Polygon Mesh Processing

Mario Botsch
Leif Kobbelt
Mark Pauly
Pierre Alliez
Bruno Lévy

Polygon Mesh Processing

Polygon Mesh Processing

Mario Botsch
Leif Kobbelt
Mark Pauly
Pierre Alliez
Bruno Lévy

A K Peters, Ltd.
Natick, Massachusetts

CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

Version Date: 20150220

International Standard Book Number-13: 978-1-4398-6531-6 (eBook - PDF)

Contents

Preface

Recent innovation in 3D acquisition technology, such as computer tomography, magnetic resonance imaging, 3D laser scanning, ultrasound, radar, and microscopy has enabled highly accurate digitization of complex 3D objects. Numerous scientific disciplines, such as neuroscience, mechanical engineering, and astrophysics, rely on the analysis and processing of such geometric data to understand intricate geometric structures and facilitate new scientific discoveries. A similar abundance of digital 3D content can be observed in other fields and industries, including entertainment, cultural heritage, geo-exploration, architecture, and urban modeling. Concurrent to these advances in 3D sensing technology, we are experiencing a revolution in digital manufacturing technology (e.g., in bio-medicine, commodity product design, and architecture). Novel materials and robotic production will soon allow the automated creation of complex, fully functional physical artifacts from a digital design plan.

Between acquisition and production lies the discipline of *digital geometry processing*, a relatively new field of computer science that is concerned with mathematical models and algorithms for analyzing and manipulating geometric data. Typical operations include surface reconstruction from point samples, filtering operations for noise removal, geometry analysis, shape simplification, and geometric modeling and interactive design. The abundance of data sources, processing operations, and manufacturing technologies has resulted in a great wealth of mathematical representations for geometric data. In this context, polygon meshes have become increasingly popular in recent years and are nowadays used intensively in

many different areas of computer graphics and geometry processing. In computer-aided geometric design (CAGD), triangle and polygon meshes have developed into a valuable alternative to traditional spline surfaces since their conceptual simplicity allows for flexible and highly efficient processing. Moreover, the consequent use of polygon meshes as a surface representation avoids error-prone conversions (e.g., from CAD surfaces to mesh-based input data of numerical simulations). Besides classical geometric modeling, other major areas frequently employing polygon meshes are computer games and movie production. In this context, geometric models acquired by 3D scanning techniques typically have to undergo post-processing and shape optimization techniques before being used in production.

This book discusses the main components of the geometry processing pipeline based on polygon meshes, as illustrated on the right. For the instructive purposes of this book, the order in which topics are described deviates somewhat from the typical processing order shown in the figure. We first discuss general concepts of surface representations in Chapter 1 and highlight the advantageous properties of polygon meshes for digital geometry processing. Chapter 2 presents efficient data structures for the implementation of polygon meshes. Chapter 3 introduces fundamental concepts of differential geometry and gives derivations for their discrete analogs. These form the basis of algorithms for mesh smoothing (Chapter 4) to reduce noise in scanned surfaces by generalizing signal processing techniques to irregular polygon meshes. Chapter 5 introduces different methods for computing surface parameterizations that are essential in many geometry processing tasks. General

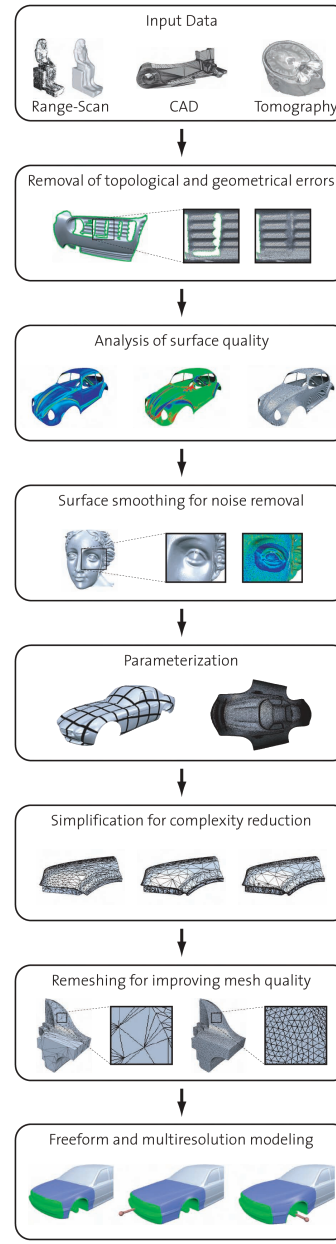

Figure 1. Geometry processing pipeline. (Image from [Botsch et al. 06b].)

remeshing methods (Chapter 6) allow optimizing the shape of triangle or polygon elements, which is important for the robustness of numerical simulations and further processing operations. Mesh simplification and approximation techniques (Chapter 7) are commonly required for error-controlled simplification of highly complex meshes acquired by 3D scanning or automatically generated along the processing pipeline. Chapter 8 describes the different sources of input data and introduces different types of geometric and topological degeneracies and inconsistencies. We discuss methods for removing these artifacts, resulting in defect-free 2-manifold meshes suitable for further processing. Chapter 9 presents techniques for intuitive and interactive shape deformation. Since linear systems appear in many of the presented mesh processing algorithms, in the appendix we describe efficient algorithms for solving linear systems and compare several existing libraries.

The idea for this book originated from a series of tutorials and courses on mesh processing and geometric modeling. In 2006, Mario and Mark organized and taught a course on polygon mesh processing for industry practitioners at ETH Zurich. The same year, Leif, as well as Christian Rössl and Stephan Bischoff, joined them for two full-day tutorials at ACM SIGGRAPH and Eurographics, respectively. The syllabus was restructured for courses at SIGGRAPH 2007 and Eurographics 2008, with Pierre and Bruno replacing Christian and Stephan as presenters.

Our thanks go to Christian Rössl and Stephan Bischoff for their contributions to the early versions of the course, to Henrik Zimmer for help with the book cover model, and to Silke Kölsch for proofreading the text. We are immensely grateful to Alice Peters of A K Peters for her encouragement, advice, and patience, to Sarah Cutler for the excellent editing, and the entire A K Peters team for their support. This book would not have been possible without the contributions of our numerous scientific collaborators and colleagues who helped shape the field of polygon mesh processing. Last but not least, a big thanks to our students. Their questions and feedback have been immensely valuable for refining the material of the book, and their enthusiasm has been the ultimate source of motivation for this project.

1 Surface Representations

Geometry processing is mostly about applying algorithms to geometric models. If the algorithms represent the *action*, then the geometry is the *object*. In this section we are going to discuss various mathematical representations for geometric objects. While these representations can be 2D or 3D, the actual geometry that we are dealing with will always be the 2D surface of a 3D solid object. As we will see throughout this book, for each specific problem in geometry processing, we can identify a characteristic set of operations by which the computation is dominated, and hence we have to choose an appropriate representation that supports the efficient implementation of these operations.

From a high-level point of view, there are two major classes of surface representations: *parametric* representations and *implicit* representations. Parametric surfaces are defined by a vector-valued parameterization function $\mathbf{f}\colon \Omega \to \mathcal{S}$ that maps a 2D parameter domain $\Omega \subset \mathbb{R}^2$ to the surface $\mathcal{S} = \mathbf{f}(\Omega) \subset \mathbb{R}^3$. In contrast, an implicit (or volumetric) surface representation is defined to be the zero set of a scalar-valued function $F\colon \mathbb{R}^3 \to \mathbb{R}$, i.e., $\mathcal{S} = \{\mathbf{x} \in \mathbb{R}^3 \mid F(\mathbf{x}) = 0\}$.

For illustration, we can define curves analogously in a parametric fashion by functions $\mathbf{f}\colon \Omega \to \mathcal{C}$ with $\Omega = [a, b] \subset \mathbb{R}$. A corresponding implicit definition is only available for *planar* curves, i.e., $\mathcal{C} = \{\mathbf{x} \in \mathbb{R}^2 \mid F(\mathbf{x}) = 0\}$ with $F\colon \mathbb{R}^2 \to \mathbb{R}$. A simple 2D example is the unit circle, which can be

defined by the range of a parametric function

$$\mathbf{f}\colon [0, 2\pi] \to \mathbb{R}^2\,, \quad t \mapsto \begin{pmatrix} \cos t \\ \sin t \end{pmatrix},$$

as well as by the kernel of the implicit function

$$F\colon \mathbb{R}^2 \to \mathbb{R}\,, \quad (x, y) \mapsto \sqrt{x^2 + y^2} - 1.$$

Similarly, in 3D, a sphere can be represented by a parametric or an implicit equation (see Section 3.2 for more details).

For more complex shapes, it is often not feasible to find an explicit formulation with a single function that approximates a given shape with sufficient accuracy. Hence, the function domain is usually split into smaller sub-regions and an individual function (*surface patch*) is defined for each segment. In this *piecewise* definition, each function needs to approximate the given shape only locally, while the global approximation tolerance is controlled by the size and number of the segments. The mathematical challenge is to guarantee a consistent transition from each patch to its neighboring ones. The most common piecewise surface definition in the parametric case is the segmentation of Ω into triangles or quadrangles. For implicit surface definitions, the embedding space is usually split into hexahedral (*voxels*) or tetrahedral cells.

Both parametric and implicit representations have their particular strengths and weaknesses, such that for each geometric problem the better suited one should be chosen. In order to analyze geometric operations and their requirements on the surface representation, one can classify them into the following three categories [Kobbelt 03]:

- Evaluation. This entails the sampling of the surface geometry or of other surface attributes, e.g., the surface normal field. A typical application example is surface rendering.
- Query. Spatial queries are used to determine whether or not a given point $\mathbf{p} \in \mathbb{R}^3$ is inside or outside of the solid bounded by a surface $\mathcal{S}$, which is a key component for solid modeling operations. Another typical query is the computation of a point's distance to a surface.
- Modification. A surface can be modified either in terms of *geometry* (surface deformation) or in terms of *topology* (e.g., when different parts of the surface are to be merged, cut, or deleted).

We will see that parametric and implicit surface representations have complementary advantages with respect to these three types of geometric operations, i.e., the strengths in terms of efficiency or robustness of the one

are often the drawbacks of the other. Hence, for each specific geometric problem, the more suitable representation should be chosen, which, in turn, requires efficient conversion routines between the two representations (see Section 1.5). In Section 1.6 we present an outlook to approaches that combine both representations in order to design algorithms that are both efficient and robust.

1.1 Surface Definition and Properties

The common definition of a *surface* in the context of computer graphics applications is "an orientable continuous 2D manifold embedded in $\mathbb{R}^3$." Intuitively, this can be understood as the boundary surface of a non-degenerate 3D solid where *non-degenerate* means that the solid does not have any infinitely thin parts or features such that the surface properly separates the "interior" and "exterior" of the solid (see Figure 1.1). A surface with boundaries is one that can be extended into a proper manifold surface by filling the holes.

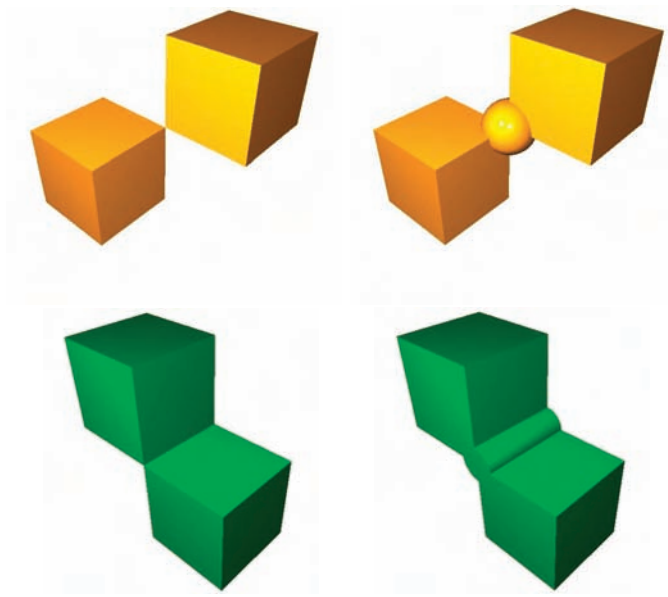

Figure 1.1. An orientable continuous 2-manifold describes the surface of a non-degenerate solid. A degenerate/non-manifold vertex (top left), which is fixed in (top right). A solid with a degenerate/non-manifold edge (bottom left), fixed in (bottom right).

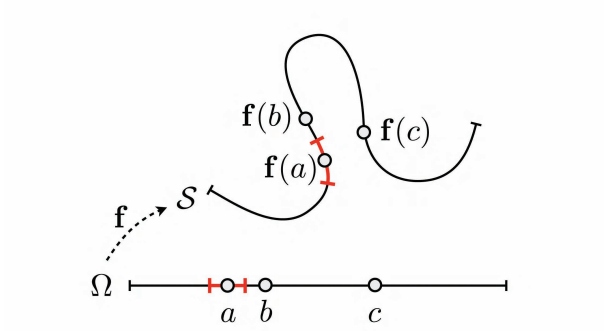

Figure 1.2. A manifold curve. While the points $\mathbf{f}(a)$, $\mathbf{f}(b)$, and $\mathbf{f}(c)$ are all in close *spatial* proximity, only $\mathbf{f}(a)$ and $\mathbf{f}(b)$ are *geodesic* neighbors since their pre-images a and b are neighbors, too. In red: The pre-image of a sufficiently small δ neighborhood around $\mathbf{f}(a)$ in $\mathbb{R}^2$ lies in an ε neighborhood of a in $\mathbb{R}$.

Since in most applications the raw information about the input surface is obtained by discrete sampling (i.e., by *evaluation* if there already exists a digital representation, or by *probing* if the input comes from a real object), the first step in generating a mathematical surface representation is to establish *continuity*. This requires building a consistent neighborhood relation between the samples. In this context, *consistency* refers to the existence of a manifold surface from which the samples are drawn.

While this so-called *geodesic* neighborhood relation (in contrast to a *spatial* neighborhood relation) is difficult to access in implicit representations, it is quite easy to extract from parametric representations in which two points on the surface are in geodesic proximity, if the corresponding pre-images in Ω are close to each other (see Figure 1.2). From this observation we can derive an alternative characterization of *local manifoldness*: a continuous parametric surface is locally manifold at a surface point $\mathbf{p}$ if, for every other surface point $\mathbf{q}$ within a sufficiently small sphere of radius δ around $\mathbf{p}$, the corresponding pre-image is contained in a circle of some radius $\varepsilon = O(\delta)$ around the pre-image of $\mathbf{p}$. A more intuitive way to express this condition is to say that the surface patch that lies within a sufficiently small δ-sphere around $\mathbf{p}$ is topologically equivalent (homeomorphic) to a disk. Since this second definition does not require a parameterization, it applies to implicit representations as well.

When generating a continuous surface from a set of discrete samples, we can either require this surface to *interpolate* the samples or to *approximate* them subject to a certain prescribed tolerance. The latter case is considered more relevant in practical applications, since samples are usually affected by position noise and the surface in between the samples is an approximation

Figure 1.3. Three examples of fair surfaces, which define a blend between two cylinders: a membrane surface that minimizes the surface area (left), a thin-plate surface that minimizes total curvature (center), and a surface that minimizes the variation of mean curvature (right). (Image taken from [Botsch and Kobbelt 04a]. ©2004 ACM, Inc. Included here by permission.)

anyway. In the next section we will consider the issue of approximation in more detail.

Except for a well-defined set of sharp feature-curves and -corners, a surface should be *smooth* in general. Mathematically this is measured by the number k of continuous derivatives that the functions $\mathbf{f}$ or F have. Notice that this analytical definition of C^k smoothness coincides with the intuitive geometrical understanding of smoothness only if the partial derivatives of $\mathbf{f}$ or the gradient of F, respectively, do not vanish locally (*regularity*).

An even stricter requirement for surfaces is *fairness*, where not only the continuity of the derivatives but also their magnitude and variation is considered. There is no general formal definition for the aesthetic concept of fairness, but a surface is usually considered fair if, e.g., the curvature or its variation is globally minimized (see Figure 1.3).

In Chapter 3 we will explain how the notion of curvature can be generalized to polygon meshes such that properties like smoothness and fairness can be applied to meshes as well (see Chapter 4).

1.2 Approximation Power

The exact mathematical modeling of a real object or its boundary is usually intractable. Hence, a digital surface representation can only be an approximation in general. As mentioned in the introduction, in order to simplify the approximation tasks, the domain of the representation is often split into small segments, and for each segment a function (a patch) is defined that locally approximates the part of the input that belongs to the segment.

Since our surface representations are supposed to support efficient processing, a natural choice is to restrict functions to the class of *polynomials* because those can be evaluated by elementary arithmetic operations. Another justification for the restriction to polynomials is the well-known *Weierstrass theorem* that guarantees that each smooth function can be approximated by a polynomial up to any desired precision [Ross 80].

From calculus we know that a C^∞ function g with bounded derivatives can be approximated over an interval of length h by a polynomial of degree p such that the approximation error behaves like $O(h^{p+1})$ (e.g., Taylor's theorem or generalized mean value theorem) [Rudin 02]. As a consequence there are, in principle, two possibilities to improve the accuracy of an approximation with piecewise polynomials. We can either raise the degree of the polynomial (*p-refinement*) or we can reduce the size of the individual segments and use more segments for the approximation (*h-refinement*).

In geometry processing applications, h-refinement is usually preferred over p-refinement since, for a discretely sampled input surface, we cannot make reasonable assumptions about the boundedness of higher-order derivatives. Moreover, for piecewise polynomials with higher degree, the C^k smoothness conditions between segments are sometimes quite difficult to satisfy. Finally, with today's computer architectures, processing a large number of very simple objects is often much more efficient than processing a smaller number of more complex ones. This is why the somewhat extremal choice of C^0 piecewise linear surface representations, i.e., *polygonal meshes*, have become the widely established standard in geometry processing.

While, for parametric surfaces, the $O(h^{p+1})$ approximation error estimate follows from the mean value theorem in a straightforward manner, a more careful consideration is necessary for implicit representations. The generalized mean value theorem states that if a sufficiently smooth function g over an interval $[a, a+h]$ is interpolated at the abscissae $t_0, \ldots, t_p$ by a polynomial f of degree p, then the approximation error is bounded by

$$|f(t) - g(t)| \;\leq\; \frac{1}{(p+1)!} \max f^{(p+1)} \prod_{i=0}^{p} (t_i - t) \;=\; O(h^{p+1}).$$

For an implicit representation $G\colon \mathbb{R}^3 \to \mathbb{R}$ and the corresponding polynomial approximant F, this theorem is still valid; however, here the actual surface geometry is not defined by the function values $G(\mathbf{x})$, for which this theorem gives an error estimate, but by the zero level set of G, i.e., by $\mathcal{S} = \{\mathbf{x} \in \mathbb{R}^3 \,|\, G(\mathbf{x}) = 0\}$.

Consider a point $\mathbf{x}$ on the implicit surface defined by the approximating polynomial F, i.e., $F(\mathbf{x}) = 0$ within some voxel. We can find a corresponding point $\mathbf{x} + \mathbf{d}$ on the implicit surface defined by G, i.e., $G(\mathbf{x} + \mathbf{d}) = 0$ by shooting a ray in normal direction to F, i.e., $\mathbf{d} = d\,\nabla F / \|\nabla F\|$. For a

sufficiently small voxel size h, we obtain

$$|F(\mathbf{x}+\mathbf{d})| \;\approx\; |d|\,\|\nabla F(\mathbf{x})\| \quad\Rightarrow\quad |d| \;\approx\; \frac{|F(\mathbf{x}+\mathbf{d})|}{\|\nabla F(\mathbf{x})\|},$$

and from the mean value theorem we get

$$|F(\mathbf{x}+\mathbf{d}) - G(\mathbf{x}+\mathbf{d})| \;=\; |F(\mathbf{x}+\mathbf{d})| \;=\; O(h^{p+1}),$$

which yields $|d| = O(h^{p+1})$ if the magnitude of the gradient $\|\nabla F\|$ is bounded from below by some $\varepsilon > 0$. In practice one tries to find an approximating polynomial F with low variation of the gradient magnitude in order to have a uniform distribution of the approximation error.

1.3 Parametric Surface Representations

Parametric surface representations have the advantage that the function $\mathbf{f} : \Omega \to \mathcal{S}$ enables the reduction of many 3D problems on the surface $\mathcal{S}$ to 2D problems in the parameter domain Ω. For instance, sample points on the surface can easily be generated by sampling the domain Ω and evaluating the function $\mathbf{f}$. In a similar manner, geodesic neighborhoods, i.e., neighborhoods on the surface $\mathcal{S}$, can easily be found by considering neighboring points in the parameter domain Ω. A simple composition of $\mathbf{f}$ with a deformation function $\mathbf{d} : \mathbb{R}^3 \to \mathbb{R}^3$ results in an efficient modification of the surface geometry.

On the other hand, generating a parametric surface parameterization $\mathbf{f}$ can be very complex, since the parameter domain Ω has to match the topological and metric structure of the surface $\mathcal{S}$ (Chapter 5). When changing the shape of $\mathcal{S}$, it might be necessary to update the parameterization accordingly in order to reflect the respective changes of the underlying geometry: a low-distortion parameterization requires the metrics in $\mathcal{S}$ and Ω to be similar, and hence we have to avoid or adapt to excessive stretching.

Since the manifold surface $\mathcal{S}$ is defined as the range of the parameterization $\mathbf{f}$, its topology is equivalent to that of Ω if $\mathbf{f}$ is continuous and injective. This implies that changing the topology of a parametric surface $\mathcal{S}$ can be extremely complicated because not only the parameterization but also the domain Ω has to be adjusted accordingly. The typical inside/outside or signed distance queries are, in general, also very expensive on parametric surfaces since they usually require finding the closest point on $\mathcal{S}$ to the query point (*foot point*). The same applies to the detection of self-collisions (i.e., non-injectivities). Hence, topological modification and spatial queries are the weak points of parametric surfaces.

1.3.1 Spline Surfaces

Tensor-product spline surfaces—often called NURBS—are the standard surface representation in today's CAD systems. They are used for constructing high-quality surfaces ("class A") as well as for freeform surface editing tasks. Spline surfaces can be described conveniently by piecewise polynomial or rational B-spline basis functions $N_i^n(\cdot)$. For more detail, see e.g., [Farin 97, Piegl and Tiller 97, Prautzsch et al. 02].

A tensor product spline surface $\mathbf{f}$ of bi-degree n is a piecewise polynomial surface that is built by connecting several polynomial patches in a smooth C^{n-1} manner. The rectangular segments are defined by two knot vectors $\{u_0, \ldots, u_{m+n}\}$ and $\{v_0, \ldots, v_{k+n}\}$ and the overall surface is then obtained by

$$\begin{aligned} \mathbf{f}\colon [u_n, u_m] \times [v_n, v_k] &\rightarrow \mathbb{R}^3 && (1.1) \\ (u, v) &\mapsto \sum_{i=0}^{m} \sum_{j=0}^{k} \mathbf{c}_{ij} N_i^n(u) N_j^n(v). && (1.2) \end{aligned}$$

The *control points* $\mathbf{c}_{ij} \in \mathbb{R}^3$ define the so-called *control mesh* of the spline surface. Because $N_i^n(u) \geq 0$ and $\sum_i N_i^n \equiv 1$, each surface point $\mathbf{f}(u, v)$ is a convex combination of the control points $\mathbf{c}_{ij}$; i.e., the surface lies within the convex hull of the control mesh. Due to the minimal support of the basis functions, each control point has local influence only. These two properties cause spline surfaces to closely follow the control mesh, thereby providing a geometrically intuitive metaphor for modeling the shape of surfaces by adjusting their control points.

A tensor-product surface—as the image of a rectangular domain under the parameterization $\mathbf{f}$—always represents a rectangular surface patch embedded in $\mathbb{R}^3$. If shapes of more complicated topological structure are to be represented by spline surfaces, the model has to be decomposed into a number of (possibly trimmed) tensor-product patches.

As a consequence of these *topological constraints*, typical CAD models often consist of a huge collection of surface patches. In order to represent a high-quality, globally smooth surface, these patches have to be connected in a smooth manner, leading to additional *geometric constraints* that have to be taken care of throughout all surface processing phases. The large number of surface patches and the resulting topological and geometric constraints significantly complicate surface construction, and in particular the later surface modeling tasks.

Another drawback of classical tensor-product spline representations is that adding more control vertices (*refinement*) is only possible by splitting parameter intervals $[u_i, u_{i+1}]$ or $[v_j, v_{j+1}]$, which affects an entire row or column of the control mesh, respectively. Here, the alternative

representation by *T-splines* can improve the situation since they enable the local refinement of the control mesh [Sederberg et al. 03].

1.3.2 Subdivision Surfaces

Subdivision surfaces [Zorin et al. 00] can be considered a generalization of spline surfaces since they are also controlled by a coarse *control mesh*, but in contrast to spline surfaces, they can represent surfaces of arbitrary topology. Subdivision surfaces are generated by repeated refinement of control meshes: after each topological refinement step, the positions of the (old and new) vertices are adjusted based on a set of local averaging rules (see Figure 1.4). A careful analysis of these rules reveals that in the limit this process results in a surface of provable smoothness [Peters and Reif 08].

As a consequence, subdivision surfaces are restricted neither by topological (other than manifoldness) nor by geometric constraints as spline surfaces are, and their inherent hierarchical structure allows for highly efficient algorithms. However, subdivision techniques are limited to producing meshes with so-called semiregular *subdivision connectivity*, i.e., surface meshes whose triangulations are the result of repeated uniform refinement of a coarse control mesh. As this constraint is not met by arbitrary meshes, those would have to be *remeshed* to subdivision connectivity in a

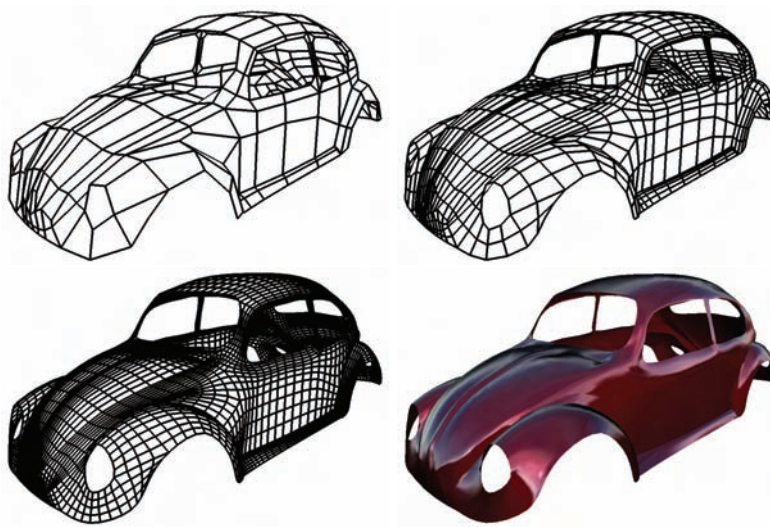

Figure 1.4. Subdivision surfaces are generated by iterative uniform refinement of a coarse control mesh. (Image taken from [Botsch 05].)

preprocessing step [Eck et al. 95, Lee et al. 98, Kobbelt et al. 99a, Guskov et al. 00]. But, since this remeshing corresponds to a resampling of the surface, it usually leads to sampling artifacts and loss of information. In order to avoid the restrictions caused by these *connectivity constraints*, our goal is to work on arbitrary triangle meshes, as they provide higher flexibility and still allow for efficient surface processing.

1.3.3 Triangle Meshes

In many geometry processing algorithms, *triangle meshes* are considered a collection of triangles without any particular mathematical structure. In principle, however, each triangle defines, via its barycentric parameterization, a segment of a piecewise linear surface representation.

Every point $\mathbf{p}$ in the interior of a triangle $[\mathbf{a}, \mathbf{b}, \mathbf{c}]$ can be written in a unique fashion as a barycentric combination of the corner points:

$$\mathbf{p} = \alpha\,\mathbf{a} + \beta\,\mathbf{b} + \gamma\,\mathbf{c}, \tag{1.3}$$

with

$$\alpha + \beta + \gamma = 1, \quad \alpha, \beta, \gamma \geq 0.$$

By choosing an arbitrary triangle $[\mathbf{u}, \mathbf{v}, \mathbf{w}]$ in the parameter domain, we can define a linear mapping $\mathbf{f} : \mathbb{R}^2 \to \mathbb{R}^3$ with

$$\alpha\,\mathbf{u} + \beta\,\mathbf{v} + \gamma\,\mathbf{w} \quad \mapsto \quad \alpha\,\mathbf{a} + \beta\,\mathbf{b} + \gamma\,\mathbf{c}. \tag{1.4}$$

Based on this per-triangle mapping, it is sufficient to define a 2D position for each vertex in order to derive a global parameterization for an entire triangle mesh. In Chapter 5 we will discuss sophisticated methods for choosing this triangulation in the parameter domain such that the distortion caused by the piecewise linear mapping from $\mathbb{R}^2$ to $\mathbb{R}^3$ is minimized.

A triangle mesh $\mathcal{M}$ consists of a geometric and a topological component, where the latter can be represented by a graph structure (simplicial complex) with a set of vertices

$$\mathcal{V} = \{v_1, \ldots, v_V\}$$

and a set of triangular faces connecting them

$$\mathcal{F} = \{f_1, \ldots, f_F\}, \quad f_i \in \mathcal{V} \times \mathcal{V} \times \mathcal{V}.$$

However, as we will see in Chapter 2, it is sometimes more efficient to represent the connectivity of a triangle mesh in terms of the edges of the respective graph,

$$\mathcal{E} = \{e_1, \ldots, e_E\}, \quad e_i \in \mathcal{V} \times \mathcal{V}.$$

The geometric embedding of a triangle mesh into $\mathbb{R}^3$ is specified by associating a 3D position $\mathbf{p}_i$ to each vertex $v_i \in \mathcal{V}$:

$$\mathcal{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_V\}, \quad \mathbf{p}_i := \mathbf{p}(v_i) = \begin{pmatrix} x(v_i) \\ y(v_i) \\ z(v_i) \end{pmatrix} \in \mathbb{R}^3,$$

such that each face $f \in \mathcal{F}$ actually corresponds to a triangle in 3-space specified by its three vertex positions. Notice that even if the geometric embedding is defined by assigning 3D positions to the *discrete* vertices, the resulting polygonal surface is still a *continuous* surface consisting of triangular pieces with linear parameterization functions (Equation (1.4)).

If a sufficiently smooth surface is approximated by such a piecewise linear function, the approximation error is of the order $O(h^2)$, with h denoting the maximum edge length. Due to this quadratic approximation power, the error is reduced by a factor of about $1/4$ when halving the edge lengths. As this refinement splits each triangle into four sub-triangles, it increases the number of triangles from F to $4F$ (see Figure 1.5). Hence, the approximation error of a triangle mesh is inversely proportional to its number of faces. The actual magnitude of the approximation error depends on the second-order terms of the Taylor expansion, i.e., on the curvature of the underlying smooth surface. From this we can conclude that a sufficient approximation is possible with just a moderate mesh complexity: the vertex density has to be locally adapted to the surface curvature, such that flat areas are sparsely sampled, while in curved regions the sampling density is higher.

As stated before, an important topological quality of a surface is whether or not it is *2-manifold* (short for *two-dimensional manifold*), which is the case if, for each point, the surface is locally homeomorphic to a disk (or a half-disk at boundaries). A triangle mesh is a 2-manifold if it contains neither non-manifold edges nor non-manifold vertices nor self-intersections. A *non-manifold edge* has more than two incident triangles and a *non-manifold*

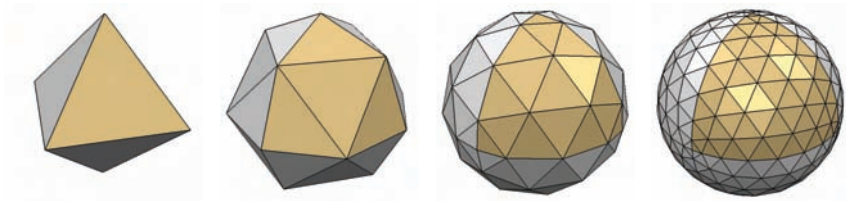

Figure 1.5. Each subdivision step halves the edge lengths, increases the number of faces by a factor of 4, and reduces the approximation error by a factor of about $\frac{1}{4}$. (Image taken from [Botsch et al. 06b].)