles, 7.61 seconds

(e) 30937 points, 61772 triangles, 20.99 seconds

(f) 83034 points, 165981 triangles, 44.41 seconds

Figure 7: *Triangulations generated by our algorithm. Below each image, we include the number of points and triangles generated, and the running time of the algorithm.*