5

Meshing of Surfaces

Jean-Daniel Boissonnat[1], David Cohen-Steiner[1], Bernard Mourrain[1], Günter Rote[2*], and Gert Vegter[3]

[1] INRIA
[2] Freie Universität Berlin
[3] Rijksuniversiteit Groningen

5.1 Introduction: What is Meshing?

Meshing is the process of computing, for a given surface, a representation consisting of pieces of simple surface patches. In the easiest case, the result will be a triangulated polygonal surface. More general meshes also include quadrilateral (not necessarily planar) patches or more complicated pieces, but they will not be discussed here. For example, Figure 5.1a shows a meshed

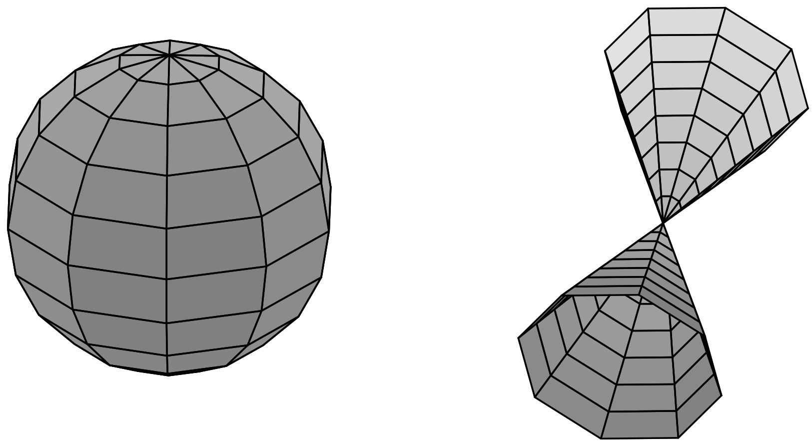


Fig. 5.1. (a) A meshed sphere. (b) A meshed double-cone.

sphere $x^2 + y^2 + z^2 = 1$. Figure 5.1b shows a good mesh of a double-cone $x^2 + y^2 = z^2$. Note that the cone has pinching point at the apex, which is represented correctly in this mesh. The automatic construction of good

* Coordinator

meshes for surfaces with singularities is still an open research area. Here we will mainly concentrate on smooth surfaces, with the exception of Section 5.4, where surfaces with singularities are also treated.

Why Meshing?

The meshing problem occurs in different settings, depending on the way how a surface is given, and on the purpose of meshing.

Usually we assume that a surface is given *implicitly*, as the solution of an equation

$$f(x, y, z) = 0.$$

The function f comes from various sources. It can be explicitly given, often as a polynomial, like in the examples above, or as a sum of exponential "blob" functions like $\exp(-\|A(x-b)\|)-1$ for a matrix A and a center b, which allow flexible modelling of shapes.

One can also try to fit data by defining an appropriate function f from the data. For example, in scattered data interpolation, a scanned image may be given a two-dimensional (or sometimes three-dimensional) grid of grey-level values. If f is a function that interpolates these values on each grid square, the equation $f(x, y) =$ const extracts a level curve (iso-curve, or iso-surface in higher dimensions). In the area of surface reconstruction from scattered data points, there are procedures for defining a function f whose zero set is an approximation of the unknown surface (natural neighbor interpolation, see Section 6.3.3, p. 261.)

The implicit representation of a surface is convenient for defining the surface as a mathematical object, for modeling and manipulation by the user, but it is not very convenient for handling the surface by computer. For drawing or displaying a surface (Computer Graphics), an *explicit* representation as a union of polygons is easier to handle. Engineering applications that perform computations on the surface (and on the volume inside and outside the surface), such as finite element analysis, also require a meshed surface.

Related Problems.

Sometimes, a surface is already given as a mesh, but one wants to construct a different, "better" mesh. For example, one may try to improve the shape of the triangles, eliminating long and skinny triangles or triangles that are too large, or one may want to produce a coarser mesh, eliminating areas that are meshed too densely, with the purpose of reducing the amount of data for storage or transmission (data compression). These are the problems of *remeshing*, *mesh refinement* and *mesh simplification*. These problems are also applicable for plane meshes, where the given "surface" is a region of the plane [284, 300]. Some of the methods that we will discuss below have been applied to this setting, and to the meshing problem for polyhedral surfaces in general, but we will only mention this briefly.

As mentioned, engineering applications also require three-dimensional *volume meshes* or even higher-dimensional meshes. Extending a given boundary mesh of a surface to a mesh of the enclosed volume is a difficult problem of its own.

In this chapter, we will concentrate on surface meshing. The other problems described above are not covered in this book. For simplicity, we restrict our attention to surfaces without boundary. Some algorithms can clip a surface by some bounding box or by intersecting it with some other surface, but we will not discuss this.

Meshing of curves, by comparison, is a much easier problem: Here we look for a polygonal chain approximating a curve. We will often discuss curve meshing because the main ideas of many meshing algorithms can be illustrated in this setting.

Meshing is related to surface reconstruction, which is the subject of Chapter 6: in both cases, the desired output is a meshed surface. However, surface reconstruction starts from a set of sample points of the surface that is given as input, and which is usually the result of some measurement process. In meshing, one also constructs a point sample of the surface, but the selection of these points is under the control of the algorithm.

There is some overlap in the techniques applied, in particular in the area of *Delaunay meshing* (Section 5.3). As mentioned above, one way of reconstructing a surface is by defining a function f whose zero set is the reconstructed surface.

Goals of Meshing Algorithms—Correctness.

There is a vast literature on meshing with many practical algorithms in the areas of Computer-Aided Geometric Design (CAGD) and Computer Graphics. In this book we concentrate on methods with proven correctness and quality guarantees. Correctness means that the result should be *topologically correct* and *geometrically close*.

The definition of what it means for a mesh to be topologically correct has evolved over the past few years. It is not sufficient to require that a surface S and its mesh S' are homeomorphic. A torus and a knotted torus are homeomorphic when viewed as surfaces in isolation, but one would certainly not accept one as a topologically correct representation of the other, see Figure 5.2. The following definition combines the strongest notions of having the correct topology with the requirement of geometric closeness.

Definition 1. *An* ambient isotopy *between two surfaces $S, S' \subset \mathbb{R}^3$ is a continuous mapping*

$$\gamma\colon \mathbb{R}^3 \times [0,1] \to \mathbb{R}^3$$

which, for any fixed $t \subseteq [0,1]$, is a homeomorphism $\gamma(\cdot,t)$ from $\mathbb{R}^3$ to itself, and which continuously deforms S into the mesh S': $\gamma(S,1) = S'$.

In addition, the approximation error D is the largest distance by which a point is moved by this homeomorphism:

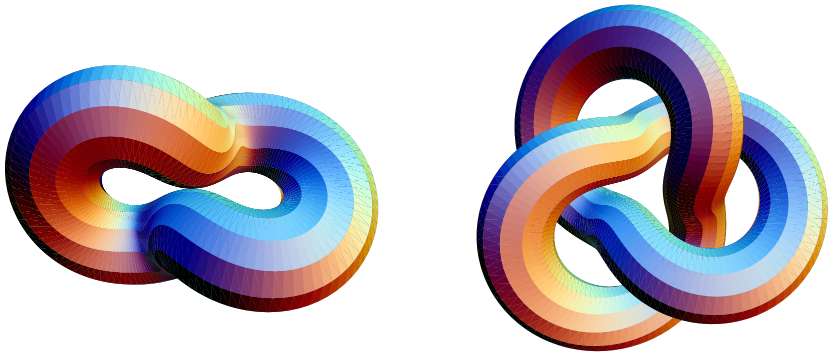


Fig. 5.2. Two homeomorphic surfaces which are not isotopic

$$\|x - \gamma(x, 1)\| \leq D \textit{ for all } x \in S$$

This implies that the surface S and the meshed surface S' are homeomorphic to each other, and their Hausdorff distance (see the definition in Section 6.2.3 on p. 249) is at most D. There is also the concept of isotopy between two surfaces, which only deforms S without deforming the ambient space $\mathbb{R}^3$, see Section 7.2 (p. 279):

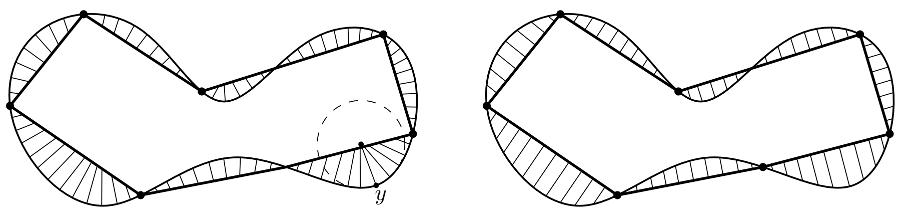
Definition 2. *An* isotopy *between two surfaces* $S, S' \subset \mathbb{R}^3$ *is a continuous mapping*

$$\gamma\colon S \times [0, 1] \to \mathbb{R}^3$$

which, for any fixed $t \subseteq [0, 1]$*, is a homeomorphism* $\gamma(\cdot, t)$ *from* S *onto its image, and which continuously deforms* S *into the mesh* S'*:* $\gamma(S, 1) = S'$.

Formally, isotopy is weaker than ambient isotopy. However, for our purposes, there is no difference between between isotopy and ambient isotopy: The isotopy extension lemma ensures that an isotopy between two smooth surfaces (of class C^1) embedded in $\mathbb{R}^3$ can always be extended to an ambient isotopy [198, Theorem 1.3 of Chapter 8, p. 180]. This does not directly apply to a piecewise linear surface mesh S', but it is easy to show that a piecewise linear surface is ambient isotopic to an approximating smooth surface, to which the theorem applies. The isotopy extension lemma cannot be used in the algorithm of Section 5.4, which deals with singular surfaces, but in this case, the ambient isotopy will be constructed explicitly.

Theorems in earlier papers have only made claims about the Hausdorff distance or about the existence of a homeomorphism, but it is often not difficult to obtain also isotopy. Typically, the mapping constructed in the proofs moves points along fibers that sweep out the space between the surface S and its approximation S'. These fibers move each point of the mesh S' to its closest neighbor on the surface S', as in Figure 5.3a. (For a curve in the plane, one can also map each curve point to its closest neighbor on the mesh, as in Figure 5.3b; this does not work in higher dimensions because it may lead to a discontinuous mapping.) Each point x on the mesh is mapped to



(a) (b)

Fig. 5.3. The isotopy between a smooth curve and the approximating polygon is indicated by fibers perpendicular to the curve (a) or to the polygon edges (b). The curve can be continuously deformed into the polygon by moving each point along its fiber.

the "correct" corresponding closest point y on the surface, if x is not too far from S, in particular, if the distance from y to x is not larger than the radius of the medial sphere, see Figure 5.3a, which shows the medial circle at $y \in S$. (See p. 109 in Section 2.7 for the definition of medial sphere.)

A common tool for establishing isotopy is a *tubular neighborhood* $\hat{S}$ of a surface S. It is a thickening of the surface such that within the volume of $\hat{S}$, the projection of a point x to the nearest point $\pi_S(x)$ on S is well-defined. The points x which have the same nearest neighbor $\pi_S(x) = p$ form a segment through p normal to the surface. These segments are called *fibers* of the tubular neighborhood, and they form a partition of $\hat{S}$.

Lemma 1 (see [285, Theorem 4.1]). *Let S be a compact closed surface of class C^2 in $\mathbb{R}^3$ with a tubular neighborhood $\hat{S}$. Let T be a closed surface (not necessarily smooth) contained in $\hat{S}$ such that every fiber intersects T in exactly one point.*

Then $\pi_S \colon T \to S$ induces an ambient isotopy that maps T to S. □

The isotopy interpolates between T and S along the fibers, and it does not move points by more than the length of the longest fiber.

Principal Approach and Primitive Operations.

Although other approaches are conceivable, all methods basically select vertices *on the surface* and connect them appropriately. The fundamental operation is to find the intersection point of a line segment with the surface. For a few algorithms, it is sufficient to compute these intersections only approximately, and thus the mesh vertices will lie only *close* to the surface. (In particular, this holds for the marching cubes algorithms in Section marching-cubes and piecewise-linear interpolation in Section 5.5.2).

Some algorithms also compute certain critical points of the surface. All these operations require access to the function f defining the surface and its derivatives. Intersecting the surface with a line segment boils down to solving a univariate equation.

In order to ensure some quality and correctness guarantees on the mesh, some algorithms need to obtain further information about the surface, for instance, bounds on the curvature, or in more algebraic terms, bounds on the derivatives of the function defining the surface.

We will discuss the required primitive operations in detail with each algorithm.

Other Quality Criteria.

Besides topological correctness and geometric closeness to the original surface, we may want to achieve other criteria.

1. Normals: The normals of the mesh should not deviate too much from the normals of the surface. Note that a "wiggly" approximation of a given surface or a curve can be isotopic and have arbitrarily small Hausdorff distance, but still have normals deviating very badly from the original normals.
2. Smoothness: Adjacent facets should not form a sharp angle.
3. Desired density. We may impose an upper bound on the size of the mesh triangles. This bound may depend on the location. For example, in fluid mechanics calculations, a region of turbulence will require a finer mesh than a region of smooth flow.
4. Regularity and Shape. We want to avoid skinny triangles with sharp angles.

There are many other criteria for individual mesh elements. All of these criteria, except the first two in the above list, also apply to plane meshes, and they have been studied extensively in the literature. The algorithms in this chapter concentrate on achieving correctness; geometric quality criteria are often considered in a secondary refinement step.

Basic Assumptions about Smoothness.

The basic assumption for most part of this chapter is that f is a smooth function and the surface has no singularities.

Assumption 1 Nonsingularity.
The function f and its gradient ∇f are never simultaneously zero.

This implies that the equation $f(x) = 0$ defines a collection of smooth surfaces without boundary. As mentioned in the introduction, meshing in the vicinity of singularities is a difficult open problem and an active area of research.

Section: Algorithm	Strategy	Topological Correctness	Scaffolding
5.2.3: Snyder [303, 304]	refinement	global parameterizability	cubes
5.2.4: Plantinga and Vegter [270]	refinement	Small Normal Variation	cubes
5.3.1: Boissonnat and Oudot [60]	refinement	sample density, local feature size	Voronoi diagram
5.3.2: Cheng, Dey, Ramos and Ray [85]	refinement	topological ball property*	Voronoi diagram
5.4: Mourrain and Técourt [247, 309]	space sweep, vertical projection	treatment of critical points	vertical planes
5.5.1: Stander and Hart [306]	parameter sweep	Morse theory	—
5.5.2: Boissonnat, Cohen-Steiner, and Vegter [57]	refinement	(Morse theory)*†	boxes†

Table 5.1. A rough taxonomy of the meshing algorithms described in this chapter, according to the overall strategy, the way how topological correctness is achieved, and the kind of spatial ground structure ("scaffolding") that is used.
* These two algorithms employ a hierarchy of conditions, which are successively tested, in order to achieve correctness.
† This algorithm uses Morse theory in a more indirect way. It works with boxes which are subdivided into tetrahedra.

5.1.1 Overview

Meshing algorithms can be roughly characterized as (i) continuation-based methods, that grow a mesh following the surface, and (ii) mesh-based methods, which build some sort of three-dimensional scaffolding around the surface. Although continuation-based methods are often used in practice, it is not easy to achieve correctness guarantees for them. Thus, all algorithms discussed in this chapter fall into the second category. There are three types of adaptive "grid structures" which are used: axis-aligned cubes, vertical planes, and the Voronoi diagram. The algorithms use different algorithmic strategies and a variety of conditions to ensure topological correctness. Table 5.1 summarizes these characteristics. One can see that the algorithms are related in various different ways. In the remainder of this chapter, we have chosen to group the algorithms mainly by the mathematical idea that underlies their correctness, but a different organization might be equally reasonable. It is perhaps rewarding to return to Table 5.1 after reading the chapter.

5.2 Marching Cubes and Cube-Based Algorithms

The *marching cubes algorithm* [325, 230] conceptually covers space by a *grid* of small cubes, and locally computes a mesh for each cube which is intersected by the surface. The algorithm computes f at all grid points. The surface must pass between the grid points with positive f and the grid points with negative f. The algorithm computes intersection points between the surface and the grid edges whose endpoints have opposite signs, and uses them as the vertices of the mesh. Depending on the desired accuracy, these intersection points can be computed by linear interpolation of f between the endpoints or some more elaborate method, or one can simply choose the midpoints of the edges.

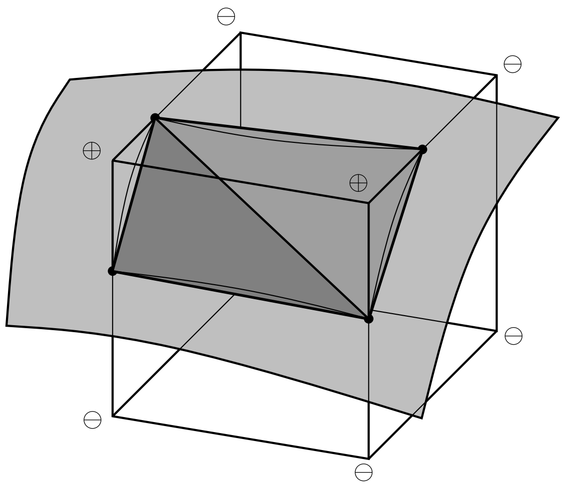


Fig. 5.4. A cube intersected by the surface $f(x, y, z) = 0$. The sign of f at the vertices is shown. Inside the cube, the surface will be represented by two triangles connecting the points where the surface intersects the edges of the cube.

These vertices have to be connected, inside each cube, by a triangular mesh that separates the positive and the negative vertices, as shown in Figure 5.4. There are different possible patterns of positive and negative cube vertices, and the triangulation can be chosen according to a precomputed table of cases. (In three dimensions, the $2^8 = 256$ cases are reduced to 15 different cases by taking into account symmetries.)

Problems with this method appear when the grid is not sufficiently fine to capture the features of the surface. In some cases, the connecting triangulation between the mesh vertices inside a cube is ambiguous, and some arbitrary decision has to be made. Figure 5.5 shows a two-dimensional instance of a curve whose constructed mesh has an incorrect topology (2 cycles instead of a

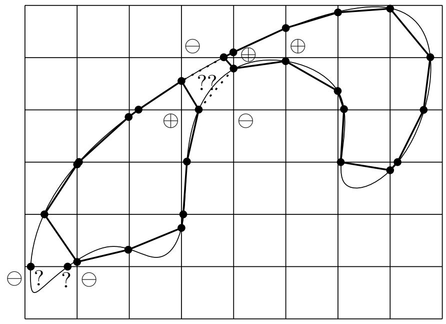


Fig. 5.5. The intersections marked with ? are not found by checking the signs of f at grid points. In the left picture, this leads to a curve with two components instead of one. The right figure shows a shifted copy of the same curve. When all 4 sides of a square are intersected, as in the square marked ??, there are two ways of connecting them pairwise. Applying a local rule to decide between the two possibilities may not always give the correct result. In addition, the meshed curve misses a large part of the protrusion in the lower left corner.

single cycle). A slightly shifted grid leads to an ambiguous situation: the sign pattern of f at the vertices is not sufficient to decide how the points should be connected.

It can also happen that components of the implicit surface or curve which are smaller than a grid cell may be completely missed.

Originally, the marching cubes algorithm was designed to find isosurfaces of the form $f(x) = \text{const}$ in a scalar field f resulting from medical imaging procedures, with one data value (gray-level) $f(p)$ for each point p in a pixel or voxel grid. In this case, one has to live with ambiguities, trying to make the best out of the available data. One can resolve ambiguities by any rule which ensures that triangles in different cubes fit together to form a closed surface. (In fact, the rules as given originally in [230] may resolve ambiguities inconsistently, leading to meshes which are not watertight surfaces.)

In our setting, however, the function f is defined everywhere, and it is not sufficient to achieve consistency: we want a topologically correct mesh; on the other hand, we are free to compute additional values at any point we like. Thus, the obvious way to solve ambiguities is to refine the grid, see Figure 5.6.

5.2.1 Criteria for a Correct Mesh Inside a Cube

The basic strategy is to subdivide the cells of the grid until sufficient information is available for forming a correct mesh. The question is now: How can we tell that a mesh within some grid cube is correct and the cube need not be further subdivided.

We present two criteria, an older one due to Snyder [304, 303], and a recent one, due to Plantinga and Vegter [270].

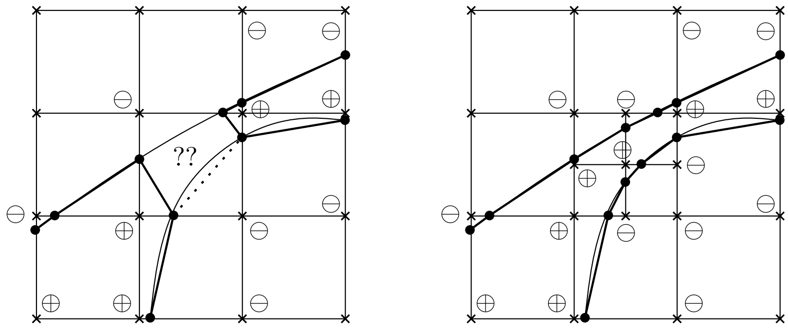


Fig. 5.6. A section from the right part of Figure 5.5 before and after subdividing the offending square into four subsquares. The ambiguity is resolved.

5.2.2 Interval Arithmetic for Estimating the Range of a Function

Both methods are based on estimating the range of values of a function $\{ g(p) \mid p \in B \}$ when the parameter p ranges over some domain B, which is typically a box. In most cases, g will be the function f or one of its derivatives, and one tries to establish that the range of g does not contain 0, i.e., $g(p) \neq 0$ for all $p \in B$.

For example, if B is a grid cube and $f(p) \neq 0$ for all $p \in B$, we know that the surface S contains no point of B, and no further processing of B is necessary.

The standard technique for estimating the range of a function is *interval arithmetic*. The idea of using interval arithmetic in connection with meshing was pioneered by Snyder [304, 303].

Interval arithmetic has been introduced in Section 3.2.2 as a method to cope with the limited precision of floating-point computation. Ideally, one would like to have the exact result x of a computation. By interval arithmetic, one obtains an interval $[a, b]$ which definitely contains the correct result x. By repeating the calculations with increased accuracy, one can decrease the size $b-a$ of the interval until it becomes smaller than any prespecified limit $\varepsilon > 0$.

Interval arithmetic is also suited to get the range of a function whose inputs $x_1, x_2, x_3, \ldots$ vary in some given intervals. Instead of starting with zero-length intervals for exact inputs, one takes the given starting intervals. Note that this approach may suffer from a systematic overestimation of the resulting interval, for example when subtracting quantities that are close, like in $f(x) = x - \sin x$. For $x = [0, 0.3]$, we get the interval approximation $\Box f([0, 0.3]) = [0, 0.3] \boxminus (\Box\sin)([0, 0.3]) = [0, 0.3] \boxminus [0, 0.2956] = [-0.2956, 0.3]$, whereas the true range of f is contained in $[0, 0.0045]$. There are techniques such as Fast Automatic Differentiation [184] which can circumvent this problem.

For the remainder of this chapter, we assume, for a given function g the availability of some operation $\Box g$ such that $\Box g([a_1, b_1], [a_2, b_2], \ldots)$ returns an interval that contains the range $\{ g(x_1, x_2, \ldots) \mid a_1 \le x_1 \le b_1,\ a_2 \le x_2 \le b_2,\ \ldots \}$. We require that the resulting interval can be made arbitrarily small if the input intervals are small enough:

Assumption 2 CONVERGENCE *(of Interval Arithmetic).*
The size of the interval $\Box g([a_1, b_1], [a_2, b_2], \ldots)$ goes to zero if the sizes of the parameter intervals $[a_1, b_1]$, $[a_2, b_2]$, … tend to zero.

This requirement is fulfilled for ordinary interval arithmetic in the vicinity of every point for which g is continuous, provided that the accuracy of the calculations can be increased arbitrarily. If the computation of g does not involve operations that may be undefined, like division by zero, and in particular, if g is a polynomial, interval arithmetic converges.

If a continuous function g satisfies $g(x) \neq 0$ for all x in a box B, convergence implies that interval arithmetic will establish this fact by subdividing B into sufficiently many sub-boxes.

5.2.3 Global Parameterizability: Snyder's Algorithm

We want to compute a mesh for a surface $f(x) = 0$ inside some box $X = [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}] \times [z_{\min}, z_{\max}]$. The method works by induction on the dimension. Thus, as a subroutine, the algorithm will need to compute a mesh for a curve $f(x) = 0$ (an approximating polygonal chain) inside some rectangle $X = [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$.

Snyder's criterion for telling when a cell need not be further subdivided is *global parameterizability* of f inside a box X in two of the variables x, y, or z. We say that $\{ (x, y, z) \in X \mid f(x, y, z) = 0 \}$ is *globally parameterizable* in the parameters x and y, if, for each pair (x, y), the equation $f(x, y, z) = 0$ has at most one solution z in the box X. One way to establish this property is to ensure that the derivative with respect to the third variable is nowhere zero:

$$\frac{\partial}{\partial z} f(x, y, z) \neq 0 \text{ for all } (x, y, z) \in X \tag{5.1}$$

An analogous definition holds for a curve $\{ (x, y) \in X \mid f(x, y) = 0 \}$ in two dimensions: it is globally parameterizable in the parameter x, if, for each value x, the equation $f(x, y) = 0$ has at most one solution y in the box X.

Global parameterizability of a curve in the parameter x means that the solution consists of a sequence of curves which can be written in the parameterized form $y = C(x)$, over sequence of disjoint intervals for the parameter x. Similarly, global parameterizability of a surface in the parameters x and y means that the solution consists of parameterized surface patches $z = S(x, y)$ over a set of disjoint domains for (x, y).

Suppose that a curve in a two-dimensional rectangle X is globally parameterizable in x. The curve has at most one intersection with the left and right edge, and an arbitrary number of intersections with the bottom and top edge. Let $x_1, x_2, x_3, \ldots$ denote the sequence of intersections, sorted from left to right, see Figure 5.7a.

Between the first two successive intersections x_1 and x_2, there can either be no solution inside X, or the solution can be an x-monotone curve in X. These two possibilities can be distinguished by intersecting the curve with a vertical line segment ℓ half-way between x_1 and x_2. More precisely, we just need to compute the signs of f at the endpoints of ℓ. Given this information, we can draw polygonal connections between the points x_i which are a topologically correct representation of the curve pieces inside X, as in Figure 5.7b. To connect two points x_i and x_{i+1} on the same edge, we can for example draw two 45° segments. Points on different edges can be connected by straight lines.

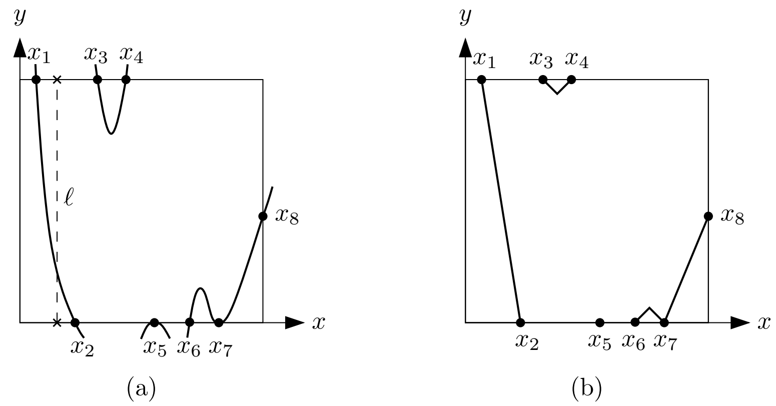


Fig. 5.7. Finding a correct mesh for a curve in a square

The following lemma summarizes this procedure, and it also formulates the three-dimensional version.

Lemma 2. *1. If a curve $f(x, y) = 0$ is globally parameterizable in x in a two-dimensional box X, and if one can find the zeros of f on the edges of the box, then one can construct a topologically correct mesh for the curve inside X.*

2. • *If a surface $f(x, y, z) = 0$ is globally parameterizable in x and y in a three-dimensional box X, and*
• *if, on the top and bottom face of X, each of the two functions $f(x, y, z_{\max})$ and $f(x, y, z_{\min})$ is everywhere nonzero or globally parameterizable in x or y, (not necessarily both in the same variable)*

then one can construct a topologically correct mesh for the surface inside X, provided that one can find the zeros of f on the edges of the box.

In each case, the only additional information required is the sign of f at a few points on the edges of the box.

We call a function f *well-behaved* with respect to the box X if the conditions of part 2 of the lemma are satisfied, possibly after permuting the coordinates x, y, and z.

Proof (Proof of part 2). Suppose the surface intersects the bottom face as in Figure 5.7a and the top face as in Figure 5.8a. Figure 5.8b shows the overlay of the two intersection patterns, like in a top view onto the box. By global parameterizability in x and y, the intersections with the top face and the bottom face cannot cross. In each region which is delimited by these intersection curves, there is either no intersection of the surface with X or there is a single surface patch. These two cases can be distinguished by checking whether an appropriate vertical line segment in the boundary of X intersects the surface, i.e., whether f has opposite signs at the endpoints of this segment.

Figure 5.8c shows the polygonal mesh for the intersection with the top face, and Figure 5.8d shows the overlay with Figure 5.7b. The shaded areas in Figure 5.8b and 5.8d represent the existing patches of the surface in X and the corresponding patches of the mesh that are to be constructed. Such a mesh can be constructed easily: we may need to find intersection points at the vertical edges of X, and we may need to add $45°$ segments on the vertical sides of X. On each vertical side, the mesh edges will look exactly as the ones that would be produced by part 1 of the lemma. This is important to ensure that the surface patches in adjacent boxes fit together across box boundaries, even if the adjacent box is parameterizable in y and z or in x and z.

Any triangulation of the grey polygons in Figure 5.8d will now lead to appropriate triangulated surface patches, as shown in Figure 5.8e. We leave it as exercise to construct an isotopy that shows topological correctness in the sense of Definition 1.

We can now present the overall algorithm. We suppose we are given an initial box containing the part of the surface in which we are interested.

The algorithm maintains a list of boxes that are to be processed. We select a box X from the list and process it as follows. First we try to establish that $f(x) \neq 0$ in X, using interval arithmetic. If this is the case, we can discard the box without further processing. Otherwise, we check if Lemma 2 can be applied. Using interval arithmetic, we try to show that one of the partial derivatives is nonzero in X, see (5.1), which implies that f is globally parameterizable in two of the parameters x, y, and z. We then also have to check the "top" and "bottom" faces of X, in a completely analogous way: Either f or one of its partial derivatives must be nonzero on the face. If this test succeeds, we know that we can mesh the surface in X. Otherwise, we subdivide X into smaller boxes and put them on the list for further processing.

The approximation error is trivially bounded by the diameter of the box, regardless of how we construct the mesh in each box. Thus, if we want to

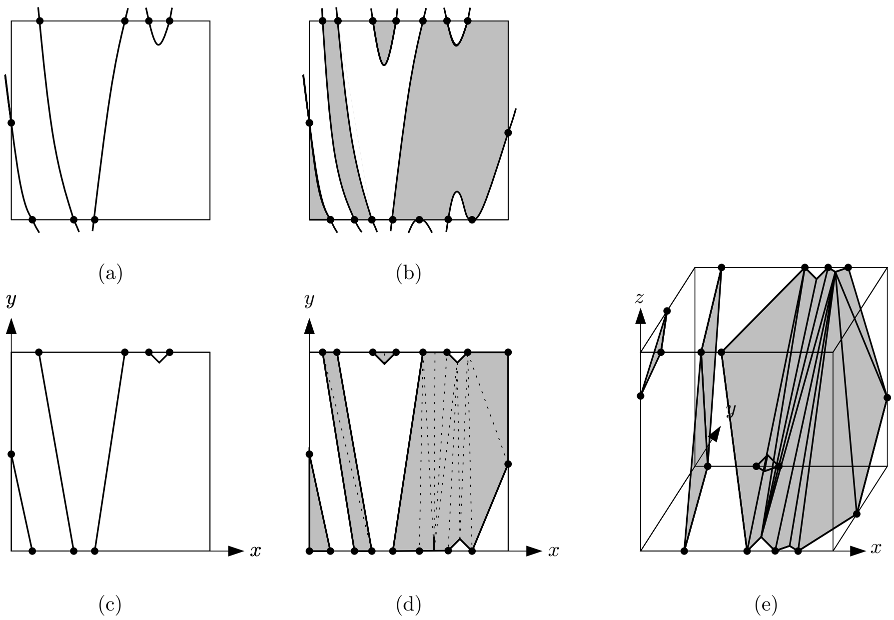


Fig. 5.8. Finding a correct mesh for a surface in a cube

guarantee a small error, we can achieve this by subdividing boxes that are too large, before checking global parameterizability.

In the end, we have a bunch of boxes of different sizes in which we have to construct meshes. Cubes of different sizes may touch, and therefore the method of Lemma 2 must be adapted: The surface is first meshed inside the smallest boxes. The pattern of intersection with the boundary is transmitted to larger adjacent boxes, and thus the mesh boundary on the sides of the boxes may look more involved than in Figure 5.8e. The largest boxes are meshed last.

We still have to discuss the assumption of Lemma 2 that the intersections of the surface with the cube edges can be found. Snyder [304, 303] proposed to use interval arithmetic also for this task. In fact, this problem is just the meshing problem in one dimension: finding zeros $f(x) = 0$ of a univariate function f. (The two- and three-dimensional versions are treated in Lemma 2.) Global parameterizability in this setting boils down to requiring $f' \neq 0$.

The basic algorithm successively subdivides a starting interval until $f \neq 0$ or $f' \neq 0$ can be established to hold throughout each subinterval, by interval arithmetic. Then one can establish the existence of a unique zero or the absence of a zero by computing the sign of f at all interval endpoints. The results is a sequence of disjoint isolating intervals $[u_1, v_1]$, $[u_2, v_2]$, …, where each interval is known to contain a unique zero (cf. Section 3.3.2).

Note that the sequence of points x_1, x_2, ... need not be exact for the algorithm in part 1 of the lemma to work. All that is required is that they have the correct order. There will be a sequence of disjoint isolating intervals for the intersections with the upper edge and another sequence of disjoint isolating intervals for the intersections with the lower edge. If any two of these intervals overlap, the intervals must be refined until they become disjoint. This will eventually happen, since, by global parameterizability, the zeros on the upper edge and on the lower edge are distinct. Then any point from the respective interval can be used to construct the mesh.

Note however, that interval arithmetic fails to converge for zeros where the function only touches the zero line or does not cross it transversally such as the points x_5 and x_7 in Figure 5.7a, or generally when $f(x) = f'(x) = 0$ (*grazing intersections*). No amount of subdivision will suffice to show the presence or absence of a zero in this case.

Thus, to cope with these cases, one has to resort to the exact methods of Chapter 3. In practice, one could of course simply stop the subdivision when the size of the intervals become smaller than some threshold and "declare" the presence of a zero in this interval, giving up any correctness claims below the precision threshold.

Another issue is termination of Snyder's algorithm. It turns out that this question is closely related to the problem of grazing intersections: the algorithm may fail to terminate in certain special situations.

Consider the elliptic paraboloid $z = x^2 - xy + y^2$. In a cube X in the first orthant with the corner at the origin, the surface is globally parameterizable in x and y, but not in any other pair of variables. However, on the bottom face $z = 0$, the partial derivatives with respect to x and to y are $2x - y$ and $2y - x$, and neither of them has a uniform sign in X. This box will never satisfy the condition of Lemma 2, and subdivision will produce a smaller box of the same type. Thus the algorithm will not terminate. Note that this surface is not in any way difficult to mesh; it presents no problems for the algorithm if the origin is inside some (small enough) cube: the surface will simply intersect the four vertical edges, and the mesh will consist of two triangles.

In both cases discussed above, the difficulty results from a special position of the grid relative to the surface: The surface is tangent to an edge (in the case of grazing intersections for the one-dimensional problem) or a face (in the case of non-termination) of a grid cube. In fact, one can show that this is the only source of difficulties: If no face of a cube that is created during the algorithm is tangent to the surface, the algorithm will terminate. Thus, a translation and rotation of the initial grid to a "generic" position guarantees termination: All cube faces are parallel to one of three given directions. A smooth surface has only finitely many points with a specified randomly chosen normal direction, the grid must be translated such that no grid plane will go through one of these critical points. This is ensured, for example, if the coordinates of the critical points are not multiplies of powers of 2, in the grid coordinate system.

In all likelihood, such a translation and rotation of the initial grid to a generic position should also remove the problem or grazing intersections with grid edges, but this has not been analyzed.

Exercise 1. If $\{ (x,y,z) \in X \mid f(x,y,z) = 0 \}$ is globally parameterizable in the parameters x and y, then the intersection curves on the vertical sides of X (parallel to the z-axis) are globally parameterizable in the parameters x or y, respectively.

Exercise 2. Construct an explicit isotopy between the original curve pieces and the polygonal approximating curve in the case of Lemma 2, part 1. (Figures 5.7a and 5.7b). Assume first that the intersections $x_1, x_2, \ldots$ with the boundary are given exactly. Then extend the construction to the case when the intersections are replaced by approximate values $x'_1, x'_2, \ldots$ etc. The only property that can be assumed is that they are ordered in the same way as the true values $x_1, x_2, \ldots$

Exercise 3. Extend the previous exercise to part 2 of Lemma 2. Assume that an isotopy from the true intersection curves to the polygonal approximation is given on each face of the cube X.

Exercise 4. Examine termination of Snyder's algorithm for a parabolic cylinder $(y-x)^2 - z = 0$ and for the hyperbolic paraboloids $x^2 - y^2 - z = 0$ and $xy - z = 0$, starting with eight unit cubes meeting at the origin. Assume that the range of f and its derivatives over any box (a) can be calculated exactly, or (b) is evaluated using interval arithmetic.

5.2.4 Small Normal Variation

Plantinga and Vegter [270] used a stronger condition than global parameterizability to guide the subdivision process, the Small Normal Variation condition:

$$\langle \nabla f(x_1), \nabla f(x_2) \rangle \geq 0, \text{ for all } x_1, x_2 \in X \qquad (5.2)$$

In other words, there is an upper bound of 90° on the angle between two gradient vectors, and in particular, between two normal vectors of the surface.

Exercises 5–7 below explore the relation to global parameterizability and Lemma 2. In particular, Small Normal Variation implies that the function is monotone in some coordinate direction, and therefore the surface (or curve) is globally parameterizable.

Condition (5.2) can be checked by interval arithmetic. We compute an interval representation $\Box\nabla f(X) = (\Box\partial f/\partial x, \Box\partial f/\partial y, \Box\partial f/\partial z)$ of the gradient and take the interval scalar product of this vector with itself. If the resulting interval does not contain 0, we have established the Small Normal Variation condition.

The algorithm starts with a given box and recursively subdivides it until, in every box X, the following *termination condition* is satisfied:

Either $f(x) \neq 0$ for all $x \in X$, or the Small Normal Variation condition (5.2) holds.

Both conditions are checked with interval arithmetic.

Theorem 1. *If the surface $S = \{ x \mid f(x) = 0 \}$ has no singular points and interval arithmetic converges, this subdivision procedure terminates.*

Proof. By the nonsingularity assumption and since f and ∇f are continuous, there is a positive minimum distance ε between the solution sets of $f(x) = 0$ and $\nabla f(x) = 0$ inside the starting box. This means that every box X which is smaller than ε has either $f(x) \neq 0$ or $\nabla f(x) \neq 0$ for all $x \in X$. Convergence implies that interval arithmetic will establish $f(x) \neq 0$ or $\|\nabla f(x)\|^2 \neq 0$, respectively, after finitely many subdivision steps. However, the interval computation of $\|\nabla f(x)\|^2 = \langle \nabla f(x), \nabla f(x) \rangle$ is identical to the calculation of $\langle \Box\nabla f(X), \Box\nabla f(X) \rangle$ that is used to check the Small Normal Variation condition.

One can see that the granularity of the subdivision adapts to the properties of the function f. In places where f and ∇f have a large variation and f is close to 0, the algorithm will have to subdivide the cubes a lot, but in regions where f is "well-behaved", not much refinement will be necessary.

We still have to show that the signs of f at the vertices of all boxes give sufficient information to construct a correct mesh. We first discuss the case of a curve in the plane, for illustration. For simplicity, let us ignore the case when f is zero at some box vertex. The algorithm will simply insert a vertex on every edge for which f has opposite signs at the endpoints. Now, the ambiguous case that caused so much headache in Figure 5.5 is excluded: If the signs alternate in the four corners, then f is neither monotone in the x-direction nor in the y-direction, contradicting the Small Normal Variation condition, see Figure 5.9a.

It may happen that the curve intersects an edge twice, and these intersections go unnoticed, as in Figure 5.9b. However, the Small Normal Variation condition ensures that the curve cannot escape too far before coming back, see Figure 5.9c.

Before trying to mesh the curve inside the boxes, the algorithm refines the subdivision until it becomes *balanced*: The size of two boxes that are adjacent via an edge differs at most by a factor of 2. As long as two adjacent boxes differ by a larger factor, the bigger box is subdivided into four boxes. (Boxes in which $f(x) \neq 0$ need not be subdivided, of course.) At this stage, we need not worry about the termination condition inside the boxes, because they are automatically fulfilled.

Finally, we insert a mesh vertex on every edge whose endpoints have different signs. We have to decide how to connect these vertices inside each square. Due to the balancing operation, there is only a small number of cases to analyze. It turns out that there can be zero, two, or four vertices on the boundary of a square. If there are two vertices, we simply connect them by a straight

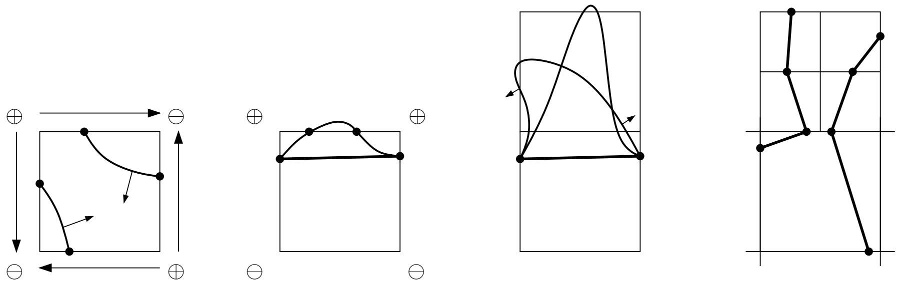


Fig. 5.9. (a) The ambiguous sign pattern cannot arise. The arrows along the sides indicate the direction in which f cannot be increasing. The little arrows indicate two normals of a hypothetical solution, which form an angle larger than $\pi/2$. (b) The two intersections with the upper edge are missed, but the straight segment between the endpoints is isotopic to the true curve. (c) In particular, the curve cannot leave the adjacent cube without violating the Small Normal Variation condition in the adjacent cube. Thus, the approximating segment is not only isotopic, it is even geometrically close. (d) The connections between endpoints in a square can be chosen by simple local rules.

line. If there are four vertices, two of them must lie on the same side, since the case of Figure 5.9a is excluded. We connect each of them to one of the other vertices, see Figure 5.9d for an example.

Theorem 2 ([270]). *The polygonal approximation constructed by the algorithm is isotopic to the curve S.* □

The algorithm works similarly for surfaces in three dimensions: The refinement step has the same termination criterion as in the plane. After balancing the subdivision, a vertex is inserted at every edge with endpoints of opposite signs. The analysis of the possible cases is now more involved. In particular, there can be an ambiguity on the *face* of a box without contradicting the Small Normal Variation condition, as in Figure 5.10a. This is because this condition does not carry over from a cube to a face: The gradient of the restricted function $f(x, y, z_{\max})$ on a face of the cube is the projection of the three-dimensional gradient vector ∇f, and two gradient vectors with angles less than π may form a larger angle after projection.

However, in this case one can insert the two edges arbitrarily in the ambiguous face. Each choice leads to a different mesh in the two boxes, see Figure 5.10b–c. But when the boxes are combined, the two choices lead to isotopic meshes.

Figure 5.10 is representative of the different cases that can arise. One just has to ensure that the choice of edges is done in a consistent manner for adjacent boxes, for example, by always favoring the edges which do not cross the diagonal in direction $(1, 1, 0)$ (Figure 5.10c) over the alternate choice, or by

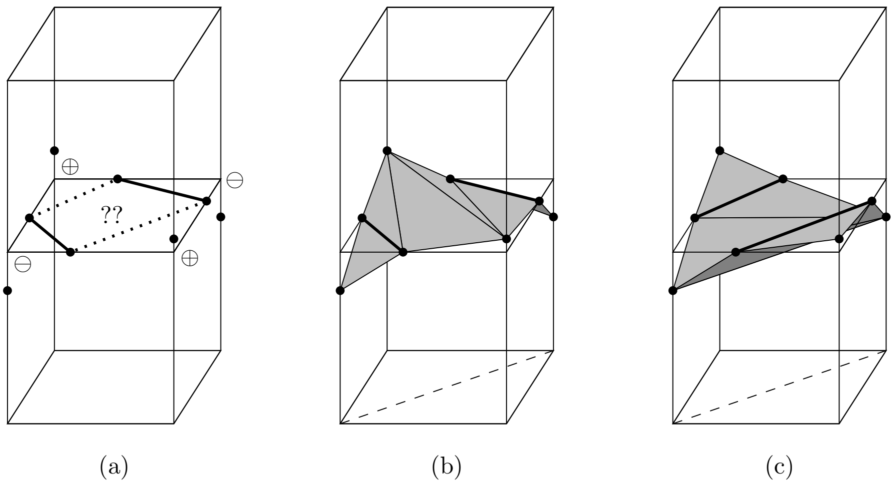


Fig. 5.10. (a) The ambiguous sign pattern can arise on a face of a cube. (b-c) The ambiguity can be resolved in two possible ways. The two resulting meshes cross the boundary face in different patterns, but they are isotopic to each other.

consulting the value of f in the middle of the square. The algorithm constructs a mesh that is isotopic to the surface S.

Comparison with Snyder's algorithm.

Looking at Figure 5.10, we can see why Snyder's algorithm of Section 5.2.3 has a harder time to terminate: it insists on topological correctness *within each single cube* separately. The example of Figure 5.10 shows that this is not necessary to get the correct topology in a global level. Snyder's algorithm may refine the grid to some unneeded precision when surface interacts with the grid in an unfavorable way.

Exercise 5. If all three partial derivatives are nonzero in a box X (and hence f is globally parameterizable in each pair of parameters out of x, y, and z), then f has Small Normal Variation.

Exercise 6. If f satisfies Small Normal Variation, then it monotone in x, y, or z, and in particular, it is globally parameterizable in some pair of parameters out of x, y, and z.

Exercise 7. Construct a function f with Small Normal Variation which is not well-behaved in the sense of Lemma 2, part 2. The function f should have the property that Small Normal Variation can be established by interval arithmetic.

Exercise 8. Construct an example of a function which is well-behaved with respect to a cube X, but is no longer well-behaved after subdividing X into eight equal subcubes. (For Small Normal Variation, this cannot happen: this condition carries over to all sub-boxes.)

Exercise 9. This exercise explores the properties of interval arithmetic for estimating the maximum angle between two gradient vectors in a box X. The algorithm of Section 5.2.4 terminates as soon as the angle between two different normals is less than 90°. Consequently, the geometric distance between the surface and the mesh can only be estimated very crudely; essentially it is proportional to the size of the box. If the angle bound is smaller, one might derive better bounds (see Research Problem 3, p. 227).

The standard way to estimate the angle is by the formula

$$\cos\alpha = \frac{\langle x, y\rangle}{\|x\| \cdot \|y\|}$$

where $x, y \in \Box\nabla f(X)$. If $\langle \Box\nabla f(X), \Box\nabla f(X)\rangle = [a, b]$ for some interval with $0 < a < b$, the standard interval arithmetic calculation leads to a bound of $\alpha \leq \arccos\frac{b}{a}$. Assuming that $\Box\nabla f(X) = ([a_1, b_1], [a_2, b_2], [a_3, b_3])$, can one derive a better bound on α by tackling the problem more directly? By how much can one improve the crude bound $\alpha \leq \arccos\frac{b}{a}$? Are there instances where the crude bound cannot be improved?

Exercise 10. Suppose that f satisfies the Small Normal Variation condition. Then there is an infinite circular double-cone C (like in Figure 5.1b) of opening angle $\alpha = 2\arcsin\sqrt{1/3} \approx 70°$ with the following property: When the apex of C is translated to any point x on the surface S, the two cones lie on different sides of S and intersect S only in x.
(Hint: α is the opening angle of the largest cone that fits into the first orthant. On the unit sphere S^2 of directions, a set of diameter π (measured in angles) is contained in a spherical disc of radius $(\pi - \alpha)/2$.)

Exercise 11. Prove that the sign pattern on the vertices shown in Figure 5.11 cannot arise, for a function with Small Normal Variation. (This pattern is configuration 13 in [270, Figure 5].)
(This exercise seems to require some geometric arguments which are not straightforward. The previous exercise may be useful.)

5.3 Delaunay Refinement Algorithms

The Restricted Delaunay Triangulation.

Given a set of points P and a surface S, the *Delaunay triangulation restricted by S* is formed by all faces of the three-dimensional Delaunay triangulation whose dual Voronoi faces intersect S. In particular, it consists of those triangles whose dual Voronoi edges intersect S. In the applications, the points of P will always lie on S.

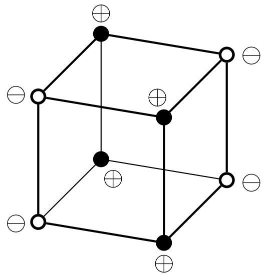


Fig. 5.11. Is this sign pattern possible when Small Normal Variation holds?

Generically, Voronoi vertices will not happen to lie on S; thus, the restricted Delaunay triangulation T contains no tetrahedra; it is at most two-dimensional. If P is a sufficiently good sample of S, then T will form a surface that is isotopic to S. A restricted Delaunay triangle xyz is characterized by the existence of an empty ball through the vertices xyz whose center p lies on the surface. We call this ball the *surface Delaunay ball.* It may happen that a vertex $p \in P$ is incident to no edge and no triangle of the restricted Delaunay triangulation T: then p must lie on a small component of S that is completely contained in the Voronoi cell $V(p)$. It can also happen that p is incident to some edges but to no triangle of T.

We will present two algorithms that use the restricted Delaunay triangulation as a mesh. They start with some initial point sample and add points until it is guaranteed that the restricted Delaunay triangulation forms a polyhedral surface that is isotopic to the given surface. The algorithms are adaptations of the greedy "farthest point" technique of Chew [86] in which the points that are added are centers of surface Delaunay balls.

The algorithms differ in the way how the correct topology is ensured, and they differ in the primitive operations that are used to obtain information about the surface. We will present an algorithm of Boissonnat and Oudot, which is based on the local feature size, and an algorithm of Cheng et al. that works towards establishing the so-called topological ball property.

5.3.1 Using the Local Feature Size

The generic form of Chew's mesh refinement algorithm is as follows, for the case of a surface without boundary.

> Select some starting sample $P \subset S$, and compute its restricted Delaunay triangulation T. If T contains a "bad" triangle xyz, insert the center of the surface Voronoi ball of this triangle into P, and update T.

Depending on the definition of "bad" triangle, the algorithm will give different results. The algorithm can also treat surfaces (and plane regions) with boundary, and in fact, this is the real challenge in the design of the algorithm. For simplicity, we will discuss only the case of smooth surfaces without boundary.

The *local feature size* of a point $x \in S$, denoted by $\mathrm{lfs}(x)$, is the distance from x to the closest point on the medial axis, see Figure 5.12. (See p. 109 in Section 2.7 for the definition of the medial axis; see also Section 6.2.2, pp. 242–245, for a more extensive discussion about the medial axis and the local feature size.)

In the case of a curve, the local feature size is small where the curve makes sharp bends, as in the region C of Figure 5.12. More generally, for a surface, the curvature radius corresponding to the maximum principal curvature (see Section 4.4.1) at x is an upper bound on $\mathrm{lfs}(x)$. The local feature size is also small when a different part of the curve comes close, as in the region around A of Figure 5.12. It is therefore a somewhat natural measure for specifying the necessary density of the mesh. There are however instances where the local feature size overestimates the density: for example, two parallel flat sheets of a surface that approach each other very closely have a small local feature size, but they can be meshed with few vertices. The local feature size is also related to the length of the fibers in a tubular neighborhood of the surface, see Lemma 1, Section 5.1.

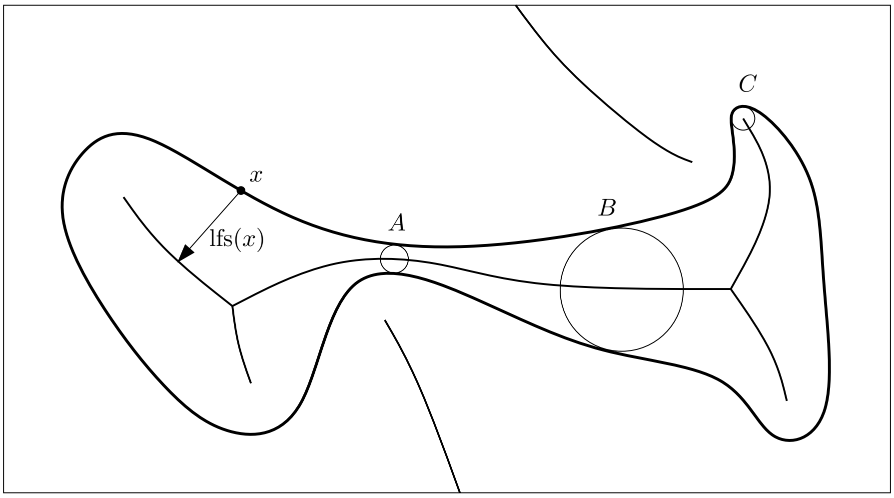


Fig. 5.12. A curve, its medial axis, and the local feature at a point x on the curve

The local feature size is nonzero if S is smooth at x. It is zero at edges or other singular points of S. The local feature size is a Lipschitz-continuous function with constant 1:

$$\mathrm{lfs}(x) - \mathrm{lfs}(y) \leq \|x - y\|$$

For a smooth compact surface, the local feature size is therefore bounded from below by a positive constant $\mathrm{lfs}_{\min} > 0$.

ε-Samples and Weak ε-Samples.

A fundamental concept is the notion of an *ε-sample* $P \subset S$ of a surface S, introduced by Amenta and Bern [20]. It is defined by the following condition: For every point $x \in S$, there is a point $p \in P$, such that $\|p - x\| \leq \varepsilon \cdot \mathrm{lfs}(x)$.

Since $\mathrm{lfs}(x)$ is difficult to obtain in practice, one replaces it by some other function ψ: A ψ-sample $P \subset S$ for a function $\psi\colon S \to \mathbb{R}^+$ is a subset with the following property: For every point $x \in S$, there is a point $p \in P$, such that $\|p - x\| \leq \psi(x)$. Thus, an ε-sample is the same as an $(\varepsilon \cdot \mathrm{lfs})$-sample. It will always be clear from the context which definition is meant.

Both of these notions are still difficult to check because the definition involves a condition for infinitely many points $x \in S$. The following concept relaxes this condition to a finite set of points.

> For every surface Delaunay ball with center x and radius r, $r \leq \varepsilon \cdot \mathrm{lfs}(x)$, or $r \leq \psi(x)$, respectively.

A point sample with this property is called a *weak ε-sample* (a weak ψ-sample, respectively). (Originally, this was called a *loose* ε-sample [60].)

The difference between ε-samples and weak ε-samples is not too big, however. It can be shown that every weak ε-sample, for small enough ε is also an ε'-sample, with $\varepsilon' = O(\varepsilon)$ [60, Theorem 1]. To exclude trivial counterexamples, one has to assume that, for each connected component C of S, the restricted Delaunay triangulation of P has a triangle with at least one vertex on C.

Theorem 3. *If $P \subset S$ is a weak ε-sample of S for $\varepsilon < 0.1$, and, for every connected component C of S, the restricted Delaunay triangulation T of P contains a triangle incident to a sample point on C, then T is ambient isotopic to S. The isotopy moves each point $x \in S$ by a distance at most $O(\varepsilon^2 \,\mathrm{lfs}(x))$.*

The theorem was first formulated for ε-samples (with the same bound of 0.1) by Amenta and Bern [20], see also Theorem 6 in Chapter 6 (p. 246) for a related theorem. The extension to weak ε-samples is given in Boissonnat [52].

We give a rough sketch of the proof, showing the geometric ideas, but omitting the calculations. Let us consider a ball tangent to S in some point $x \in S$. The definition of local feature size implies that such a ball contains no other point of S as long as its radius is smaller than $\mathrm{lfs}(x)$. Thus, when we draw the two balls of radius $\mathrm{lfs}(x)$ tangent to S, it follows that the the surface must pass between them, and hence, in the neighborhood of x, S must be more or less “flat”, see Figure 5.13. By the Lipschitz continuity of lfs, the same property (with slightly smaller balls) must hold for all other surface points in the vicinity of x. Consider three points $a, b, c \in S$ at distance $r = \varepsilon \cdot \mathrm{lfs}(x)$ from x. Then the normal of the plane through these points differs from the surface normal at x only by a small angle, which can be shown to be $O(\varepsilon)$.

We now apply this observation to the situation when x is a center of a surface Voronoi ball through the vertices a, b, c of a restricted Delaunay

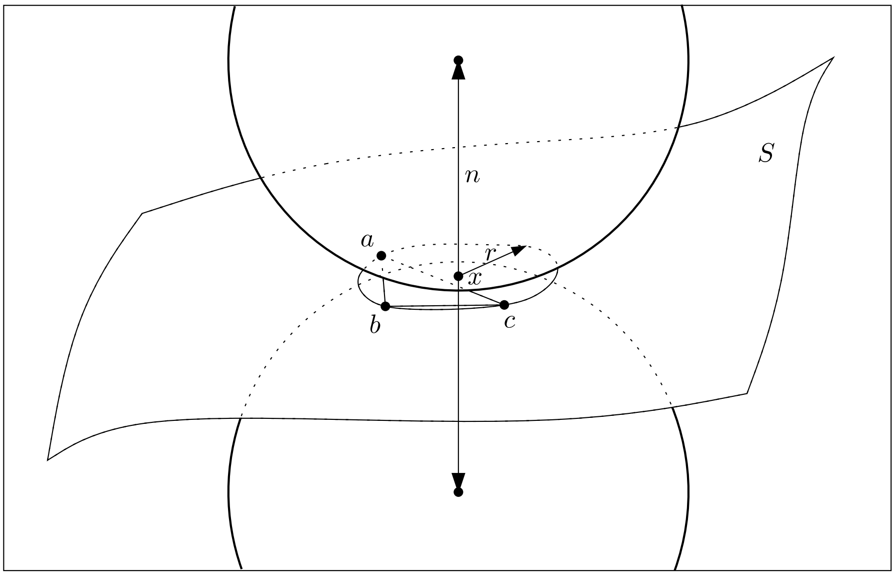


Fig. 5.13. The surface must squeeze between two tangent balls of radius $\mathrm{lfs}(x)$.

triangle Δ. Since the variation of surface normals is bounded, one can show that the maximum distance from a point of Δ to the closest point on S is $O(\varepsilon^2 \,\mathrm{lfs}(x))$. It follows that the projection π_S that maps every point of Δ to the closest point on S is injective: if we extend an open segment of length $\mathrm{lfs}(y)$ from every surface point y to both sides of S in the direction normal to S, these segments do not intersect, and they can be used as the fibers of a tubular neighborhood $\hat{S}$ of S. Each point of such a segment has y as its unique closest neighbor on S. For small enough ε, the triangle Δ is contained in $\hat{S}$. Thus, the mapping π_S defines an isotopy between Δ and a corresponding surface patch, as in Lemma 1 and Figure 5.3a.

One can show that two triangles of T that share an edge or a vertex have normals that differ by at most $O(\varepsilon)$, and the mapping π_S extends continuously across the edge or the vertex. It follows that the projection π_S is a homeomorphism that is invertible locally. (In topological terms, $\pi_S\colon T \to S$ is a covering map, if we can establish that it is surjective.) By assumption, on every component, there is at least one vertex contained in a triangle of T. This ensures that $\pi_S(T)$ contains that vertex, and since the mapping can be continued locally, it follows that every component of S is covered at least once. It is now still possible that some component is covered more than once by π_S. This would imply that some sample point $p \in P$ is covered more than once. However, one can show quite easily that no point of T except p itself has p as its closest neighbor on S.

With the help of Lemma 1, one can obtain the desired ambient isotopy between T and S. $\square$

The Delaunay refinement algorithm of Boissonnat and Oudot [60] applies this theorem to obtain a topologically correct mesh. It requires some function $\psi(x)$ for determining the necessary degree of refinement. The function ψ must be a lower estimate for the local feature size. To obtain a bound on the running time, it should be a Lipschitz function with constant 1:

$$\psi(x) - \psi(y) \leq \|x - y\|$$

The refinement algorithm is an instance of the general Delaunay refinement algorithm described on p. 201. A triangle with surface Delaunay ball of radius r centered at x is declared "bad" if $r > \psi(x)$.

Thus the main loop of the algorithm runs as follows. We have to intersect every Voronoi edge with the surface S. If there is an intersection point x, it is the center of a surface Delaunay ball, and it is the witness for a corresponding triangle in the restricted Delaunay triangulation. If the radius r of the ball is larger than $\psi(x)$, we insert x into P and update the Voronoi diagram. It may happen that a Voronoi edge intersects the surface more than once. Then we carry out the test for all intersection points. If follows from the arguments in the proof of Theorem 3 that at least one point x has $r > \phi(x)$ and can be inserted into P.

The following lemma helps to prove termination of this algorithm.

Lemma 3. *Let ψ be a Lipschitz function with $0 < \psi(x) \leq \mathrm{lfs}(x)$. Consider an algorithm which successively inserts points into a sample P with the property that every point p has distance at least $\psi(p)$ from all previously inserted points. Then the number of points is at most $O(H(\psi, S))$, with*

$$H(\psi, S) := \int_{x \in S} \frac{1}{\psi(x)^2}\, dx.$$

Proof. This is proved by a packing argument: Let L be the Lipschitz constant of ψ. First we prove that

$$\frac{1}{L+2}\psi(p) + \frac{1}{L+2}\psi(q) \leq \|p - q\| \tag{5.3}$$

for any two points $p, q \in P$. Assume that p was inserted before q. Then, by assumption, $\|p - q\| \geq \psi(q)$. By the Lipschitz property, we have

$$\psi(p) \leq \psi(q) + L \cdot \|p - q\| \leq (L + 1) \cdot \|p - q\|,$$

which implies $\psi(p) + \psi(q) \leq (L + 2) \cdot \|p - q\|$ and hence (5.3). It follows that we can draw disjoint disks D_p of radius $\psi(p)/(L+2)$ around all points $p \in P$. It is not difficult to show that the integral over these disks is bounded from below:

$$\int_{x \in D_p} \frac{1}{\psi(x)^2}\, dx \geq \Omega\Big(\frac{1}{(L+2)^2}\Big) \tag{5.4}$$

The argument is a follows. The area of D_p is $\Omega(\psi(p)/(L+2))^2$: it can be somewhat smaller than a plane disk of radius $\psi(p)/(L+2)$, since D_p lies on the curved surface S, but since the radius is bounded in terms of the local feature size, it cannot be smaller by more than a constant factor. By the Lipschitz property, the integrand cannot deviate too much from the value $1/\psi(p)^2$ at the center of the disk. Multiplying the integrand by the area of integration yields the lower bound (5.4). Since the disks D_p are disjoint, it follows that the number of points is $O((L+2)^2 H(\psi, S))$.

In a similar way, but using a covering argument instead of a packing argument, one can prove a lower bound on the size of a weak ε-sample (an hence on an ε-sample), cf. [142]:

Lemma 4. *Let ψ be a Lipschitz function with $0 < \psi(p) \leq \operatorname{lfs}(p)$. Any ψ-sample of S with respect to ψ must have at least $\Omega(H(\psi, S))$ points.* □

We still have to ensure that the restricted Delaunay triangulation contains a triangle on every component. A *seed triangle* is a triangle in the restricted Delaunay triangulation with a surface Delaunay ball of radius $r \leq \psi(x)/3$, where x is the center. Since this triangle is so small, one can show that the refinement algorithm will never insert a point in its surface Delaunay ball.

Lemma 5. *If the sample P contains a seed triangle, this triangle will remain in the restricted Delaunay triangulation.* □

The algorithm starts with sample P that consists of a seed triangle on every component of the surface. The lemma ensures that this triangle remains in the restricted Delaunay triangulation till the end, thus fulfilling the last assumption of Theorem 3. Since $\int 1/\operatorname{lfs}(x)^2\, dx \geq \Omega(1)$ on every closed surface, the extra seed points fall withing the asymptotic bound of Lemma 3.

The following theorem summarizes the conclusions from the above theorems and lemmas about the Delaunay refinement algorithm.

Theorem 4. *Let ψ be a Lipschitz-continuous function with Lipschitz constant 1 and $0 < \psi(x) \leq \operatorname{lfs}(x)$. Suppose that P is initialized with seed triangle on every connected component of S. Then the Delaunay refinement algorithm computes a sample of $\Theta(H(\psi, S))$ points. The resulting mesh is a weak ψ-sample. If $\psi(x) \leq \varepsilon \cdot \operatorname{lfs}(x)$ for $\varepsilon \leq 0.1$, it is ambient isotopic to S. The isotopy maps every point to a point of distance at most $O(\varepsilon^2 \operatorname{lfs}_{max})$.* □

Geometry Improvement.

After obtaining the correct topology, one can improve the geometry of the mesh by eliminating triangles with small angles (with bad "aspect ratio").

A triangle is declared "bad" if the minimum angle is below some bound. This can be done concurrently with the size criterion specified by ψ. If the bound is not too strict (less than $\pi/6$), termination of the algorithm is guaranteed. For further details, we refer to [60].

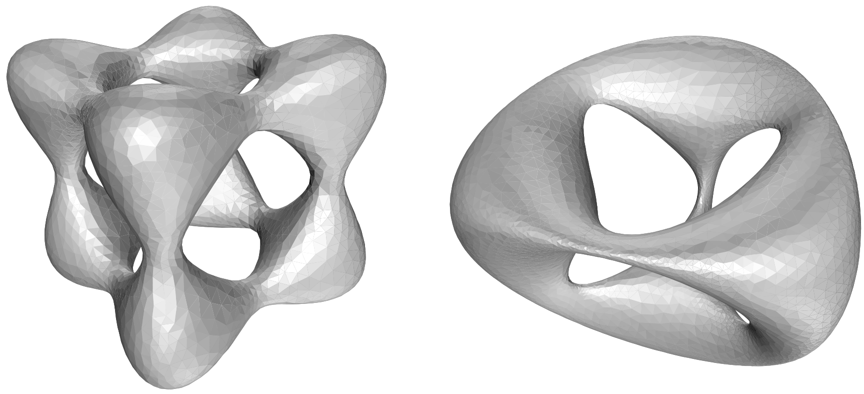


Fig. 5.14. Two meshes constructed by the Delaunay refinement algorithm of Boissonnat and Oudot [60]

Primitive Operations.

The algorithm needs some lower estimate ψ on the local feature size as external input. In some applications, such information can be known in advance. As a heuristic without correctness guarantee, one can also use the distance to the poles of the Voronoi diagram (roughly, the largest distance to a Voronoi vertex in the Voronoi cell of each sample point on each side of the surface (see the definition on p. 234 in Section 6.2.1). This is a suitable substitute for the local feature size, once a reasonably fine starting mesh has been obtained (Theorem 7 in Chapter 6, p. 247). The distance to the closest pole should be scaled down by a constant factor, to obtain a lower bound on the local feature size with some safety margin.

The essential primitive operation during the algorithm is the intersection of the surface with a Voronoi edge. If the surface is given as a solution set of an equation, this can be written as an equation in one variable. Depending on the type of the equation, it can be solved exactly with the techniques of Chapter 3 or numerically with a bisection method or interval arithmetic.

If a Voronoi edge happens to be tangent to the surface or have a "grazing intersection", this causes numerical or algebraic difficulties. This is a drawback that is shared by all algorithms that compute intersections: They may have numerical or algebraic problems that are not inherent in the problem itself, but are caused by the added "scaffolding" structure that the algorithm puts around the surface (a Voronoi diagram, or a grid of cubes like in the previous section).

On the other hand, once the mesh is sufficiently fine, Voronoi edges that intersect the surface will do so transversally, and the smaller the mesh becomes, the more perpendicular and numerically well-behaved will be the intersection.

Also, in connection with appropriate conditions like Small Normal Variation, one can avoid attempts to compute difficult intersections.

Besides the estimate ψ and the computation of intersection points, the algorithm requires only a small initial seed triangle on every component. One can construct examples with some tiny components that are missed completely unless they are specified initially. For instances which are not so complicated, it is often sufficient to intersect a few random lines with the surface to provide a starting sample. An alternative approach is to look for critical points on the surface in some arbitrary direction d, by solving for points where ∇f is parallel to d. This will insert at least two seed vertices on every component of S. Then one can test the incidence criterion of Theorem 3 at the end, by checking whether the seed points are incident to some triangles of the restricted Delaunay triangulation, and if necessary, restart the algorithm after a few additional refinements. Figure 5.14 shows meshes that were obtained by this algorithm.

Theoretically, the algorithm is restricted to smooth surfaces, because the local feature size is 0 at non-smooth points. In practice, however, the algorithm works reasonably well also on polyhedral surfaces as long as the edges are not too sharp, and even for surfaces with singularities, see [60] for some examples.

Exercise 12. Assume we know a Lipschitz continuous function $\phi(x)$ which is a lower estimate on the local feature size $\mathrm{lfs}(x)$, with $\phi(x) \geq \phi_{\min} > 0$. Refine the analysis leading to Theorem 4 to show that an approximating mesh with Hausdorff error $O(D)$, for $D \leq \phi_{\min}$, can be obtained by running the Delaunay refinement of Section 5.3 with threshold function $\psi(x) := \sqrt{D\phi(x)}$. Show that ψ satisfies the assumptions of Theorem 4.

Exercise 13. 1. Take a set P of at least 4 points from a sphere S, but otherwise in general position. Show that the Delaunay triangulation of P restricted by S is homeomorphic to S.

2. Find a point set P which lies very close to the sphere S, whose restricted Delaunay triangulation is not homeomorphic to S.

5.3.2 Using Critical Points

Another meshing algorithm of Cheng, Dey, Ramos and Ray [85] uses a different criterion for topological correctness of the restricted Delaunay triangulation. We say that a point set P on a surface S satisfies the *topological ball property* if every Voronoi face F of dimension k in the Voronoi diagram of P intersects S in a closed topological $(k-1)$-ball or in the empty set, for every $k \geq 0$. (For example, the non-empty intersection of a 2-dimensional Voronoi face F with the surface must be a curve segment.) Moreover, the intersection must be nondegenerate, in the following sense: the (relative) boundary of the intersection must coincide with the intersection of S with the boundary ∂F. (For example, it is forbidden that an interior point of the curve segment in the above example touches the boundary of F.)

Theorem 5 ([132]). *Let P be a non-empty finite point set. The Delaunay triangulation of P restricted by a surface S is homeomorphic to S if the topological ball property is satisfied.*

In the paper [132], where this property was introduced, it was called the *closed ball property*, see also Theorem 1 in Chapter 6 (p. 239).

The topological ball property is by itself not restricted to smooth surfaces. However, the termination proof of the algorithm and the methods that are used to establish the topological ball property are restricted to surfaces without singularities, defined by a smooth function f.

Instead of using some quantitative argument as in Section 5.3.1, the algorithm that we are going to describe tries to establish the topological ball property directly. For this purpose, we need some computable conditions which imply the topological ball property for a given surface $f(x) = 0$. These conditions will be given in the sequel. In each case, testing the condition involves finding a point on the surface with certain criticality properties, which reduces to the task of solving a system of equations involving f and its derivatives. There will be three equations in three unknowns, and thus generically a zero-dimensional set of solutions. If f is a polynomial, the number of solutions will be finite and the techniques of Chapter 3 can be applied to solve the problem exactly, thereby leading to a provably correct algorithm. In this section, however, we will restrict ourselves to the task of setting up the geometric and topological conditions and deriving the corresponding systems of equations.

The following lemma gives a sufficient condition for a two-dimensional surface patch to be isotopic to a disk.

Lemma 6 (Silhouette Lemma). *Let $M \subset S$ be a connected, compact 2-manifold whose boundary is a single cycle, and let $d \in \mathbb{S}^2$ be an arbitrary direction. If M contains no point whose normal is perpendicular to d, then it is a topological disk.*

See Exercise 14 for a proof. The set of points whose normal is perpendicular to d is called the *silhouette* of the surface S in direction d. Algebraically, it is defined by the condition $\langle \nabla f, d \rangle = 0$ (and $f = 0$), which specifies a one-dimensional family of solutions. Generically, it is a set of smooth curves.

The algorithm performs a sequence of four tests to establish the topological ball property. If any test fails, a new sample point is identified. This point is inserted into the sample, and the restricted Delaunay triangulation is updated. We assume throughout that no Voronoi vertex lies on S. (If this case should happen, it can be resolved by perturbing the set P slightly, either by a small random amount, or conceptually with symbolic perturbation [130, 327]. In this way, one can also ensure that P contains no 5 co-spherical points, and thus the Delaunay triangulation is indeed a proper triangulation.)

(1) For a Voronoi edge e, the topological ball property amounts to requiring that e intersects S in at most one point (but not in one of its endpoints).

Thus, we look at each Voronoi edge e in turn, and intersect e with S. If there is more than one intersection, then insert the intersection point p^*

which is farthest from p, where p is the sample point in any Voronoi cell to which e belongs. (The choice of p has no influence on the distances to p^*.) The possibility that the intersection point is an endpoint of e, and thus a Voronoi vertex, has been excluded above.

(2) The second check is some topological consistency check for the restricted Delaunay triangulation T. (The purpose of this test will become apparent later.) We check if T is a two-dimensional manifold: each edge must be shared by exactly two triangles, and the triangles incident to each vertex p must form a single cycle around p. If this holds, these triangles form a *topological disk* around p. If the test fails for some vertex p or for some edge pq, we intersect all edges of the Voronoi cell of p with S and insert into P the intersection point p^* which is farthest from p.

(3) Next, we look at each Voronoi facet F. The topological ball property requires that F intersects S in a single curve connecting two boundary points. In general, the intersection $F \cap S$ could consist of several closed curves or open curves that end at the boundary of F.

Suppose that F is the intersection of the Voronoi cells $V(p)$ and $V(q)$. First, we exclude the possibility of some closed loop. If there is a closed loop, then it must have an extreme point in some direction. Thus, we choose some arbitrary direction d in F and compute the points of $F \cap S$ with tangent direction d. Algebraically, this amounts to finding a point x on $F \cap S$ where the surface normal ∇f is perpendicular to d in $\mathbb{R}^3$. We take the line l in F that is perpendicular to d and goes through x, and we intersect l it with S. If x is part of a cycle, then l must intersect S in a point x^* of F different from x. Among all such points x^*, we choose the one with largest distance from p and insert it into P. If necessary, we have to repeat this test for each critical point x which lies in F.

If no point p^* is found in this way, we have excluded the possibility of a closed cycle in $F \cap S$. $F \cap S$ might still consist of more than one topological interval. However, since no Voronoi edge intersects S in more than one point, S must intersect more than two Voronoi edges of F. This means the dual Delaunay edge pq of F is incident to more that two triangles in the restricted Delaunay triangulation, violating the topological disk condition for p and q that was established in (2).

Thus, we have established the topological ball property for all Voronoi edges and for all two-dimensional Voronoi faces.

(4) Finally, we have to check that the surface forms a topological disk in each Voronoi cell $V(p)$. Because of the test (2), we already know that S intersects the boundary of $V(p)$ in a single cycle. However S could still contain a handle or some more involved topological structure inside $V(p)$. To exclude this possibility, we use the Silhouette Lemma (Lemma 6). We take the normal direction $d = \nabla f(p)$ in the point p and look at the silhouette in that direction, defined as the set of points whose normal is perpendicular to d. Arguing in the same was as in (3), a silhouette can either form a cycle inside $V(p)$, or it must intersect the boundary. We choose an arbitrary direction $d' \perp d$, and we look

for a critical point in direction d' on the silhouette curve; we also intersect the silhouette with all boundary facets of $V(p)$. If we find any point in $V(p)$ in this way, we insert it into the sample. (It is not necessary to choose the point farthest from p.) Otherwise, we know that no silhouette exists in $V(p)$, and the surface must be a topological disk.

For intersection of the silhouette with a plane, we have the two equations characterizing the silhouette:

$$f(x) = 0,\ \langle \nabla f(x), d \rangle = 0,$$

and the plane equation, leading to a 0-dimensional solution set. The third equation for defining the critical point in direction d' can be worked out as the condition that the three vectors $H_f(x) \cdot d$, d', and the gradient $\nabla f(x)$ should be coplanar:

$$\det(H_f(x)d, d', \nabla f(x)) = 0, \tag{5.5}$$

where $H_f(x)$ is the Hessian matrix at x.

If we start the algorithm with an empty sample set $P = \emptyset$, the algorithm will directly proceed to step (4) to find a critical point on some silhouette (in an arbitrary direction). Any component of S without sample points has a silhouette with critical points (for any directions d and d') and will thus eventually be detected. Of course, any other method to find some seed points on the surface is also appropriate for starting the algorithm. Cheng et al. [85] propose to initialize P with all critical points of S in the vertical direction.

Here is a summary of the *Topology Refinement Algorithm* to obtain a topologically correct mesh.

> Check the conditions given in (1)–(4), in this order. If any condition fails, it defines a new point. Insert it into P and update the Voronoi diagram. Continue the process till no new point is inserted.

As discussed above, this process ensures that the topological ball property holds when the algorithm terminates, and therefore, the restricted Delaunay triangulation is a homeomorphic mesh for S. It is however, not known whether the mesh is actually isotopic to S, see Research Problem 4 on p. 228.

The proof of termination assumes that f is smooth and defines a smooth compact surface S, and it is based on the local feature size. One can show that a new point p^* that is inserted has distance at least $k \cdot \mathrm{lfs}(p)$ from all previous points, for $k = 0.06$. Therefore, Lemma 3 applies and gives the explicit bound $|P| = O(H(S, \mathrm{lfs}))$. (Note that the algorithm does not have to know the local feature size.)

The tests in (1) and (2) insert centers of surface Delaunay balls and fall within the framework of Delaunay refinement. The tests in (3) and (4) are different and require more involved computations.

The above discussion has ignored a few degenerate cases which do not happen in practice, but which need to be excluded nevertheless, to obtain a certified method. For example, in step (3), the intersection $S \cap F$ with a

Voronoi face might be tangent to a third edge between its two endpoints. This would violate the topological ball property for F because the boundary of the intersection $S \cap F$ does not coincide with the intersection $S \cap \partial F$ with the boundary. However, this situation would lead to an extra triangle incident to the Delaunay edge pq and be detected in step (2). It is possible that the intersection $S \cap F$ is formed by S being tangent to F from one side, either in a point, in a curve, or even in a two-dimensional area. This would be in violation of the topological ball property for one of the incident cells $V(p)$ or $V(q)$. However, such a point of tangency is detected by the geometric condition of having a surface normal ∇f perpendicular to d, in step (3), and can be inserted into the sample.

Improving the Geometry.

The mesh obtained in this way has the correct topology, but it may still be a very rough approximation. As in the Delaunay refinement algorithm of the previous section, one can refine the mesh in order meet various geometric quality criteria. One can eliminate triangles with small angles or edges with sharp face angles between adjacent triangles. This refinement may destroy the correct topology, hence it is necessary to go back to steps (1)–(4) and then come back for another round of geometry improvement, and so on, until the process stabilizes. It can be shown that the asymptotic upper bound $|P| = O(H(S, \psi))$ from Lemma 3 still applies (with a different constant), and hence, the method terminates, for smooth surfaces.

Primitive Operations.

The only primitive operation is the solution of systems of equations involving the function f and its derivatives, and various plane or line equations that come from the Voronoi diagram. These equations have generically a zero-dimensional set of solutions, and thus are amenable to techniques of Chapter 3 and to software such as SYNAPS, in the case of an algebraic function f. In contrast to the Delaunay refinement algorithm of the previous section, the equations go beyond intersecting a line segment with the surface: they involve derivatives of f up to second order.

On the other hand, no a-priori knowledge or estimate about the local feature size or the location of different connected components is required by this algorithm.

Polyhedral Input.

This Delaunay refinement algorithm has been extended to the case when the input is already a polygonal surface S whose dihedral angles are not too sharp (bigger than π) [121]. For example, S can come from a sufficiently fine sample of a smooth surface. Actually this falls into the area of remeshing, which is not within the scope of this chapter, but the algorithm is based on the same ideas as for the smooth case. It is is implemented in SurfRemesh software by Tamal K Dey and Tathagata Ray[4]. Of course, for a polyhedral input surface, some simplifications are possible. The algorithm refines the surface with Delaunay triangles that have bounded aspect ratio, and it achieves a small approximation error.

Exercise 14. Prove the Silhouette Lemma (Lemma 6) by projecting M in direction d onto a plane.

Exercise 15. Prove that the points which are critical in direction d' on the silhouette in direction d are defined by (5.5).

5.4 A Sweep Algorithm

Mourrain and Técourt [247, 309] have proposed a meshing algorithm for algebraic surfaces that is based on sweeping a vertical plane over the surface. We have already seen in previous sections that critical points play a crucial role in determining the topological structure of a surface. Accordingly, the algorithm uses certain critical points to guide the sweep. In contrast to previous methods discussed in this chapter, this algorithm makes no smoothness or regularity assumptions about the input surface (other than those which follow from being an algebraic surface). The algorithm works for surfaces with self-intersections, fold lines, or other singularities. On the other hand, it makes no guarantees about the geometric accuracy of the mesh, and it cannot be extended in a straightforward way to provide a more accurate mesh.

We first give a rough overview, concentrating on the geometric ideas. We will then discuss the primitive geometric operations that are necessary. In the case of algebraic surfaces, these operations can be carried out exactly.

The algorithm cuts the surface M into vertical slabs by a series of planes parallel to the y-z-plane in such a way that, between two successive planes, the surface has a "trivial" structure that can be constructed easily. Figure 5.15 shows a mesh for a torus shape that is constructed in this way. Before meshing the surface between the planes, we first have to construct the intersection of M with each plane. This is the meshing problem for a plane curve. The algorithm solves this problem in an analogous way, cutting it by vertical lines through certain critical points, finding the intersections with these lines, and connecting the points between them. Thus, the algorithm proceeds by induction on the dimension.

We first discuss the problem of finding a topologically correct mesh for a planar curve C, in the form of a planar straight-line graph which is ambient isotopic to C. This algorithm is due to González-Vega and Necula [176]. We will assume throughout that C and M contains no straight line segment which is vertical. (This can most conveniently be achieved by a sufficiently random

[4] http://www.cse.ohio-state.edu/~tamaldey/surfremesh.html

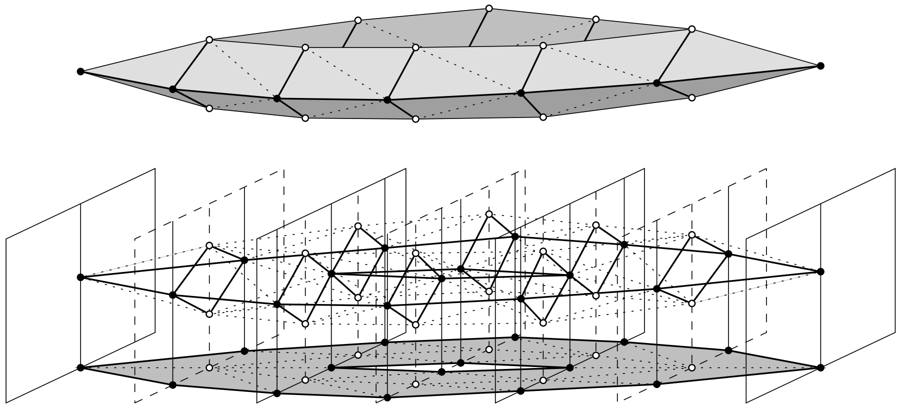


Fig. 5.15. A triangular mesh for a torus constructed by connecting a sequence of vertical cuts. The mesh is shown both as a surface and as a transparent skeleton. The vertices and edges are marked according to the conventions of Figures 5.16 and 5.17 below.

rotation of the coordinate system; alternatively, one can handle vertical parts explicitly.) For simplicity of exposition, we will also assume that the curve or surface is bounded. The algorithm can also deal with curves and surfaces that are clipped by a bounding box or sphere, and it can be extended to handle algebraic surfaces with infinite parts.

5.4.1 Meshing a Curve

We want to construct an embedded planar straight-line graph which is isotopic to the solution set of the equation $f(x, y) = 0$. First we compute the x-coordinates of all potential "critical points": points where the curve has a vertical tangent, or where it crosses itself or has some other singularity. These points are characterized by the equation

$$f_y(x, y) = 0, \quad f(x, y) = 0. \tag{5.6}$$

Let X be the finite set of x-coordinates of the solutions of this system of equations. In the x-interval between two successive critical points, the solution consists of a constant number of x-monotone curves, since $f_y(x, y) \neq 0$, and by the implicit function theorem, the curve can be locally written as a graph of a function $y = h(x)$, see Figure 5.16. To make life easier, we insert an intermediate x-value between each pair of successive values of X. Now, for each value x in this enlarged set, we intersect the curve with the vertical line at x. (This is nothing but the *one-dimensional* version of the meshing problem.) We get a discrete set of points on each vertical line. By construction, the intersection equation has only simple roots at each intermediate value x, whereas it has at least one multiple root at each value of X.

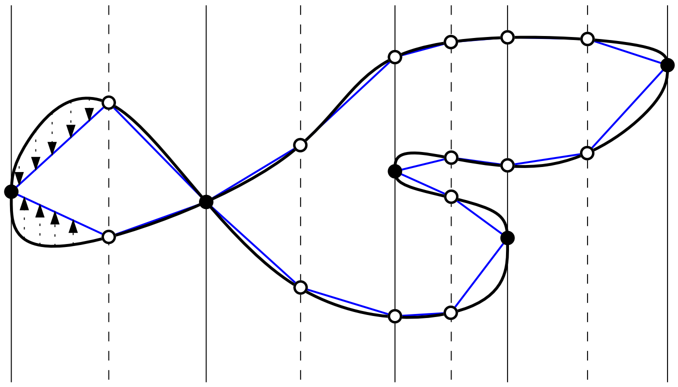


Fig. 5.16. A plane curve and its critical points (full circles). The vertical lines through the critical points and the additional lines between the critical points (dotted) define the polygonal mesh. In the left part, the isotopy that maps the curve to the meshed curve is indicated.

Now that we have the points on each vertical line, we need to figure out how to connect them. For each point, we need to know how many parts of the curve emanate on each side. For a simple root (including all points on the intermediate lines) there is one piece emanating on each side. For the critical points (multiple roots), we have to use algebraic tools to find this information. This is described in detail in Sections 3.6.1 and 3.6.2. In each vertical slab, we now scan the points from bottom to top and connect them simply by a straight segment, taking into account the multiplicity of emanating arcs, see Figure 5.16. By adding the intermediate lines, we have ensured that these line segments are disjoint. (In the example of Figure 5.16 we could not connect points on the first and third line directly by straight segments if the second line were not there.)

The resulting graph is ambient isotopic to the given curve. We can even explicitly construct an isotopy of a very restricted kind, because all we have to do is to move points vertically upward or downward from each curve piece to the corresponding segment, as indicated in Figure 5.16. The points on the vertical lines remain fixed.

Finally, we summarize the geometric primitives that we need to provide in order to make this algorithm work:

- We must be able to find all critical points by solving (5.6).
- We must find the intersection of the curve with a vertical line. (Some of these lines are defined by going through a critical point.)
- For each point on one of the lines, we must be able to determine how many branches of the curve emanate to the left and to the right.

For the case of an algebraic curve, Sections 3.6.1 and 3.6.2 describe how this can be done in an exact manner.

5.4.2 Meshing a Surface

Now that we know how to mesh a curve, let us proceed to a surface $f(x, y, z) = 0$ in space. Our goal is to find uniform regions in the x-y-plane where the surface can be regarded as a family of a constant number of function graphs of the form $z = h(x, y)$. We therefore have to cut the surface at the points of self-intersection and the points that have vertical tangents (the *silhouette*). These points form the *polar variety* C, which is defined by

$$f_z(x, y, z) = 0, \quad f(x, y, z) = 0. \tag{5.7}$$

This system of two equations in three variables will in general have a one-dimensional solution space, consisting of curves on the surface M. (The points of self-intersection, as well as other sorts of critical surface points, satisfy in fact the stronger condition $f_x(x, y, z) = f_y(x, y, z) = f_z(x, y, z) = 0$, and hence they are included in the solution of (5.7).) When we cut the surface at the polar variety, we obtain x-y-monotone surface patches that can be parameterized in x and y.

As discussed in the beginning, we partition the surface into vertical slabs by planes perpendicular to the x-axis, in such a way that, between two sections, the topology of the surface "does not change". The points at which we cut will be called *slab points*. These points include all x-critical points of the polar variety, as well as all points where the projection of the polar variety on the x-y-plane intersects itself.

The system of equations that characterize the x-critical points has been given in (5.5) for two general directions d and d'. In our case, d is the z-direction and d' is the x-direction. Thus, the critical points are given by the system

$$(f_z \cdot f_{yz} - f_y \cdot f_{zz})(x, y, z) = 0, \quad f_z(x, y, z) = 0, \quad f(x, y, z) = 0. \tag{5.8}$$

This includes the x-critical points of the surface itself, i. e., the points where x has a local extremum: these points have a tangent plane perpendicular to the x-axis, and a fortiori a vertical tangent line, and therefore they lie on the silhouette. There are cases when the system (5.8) does not have a zero-dimensional solution set, and therefore it cannot be used to define slab points. (The example of Figure 5.20 below is an instance of this.) In these cases, one must modify the system to obtain a finite set of slab points, as described in [247, 309].

The points where the vertical projection of the polar variety onto the x-y-plane crosses itself are the points (x, y) for which (5.7) has more than one solution z. For a polynomial f, these points can be found by computing the resultant of the polynomials in (5.7), see Chapter 3 for details. A slab point (x, y) of this type will be called a *multiple slab point* if more than two curves of the polar variety pass through the vertical line at (x, y) without going through the same point in space.

We make the following important *nondegeneracy assumption*:

> There is a finite set of slab points, there are no multiple slab points, and no two slab points have the same x-coordinate.

This assumption excludes for example a surface which consists of two equal spheres vertically above each other. The two silhouettes (equators) would coincide in the projection. It also excludes a torus with a horizontal axis, or a vertical cylinder (for which the polar variety would be two-dimensional), for the same reason. Such cases are very special, and they can easily be avoided by a random transformation of the coordinate system. Still, any number of curves of the polar variety may go through the same point in space, and in particular, the surface can have self-intersections of arbitrary order. Thus, the nondegeneracy assumption is no restriction on the generality of the surface M.

Now we proceed as in the planar case. We take the x-coordinates of all slab points, we add intermediate "regular" x-values between them, and we compute all vertical cross-sections at these values, using the algorithm of Section 5.4.1 for plane curve meshing. Note that the intersections of the polar variety with the vertical planes become critical points for the two-dimensional meshing problem. This can be seen by comparing (5.7) with (5.6), noting that the z-coordinate of our three-dimensional problem becomes the y direction of the two-dimensional problem. The algorithm produces in each vertical plane a planar graph that is ambient isotopic to the cross-section. The isotopy has only deformed the curves vertically.

Now, as we look at a slab from the top, the polar variety will form x-monotone non-crossing curves from one plane to the next, as in Figure 5.17a. The strip between the boundaries is divided into triangular and quadrangular *regions* that are bounded by two curves of the polar variety C, and one or two straight pieces from the boundary walls. (In addition, there are the unbounded regions at the extremes, but by the boundedness assumption on the surface M, there cannot be any part of M in these areas.) We must find the correct assignment between the critical points on the two planes that have to be connected by the polar variety in the projection. By construction, one of the planes is an "intermediate" plane without a slab point; so each critical point is incident to one piece of C. By assumption, the other plane contains at most one slab point, and we know which one it is. We can therefore find the correct connections by assigning critical points in a one-to-one manner, with the projected slab point absorbing the difference between the number of critical points on the two sides. In the mesh, these pieces of C will be replaced by straight line segments, see Figure 5.17b. Figure 5.17 shows an example where the critical points in the regular cross-section outnumber the critical points on the other side, and thus s has to accept two connections. A different case arises if s is a local x-minimum of the surface, or in the situation of Figure 5.19: s receives no connections at all from the left.

Now we have to construct the surface pieces. *Above each region of the projected picture, the surface M consists of a constant number of x-y-monotone surface patches.* The number of patches is determined by any point in the x-y-

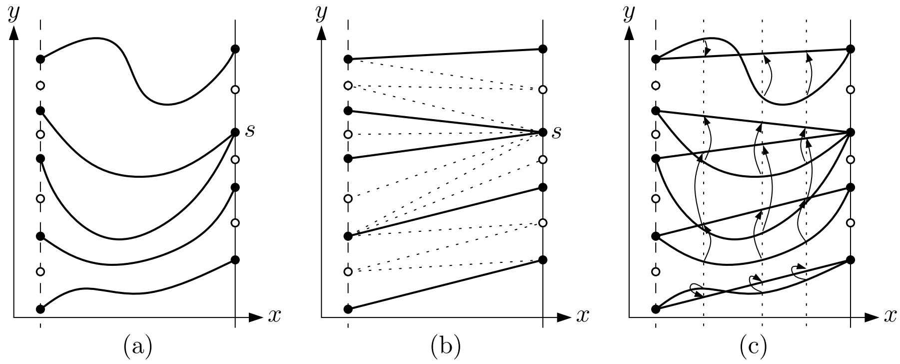


Fig. 5.17. (a) Vertical projection of the polar variety between two planes. The critical points in each vertical plane are marked by full circles. The plane on the right contains a slab point s, the plane on the left is a "regular" cross-section. (b) Vertical projection of the resulting mesh. (c) The horizontal component of the isotopy.

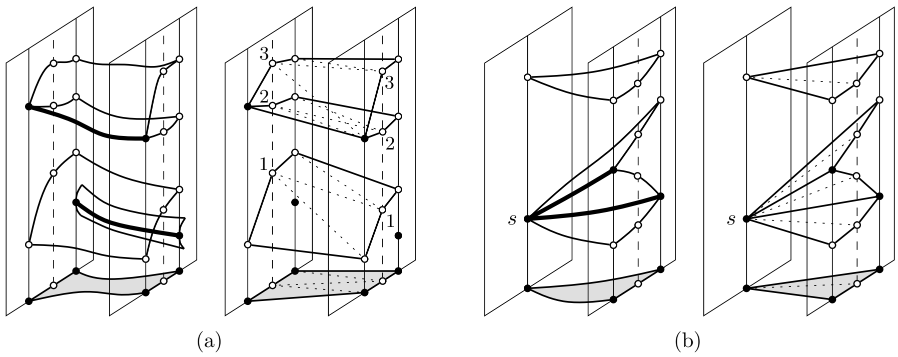


Fig. 5.18. Connecting a region in several layers: (a) A simple situation with three layers above the projection and a one-to-one assignment between two successive cross-sections. The pieces of the polar variety are shown in thick lines. The figure on the left includes a piece of the surface from an adjacent region, to show how the segment projected polar variety in the projection arises. This part of the surface will be meshed as part of the adjacent region. (b) Four layers over a triangular region. Three parts of the surface intersect in the point s, which is therefore a slab point.

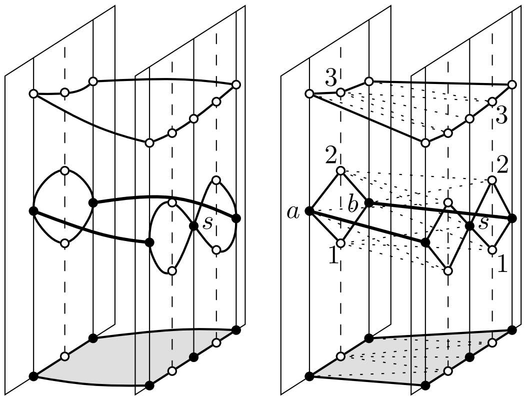


Fig. 5.19. Connecting a quadrilateral region in several layers: The triangulation of the region must avoid to connect the critical point s with the boundary points a and b on the other side, because otherwise the first and second layer of the surface would touch along this diagonal. This situation occurs for example in the second slab for the torus of Figure 5.15.

plane which does not lie on the projection of C, for example, at the "intermediate" vertical lines from the cross-sections (the open circles in Figure 5.17b). It is now a straightforward matter to connect the cross-sections above each region. We choose some triangulation of the region (as indicated in Figure 5.17b) and use this triangulation to connect the pieces in all layers. Over a quadrilateral region, one can simply connect the curve pieces in the two cross-sections one by one from bottom to top, see Figure 5.18a. The situation can be more involved over a triangular region, see Figure 5.18b for an example. However, there is always a unique way to connect the cross-sections, if one takes into account the information from adjacent regions. Figure 5.19 shows a situation where the triangulation of the region cannot be chosen arbitrarily. There are degenerate situations which are more complicated, for example when more than three surface patches intersect in the same point, or when an x-minimal point on a self-intersection curve has at the same time a vertical tangent plane. Since we know that there is only one slab point on every vertical line and we know which point it is, these cases can also be resolved.

It is clear that the resulting triangles do not cross, and hence form a topologically correct mesh of the surface above each region. One can even write down the ambient isotopy between the surface and the mesh: In a first step, one transforms only the y coordinates to deform Figure 5.17a into Figure 5.17b, see Figure 5.17c:

$$(x, y, z) \mapsto (x, g(x, y), z),$$

for some continuous function $g\colon [x_1, x_2] \times \mathbb{R} \to \mathbb{R}$ that is monotone in y for each value of x, similarly to the two-dimensional case. More explicitly, g is defined for all points on the projection of C by the condition that they must be mapped to the corresponding straight line segments. Between these points, g is extended by linear interpolation in y. For $x = x_1$ and $x = x_2$, we have $g(x, y) = y$: the two boundary planes are left unchanged.

In a second step, we only have to deform the surfaces vertically. Note that this coincides with the isotopy that is defined for each vertical slab by the planar curve meshing procedure. Thus, by concatenating the two isotopies (first in the y-direction and then in the z-direction) and gluing them together across all slabs, we get the isotopy between M and the mesh.

Theorem 6. *The mesh constructed by this algorithm is ambient isotopic to the surface M.*

For an algebraic surface, one can analyze the number of solutions that the equations arising in the course of the solution might have [247, 309]:

Theorem 7. *For an algebraic surface of degree d, the algorithm constructs a mesh with at most $O(d^7)$ vertices.*

Note that the solution set M of the equation $f(x, y, z) = 0$ may not be a surface at all. Of course, without any smoothness requirements whatsoever, M could be some "wild" set. But even when f is a polynomial (the case of an *algebraic "surface"*), M can be a space curve or a set of isolated points. It can even be a mixture of parts of different dimensions, for example the union of a sphere and a line through the sphere, plus a few isolated points. The algorithm can be extended to handle these cases.

In particular, if the set M contains a space curve C, then all points on that curve will automatically form part of the polar variety. Figures 5.20–5.21 show an example of a sphere and a line that are defined by the equation

$$(x^2 + y^2 + z^2 - 1)\left((x + z)^2 + (y + z)^2\right) = 0.$$

In such cases, the connection between two vertical sections will contain edges with no incident triangles.

In fact, when the curve meshing problem (Section 5.4.1) is used as a subroutine for the surface meshing problem, degenerate cases of this type *will* occur. For example, an x-critical point p of M which is a local minimum or maximum in the x-direction will become an isolated point in the vertical plane through p. A saddle point in the x-direction will become a double point of the curve.

Finally, let us recall the geometric primitive that is needed, in addition to those that are necessary for the curves in the two-dimensional vertical cross-sections:

- We must be able find all slab points.

It is implicit that we can check whether a finite set of slab points exists, whether two slab points have the same x-coordinate, or when a multiple slab point occurs. Thus, when at any time in the algorithm, we find that our

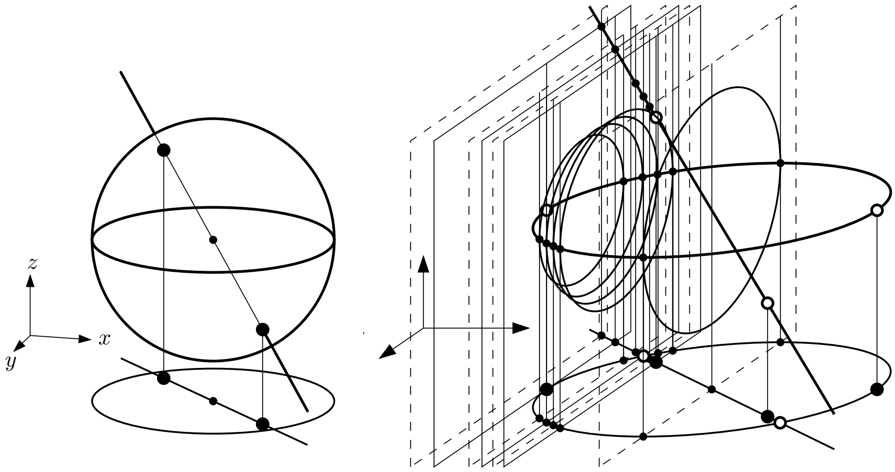


Fig. 5.20. The union of a sphere and a line, and the first half of the vertical cross-sections. The cross-sections in the right half are symmetric. The slab points are marked white.

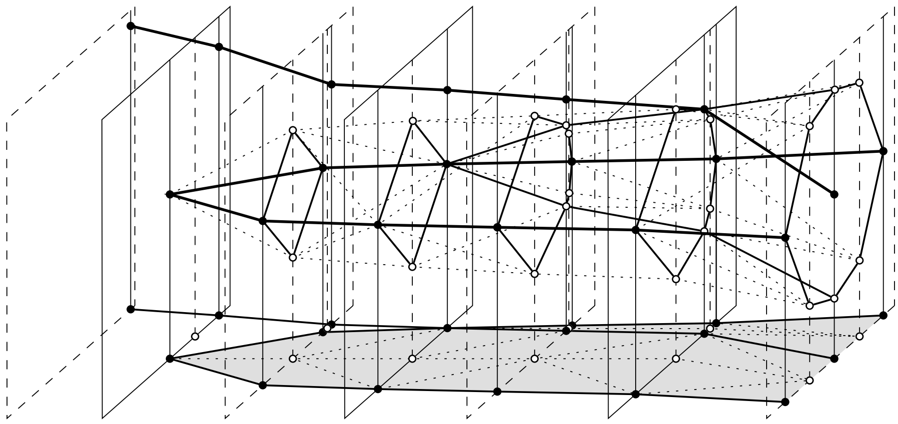


Fig. 5.21. The mesh for the example of Figure 5.20. For better visibility, the vertical sections have been separated by a large amount. Again, only the left half of the mesh is shown.

basic assumption is violated, we can simply perform a sufficiently generic transformation of the coordinates and start from scratch. For details about how this primitive can be carried out for the case of an algebraic surface, we refer to [247, 309].

The two-dimensional subproblems arise from intersecting M with a vertical plane, i. e., by substituting the variable x by some constant (which is often the x-coordinate of some slab point).

As a by-product, the algorithm produces a mesh of a space curve, namely the polar variety on M, defined by two polynomial equations (5.7). The algorithm can be extended to construct a topologically correct polygonal approximation for a space curve that is defined by two *arbitrary* polynomials [168].

Finally, let us step back and look at the algorithm from a broader perspective. Some ideas recur that we have already seen in connection with Snyder's algorithm (Section 5.2.3): the algorithm proceeds by induction on the dimension, and the condition when it is safe to construct a mesh is very similar to global parameterizability, except that there are several curve pieces (a constant number of them), each of which is parameterizable.

Silhouettes and the polar variety, which play an important part in this algorithm, are also used in the algorithm of Cheng, Dey, Ramos and Ray [85] of Section 5.3.2 to avoid complicated topological situations.

Exercise 16. By applying a random transformation of coordinates, one can assume in the meshing algorithm for an algebraic curve (Section 5.4.1) that no two critical points have the same x-coordinate. Is this statement still true when the curve meshing algorithm is used as a subroutine for the vertical sections of the surface meshing algorithm (Section 5.4.2)?

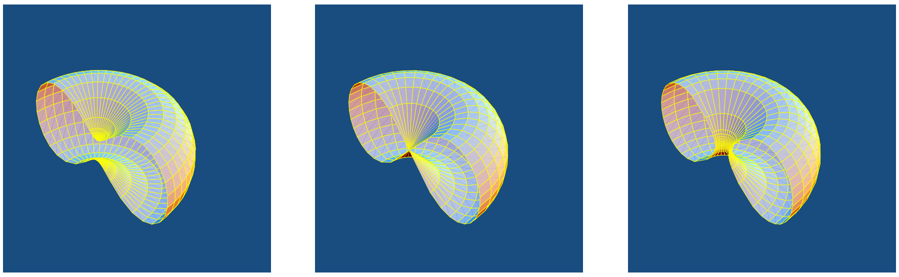
5.5 Obtaining a Correct Mesh by Morse Theory

5.5.1 Sweeping through Parameter Space

Stander and Hart [306] proposed a method for obtaining a topologically correct mesh that is based on sweeping through the family of surfaces $f(x, y, z) = a$ for varying parameters a and watching the critical points where the topology changes. Morse theory (see Section 7.4.2 on p. 298) classifies these changes. This method works theoretically, but there is no completely analyzed guaranteed finite algorithm to implement it. We sketch the main idea of this method.

For a given parameter a, the surface $f(x, y, z) = a$ can be interpreted as the *level set* of a trivariate function $f : \mathbb{R}^3 \to \mathbb{R}$. The idea is to start with a very small (or very large) value a for which $f(x, y, z) = a$ has no solution, and to gradually increase a until $a = 0$ and the surface in which we are interested is at hand. This is related to the space sweep method of Section 5.4, except that it works in one dimension higher: It sweeps a hyperplane $a = \text{const}$ through the four-dimensional space of points (x, y, z, a) and maintains the intersection with the hypersurface $f(x, y, z) = a$.

As a varies, the surface "expands" continuously, except when a passes a *critical value* of f, where the topology changes. A critical value is the value of f at a *critical point*, i. e., at a point x where $\nabla f(x) = 0$. (These are precisely the values that we have avoided in the discussion so far, by assuming that the surface has no critical points.) At a *non-degenerate* critical point x, the Hessian H_f has full rank, and the number of its negative eigenvalues (the Morse index) gives information about the type of topology change. A critical value of Morse index 0 or 3 is a local minimum or maximum of f, and it corresponds to the situation when a small sphere-like component of the surface appears or disappears as a increases. The more interesting cases are the *saddle points*, the critical points of Morse index 1 and 2. Generically, they look like a hyperboloid $x^2 + y^2 - z^2 = a$ in the vicinity of the origin, for $a \approx 0$. For $a > 0$, we have a hyperboloid of one sheet, and for $a < 0$, we have a hyperboloid of two sheets, see Figure 5.22. The transition occurs at $a = 0$, where the surface is a cone. Depending on the Morse index (1 or 2),



(a) (b) (c)

Fig. 5.22. The change of the surface at a saddle point of f. Two separate pieces of the surface (a) come together in a pinching point (b) and form a tunnel (c).

the transition in Figure 5.22 takes place from left to right or from right to left as a increases. The eigenvectors of the Hessian give the coordinate frame for rotating and scaling the picture such that it looks like the standard situation in Figure 5.22.

Degenerate critical points, where the Hessian H_f does not have full rank, would pose a difficulty for this approach. They can be avoided by multiplying f by some suitably generic positive function g like $g(x) = a + \|x - b\|$ for some arbitrarily chosen scalar point b and scalar $a > 0$.

The algorithm of Stander and Hart [306] proceeds as follows: First we compute all critical points and critical values. This amounts to solving a 0-dimensional system of equations. Then we let a vary from $a = a_{\min}$, where the surface is empty, to $a = 0$ in small steps. At each step, we maintain a mesh of the surface $f(x, y, z) = a$. Between critical values, we simply update the mesh. We know that the surface has no singularities, and we know that the topology is unchanged from the previous step. Any standard continuation method that builds a mesh on each component of the surface, taking into account Lipschitz constants for ∇f, can be applied.

At a critical point, we have to implement the appropriate topological change in the surface. A critical point of index 0 is easy to handle: One just has to generate a small spherical component of the surface. A critical point of index 3 is even easier: a small spherical component is simply deleted.

At a critical point, we have to implement the topological change indicated in Figure 5.22. Going from left to right, two surface patches meet, forming a tunnel. We shoot rays from the origin in the positive and negative z direction (which is given by one of the eigenvectors of the Hessian), and remove the two mesh triangles that we hit first. Connecting the two triangles by a cylindrical ring establishes the new topology.

Going from right to left corresponds to closing a tunnel and separating the surface into two pieces which are locally disconnected. We intersect the x-y-plane with the surface and remove the ring of intersected triangles. By triangulating the two polygonal boundaries that are formed in the upper and in the lower half-plane, the two holes are closed.

To make a rigorous and robust method, one has to analyze the required step length that makes the approximations work, but this has not been done so far. Also, the complexity of the resulting mesh has not been analyzed.

5.5.2 Piecewise-Linear Interpolation of the Defining Function

The method of Boissonnat, Cohen-Steiner, and Vegter [57] also uses Morse theory, but in a more indirect way. The basic idea is to output the zero-set of a piecewise-linear interpolation of the defining function f. More precisely, let $S = f^{-1}(0)$ denote the surface that we want to mesh, and assume S is contained in some bounding box. Let T denote a tetrahedral mesh of this bounding box, $\hat{f}$ be the function obtained by linear interpolation of f on T, and set $\hat{S} = \hat{f}^{-1}(0)$. The algorithm consists in building a tetrahedral mesh T such that the output mesh $\hat{S}$ is isotopic to S.

To ensure that this is the case, the mesh T must of course satisfy certain conditions. From Morse theory, one might require that f and $\hat{f}$ have the same critical points, the same value at critical points, and the same types of critical points. Unfortunately, this is not sufficient even for implicit curves in the plane. Indeed, the situation in figure 5.23 is a two-dimensional example of two zero-sets $S = f^{-1}(0)$ and $S' = g^{-1}(0)$ (boundaries of the grey regions) which are not homeomorphic, though their defining functions have the same critical points, with the same values and indices. In this example, g cannot be obtained from f by piecewise-linear interpolation, but it is possible to design examples where this is the case.

Therefore, additional conditions are required. A sufficient set of conditions is given in the theorem below, which is the mathematical basis of the algorithm. The theorem is based on Morse theory for *piecewise-linear functions*, see [38, 39, 57]. We present a simplified version here. We assume that every

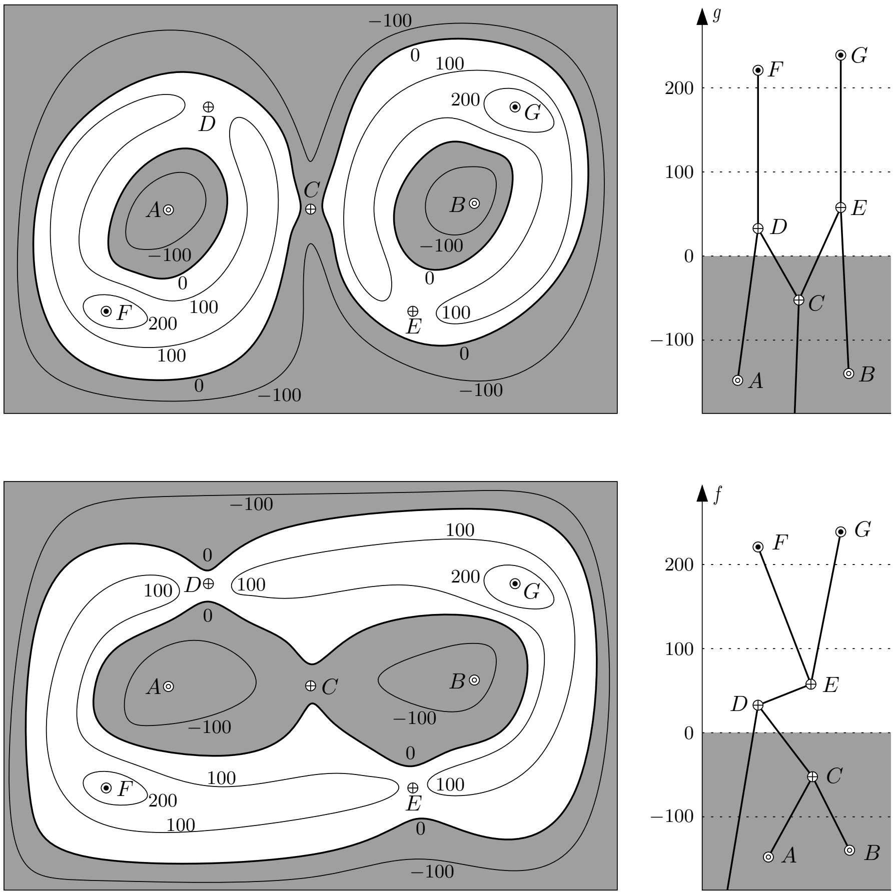


Fig. 5.23. Critical points do not determine the topology of level sets. The two functions have the same critical points of the same types at the same heights, but different level sets at level 0. Minima and maxima are indicated by empty and full circles, and crosses denote saddle points. On the right, the corresponding contour trees (Section 7.4.2) are shown.

critical point of f is a vertex of T. The local topology at a critical point s of f (or $\hat{f}$) is characterized by the Euler characteristic of the *lower link* at s. Loosely speaking, the lower link can be defined as the intersection of the lower level set $f^{-1}((-\infty, f(s)])$ with a small sphere around s. The lower link is actually defined only for a piecewise linear function $\hat{f}$ on a triangulation T, as a certain subcomplex of T. If f is a Morse function and s is a critical point with Morse index i, the Euler characteristic if the "lower link" according to the definition above is $1 - (-1)^i$, see Exercise 3 in Chapter 7 (p. 309).

Theorem 8. *Assume f and $\hat{f}$ have the same critical points. At each critical point s, f and $\hat{f}$ have the same value, and the lower link of s for f has the same Euler characteristic as the lower link for $\hat{f}$. Suppose there is a subcomplex W of T satisfying the following conditions:*

1. *f does not vanish on ∂W.*
2. *W contains no critical point of f.*
3. *W can be subdivided into a complex that collapses onto $\hat{S}$ (see Section 7.3, p. 290).*

Then S and $\hat{S}$ are isotopic.

An example is shown in Figure 5.24.

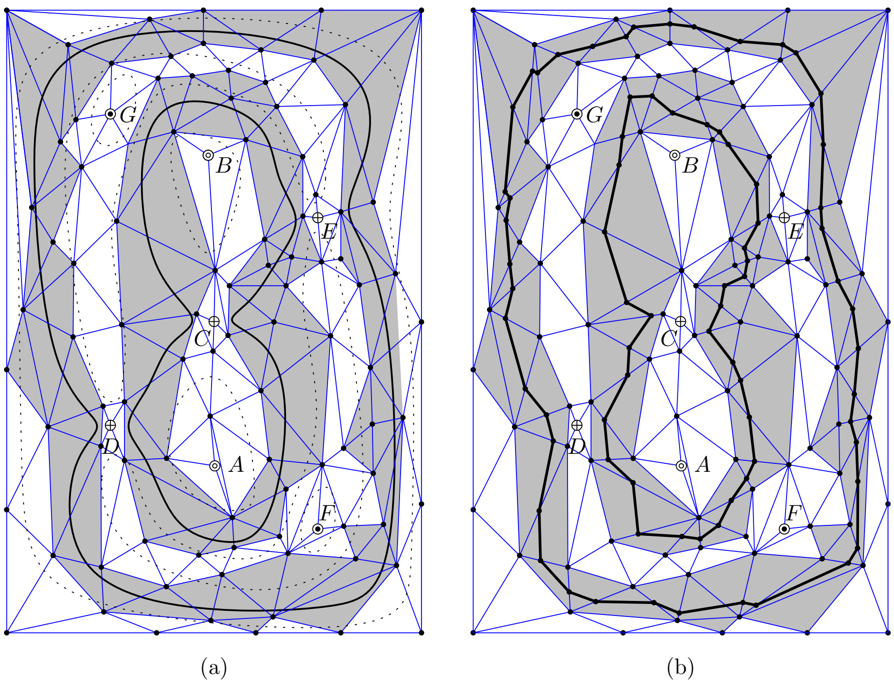


Fig. 5.24. (a) a triangulation T for the function f of Figure 5.23, rotated by 90°. The subcomplex W is shaded. Since W must collapse to S, it must form two bands that enclose the two components of S, without common vertices. (b) the zero-set of the piecewise linear function $\hat{f}$.

The algorithm that is based on this theorem works with an octree-like subdivision of the bounding box into boxes, which are further subdivided into a tetrahedral mesh T. The complex W is taken to be the "watershed" of $\hat{S}$ in the graph of $|\hat{f}|$: W is grown outward from the set of tetrahedra which have vertices with different signs of f. Tetrahedra are added to W in order to fulfill Condition 1, while trying to avoid the inclusion of critical points (Condition 2). If a set W cannot be found, the mesh T is refined. Note that fulfilling the conditions requires to compute all critical points of f exactly, which is difficult, in particular in the case of nearly degenerate critical points. This is why the algorithm actually uses a relaxed (but still sufficient) set of conditions that permits an implementation within the framework of interval analysis. This algorithm is not meant to provide a geometrically accurate approximation of S, but rather to build a topologically correct approximation using as few elements as possible.

5.6 Research Problems.

1. It was mentioned in Section 5.2.4 that the behavior of the Small Normal Variation refinement algorithm Plantinga and Vegter [270] adapts the refinement to the properties of f. Estimate the number of cubes generated by the algorithm in terms of properties of the function f, like total variation of f and ∇f, etc.
2. Can the balancing operation be eliminated in the algorithm of Section 5.2.4? Try to define rules for constructing a mesh when a cube may have an arbitrary number of small neighboring boxes.
3. The algorithm in Section 5.2.4 stops as soon as the angle between two surface normals inside a cube is bounded by $\pi/2$. If we impose some smaller bound α on the angle, what can be said about the distance between the surface and the approximating mesh? How should the mesh be chosen to obtain a good approximation?

The Delaunay refinement algorithm requires the knowledge of a lower estimate $\psi(p)$ on the local feature size. The *minimum* local feature size $\mathrm{lfs}_{\min}$ corresponds to points of maximum principal curvature or to medial spheres that touch the surface in two or three points and have a locally minimum radius. In the case of an implicit surface $f(x, y, z) = 0$, the points where these extrema are attained can be found by solving appropriate systems of equations involving f and its derivatives. Generally, these systems have a finite set of solutions, which includes all *local* minima and maxima. By checking and comparing these solutions, one can compute $\mathrm{lfs}_{\min}$ and use this constant as a global lower estimate $\psi(p)$. This yields a theoretically guaranteed and reliable meshing algorithm for smooth surfaces, provided that the equations that are involved can be solved (for example, when f is a polynomial).

However, in this case, the necessary mesh density is dictated by the global minimum of the local feature size, and thus it does not adapt to different parts of the surface. There is no reliable way to find a good *individual* lower estimate $\psi(p)$ on the local feature size $\mathrm{lfs}(p)$ beforehand, short of computing the medial axis. The next two questions address this question from a theoretical viewpoint.

1. The algorithm may rely on the user to specify the function $\psi(p)$, which can as well simply be a global constant $\psi_{\min}$ independent of the location. Suppose the algorithm terminates, for a given function ψ, and constructs a mesh. Is there a way of deciding if the constructed mesh is at least *consistent*, in the sense that *there exists* a hypothetical surface S' for which ψ is a lower bound on the local feature size, and for which the same mesh would be obtained? (This idea of having a "certificate" of consistency is similar to the approach of [122] for curve reconstruction.)
   Note that in practice, one may apply the algorithm to a non-smooth surface and be perfectly happy with the resulting mesh; however, the hypothetical surface S' in the above question would necessarily have to be smooth. Otherwise it would contain points with $\mathrm{lfs} = 0$.
2. For an implicit surface $f(x, y, z) = 0$, is there a way of estimating the local feature size within some given range for the variables x, y, z, by looking at the function f and its derivatives? Can one use interval arithmetic to obtain a conservative lower bound ψ?
3. The test (5.5) for critical points in a silhouette involves second derivatives (cf. Exercise 15). Is there a zero-dimensional system of equations for establishing the topological ball property that only involves f and its first derivatives?
4. **The topological ball property and isotopy.**
   The topological ball property only guarantees a homeomorphism between the original surface and the reconstruction, it does not provide an isotopy. In fact, Theorem 5 can be extended to manifolds in arbitrary dimension k (and even to non-manifolds [132]): For a k-dimensional manifold $M \subset \mathbb{R}^n$, the topological ball property means that every Voronoi face F of dimension d intersects M in a closed topological $(d - n + k)$-ball or in the empty set. For manifolds of codimension at least 2, the topological ball property is not sufficient to establish isotopy. For example, the topological ball property for a point sample P on a curve C in $\mathbb{R}^3$ ($k = 1$, $n = 3$) will not detect whether C is knotted inside a Voronoi cell, and thus the restricted Delaunay will not always be isotopic to C.
   a) Does the topological ball property for a surface S in $\mathbb{R}^3$ (or more generally, for an $(n-1)$-manifold embedded in $\mathbb{R}^n$) imply that the restricted Delaunay triangulation is isotopic to S?
   b) Find an appropriate strengthening of the topological ball property that ensures isotopy of the restricted Delaunay triangulation.
5. For a curve $f(x, y) = 0$, the critical points in direction $\binom{u}{v}$ are given by the equation

$$u \cdot f_x(x, y) + v \cdot f_y(x, y) = 0, \quad f(x, y) = 0.$$

   If f is a polynomial of degree d, give an upper bound on the number of directions for which two distinct critical points lie on a line parallel to $\binom{u}{v}$.