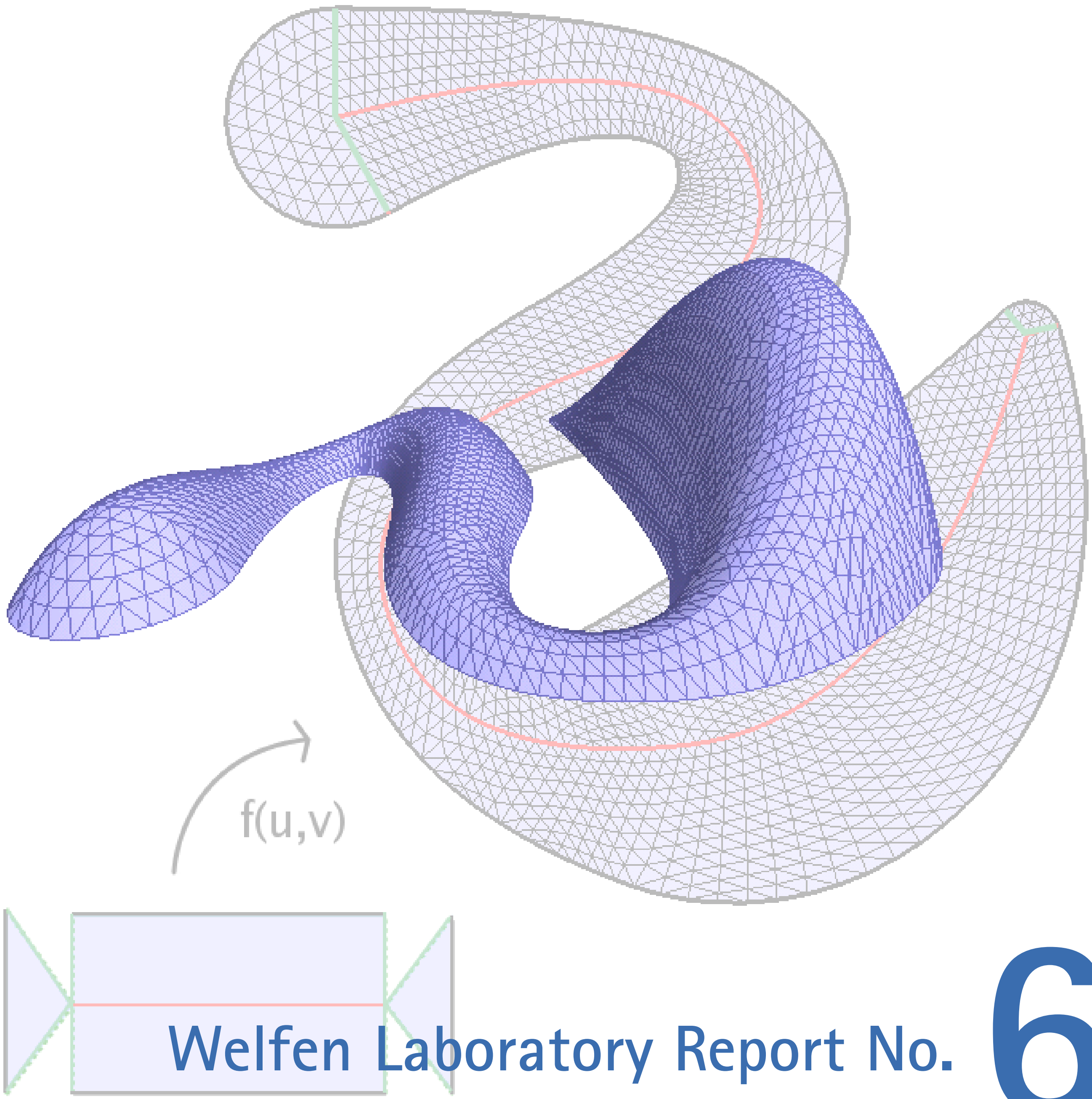
Welfenlab
Institut für Mensch-Maschine-Kommunikation

Leibniz
Universität
Hannover



Welfen Laboratory Report No. 6

Geometric Modeling for Engineering Applications

Franz-Erich Wolter, Martin Reuter, Niklas Peinecke

Herausgeber:
Gottfried Wilhelm Leibniz Universität Hannover
Institut für Mensch-Maschine-Kommunikation
Lehrstuhl Graphische Datenverarbeitung
Prof. Dr. Franz-Erich Wolter

August 2009
ISSN 1866-7996

This preprint was published as

Franz-Erich Wolter, Martin Reuter, Niklas Peinecke
"Geometric Modeling for Engineering Applications"
in Encyclopedia of Computational Mechanics. Part 1: Fundamentals. E. Stein, R. de Borst & T.J.R. Hughes (eds.), John Wiley & Sons, 2007.

Abstract

A geometric model of an object – in most cases being a subset of the three dimensional space – can be used to better understand the object's structure or behavior. Therefore data such as the geometry, the topology and other application specific data have to be represented by the model. With the help of a computer it is possible to manipulate, process or display these data.
We will discuss different approaches for representing such an object: Volume based representations describe the object in a direct way, whereas boundary representations describe the object indirectly by specifying its boundary. A variety of different surface patches can be used to model the object's boundary. For many applications it is sufficient to know only the boundary of an object.
For special objects explicit or implicit mathematical representations can easily be given. An explicit representation is a map from a known parameter space for instance the unit cube to 3D-space. Implicit representations are equations or relations such as the set of zeros of a functional with three unknowns. These can be very efficient in special cases.
As an example of volume based representations we will give a brief overview of the voxel representation. We also show how the boundary of complex objects can be assembled by simpler parts such as surface patches. These come in a variety of forms: planar polygons, parametric surfaces defined by a map from 2D-space to 3D-space, especially spline surfaces and trimmed surfaces, multiresolutionally represented surfaces (for example wavelet-based) and surfaces obtained by subdivision schemes.
In a boundary representation only the boundary of a solid is described. This is usually done by describing the boundary as a collection of surface patches attached to each other at outer edges. One of the (topologically) most complete schemes is the half-edge data structure as described by Mäntylä.
Simple objects constructed via any of the methods above can be joined to build more complex objects via Boolean operators (constructive solid geometry, CSG). Constructing an object one has to assure that the object is in agreement with the topological requirements of the modeling system. Notoriously difficult problems are caused by the fact that most modeling systems can compute surface intersections only with a limited precision. This yields numerical results that may finally cause major errors such as topologically contradictory conclusions.
The rather new method of "Medial Modeling" is also presented. Here an object is described by its medial axis and an associated radius function. The medial axis itself is a collection of lower dimensional objects, i.e. for a 3D-solid a set of points, curves and surface patches. This medial modeling concept developed at the Welfen-

lab yields a very intuitive user interface useful for solid modeling, and also gives as a by-product a natural meshing of the solid for FEM computations.
Additional attributes can be attached to an object, like attributes of physical origin or logical attributes. Physical attributes include photometric, haptical and other material properties, such as elasticity or roughness. Physical attributes are often specified by textures. These texture are mapped to the surface to relate surface points to certain quantities of the attribute. The most common use for these are photometric textures, although they can also be used for roughness etc. Logical attributes relate the object to its (data-)environment. They can for example group objects which are somehow related, or they can associate scripts to the object, such as callbacks for user interactions.

Contents

1 History and Overview

Basically the history of Computer Aided Design (CAD) dates back to the late 1950s, when computer systems became available to help constructing 3D shapes out of blocks of wood or steel. These objects could then be used to stamp products (such as car body panels). Still it was difficult to create shapes that coincided exactly with the curves on the drawing board. Therefore the final standard was still a "master model". A method had to be found to describe free form shape in a convenient, accurate and complete way to produce an indisputable definition. Drawing would still be created for the purpose of design or explanation, but numbers should stipulate the shape.

Introducing the mathematical geometric concepts of curves and surfaces lead to the birth of Computer Aided Geometric Design (CAGD). Even though basic work on the description of curves was already published in the early 40's by Schoenberg and Liming (working for the American aircraft industry), many concepts were developed independently and not published until many years later. First only simple forms were used: of course lines, then circular arcs that could be transformed into elliptic arcs defined by only three points. However connecting elliptic arcs was far too restrictive. Using a projector to project slides of different curve types onto a wall (not necessary orthogonally) to create a larger variety of shapes turned out to be not very practical. Nevertheless this idea lead to the introduction of mathematical models employing projective geometry and matrix computations.

To define the shape of a curved surface parallel cross-sections through the object were used. The surface in between these curves could then be defined by using a template (a curved plank tool) that would be swept along two neighboring curves. As a next step one tried to change the shape of the template while it was simultaneously moved along the curves. This lead to a mathematical concept using polynomial curves and a "characteristic net". See [16] for a detailed description of the historical aspects mentioned above.

The breakthrough in CAGD was the development of Bezier curves and surface patches by de Casteljau's at Citroen and by Bezier at Renault. Casteljau's work which was slightly earlier than Bezier's was kept secret (a technical report by Coons did not appear until 1967). Therefore the whole theory is named after Bezier. Further important work was done by Coons (MIT, Ford), Ferguson (Boeing), deBoor (GM), Birkhoff and Garibedian (GM) in the 1960s and Gordon (GM) and Reisenfeld in the 1970s. A continuation of this work was the combination with B-spline methods and later with rational B-splines: NURBS (see Section 4.4 and 4.5).

After the basic mathematical concepts were laid software and commercial applications could be developed. The SCETCHPAD system created by Sutherland 1963 at MIT was a (arguably the) prototype of a graphical user interface, with which a designer could interact with the computer graphically by drawing on a CRT display with a light pen. Since then commercial modeling systems have evolved drastically. The early 2D systems in the 1970s were only capable of creating virtual drawings. Advances in hardware, programming and solid modeling in the 80's lead to the boom of more sophisticated systems like CATIA (Dassault Systemes), AutoCAD (Autodesk) or Pro/ENGINEER (PTC) in 1988 (to name only a few). The development of boundary representation architectures (see Section 5) further advanced CAD by aiding the designer in the process to create topologically and geometrically correct 3D objects.

By now the field of CAD has evolved into different directions and has interwoven with many other areas. Advances in scientific sensor technology, the explosive growth in the number and resolution of instruments, have given us the possibility to obtain massive amounts of digital data describing our world (the earth's surface, the atmosphere, the oceans and even the structure of our human body). Geometric entities are constructed from scattered point clouds (for instance obtained from laser scans) or from voxel representations of some density distributions (see Section 3). Methods have been developed to store and compress such data efficiently on multiresolutional scales (for example wavelets, see Section 4.7).

Furthermore an object is not anymore simply designed for the purpose of manufacturing. In entertainment and virtual reality 3D objects are created to exist only virtually. They can be animated to reflect realistic behavior. Photorealistic rendering for example tries to create images of a virtual scene that can hardly be distinguished from reality. Typical problems are how to model hair or skin (we have already seen nice results in the recent animated movies), or complex scenes involving fire or smoke.

To accomplish these task differential equations governing these processes have to be solved, which leads to another field with an increasingly tight connection to CAD: Computer Aided Engineering (CAE) employing methods like the Finite Element Analysis (FEA). CAE aids a designer not just to create a virtual model, but to analyze its potential behavior (where it might break, how air flows around it, how it reacts to heat etc.) in order to optimize the design. Virtual crash test of a car for example lead not only to a better understanding of the process and consequently to a safer design but also save much money for conducting real tests with prototype models.

Future modeling systems will need to create a virtual reality allowing the designer to naturally and intuitively interact with his virtual model. Moreover they will have to be able to combine CAD with CAE in a way that allows the designer/engineer to apply his analysis directly on the exact geometry representation (without any conversion or approximation) and to optimize the shape afterwards by integrating the results (for instance from FEA) directly into the original model.

2 Architecture of modeling systems

It is not easy to define a modeling system. A modeling system can be every system to model a 2D or 3D object. Still many designers model with clay, hence from their point of view a modeling system would be pencil, paper, clay and the designer himself. In the area of computer graphics one is mainly interested in a virtual model of the object, which can be viewed from different perspectives, modified and processed further, to simulate the behavior of the object in

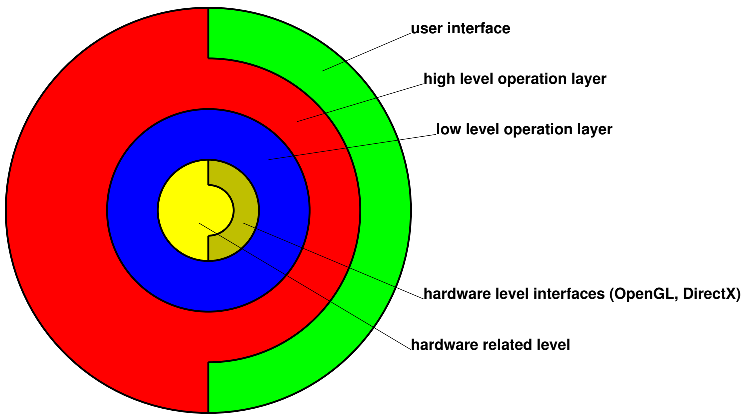

user interface
high level operation layer
low level operation layer
hardware level interfaces (OpenGL, DirectX)
hardware related level

Figure 1: Software levels of a modeling system

reality. Here a modeling system consists of the computer hardware and software and of the user.

Before choosing or building a modeling system an appropriate model has to be found for the design problem at hand. We will discuss different types of models in the following chapters but we will not go into detail on how to map a given real world problem onto one of these models. Please refer to [25] for some insights on how to accomplish this.

Today a strict separation of physical modeling for example with clay and virtual modeling with a computer cannot be sustained, since many mixtures are used in practice. Clay modelers for example often use a 3D-scanner to create a virtual model and on the other hand virtual models can easily be printed with a 3D-printer to create three dimensional prototypes. Recently even stronger connections are made using haptical devices and 3D-glasses to enable the user to feel and see the object in 3D space.

Since the user is still the most important part of a modeling system, the interaction between the human and the computer plays a crucial role. Therefore different hardware tools like scanners, printers, viewing, and input devices have been developed to interact with the user. The software is then needed to ensure the smooth interaction of all components.

The software of a modeling system can be divided into four abstraction layers (see figure 1):

1. The *user interface* (UI) is the part of the software that interacts directly with the user. The UI is mostly graphical and presents the user with many options to create, modify, analyze and view the object. Constructing a graphical UI is a complex venture where not only the demands of the user have to be taken into consideration, but also the possibilities of the hardware. It is important, that frequently repeated operations do not consume too much time and that the user is constantly informed about the status of any operation. An intuitive layout (of buttons, menus ...) should also be kept in mind.
2. The *high level operation* layer hosts mainly complex operations like intersecting, cutting, modifying, analyzing and post processing of objects. These operations can be accessed

through the user interface and can be understood as the main modeling tools. They should be robust and efficient to supply powerful tools to achieve every option the user has in mind.

3. On the *low level operation* layer the data structure is located together with its low level operators. These operators provide the next higher level with the controlled access and modifying options of the data structure. They keep the data in an organized state. Since the data structure and its operators are strongly connected an object oriented programming language like C++ is well suited for the implementation.

4. The *hardware related level* is the lowest layer. Here the interaction with the input- and output-hardware devices is implemented. Sometimes it is necessary to directly program the hardware (driver programming, assembly language, etc.) to elicit the needed features, but most of the time it is sufficient to use existing drivers and interfaces (e.g., OpenGL or DirectX).
   Another important aspect that needs to be dealt with on the lowest layer is the precision of operations. Since floating-point arithmetic is only approximate, but not precise, small errors may accumulate and possibly lead to catastrophic failure. Therefore provisions have to be made to prevent this failure or an exact arithmetic has to be implemented, unfortunately leading to a slowdown of the entire system.

We have seen that the data structure and its operators form the heart of the modeling system (level 3). Therefore the data structure determines the feasibility and performance of the high level operations. Many different types of data structures exist, each with its own advantages and disadvantages.

More on modeling systems (with an approach slightly different from the one presented here) can be found in [19].

3 Voxel representation

A typical volume based approach in modeling is the *voxel representation*. [25] refers to these kind of models as "sugar cube blobs" since these models can be thought as a set of sugar cubes glued together appropriately. This concept is a straight forward generalization of pixel graphics as known from computer graphics. Whereas in pixel representations a 2D image is discretized into a set of squares with integer coordinates (the pixels), in voxel representations 3D space is split into a regular cubic grid consisting of voxels. The easiest way of representing an object like this is to assign to each voxel a Boolean value, deciding if the volume described by the voxel is part of the object or not. Figure 2 shows a typical $12 \times 12 \times 12$ representation of a full sphere. A problem of the voxel based approach is, that the approximation of the objects volume at low resolutions is usually relatively poor, while higher resolutions increase memory consumption at a cubic rate. As a compromise for voxels intersecting the boundary of the object, the Boolean values can be changed to fuzzy numbers depending on the volume of the intersecting part of voxel and object. Additional attributes as described later can also be assigned to voxels. In medical imaging applications or geology for example density distributions are often represented as gray values assigned to each voxel. Furthermore voxels do not

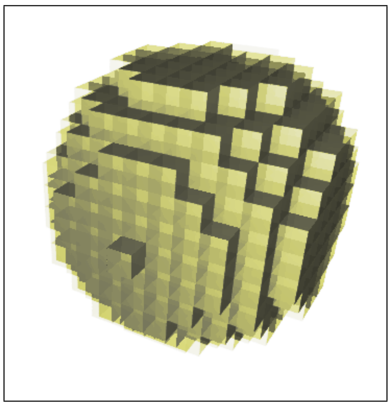


Figure 2: Voxel representation of a full sphere

necessarily need to be cubes, but can be cuboids with different sidelengths.

Voxel representations make Boolean operations like intersection or union of two objects extremely easy. Only the corresponding Boolean operations for their assigned voxel values need to be carried out, for example the logical **"and"** for the intersection. Again this is relatively costly at a higher resolution due to the enormous number of voxels involved.

The voxel representation method can be viewed as a special case of the CSG technique discussed in section 6 with only one primitive (the cube at integer coordinates) and one operator (the union).

Voxel representation is also known as *spatial-occupancy enumeration* (see [17]).

3.1 Octrees

Voxel representations can become memory consuming if a greater level of detail is desired. Thus sometimes voxels are organized into octrees. These are trees where each node is of degree eight or zero. Octrees are obtained by starting with one voxel large enough to enclose the whole object, representing the root node of the octree. In general this voxel is a poor approximation of the object, therefore this voxel is divided into eight equal sized smaller voxels, representing the child nodes of the root node. For each voxel an approximation criterion is checked, for example if the voxel intersects the boundary of the object. If this criterion is met, it is subdivided further, otherwise subdivision is omitted. This process is repeated for those voxels, that require further subdivision until the desired level of approximation is reached.
To understand the way an octree is obtained refer to figure 3. Here the octree's 2D analogue, a quadtree, for a triangle is constructed. The resulting quadtree is shown in figure 4. Note that only squares (and thus nodes) contributing to a higher level of detail are to be refined in a following step. Hierarchical representation schemes like octrees make tasks like collision detection particularly easy: First the two root nodes need to be checked for intersection. If and only if an intersection is found, the child nodes belonging to the respective objects are

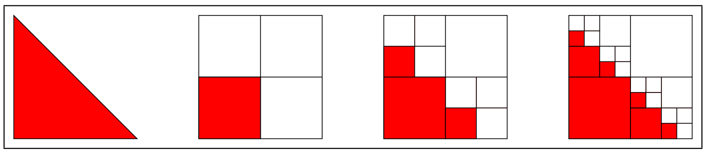


Figure 3: A quadtree for a triangle

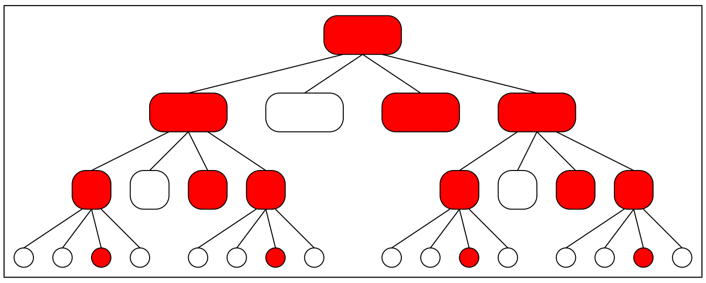


Figure 4: Resulting quadtree for the triangle

checked and so on. This is almost as easy as in the voxel case while being far more efficient. For an overview on how to implement Boolean operations for octrees see [17]. For a comprehensive survey of octree related techniques see [44].

Both voxel and octree representations may require conversion to a boundary representation before FEM computations can be carried out. This conversion can be accomplished using the famous *marching cubes algorithm* ([30]). Figure 5 depicts the result of such a conversion.

4 Surface patches

Surface patches form the base for boundary representation schemes. Therefore before we can discuss the foundations of boundary representations in section 5 we need to know, how to model a surface – the boundary of our object.

We define a *surface patch* to be a connected two-dimensional manifold in 3D. A surace patch is basically a set of points in 3D, where each inner point has a small surrounding neighborhood homeomorphic to the 2D open disc. We define the boundary of this set to be part of the surface patch.

There exists quite a huge variety of surface patches matching this definition and most of them are not easily represented by a data structure. Furthermore we should keep in mind that surface patches are usually meant to be ”glued” together or identified along their boundary (see section 5) in order to form more complex surfaces. Therefore they are mainly simple bounded and bordered manifolds. Nevertheless we shall give no formal definition of simplicity but rather present a selection of commonly used techniques for implementing special classes of

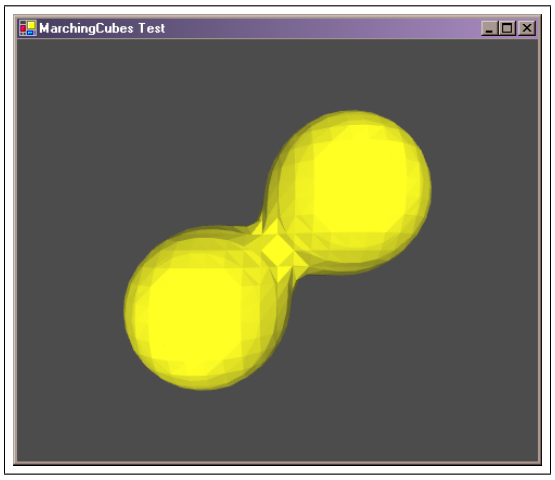


Figure 5: Result of a marching cubes conversion

surface patches.

For a detailed discussion of many of the topics mentioned in this section refer to [20]. Also refer to [25] for some deeper insights on surface patches.

4.1 Polygonal patches

Given a sequence of coplanar points $p_0, \ldots, p_n$ in 3D we define the sequence of edges joining two points $\overline{p_i, p_{i+1}}$ plus the edge $\overline{p_n, p_0}$ to be the *closed polygon* of the points. We define the *geometric interior* of the polygon to be the set of all points in the same plane, that cannot be reached by an arbitrary path from a point far away in the same plane (from a point outside the convex hull of the polygon) without crossing the polygon. We will consider every geometric interior point plus the boundary of these points to be part of the polygonal patch.

Of course this definition gives no efficient algorithm for testing, if a point belongs to the polygonal patch. There exists a variety of methods to do this ([17]), each meeting our definition in special cases (and not in others). For instance some efficient algorithms fail, if the polygon is not simple, meaning it possesses self intersections.
Figure 6 shows different polygons with their geometric interior painted red. It is often desirable to allow polygons to have inner boundary components (i.e., inner parts of the boundary that are not directly connected to the outer boundary). We will refer to these as *inner loops* (see also section 5). Figure 7 shows a rectangular polygon with two inner loops. Note that these polygons also match our definition of a surface patch.

Because every polygonal patch can be decomposed into a set of triangular patches, it is sometimes sufficient to consider only triangular patches, yielding its *triangulation*. These can be handled very efficiently since they are always convex and planer. Especially inside-outside testing and various other calculations are easily carried out for triangles. Nevertheless since

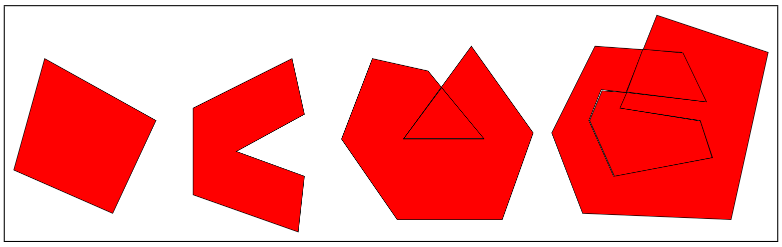


Figure 6: Geometric interior of polygons

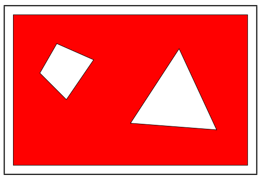


Figure 7: Polygon with inner loops

a triangulation is not unique for a patch, a chosen triangulation sometimes introduces difficulties into these calculations. Often elaborate meshing techniques introducing new vertices and yielding almost equilateral triangles need to be applied. Therefore more general schemes allowing also non-triangular patches should be carefully considered as well.
Triangulation is a special case of general *meshing techniques*, approximating a curved surface with planar polygons. Figure 8 shows an example of an object composed of polygonal patches (here only triangles).

4.2 Parametric surfaces

Often we want the surface to be really curved instead of just (piecewise) planar like in the polygonal case. This can be achieved employing *parametric surfaces*.
Let D be a subdomain of 2D space and let $f : D \to \mathbb{R}^3$ be a continuous map. Often we require f to be differentiable, mostly f will be a homeomorphism onto its image set $f[D]$ and generally we assume the differential of f having maximal rank. We will call the pair (D, f) a parametric surface with parametrization f, the surface patch is represented by the image of f[1].

Note that we assume no further restrictions for D, allowing explicitly every planar polygon (in 2D), even with inner loops. This is because it can be sometimes intricate to find parametrizations for special surfaces. For instance a ring-like structure as depicted in figure 9 can easily be represented by the domain

$$[-1,1] \times [-1,1] \backslash \left[-\frac{1}{3}, \frac{1}{3}\right] \times \left[-\frac{1}{3}, \frac{1}{3}\right]$$

[1]Topologists call $f[D]$ the image set of f and $\mathbb{R}^3$ the range of the map. Analysts often call $f(D)$ the range of the map and do not introduce a special name for the image set of f.

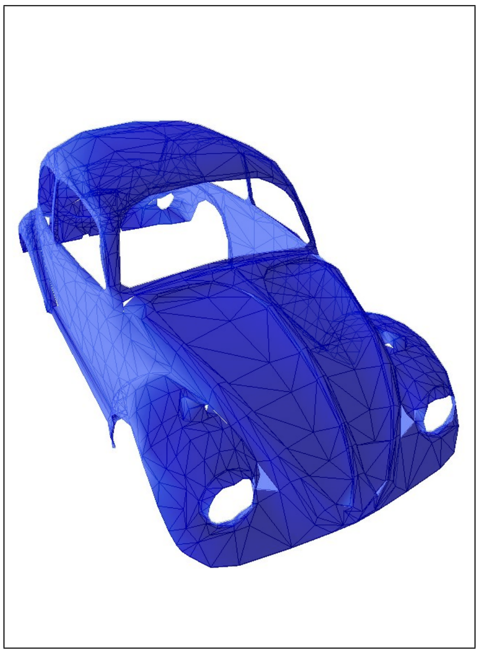


Figure 8: Triangulated object

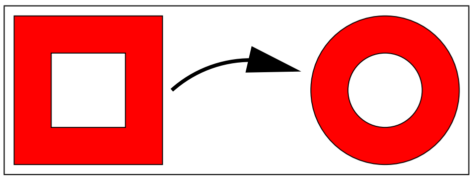


Figure 9: Parametrization of a planar ring

with parametrization

$$f(x,y) := \frac{\max\{|x|,|y|\}}{\sqrt{x^2+y^2}} \begin{pmatrix} x \\ y \\ 0 \end{pmatrix}$$

Nevertheless most of the time D will be polygonal or, for convenience, the unit square. The spline surfaces discussed in section 4.4 are a popular example of parametric surface patches. An alternative method is to model using partial differential equations. This elaborate technique was developed by M. Bloor and M. Wilson at the University of Leeds and is described in [36].

4.3 Trimmed surfaces

We have just seen that the domain of a parametric surface is not necessarily the unit square. We can generalize this principle by cutting out polygonal and even non-polygonal subdomains (e.g., domains bounded by splines curves) from the parameter space and thus trimming the surface patch itself. This process is depicted in figure 10, where a user selects a closed curve in parameter space (green) which is then removed from the parameter space and thus trimmed

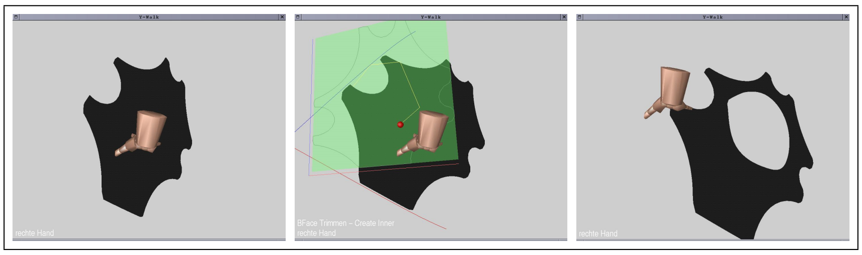


Figure 10: Trimming of a surface patch: the surface patch (left), the superimposed parameter space is being edited (middle), the resulting trimmed surface patch (right)

out of the surface patch (black). Note that trimming the surface patch directly (instead of the parameter domain) is a rather complex task since it theoretically involves the computation of the inverse f^{-1} of the parametrization f (see [37]).

4.4 Spline surfaces

It is well known that for a given set of 3D points $p_0, \ldots, p_n$ there is a unique polynomial curve

$$\alpha(t) = \sum_{i=0}^{n} c_i t^i$$

with $c_0, \ldots, c_n \in \mathbb{R}^3$ such that α interpolates every point p_i. We refer to the points c_i as *control points* of α.
Polynomial interpolants suffer from three major drawbacks:

- They tend to form unexpected "swinging" curves that can move far away from the interpolation points.
- Construction and evaluation of these curves are numerically unstable.
- There is no intuitive interrelation between coefficients of a polynomial and the shape of the resulting curve or surface.

Splines try to overcome these problems by two basic techniques:

- Use a type of curve, that is numerically and visually more "tame" (i.e. closer to its interpolation points).
- Compose the curve piecewise from sub-curves.

All different types of splines are obtained from piecewise sub-curves, they only differ by the base type chosen for these curves. Best known and widely used are Hermite-splines, Bezier-splines, B-splines and NURBS, and of course monomial-splines (where each sub-curve is an ordinary polynomial curve). These are all curves of piecewise polynomial type (except the NURBS which are piecewise rational polynomial). Furthermore there are non-polynomial types like trigonometric splines, exponential splines or splines based on sub-curves obtained from other subdivision processes (which are not necessarily polynomial), although these are more rarely used as they may be computational costly.
Formally we will call a curve a spline (of degree n) if it is

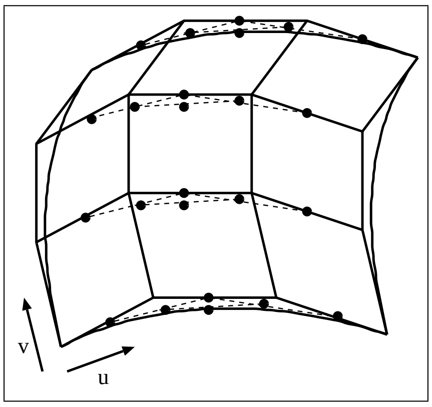


Figure 11: Control array of a spline surface

a) piecewise composed of the same type of sub-curve belonging to the same finite dimensional vector space of functions[2],

b) at least $n-1$ times continuously differentiable.

Note that depending on the type of spline chosen we often need additional control points besides the interpolation points to characterize the curve completely.

Using the techniques from subsection 4.2 one can easily obtain *spline surface patches* from spline curves. Given an array of control points (c_{ij}) with $i \in \{1, \ldots, n\}, j \in \{1, \ldots, m\}$ each sequence $c_{1j}, \ldots, c_{nj}$ defines a spline curve α_j, which evaluated at a certain point x yields a further sequence of control points $\alpha_1(x), \ldots, \alpha_m(x)$. These form a spline β_x which can then be evaluated at a point y, this way giving a resulting range point. This process describes a map

$$f(x, y) := \beta_x(y)$$

where f in fact is a parametrization of a surface patch.
Figure 11 shows a control array and the underlying spline surface.

For a detailed overview on spline techniques see for example [16], [20] and [54]. A recent reference can be found in [37], this book also deals with problems of spline surface intersections, which are important when splines are combined with CSG representations (see section 6). Certainly the most elaborate spline technique is based on *NURBS* (non uniform rational B-splines), see [41] for a comprehensive overview.

Figure 12 shows a wrench modeled with piecewise B-spline patches.

4.5 B-splines and NURBS

We will now explain the construction of spline curves and surfaces in more detail at the example of B-splines. B-splines and NURBS are commonly used in current CAD systems. Their ability

[2]Note that the subcurve is in most cases C^∞.

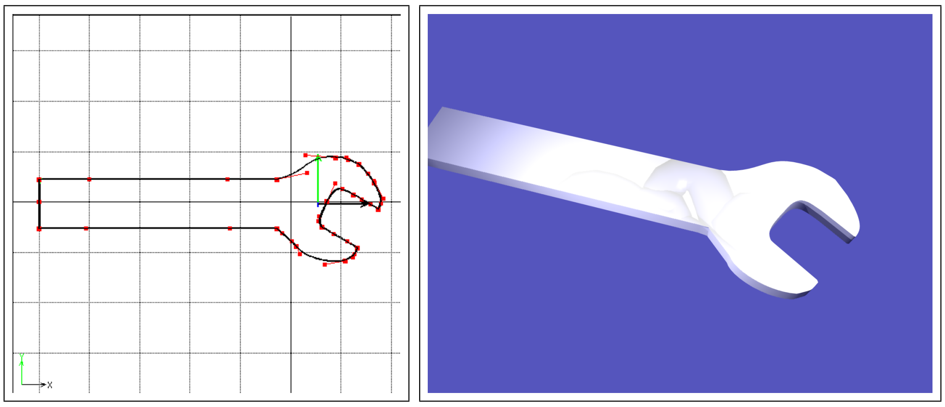


Figure 12: Wrench composed of B-spline patches

to model smooth objects as well as sharp corners suites them perfectly for design applications. NURBS (non uniform rational B-splines) advance the B-spline technique to allow rational weights, thus allowing even more freedom for the designer to model, for instance, perfectly round disks or tori, that cannot be created by piecewise polynomial curves or surfaces. B-spline as well as NURBS curves and surfaces can be controlled and modified easily by adjusting the corresponding *control points* and their *knot vector*. We will explain these terms in the following sections.

4.5.1 Basis functions and knot vectors

Let $T = (t_1, t_2, \ldots, t_{n+d+1})$ be the so called *knot vector* with $t_i \in \mathbb{R}$ and $t_0 \leq t_1 \leq \ldots \leq t_{n+d+1}$. The components of this vector t_i will be called *knots*. They represent a set of coordinates in the parametric space. Here n is the number of basis functions and d is their polynomial order, as we will see the number n will correspond with the number of control points of a B-spline that we will construct with the help of the basis functions later. We now define the basis functions starting with piecewise constant

$$N_i^0(t) \quad := \quad \left\{ \begin{array}{ll} 1 & \text{for } t_i \leq t < t_{i+1} \\ 0 & \text{else} \end{array} \right\} \quad \text{for } i = 1, \ldots, n+d$$

and with higher degrees d

$$N_i^d(t) \quad := \quad \frac{t - t_i}{t_{i+d} - t_i} N_i^{d-1}(t) + \frac{t_{i+d+1} - t}{t_{i+d+1} - t_{i+1}} N_{i+1}^{d-1}(t)$$

for $i = 1, \ldots, n$. These recursively defined functions $N_i^d(t)$ are called *basis functions* of polynomial degree d.

We see that the knots define the intervals of the parameter space on which the basis functions live. If the knots are equally spaced, they are called *uniform* otherwise *non-uniform*. If we use the convention $\frac{0}{0} := 0$ in the definition of the basis functions, we can allow knots to be located

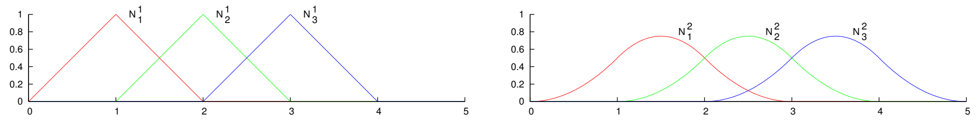


Figure 13: Basis functions of degree 1 and 2 on uniform knot vector

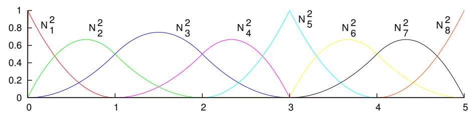


Figure 14: Basis functions of degree 2 on non-uniform knot vector

at the same position. Such knots are called *repeated* knots. If the first and last knot are each repeated $d + 1$ times the knot vector is said to be *open*.

Figure 13 shows the basis functions of degree 1 and 2 on a uniform knot vector. For uniform knot vectors the functions $N_i^d(t)$ are identical except for translation. Generally basis functions of degree d have $d - 1$ continuous derivatives, for example, the quadratic functions are C^1 continuous everywhere. This continuity can be decreased by repeating a knot in the knot vector. The basis function becomes interpolatory if the multiplicity of an inner knot reaches d as can be seen in the example in figure 14 where we used a non-uniform, open knot vector. Note also that because the multiplicity of the first and last knot is $d + 1$ the functions are interpolatory at the ends.

B-spline basis functions have many practical properties. We will only present the two most important properties here:

1. Support interval: The support of N_i^d is $[t_i, t_{i+d+1})$ meaning that $N_i^d(t) = 0$ for $t \notin [t_i, t_{i+d+1})$.
2. Partition of unity: $\sum_{i=1}^{n} N_i^d(t) = 1$ for $t \in [t_{d+1}, t_{n-d})$

4.5.2 B-spline Curves

By constructing linear combinations of the basis functions, it is possible to construct *B-spline curves*:

$$C(t) = \sum_{i=1}^{n} c_i N_i^d$$

The n coefficients c_i are called *control points* and are normally points in $\mathbb{R}^2$ or $\mathbb{R}^3$ defining 2D or 3D curves (even though they could be of higher dimension as well). If we connect the control points with line segments, we speak of the *control polygon*. In Figure 15 you see a B-spline curve of degree $d = 2$ together with its control polygon. We used the same open knot vector

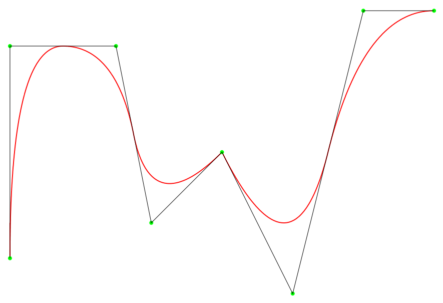


Figure 15: B-spline curve of degree 2 with basis functions and knots as in Fig. 14

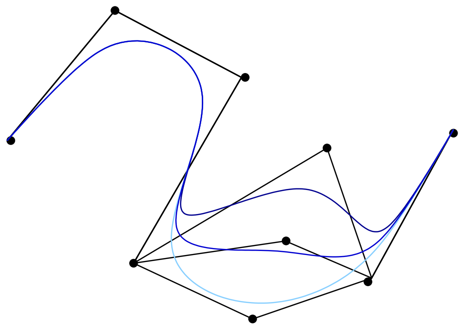


Figure 16: Local influence of a control point

described earlier to demonstrate the interpolation of the first and last control point and of the point c_4 where we raised the multiplicity of the corresponding knot to the polynomial order $d = 2$.

Some important properties of B-spline curves are:

1. Continuity: Generally B-spline curves of degree d are C^{d-1}. If the multiplicity of a knot is raised to k, the continuity decreases to C^{d-k}.

2. Convex Hull: A B-spline curve is always contained in the convex hull of its control points.

3. Locality: Moving a control point changes the curve only locally, that is, changing c_i changes $C(t)$ only for $t \in [t_i, t_{i+d+1})$ (see figure 16).

4. Affine Transformation: Applying an affine transformation to the curve can be done by applying it to the control points.

5. End Point Interpolation: The choice of knots in case of an open B-spline curve ensures start and end point interpolation.

It is possible to represent the same B-spline curve with a higher amount of control points by applying *knot insertion* or *degree elevation*. Knot insertion simply represents the same curve with one additional control point by adding a new knot in the knot vector (resulting in $n + 1$ basis functions). The degree elevation on the other hand increases the polynomial degree of the basis functions using the recursive definition. In this case the unique knot values need to be repeated in order to keep the discontinuities in the derivatives of the curve. Therefore the multiplicities of existing knots accounts for the number of newly created control points. For a detailed description of knot insertion and order elevation refer to [16] and [20]. It should be noted that both methods yield a control polygon that approximates the curve much better than the original.

4.5.3 B-spline Surfaces and Solids

Similar to B-spline curves *B-spline surfaces* can be constructed. For this purpose we need a *control net* $c_{ij},\ i = 1, \ldots, n,\ j = 1, \ldots, m$ and two knot vectors $U = (u_1, u_2, \ldots, u_{n+d_1+1})$ and $V = (v_1, v_2, \ldots, v_{m+d_2+1})$. Using a tensor product of B-spline basis functions $N_i^{d_1}$ and $M_j^{d_2}$ the B-spline surface can be defined by:

$$S(u, v) = \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} N_i^{d_1} M_j^{d_2}$$

This method can easily be continued to define tensor product *B-spline solids* using a 3D control net, three knot vectors and three basis functions. Sometimes it makes sense to use even more variables and high dimensional control points to represent multivariate non linear polynomial systems geometrically (see [37] for an application using Bezier curves). In order to solve such systems one can take advantage of the B-spline properties (especially the convex hull property).

4.5.4 NURBS

NURBS is short for *non uniform rational B-splines.* NURBS curves are an extension of B-splines that are superior mainly because of two aspects. First, non-uniformity allows better approximation of curves and surfaces while using less control points (see [52]). Second, important geometric objects like circles, ellipses (conic sections in general) cannot be represented by a B-spline curve. Nevertheless they can be constructed exactly by projecting three dimensional piecewise quadratic B-spline curves into $\mathbb{R}^2$ (see figure 17), which can be done by representing the curve as a rational polynomial $C(t) = g(t)/h(t)$ with two piecewise polynomial functions g and h.

In order to construct a NURBS curve in $\mathbb{R}^k$ we use a set of control points c_i^w (called *projective control points*) for a B-spline curve in $\mathbb{R}^{k+1}$ with corresponding knot vector T. To get the control points of the NURBS curve in $\mathbb{R}^k$ we simply take the first k coordinates of c_i^w and divide them by the $k+1$ coordinate that we will call w_i (the i-th weight):

$$\begin{array}{lll} c_i^w & = (x_1, x_2, \ldots, x_k, w_i) & \in \mathbb{R}^{k+1} \\ c_i & = (\frac{x_1}{w_i}, \frac{x_2}{w_i}, \ldots, \frac{x_k}{w_i}) & \in \mathbb{R}^k \end{array}$$

This is in fact a central projection of the points c_i^w onto the plane at $x_{k+1} = w_i = 1$. We can now define the rational basis functions and the NURBS curve:

$$R_i^d(t) = \frac{w_i N_i^d(t)}{\sum_{j=1}^n w_j N_i^d(t)} \quad \text{and} \quad C(t) = \sum_{i=1}^{n} c_i R_i^d$$

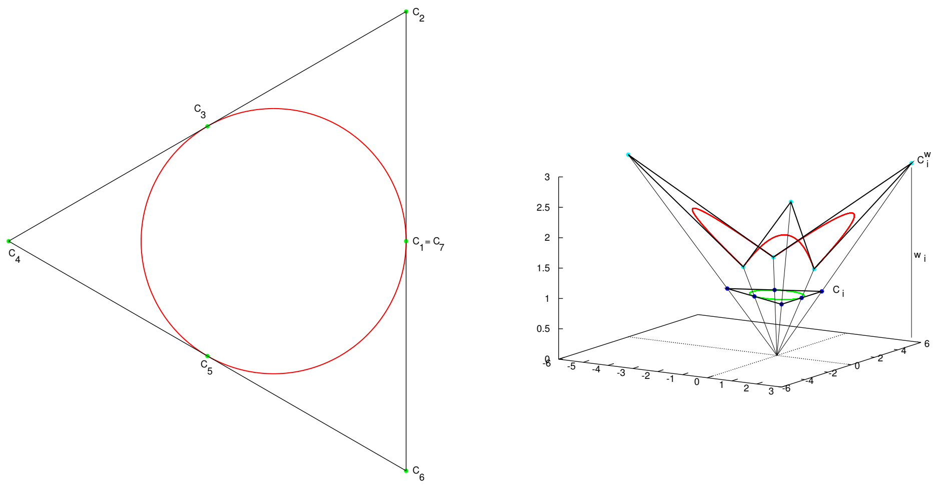


Figure 17: NURBS curve in projective space with weights w_i in the z-coordinate

NURBS surfaces (and solids ...) can be constructed in a similar way by employing the basis functions:

$$R^d_{ij}(u,v) = \frac{w_{ij}N_i^{d_1}(u)M_j^{d_2}(v)}{\sum_{k=1}^{n}\sum_{l=1}^{m} w_{kl}N_k^{d_1}(u)M_l^{d_2}(v)}$$

4.6 Subdivision surfaces

B-splines address most of the problems arising with the use of simple polynomial curves and surfaces. In particular they provide a set of control points that is closely related to the final shape of the computed curve or surface. This idea of generating the final shape in a geometrical sense from a set of underlying control points can be generalized to the concept of *subdivision schemes.*

Let there be a (possibly infinite) sequence $r = (\dots, r_{-1}, r_0, r_1, \dots)$ of real weights and a sequence $c_0 = (\dots, c_{0,-1}, c_{0,0}, c_{0,1}, \dots)$ of real numbers. Then a function can be iteratively defined by:

$$f_0(i) := c_{0,i}$$
$$f_j(i/2^j) := \sum_{k=-\infty}^{\infty} f_{j-1}((i+k)/2^j)$$

Each f_j is defined on successively finer subsets of $\mathbb{R}$ (i.e., f_0 is defined on the integers only, f_1 is additionally defined on the midpoints between two integers and so on). One should notice that for odd k the value of $f_{j-1}((i+k)/2^j)$ is initially undefined. Thus we set this value to $1/2(f_{j-1}((i+k-1)/2^j) + f_{j-1}((i+k+1)/2^j))$.
We can now define a limit function

$$f := \lim_{j\to\infty} f_j$$

over a dense subset of $\mathbb{R}$, which means that for each $x \in \mathbb{R}$ we find an arbitrary close $\tilde{x}$ where f is defined. We call this function f the **subdivision function** for given control points c_0 and subdivision weights r. It is well defined in case the weights admit pointwise convergence for $f(x)$.

The best known example of a subdivision scheme is the *Chaikin subdivision* with weights $r = (1/2, 1/2)$. Figure 18 shows an example of four iterations of a Chaikin subdivision. We

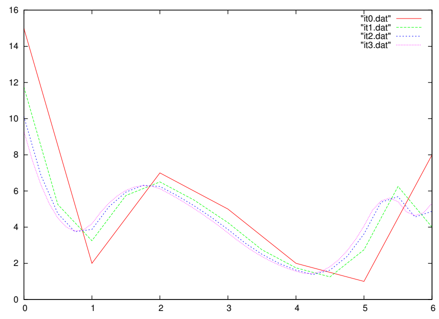


Figure 18: Four iterations of a chaikin subdivision

observe that the resulting curve is smoothed by further iterations, while still resembling the original control points. However, this does not apply to all kinds of subdivision schemes.

The following classes of subdivision schemes are distinguished:

- *Stationary schemes* where the weights r do not depend on the iteration j,
- *uniform schemes* with the weights independent of the position of evaluation i.

Most of the uniform schemes are not endpoint interpolating, which would be highly desirable in most applications. Therefore they have to be modified at the boundaries of the parameter space, to yield non uniform but endpoint interpolating schemes. Very little is known about non stationary schemes, most schemes applied today are in fact stationary.

Subdivision schemes are deeply interlinked with wavelets that we will look at in section 4.7. We should only stress here that certain subdivision schemes yield classes of well known functions. For example for $r = 1/2(1,1)$ we get the Haar functions (see figure 22) and for $r = 1/4(1,2,1)$ cubic B-splines are obtained. In fact, it turns out that functions representable by wavelets (see section 4.7) are just the subdivision functions.

So far we have only dealt with constructing functions from a set of control points. Nevertheless, building curves and surfaces from the obtained functions is straightforward. Just like with B-splines, one can define a parametric curve as a vector (f_x, f_y) of two subdivision functions f_x, f_y. A surface patch can be generated using tensor products, as described in section 4.4.

A disadvantage of using tensor product surfaces is that one is limited to surface patches generated by regular quad meshes. In order to extend the ideas of subdivision to triangular meshes

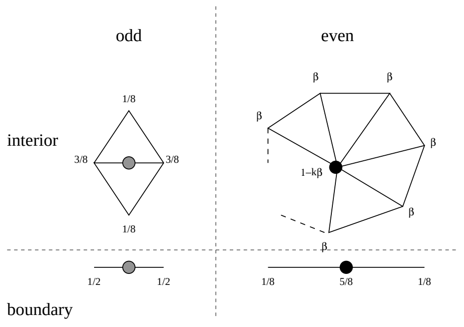


Figure 19: Masks for Loop subdivision

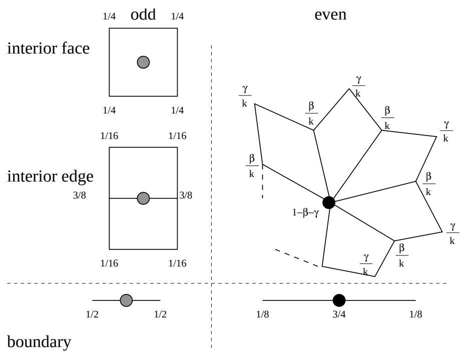


Figure 20: Masks for Catmull-Clark subdivision

a variety of special subdivision schemes for surface patches has been developed.

For subdividing a mesh two principal approaches exist: subdividing the faces of a mesh (primal subdivision), or subdividing the vertices (dual subdivision). Furthermore, most schemes make a distinction between vertices that are generated in the most recent subdivision step (odd vertices) and vertices that were already there (even vertices). To gain a limit surface that is, for example, C^1-smooth or interpolating one has to treat these vertices differently.

One of the most common schemes is the Loop subdivision. It is a primal scheme. Figure 19 shows the subdivision masks for splitting edges (generating odd vertices) and treating even vertices. The constant β in the Loop subdivision scheme can be chosen as $\beta = \frac{1}{k}(\frac{5}{8} - (\frac{3}{8} + \frac{1}{4}\cos\frac{2\pi}{k})^2)$ (see [29]) or $\beta = \frac{3}{8k}$ (see [48]). Loop subdivision results in an approximating, C^1-smooth limit surface ([55]). For a regular mesh the limit surface is even C^2-smooth.

While Loop subdivision is the best known scheme for triangular meshes, the most widespread technique for quad meshes is Catmull-Clark subdivision. Figure 20 shows the masks for this scheme. Catmull and Clark ([12]) suggest to choose $\beta = \frac{3}{2k}$ and $\gamma = \frac{1}{4k}$. Note that Catmull-Clark subdivision is defined for quad meshes only. This means if there are triangles or polygons with more than four vertices the mesh has to be subdivided into quads first (see figure 21). The limit surface for this scheme yields surfaces that can be represented by bicubic tensor product splines. For a regular vertex the surface is C^2-smooth, everywhere else C^1-smooth.

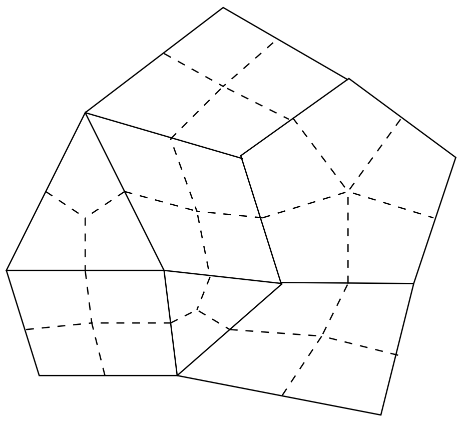


Figure 21: Decomposing a mesh into quads

Aside from Loop subdivision and Catmull-Clark subdivision a variety of different schemes has been developed in order to overcome specific disadvantages of the aforementioned methods, for example Doo-Sabin, Butterfly and Kobbelt. For an overview and some implementation details refer to [56].

4.7 Multiresolutional approaches

As we have seen in subsection 4.4 splines bring great improvements in curve and surface design. Nevertheless modern design and modeling applications may demand further features from a surface representation which are in detail (see [46]):

- easy approximation and smoothing of a representation, gained from external sources (such as scan points from a digitizer)
- changing the macro scale appearance without changing the micro scale appearance (the "character") and vice versa
- edit a representation on various, preferably continuous levels of detail

These issues can be addressed by using a relatively new idea, the so called *B-spline wavelets* instead of ordinary B-splines for curve and surface modeling.

The idea of a a wavelet-representation is to represent a curve using linear combinations of functions given in different levels of detail. Wavelets are employed using nested vector spaces of functions, that is, vector spaces V_i fulfilling

$$V_0 \subset V_1 \subset \ldots \subset V_n \subset \ldots$$

For each pair of spaces V_i and V_{i+1} there is a set of functions that enriches a base of V_i to a base of V_{i+1}. In other words, these additional functions span the othogonal space W_i of V_i with respect to V_{i+1}. These additional base functions are called *wavelets*, while the base functions of the hierarchical spaces V_i are called scaling functions. The *Haar base* functions shown in figure 22 are the best known examples. Thus we get a representation where a manipulation of a single coefficient has only relatively local impact, depending on the level. Since B-splines also

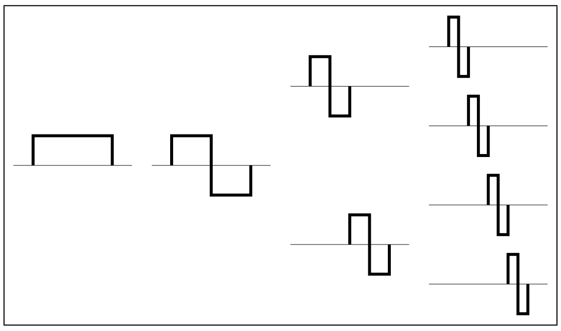


Figure 22: Haar base wavelets

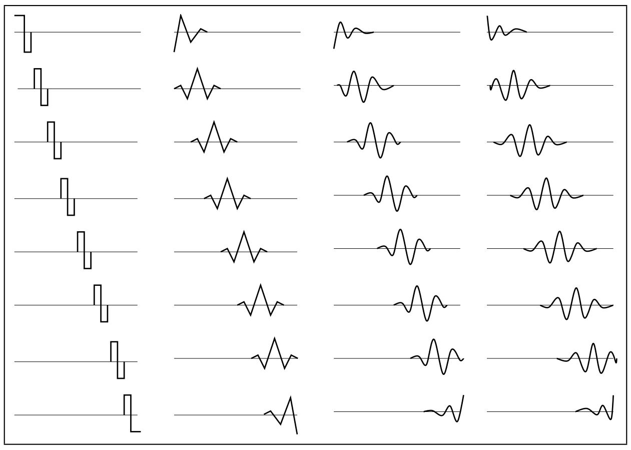


Figure 23: Some B-spline wavelets, adapted from [46]

form a vector space, wavelets can be built from them. Figure 23 depicts B-spline wavelets for degree 0 up to degree 3 for the third level of detail (thus giving $2^3 = 8$ functions per degree). For a comprehensive overview on wavelets from an analytical point of view refer to the book by [31].

5 Boundary representation

A boundary representation (B-rep) of a solid describes the solid only by its oriented boundary surface. The orientation is needed to decide easily which side of the boundary is the top side and which is the bottom side (even if the object is not closed). Since a normal vector is known everywhere B-rep solids can be visualized very easily.

Generally it is possible to use a variety of different surface patches to model the boundary. These patches (e.g., NURBS-patches, parameterized surfaces or simply planar polygons, see section 4) have to be connected with each other at their boundaries. The orientation must not be destroyed during this step.

In most applications planar polygons are used as patches (very often only triangles are permitted). These patches are called *faces*. Their borders consist of *vertices* and *edges*. Different data structures have been developed to hold the information necessary to create and work with a B-rep solid. The location and orientation (normal vector) of a plane containing a face have to be known and also the correspondence of the vertices and adjacencies of the edges and faces need to be controlled. Furthermore it is necessary to ensure the feasibilty (topological integrity) of a virtual object (guaranteeing that it can actually be build). A solid, for example, should not contain any holes, or its boundary patches should not be twisted like the Moebius strip.

The boundary representation of a solid therefore has two parts:

- the *topological description* of the connectivity and orientation and
- the *geometric description* to place all elements in space.

To understand how the topological data is maintained and how topological integrity can be ensured it is useful to understand *planar models* and *Euler operators* first. Later on a *half-edge data structure* is introduced as an example on how to implement a B-rep model.

5.1 Planar models

Planar models are useful to represent the topology of a solid. A planar model is a planar oriented graph $\{F, E, V\}$ consisting of a set of faces $F = \{f_1, f_2, ...\}$, edges $E = \{e_1, e_2, ...\}$ and vertices $V = \{v_1, v_2, ...\}$. Every edge has its orientation. If different faces share the same edges or vertices, they have to be identified with each other. Identified edges must show in the same direction. In other words, the directions of the edges imply the way they have to be identified. Note that with the half-edge data structure described in the next chapter one edge always consists of two half-edges showing in opposite directions. Here we only have a single edge, which can appear several times in a planar model. Figure 24 shows an example of the planar model of a tetrahedron. Figure 25 shows a model of the torus and the Klein bottle.

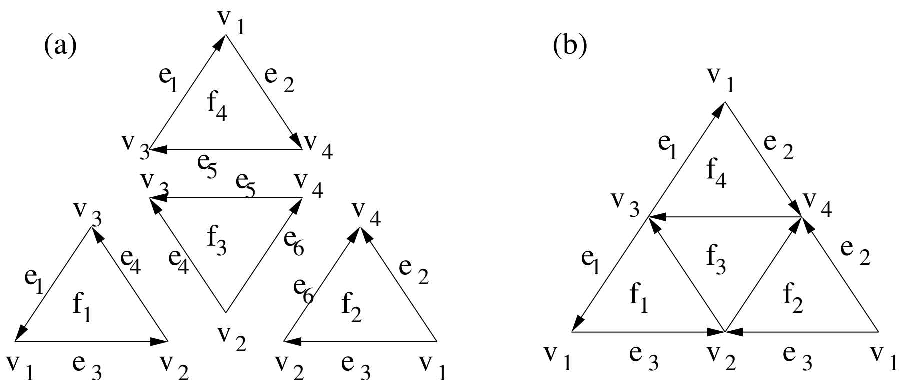


Figure 24: Planar model of a tetrahedron

The only difference between the two models in figure 25 is, that one of the edges e_2 points in the opposite direction. This results in a different identification. The two models therefore describe different objects.

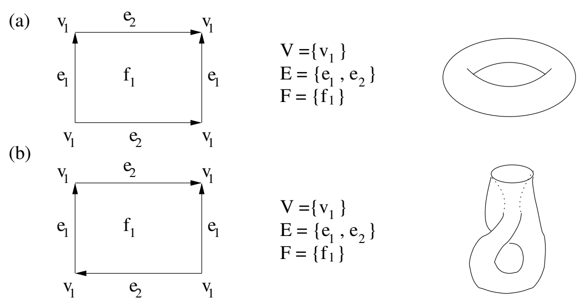


Figure 25: Planar model of the (a) torus (b) Klein bottle

A solid can have different planar models. An example can be found in figure 26, where two different planar models of the sphere are presented.

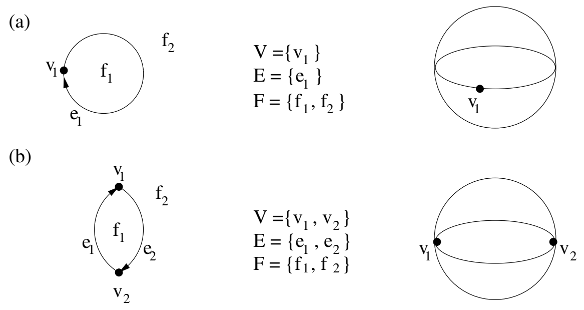


Figure 26: Planar models of the sphere

From a planar model the *Euler characteristic* can be calculated quickly: $\chi = |V| - |E| + |F|$. Here it does not matter which particular planar model of a solid is used. The Euler characteristic of the tetrahedron is $\chi_T = 4 - 6 + 4 = 2$, the characteristic of the torus and Klein bottle (figure 25) is $\chi_B = 1 - 2 + 1 = 0$, and of the sphere (figure 26) is $\chi_S = 1 - 1 + 2 = 2 - 2 + 2 = 2$.

Based on the planar models a set of topological operators was developed to manipulate models in a way that leads to all models of physical significance while making sure that only feasible models[3] can be created. These topological operators are split into two classes, the local and global operators.

Local operators work on planar models without modifying the Euler characteristic while global operators can create objects of higher genus (such as the double torus), thus changing the Euler characteristic. A detailed description and proofs of the properties of these operators can be found in [33].

[3]For instance non-orientable models cannot be created.

5.2 Half-edge data structure

In order to work with a B-rep model one needs to construct a data structure, combining geometric and topological data. Probably the oldest formalized structure, the so called *winged-edge* data structure, was introduced by [5]. The half-edge data structure (see 27) is a variation by [33] that permits multiple connected faces and sustains a close relationship to the planar models.

The half-edge data structure utilizes the fact that each edge of the boundary surface of a closed solid belongs to exactly two faces. So every edge is split into two half-edges which are oriented in opposite directions. Every face has exactly one outer boundary (outer loop) consisting of counterclockwise oriented half-edges (if viewed from above) and possibly further inner loops consisting of half-edges which are oriented clockwise. The orientation of the loops makes it possible to determine the top and the bottom side of each face. All vertices of a face have to lie on the same plane and are saved in three dimensional homogeneous coordinates. Every vertex has to be unique and can be referenced by many half-edges (depending on how many faces share that vertex). Since all half-edges know their neighbor and their parent loop, which again knows the parent face, finding neighbor faces and running over the data structure is quite easy.

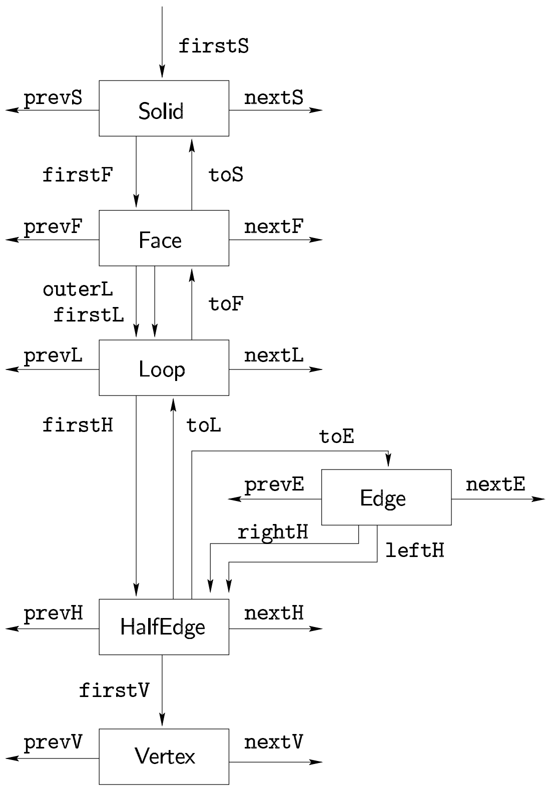


Figure 27: Half-edge data structure

A set of low- and high-level operators (the so called *Euler operators*) can be derived from the

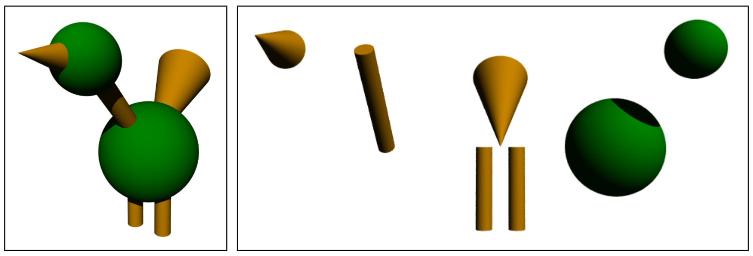


Figure 28: An object composed of CSG primitives

topological operators of the planar model (see the previous subsection). This permits operations on the data structure in an ordered manner. Any further operators can be implemented using the Euler operators, thus granting the technical feasibility of the modeled object.

6 Constructive solid geometry

One of the best known volume based approaches to modeling is the *constructive solid geometry (CSG)* approach. Again, refer to [17] or [19] for an overview on the subject.

In CSG every object is either one of a set of simple objects, the *primitives*, or it is derived from these by a sequence of operations. Various CSG schemes exist. They are different with respect to their sets of primitives and operations. In 3D modeling the most commonly used primitives are:

- ball,
- cylinder,
- box,
- cone.

Further possibilities include surfaces of revolution, implicit bodies and boundary representation objects. This shows that CSG can be combined with other methods (discussed above) to gain a greater variety of primitives.
A suitable set of operations must include:

- Euclidean motions (translation, rotation)
- union,
- intersection,
- difference.

The latter three are called *regularized Boolean operations* because they are analogous to the well known Boolean set operations with a slight difference we will discuss later. Let us first consider the example shown in figure 28. The object on the left side is composed of the primitives on the right side via the union operation. Note that parts of the objects that are inside

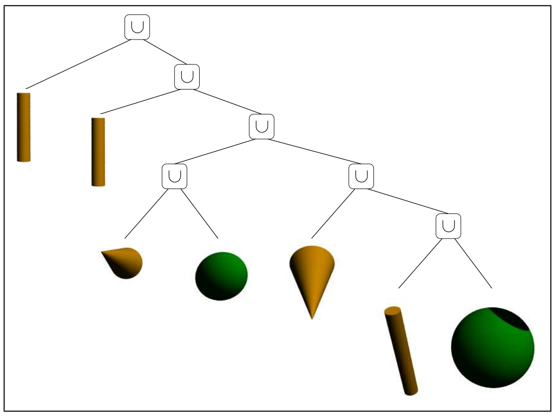


Figure 29: CSG tree for the bird object

other objects are "swallowed", there are no more overlaps or double points.

Internally composite objects are kept as binary operator trees. Figure 29 shows one of such trees, $\cup$ denotes the union operator. Obviously neither the sequence of operators nor the resulting tree is unique for a given result. Nevertheless by this way CSG keeps a history of the construction steps, hence every complex object can be decomposed into primitive parts again. This is not possible with most other methods.

Figure 30 shows another example, a wrench composed of the primitives shown on the right. Formally the regularized Boolean operators are defined as follows: Given two objects A and B and a Boolean set operator $\circ$. The result of the corresponding regularized Boolean operator $\bar{\circ}$ is defined to be $A \bar{\circ} B := \overline{A^\circ \circ B^\circ}$ where A° denotes the interior of A and $\overline{A}$ denotes the closure. This definition avoids the problem of Boolean operators to yield results that do not represent 3D objects. For example, the intersection of two adjacent boxes sharing one side would normally result in just that side, returning a non 3D object.

Sometimes further operations are included like non-Euclidean matrix operations (scaling, skew transforms) or surface sweep operations. One has to take care that these additional operations are applicable for the given primitives. It is mostly impossible to apply sweep operations to objects given in implicit representations while keeping their implicit representation.

CSG is best suited for ray tracing and other applications that require only inside-outside testing as these can be easily carried out in a CSG representation. Also voxel representations can be gained easily from a CSG representation. On the other hand CSG can not easily be applied in situations that require a meshing of the given object, for instance, for FEM computations since this demands the elimination of unnecessary ("swallowed") object parts first which can be costly. A method that avoids these computations is to apply the marching cubes algorithm ([30]). This only requires testing if an arbitrary point is inside the interior of an object, which is easily possible for a CSG representation. One major drawback of this method is that

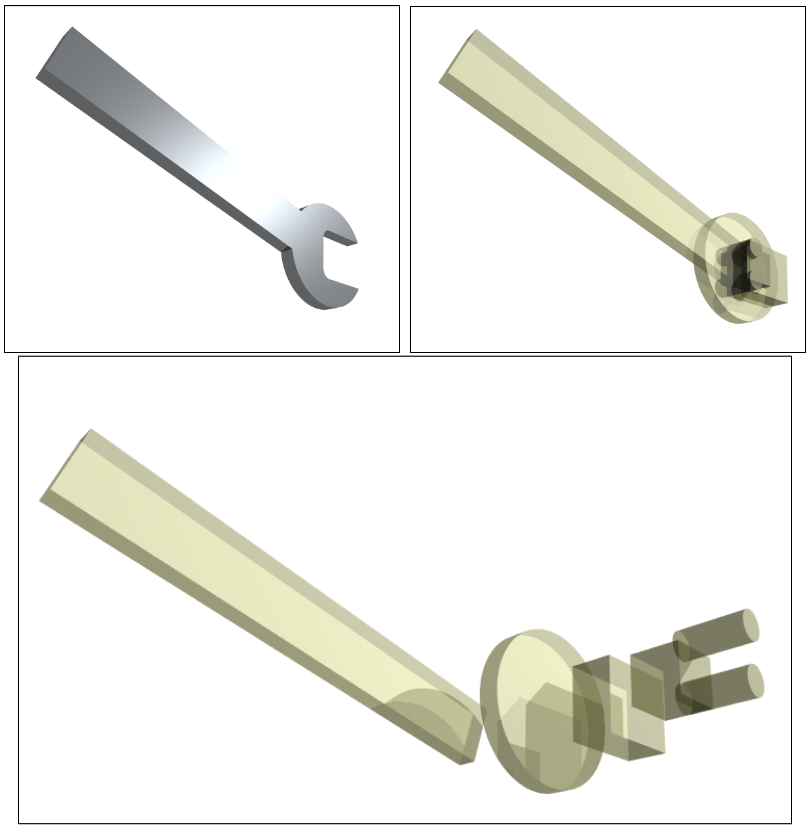


Figure 30: A CSG wrench model

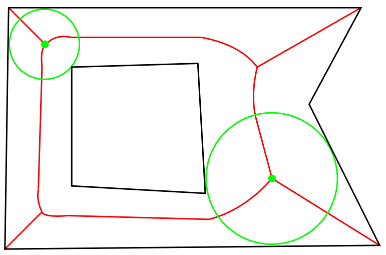


Figure 31: Medial axis of a domain

important details of the model might be lost. Another method would be to mesh each primitive separately[4] and join these[5] to form the mesh.

7 Medial modeling

Medial modeling is the newest one among the modeling concepts presented in this survey paper and is essentially based on research of the Welfenlab, the authors research lab at the University of Hannover (see [53] for a brief overview of results). The suggestion to use the medial axis concept as a tool for shape interrogation appears to have been discussed first quite extensively by [7]. For a detailed mathematical analysis of the underlying mathematical concepts (in the generalized context as cut loci) refer to [49], [50] and [51]. The latter paper presenting an extended analysis of mathematical foundations of the medial axis contains also early results (e.g. the one stated below in formula (1)) indicating already the possibility to employ the medial axis as a geometric modeling tool. The latter aspect will be a subject discussed in this section.

One could perhaps summarize the most relevant points of medial modeling as follows: In medial modeling a solid is described by its medial axis and an associated radius function. The medial axis being a subset of the solid is a collection of lower dimensional objects. For a general 3D-solid the medial axis mostly consists of a connected set built by a collection of surface patches and curves. Medial representations often simplify the process of gaining volume tessellations for the given object, supporting the meshing of solids, see section 7.4. They also offer new possibilities for the construction of intuitive haptic user interfaces that are useful to mold a solid's shape.

The basis of medial modeling can be summarized in a few geometric observations, ideas and definitions that are outlined here. Let K be a solid in the 3-dimensional or 2-dimensional Euclidean space. The *medial axis* $M(K)$ of K being a subset of the solid contains all points in K that are centers of maximal discs included in K. One usually includes in the medial axis set $M(K)$ its limit points. Figure 31 shows the medial axis (green) of a domain (black) with some maximal discs given (red). We assign to any point p in the medial axis $M(K)$ the radius $r(p)$ of the aforementioned maximal disc with center p and radius $r(p)$. This disc is denoted

[4]Most of the time it is relatively easy to give a mesh for each primitive.
[5]This is the difficult part.

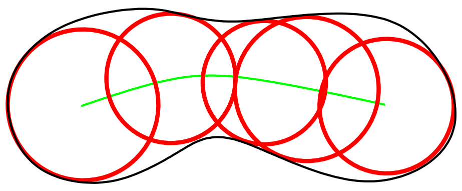


Figure 32: Maximal discs defining a shape

with $B_{r(p)}(p)$. The pair $(M(K), r)$ described by the medial axis $M(K)$ of a solid K and the associated maximal disc radius function

$$r : M(K) \to \mathbb{R}^+$$

is called *medial axis transform*, where R^+ denotes the non-negative real numbers. This pair $(M(K), r)$ yields a new possibility to represent the solid K simply as the union of the related maximal discs:

$$K = \bigcup_{p \in M(K)} B_{r(p)}(p) \tag{1}$$

For details see [51]. Figure 32 shows how the union of maximal discs defines the shape of a planar domain. The general reconstruction statement expressed in equation (1) already holds for solids with merely continuous boundary surfaces. However if the solid has merely continuous boundary surfaces (or continuous boundary curves for solids being closed 2-dimensional domains) then the medial axis may have a "wild" structure that may be presented by a set being dense in some open 3-dimensional sets (containing 3-dimensional discs).

In case one poses some regularity conditions for the solid's boundary surface ∂K, such as curvature continuity, the respective medial axis $M(K)$ will have a more benign structure. For instance if ∂K is curvature continuous then $M(K)$ will not be dense in any open set in $\mathbb{R}^3$.

Let us assume that ∂K is built by a finite collection of finitely many B-spline patches then $M(K)$ could be constructed by a union of finitely many medial sets. Each of the latter consisting of points being equidistant to two appropriately chosen boundary parts. Hence each medial set can be viewed as subset of zero sets defined implicitly by the condition stating that the difference of the distances of a point in the medial set to the respective parts of ∂K is zero.

This insight can be used to develop strategies to compute (approximately) the medial axis by assembling it from medial sets (see figure 33). In case the boundary parts are given implicitly by solutions to polynomial equations, then the medial sets can be described in principle by implicit polynomial equations as well.

The reconstruction result stated above in equation (1) can be used also to model shape for families of objects. Clearly in equation (1) the shape of the object depends on the medial axis set $M(K)$ and the function r.

Intuitively a continuous deformation $M(K)_t$, $t \in \mathbb{R}$ of the medial axis $M(K) = M(K)_0$ combined with a continuous change of the function r described via a continuous family of functions $r(t, s) : M(K)_t \to \mathbb{R}^+$ with $r(0, 0) : M(K) \to \mathbb{R}$ (with $r(0, 0) = r$) should yield a continuously deformed family of objects

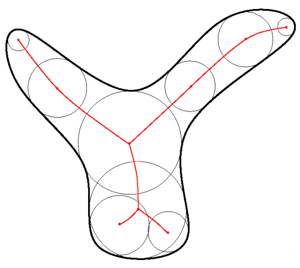


Figure 33: Assembled shape

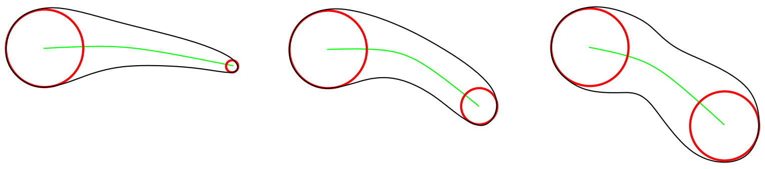


Figure 34: Continuous deformation of an object

$$K_{(t,s)} = \bigcup_{p \in M(K)_t} B_{r(t,s)(p)}(p)$$

The two control parameters t, s indicate that the change of the radius function $r(t,s)$ depends on the chosen respective domain of definition controlled by the parameter t and for a fixed domain of definition $M(K)_t$ the radius function depends on the parameter s. Figure 34 shows such a deformation. Note that the medial axis and the radius function are both modified. In order to present a well defined concept for the continuity of the deformation outlined here we need some formal requirements due to complications that may occur during the deformation process. We observe that a continuous (differentiable) deformation of the solid's boundary may result in a family of medial axes $M(K)_t$ whose homeomorphic type may change during the deformation process. Such a metamorphosis will occur when a (new singularity) curvature center of the boundary will meet the family of medial axes occurring during the deformation process of the solid. See figure 35 for an example. Therefore it makes sense to consider continuously changing families of functions $r(s,t)$ under the provision that for a varying parameter s the domain of the function family (here the medial axis set $M(K)_t$) should be fixed. This will allow to consider (for a fixed parameter t_0 and a variable parameter s) the family of continuous

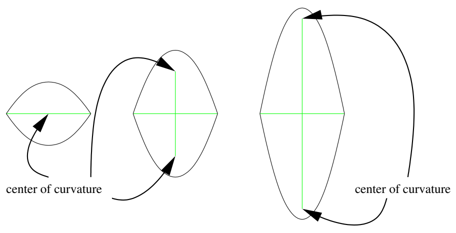


Figure 35: Non homeomorphic deformation of a medial axis

functions $r(t_0, s) : M(K)_{t_0} \to \mathbb{R}$ as a continuous path in a vector space of real valued functions defined on the compact set $M(K)_{t_0}$. That vector space will be endowed with an appropriate topology or norm. Here fixing the domain $M(K)_t$ makes it easy to define a distance between two radius functions $r(t_0, s_1)$ and $r(t_0, s_2)$ by

$$d(r(t_0, s_1), r(t_0, s_2)) := \max_{p \in M(K)_{t_0}} \{|r(t_0, s_1)(p) - r(t_0, s_2)(p)|\}$$

In order to express the continuous deformation of the medial axis family in a formally precise setting we need to endow the collections of all medial axes with a topology as well. Here it makes sense to use the Hausdorff-metric $d_H(\cdot, \cdot)$ defined on all compact subsets of the respective 2-dimensional or 3-dimensional domain. Let $A, B \subset \mathbb{R}^3$ then

$$d_H(A, B) := \inf\{\varepsilon : A \subseteq N_\varepsilon(B) \text{ and } B \subseteq N_\varepsilon(A)\}$$

with

$$N_\varepsilon(A) := \{y : |x - y| < \varepsilon \text{ for some } x \in A\}$$

A continuously deformed family of medial axes (depending on a parameter t) can now be viewed as a continuous path ϕ in the Hausdorff space H_{d_H} of compact sets in $\mathbb{R}^2$ or $\mathbb{R}^3$. Here we have

$$\phi(t) : \mathbb{R}^+ \to H = \{A \subset \mathbb{R}^3 : A \text{ compact}\}$$

Examples may be given by families of spline patches controlled by continuously moving control points $c_i(t)$, see section 4.4.

7.1 A metric structure for medial modeling

In the previous setting we compared radius functions only in the simplified special case where they were defined on a common medial axis set. It is desirable to formulate the continuous change of the medial axis set together with the change of the radius function in a common topology. For this purpose it is also possible to consider the preceding continuous deformation concept as a whole being describable within a general setting employing Hausdorff-topology and spaces of functions endowed with appropriate topologies. For this we define a metric on the product space built by the product of the two spaces $\tilde{H} \times F$ one of them being the above Hausdorff space

$$\tilde{H} = \{A : A \text{ compact subset of } \mathbb{R}^3 \cap B_h(0)\}$$

$(\tilde{H}, d_H)$ being endowed with the Hausdorff metric d_H defined above. The other space F in the product $\tilde{H} \times F$ is given by the space of all continuous real valued functions defined on the compact set $\overline{B_h(0)}$. On the latter space of continuous functions we can define a metric

$$d_S(f, g) := \max_{x \in \overline{B_h(0)}} \{|f(x) - g(x)|\}$$

for any pair of continuous real valued functions f, g defined on $\overline{B_h(0)}$.

In this context it is quite important to understand that any continuous function defined on a compact subset $A \subset \overline{B_h(0)}$ can be viewed as a restriction of an appropriately chosen function

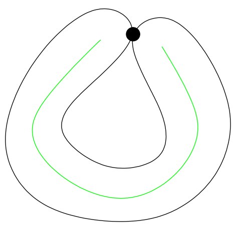


Figure 36: Tangential contact of envelopes

being continuous on all $\overline{B_h(0)}$. This holds since the space $\overline{B_h(0)} \subset \mathbb{R}^3$ fulfills appropriate separation properties, see also T_4 axiom of [18].[6] Clearly the metric on the product space is now defined by

$$d_\pi((A, r_1), (B, r_2)) := d_H(A, B) + d_s(r_1, r_2)$$

.

It can be shown that if

$$d_\pi((A_n, r_n), (A_0, r_0)) \rightsquigarrow 0 \text{ then}$$
$$d_H((\bigcup_{p \in A_n} B_{r_n}(p)), (\bigcup_{p \in A_0} B_{r_0(p)}(p)) \rightsquigarrow 0$$

The sequence of objects (each of which modeled by the union of discs) converges in the Hausdorff metric to the related limit object. Unions of discs obtained from members of a sequence of medial axis transforms converge against the discs union of the limit (medial axis transform). Clearly if $\psi(t) = (A(t), r_t)$ is a continuous deformation path in $(H \times C)$ with (A_0, r_0) being the medial axis transform of a solid, then for the respective discs unions related to $\psi(t)$ we have Hausdorff convergence towards the solid corresponding to (A_0, r_0).

1. However note that not every pair (A, r) will define a solid via the union $(\bigcup_{p \in A} B_{r(p)}(p))$
2. In case $(\bigcup_{p \in A} B_{r(p)}(p))$ defines a solid it may not have A as medial axis and $r : A \to R$ may not be maximal disc radius function.

Examples in the context of statement 1 above may be constructed easily in case we use a radius function r, that may attain the value zero as then parts of the object might agree with the axis A that may be chosen deliberately wild. In case we assume that the radius function $r > 0$ then we may still have delicate situations where the boundary of a domain obtained from a union of closed discs may at some points be locally homeomorphic to two arcs having tangential contact of a single point (see figure 36). Figure 37 shows an example illustrating the claim in 2. Here $\bigcup_{p \in A} B_{r(p)}(p)$ defines a solid whose medial axis contains a topological circle while A does not.

[6]This consideration shows that any radius function being a continuous function on a compact subset A of $\overline{B_h(0)}$ is a restriction of some continuous function defined on the set $\overline{B_h(0)} \supset A$.

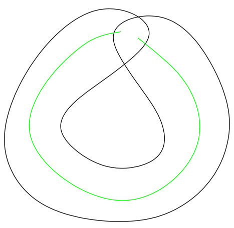


Figure 37: Self intersection of envelopes

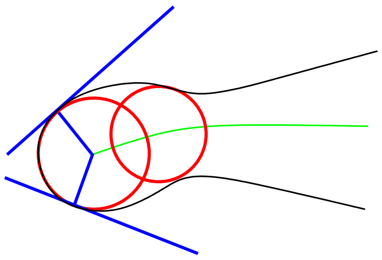


Figure 38: Construction of the envelope

7.2 Boundary representation in Medial Modeling

So far the Medial Modeling concept has been built on the idea to mold the solid by a union of discs. It may be preferable to represent the respective solid rather by appropriate boundary surfaces, see section 5. The latter ones arise quite naturally in the medial modeling context. Here the boundary surface (curve) is created as the envelope surface of the family of discs belonging to the specific medial axis transform (see figure 38). Let us assume that the medial axis is presented locally by a differentiable curve or surface patch being presented by parametric functions $m(u)$, with the radius function $r(u)$ depending on the parameter u as well.

It is possible to express the envelope surface using functions $en(u)$ in terms of expressions involving $m(u), m'(u), r(u), r'(u)$. It is also possible to compute $en'(u)$ and the curvature of the envelope curve. The latter computations need higher order derivative information of the functions representing the medial axis and of the radius function, refer to [53].

Employing the concepts outlined above different systems have been developed at the Welfenlab that can compute the envelope surface yielding a boundary representation of a solid whose medial surface and whose radius function have been given. More precisely the aforementioned medial modeling system computes for a parametric spline surface patch $m(u) : [0,1] \times [0,1] \to \mathbb{R}^3$ and for an associated radius function $r(u)$ the boundary surface of the corresponding solid whose medial surface is given by $m([0,1] \times [0,1])$ being a deformed rectangle embedded without self intersections into $\mathbb{R}^3$, see figure 39. Figure 40 illustrates the simplified special case where $m : [0,1] \to \mathbb{R}^2$ is a planar arc and the now two dimensional solid corresponds here to a planar domain. At those positions where the center points of the maximal discs are located on the boundary of the medial patch we get the related boundary surface of the solid from appropriate parts of maximal spheres there. Here the construction (using the modeler) is valid

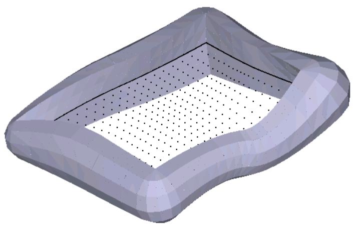


Figure 39: A medial patch inside its associated solid

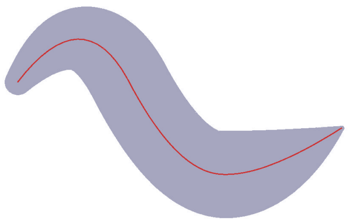


Figure 40: The 2D case

if the normal segment (joining the point $env(u)$ of the envelope surface with the medial axis point $m(u)$) does not meet a curvature center of the point $env(u)$ of the envelope surface prior to meeting $m(u)$. Using the curvature formula for the envelope surface mentioned above and some additional criteria then it is easily possible to check if the radius function is admissible. This means that the previously stated curvature center condition must hold. Under those assumptions it can be shown that the related envelope surface which we assume to be free of self intersections yields the boundary surface of a solid

$$\bigcup_{m(u)\in m([0,1]\times[0,1])} B_{r(u)}(m(u))$$

with $m([0,1]\times[0,1])$ being the medial axis of the solid being homeomorphic to a 3D-cube.

This result can be generalized to situations where the medial axis is built by a connected finite collection of patches. Again that collection of patches denoted by A will constitute the medial axis of a solid whose boundary surface is given by the envelope surface obtained via the disc radius function being defined on the collection of patches. Again we must assume that the envelope surface has no self intersections and that the above mentioned curvature center assumption holds for the envelope surface.

The situation where the medial axis is built by a collection of patches is far more complicated than the case where the medial axis is given by a single patch. Therefore we shall not go into a detailed discussion on this case in this survey paper. Suffice to say in order to deal with that complicated case envelope surfaces related to adjacent medial patches are joined along curves. The geometry of the intersection of medial surface patches (i.e., the angles between intersecting medial patches at an intersection point) poses conditions that can be used to appropriately blend adjacent envelope surfaces that are related to adjacent medial surface patches. These blended envelope surfaces are used to construct the boundary surface of a solid containing the aforementioned medial surface patches.

7.3 Medial Modeling and topological shape

One of the major reasons why the medial axis can be considered a shape descriptor of a solid is because it essentially contains the homotopy type of a solid because it is a deformation retract of the solid (refer to [51], [53]). The following *topological shape theorem of the medial axis* applies: The Medial Axis $M(D)$ contains the essence of the topological type of a solid D.

Let ∂D be C^2- smooth (or let ∂D be 1-dimensional and piecewise C^2- smooth, with $D \subset R^2$) Then the Medial Axis $M(D)$ is a deformation retract of D thus $M(D)$ has the homotopy type of D.

The proof of this theorem shows that it is possible to define a homotopy $H(x,t)$ as explained below the next figure describing a continuous deformation process of the solid D. This deformation process is depending on the time parameter t. The deformation starts with the solid being in the figure a rectangle with a circular hole. During the deformation points are moved along the shortest segments starting at the solid's boundary ∂D until the segments meet the dotted Medial Axis. The shortest segments are indicated by arrows in figure 41.

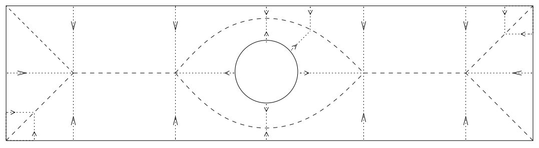


Figure 41: Deformation retract

We describe a homotopy

$$H(x,t) : (D\backslash\partial D) \times [0,1] \to (D\backslash\partial D)$$

such that

$$\begin{aligned} H(x,0) &= x \ \forall x \in D\backslash\partial D \\ H(x,t) &= x \ \forall x \in M(D) \\ H(x,1) &= R(x) \text{ with } R : D\backslash\partial D \to M(D)\backslash\partial D \end{aligned}$$

For this we define the homotopy as follows:

$$H(x,t) := x + td(x,\psi(x))\nabla d(\partial D, x)$$

Here $d(x,y)$ denotes the function describing the distance between variable points x, y; $\nabla d(x,y)$ describes the gradient of the distance function $d(x,y)$. $\psi(x)$ is defined as point where the extension of a minimal join from ∂D to $x \in (D\backslash\partial D)$ meets $M(D)$.

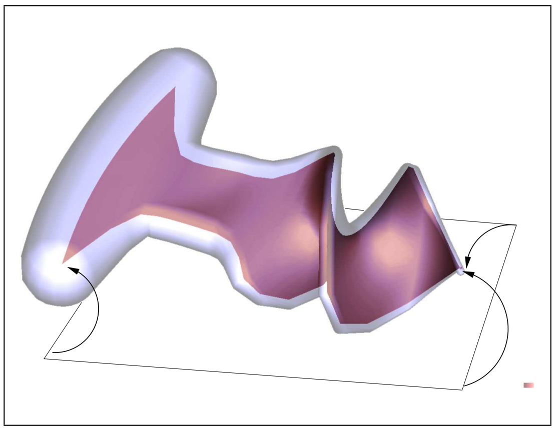


Figure 42: Medial axis of a solid

7.4 Medial Modeling and meshing of solids

In the preceding section on medial modeling we outlined geometrical concepts that were used to explain the deformation retract property stated in the topological shape theorem. We outlined also how to look at the solids boundary surface as an envelope surface that can locally be presented by a nonsingular parametrization map defined on the medial axis, see [53] for more details. All those geometric considerations immediately lead to insights that can be used to construct a meshing partition for a given solid that is naturally associated with the solid's medial axis.

We shall outline possibilities to use the medial axis for the meshing of solids by sketching some examples presented subsequently in several figures further down. In this context some observations are relevant. In case the solid S is created with the medial modeler say with a medial axis being diffeomorphic to a square Q then we immediately obtain a quite simple parametrization of the solid. That parametrization map can be described by a differentiable map defined on a solid PS containing all points in 3-space whose Euclidean distance to the unit square Q in the xy-plane is not larger than one. Here we identify the latter unit square with the parameter space of the medial axis surface. Our definition of the parametrization map is obtained from the differentiable function $env(u)$ describing the envelope surface (the solid's boundary surface, see 7.2). The respective parametrization map of the solid S maps an Euclidean segment in PS (joining any point u in the interior of Q orthogonally with the boundary of PS) linearly onto an Euclidean segment in S. The latter segment joins the medial axis point $m(u)$ with one of the two corresponding boundary points $env(u)$ in S. This segment in S meets the boundary surface of S orthogonally, see section 7.2. The outlined parametrization map of the solid yields a differentiable map f from PS onto S with a differentiable inverse f^{-1}. Figures 39 and 42 show the correspondence between the PS and S. In the simplified (lower dimensional case) the solid S is a 2D-domain with its medial axis being now an arc instead of a 2D-surface. The maps f and f^{-1} can be used to map certain convex sets in PS onto convex sets in S. This can be used to get a partition of an approximation of the solid S into convex subsets. Figure 43 shows examples where engineering objects modeled with our medial modeling system have now obtained tetrahedral meshes that have been constructed employing the geometrical concepts

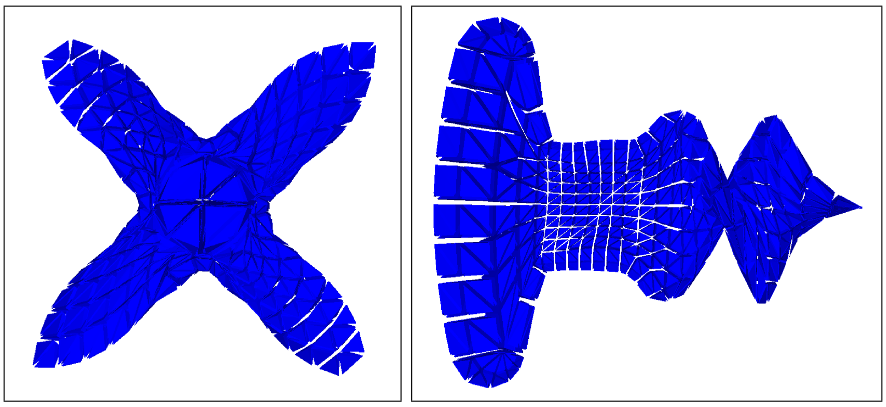
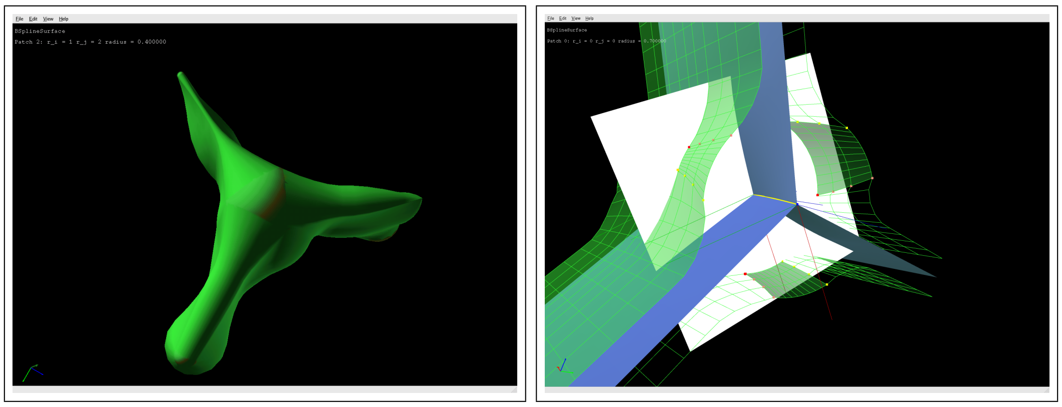


Figure 43: Meshes obtained from a medial axis representation



BSplineSurface

Patch 2: r_i = 1 r_j = 2 radius = 0.400000

BSplineSurface

Patch 0: r_i = 0 r_j = 0 radius = 0.700000

Figure 44: Multiple branches in a 3D medial axis representation

as explained above.

It is possible to extend these methods to more complex shapes where several branches of the medial axis are involved. See for example figure 44 (images from [8]) for the case where three different medial patches branch along a common line.

7.5 Medial modeling and Finite Element Analysis

Models created in CAD systems are often supposed to fulfill certain physical properties in addition to mere design aspects. It is for example of interest, how fast an objects breaks under load or how air or water flows around it. In order to be able to avoid expensive analysis of real prototypes, a thorough analysis is rather performed on virtual models. For this purpose differential equations are solved on the virtual model employing commonly a finite element analysis (FEA). For the FEA the objects need to be tessellated into small elements (for instance triangles or tetrahedra) first. This tessellation/meshing process often only approximates the geometry of the object at the boundary. After that a differential equation with certain boundary conditions is solved on the tessellation. If a design optimization is desired, the FEA

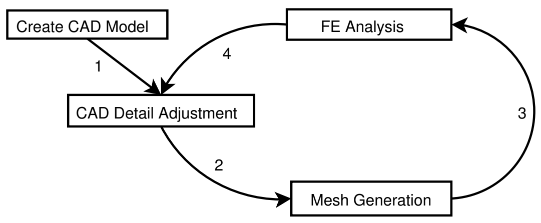


Figure 45: Design optimization cycle

solutions even need to be fed back into the CAD engine, to allow the (possibly automatic) modification towards a better design based on the results of the FEA. See figure 45 for a flow chart of the design optimization cycle. In contemporary systems and approaches still a huge gap between the CAD on the one side and the FEA on the other side exists. Future systems will need to join both areas under a single concept, that does not need the conversion from the CAD system into the approximative FEA data or the even more complicated integration of the FEA results back into the CAD engine.

We think the parametrization of an object via its medial axis yields such a common concept, for it can be used to model, to mesh, to solve differential equations on the exact geometry and therefore to re-integrate solutions into the design optimization process. A medial meshing method (see [43] and figure 46) can construct a mesh in the parameter space of an object with a given medial axis, instead of in the range. Thus the geometry of the object will be represented exactly. This can be of great advantage when differential equations with complex boundary conditions are to be solved. If the boundary is only approximated (linearly or with higher polynomial degree) the representation error at the boundary might yield huge errors in the FEA solution. These errors can be prevented by transforming the equations and solving them in the parameter space (i.e. with the exact geometry). See [42] for an example of such a computation. This approach has for example been used by Hughes ([21]), who represents the exact geometry (and even the basis functions of the FEA) with NURBS. Hughes approach (the "isogeometric analysis"), as well as ours (the "medial axis parametrization", maintain the exact geometry at all refinement levels without the need to communicate with a CAD description.

Because of its skeletal structure with corresponding thickness information the medial axis representation describes an object very intuitively. Hence the transition from the FEA solution back to the design optimization can be achieved smoothly. The FEA results are obtained directly on the exact geometry therefore the optimization process can easily adjust local parameters such as the thickness or the trend of the axis in areas where the object has bad characteristics (for instance where it might break or crack easily). As an example we will look at the construction of a 2D longitudinal section through a dental bridge (see figure 47). The goal is to analyze the stresses for the 2D section first and later to optimize the shape of the bridge in such a way that the stresses will be minimal.

Since the original 3D bridge object is given as a triangulated surface the 2D cut will be a linearly bounded polygon. Therefore the medial axis of the 2D domain will have many unnecessary branches leading into every convex corner. Since most of these branches do not contribute much to the shape of the bridge (but rather model the unwanted tiny corners in the boundary), they

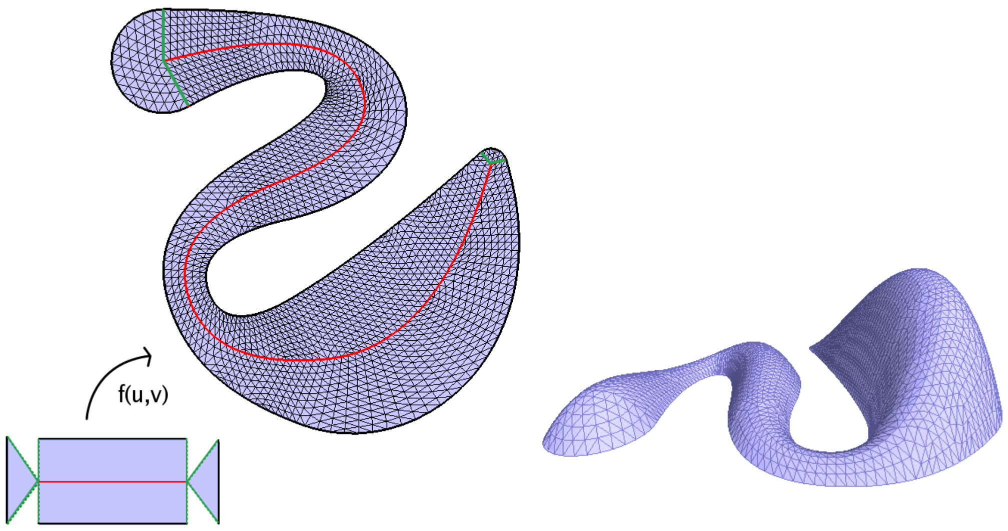


Figure 46: Triangulation and solution of the Poisson equation $\Delta f = -2$ with Dirichlet boundary condition

can be discarded. The structure will be reduced to only a few significant branches (see figure 48). The level of simplification can be controlled by a parameter.

After this simplification the object is successfully converted into the MAT representation. From this point on the geometry (especially the boundary) will be represented exactly throughout the computation and optimization process. Instead of converting an object into the MAT representation, it is possible to use the medial modeler described earlier to model it manually. Once an object is represented by its MAT parametrization, a mesh can be constructed in the parameter space (as described earlier). Figure 49 demonstrates this process for the main branch of the axis. It can be seen that the generated mesh follows the trend of the axis. The resolution of the mesh can again be controlled by a parameter.

After the mesh construction a FE-analysis can be performed to find areas with large stress. A load is applied at the upper side of the bridge and the bridge is fixed at the two anchor teeth. Figure 50 shows a possible solution where the isolines of the stresses give information about the stress distribution. With this data it is finally possible to modify the shape of the bridge. It can for example be thickened in the middle or at the transition to the anchor teeth. A recomputation of the stresses shows, whether the shape modification has led to a better distribution of the load.

8 Attributes

Sometimes there is a need to store additional information associated with a geometric model. We have already seen such an example: topological information in a boundary representation such as adjacency information.Another example might be the case where additional data has to be stored together with a model, that enriches the information but can somehow be precomputed. For example in a data base containing several hundreds of models each of them

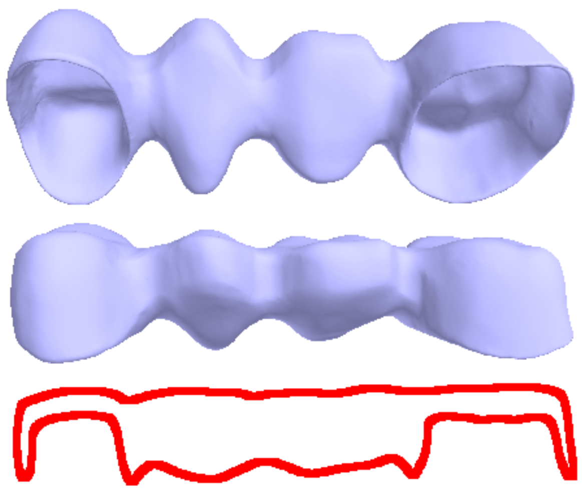


Figure 47: 3D views and 2D section of a dental bridge

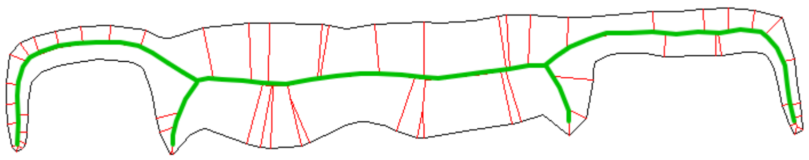


Figure 48: Simplified MAT of the bridge polygon (thick green line) with some unnecessary branches (thin red lines)

has to be identified with a unique key (see [43, 39]). These keys can be stored with the models thus making tasks like finding or comparing particular objects easier. There is a variety of other data that can be associated to a model, we will refer to all of these as *attributes of the model*. There can be attributes of physical origin, which alter the reception of the object by the user, or logical attributes, which relate the object to other objects or data. Physical attributes include photometric, haptical and other material constraints, such as elasticity or roughness.

8.1 Textures

If attributes are quantifiable (which is true for most of the physical attributes) then they are often specified by textures (i.e., functions that relate surface points to certain quantities of the attribute). Formally a texture is defined by:

$$t : M \to V$$

where M is the set of points of the model and V is the set of possible values of the texture. M can consist either of the entire volume of the model or only the surface points depending on the nature of the attribute.

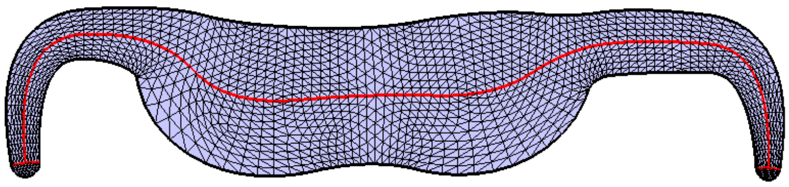


Figure 49: Mesh, generated only for the main branch

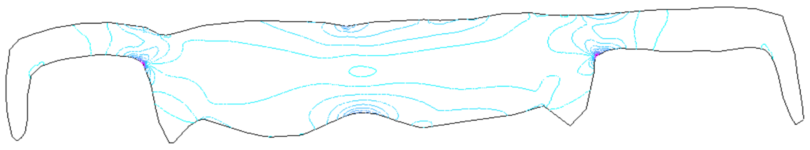


Figure 50: Example of iso stresses of a dental bridge

Normally textures are implemented using two maps

$$p : \mathbb{R}^2 \to M_S$$

and

$$v : \mathbb{R}^2 \to V$$

with p being a parametrization of the surface points M_S. Then the texture t is given by

$$t := v \circ p^{-1}$$

Note that in practice one does not need to compute the inverse of p since the coordinates in parameter space (here identical to the texture coordinates) are known during the process of painting. Nevertheless often it is rather difficult to find appropriate (non singular) mappings from the plane onto a given surface[7]. This results in distortions of the texture near the singularities. To avoid this, one can use *solid textures* (see [38] and [40]). Here the map v is defined as

$$v : \mathbb{R}^3 \to V$$

and thus $t := v$ since $M \subset \mathbb{R}^3$. Note that while ordinary textures are implemented as pixel images, solid textures are represented by voxel spaces (cf. section 3). This approach is slightly faster and avoids distortions induced by the parametrization, on the other side it consumes far more memory. Furthermore ordinary 2D textures can often easily be derived from photographs whereas this is much more complicated for solid textures (see [14]). Modern graphics hardware supports mapping solid textures to surface models using so called *3D textures*.

[7]In fact for most surfaces it is impossible as stated by the Hopf index theorem

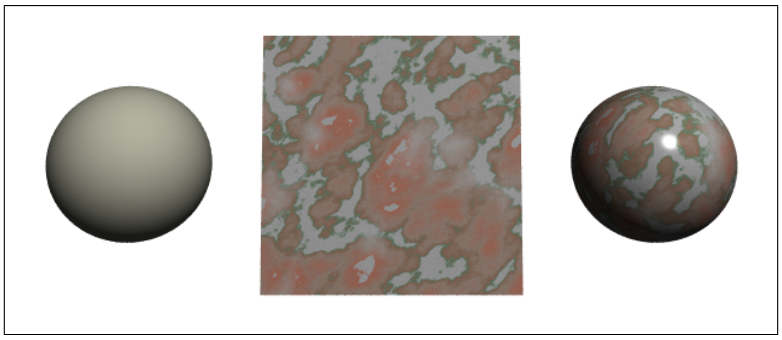


Figure 51: Photometric marble texture

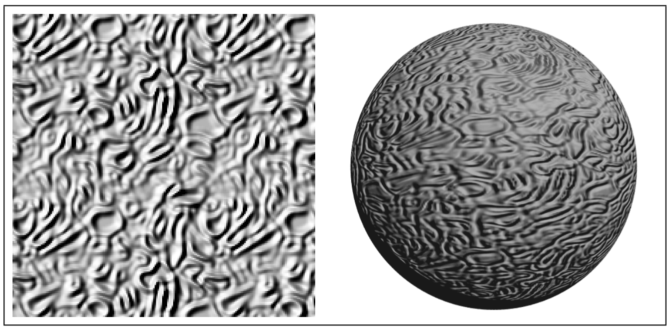


Figure 52: Bump-map on a sphere

Solid textures are easily applied in areas, where the texture data itself result from real world data, for instance a spatial scan of a material density or the like. Nevertheless modeling a spatial texture can be intricate. The approach presented in [6] shows how to combine traditional modeling techniques like CSG with the theory of distance functions to model arbitrary 3D textures. For each textural attribute (here referred to as *feature*) its extremal sets are modeled as separate solids, then the gaps in between are filled via distance interpolation methods. A slightly different and more general approach can be found in [22] and [28]. These techniques are commonly referred to as *heterogeneous* or *inhomogeneous* modeling. The texture can then be kept in its quasi-continuous representation to benefit from the representations superior analytic properties, or it can be easily converted to a voxel space representation for faster rendering etc.

The most common use for textures are photometric textures (i.e., maps that modify the color of a surface point). Figure 51 shows a sphere, a photometric texture resembling marble and the sphere "wrapped" with the texture.

Note that the use of textures is not limited to color (although this is assumed widely in the literature), other common applications include *bump maps* which are vector fields that alter the direction of the point normal (and thus alter the appearance of the surface locally near the point). Figure 52 shows a texture and the result of bump-mapping it onto a sphere. Note that not the sphere's geometry itself is changed only the face normals.

Textures are especially useful to model micro scale aspects of surfaces where detailed polyhedral modeling is too costly. For example a micro scale roughness of a stone surface can more efficiently be simulated by photometric and haptical textures ([1]) than by subdividing the (macro scale) smooth surface into tiny triangles. Furthermore material properties like elasticity or particle density can be represented by 3D textures. Rather than simulating the position of single, individually invisible particles, a quasi continuous texture is applied to the model space.

8.2 Model parameters

A special class of attributes are the *model parameters*. As we have seen in the preceding chapters, most model representations have a set of parameters associated with them, for example the set of control points for a spline patch. Sometimes it makes sense to view some of these parameters as attributes. Additional parameters can be added to most models, these include Euclidean motion matrices, stiffness constraints and the like. It is then sometimes more appropriate to allow these parameters to change over time, making them effectively attached *functions* rather than constants. These techniques lead to the theory of *physics-based modeling*, refer to [34] for a comprehensive overview of this topic.

8.3 Scripts

Examples of logical attributes are scripts. These are parts of code or methods that can be invoked when certain constraints of the model are met. A 3D object can have scripts attached that react to user interaction, when the object is selected in an interactive scene. Another example would be a script that is activated on collisions of the object with other parts of the scene. Scripts are especially useful in applications like physical modeling, where the modification of one object may require also modifications to associated objects. Rather than attributing these dependent objects to the calling object passively and letting the main program do the work, the tasks are carried out directly by the objects involved.

The idea of scriptable attributes has now been around for several years without finding a broad acceptance. Nevertheless there have been some prototype implementations like the *Odyssey Framework* (see [10], [13]).

9 Outlook and concluding remarks

A theoretically and practically difficult topic that we barely touched upon in this paper considers aspects related to the analysis and computation of singularities of geometric loci. Those singularities may come up on various occasions. They quite often concern the structure of geometrically defined solutions of non-linear equations being crucial to define precisely the local and global topological structure of solids and their parts. Those singular sets very naturally come up when we are dealing with surface intersections that may be related to Boolean operations carried out for solids bounded by surfaces (see [26], [37]). Similar problems also cause major difficulties in the context of CSG modeling, see section 6 and [19]. Simply spoken whenever a set under consideration has not the structure of a topological or of a differentiable manifold then it will have a singular structure at some locations. For an important class of singular sets this can be rephrased by saying singular sets in the Euclidean space cannot be represented by solutions of equations corresponding to some differentiable functions whose differential has a maximal rank at all points belonging to the "singular set" under consideration. Computations and constructions related to the medial axis in section 7 of this paper often contain as their most difficult part computations and representations of the singular subsets of the medial axis (see section 6). In general analyzing and understanding the mathematical structure of singular sets is sometimes quite difficult and may require the use of sophisticated and fairly advanced mathematical methods related to singularity theory. The mathematical and computational trouble caused by mathematically singular sets is enhanced by an additional fundamental problem in this context. One of the crucial difficulties that we encounter in geometric modeling is caused by the fact that all our models are usually represented in a discrete space and they only use points on a finite 3D-grid having a limited resolution. This implies that even in cases where we are dealing with solids, bounded by surfaces consisting of triangular facets only, we still may have difficulties carrying out Boolean operations. Those difficulties are caused by the fact that for certain geometric configurations we cannot properly compute the intersection set of two triangular facets. The latter problem may result in a (wrong) decision assuming the intersection point to be falsely classified inside or outside of some triangular facets. In the end all this may contribute to major topological inconsistencies and contradictions causing a failure of the system. In our view the state of the art in geometric modeling related to all of the aforementioned areas still needs substantial improvements by innovative concepts. Those new concepts to be developed should benefit from ideas inspired by advanced mathematical concepts from computational differential geometry and from singularity theory, cf. the pioneering work of the late [47], [11], [3], [4]. New exciting research by [27] uses the medial axis concept (section 7) in combination with ideas resting on a singularity analysis of distance wave fronts as to develop new methods for 3-dimensional shape representation that are applicable in a context of discrete point sets.

Another currently very active area related to geometric modeling is dealing with data compression. Often huge amounts of data points may arise from measurements or from construction procedures when large objects are constructed by many patches. Those collections of many patches need to be simplified, reduced or approximated by a surface whose description needs far less data (see [9]). However this approximation often must fulfill some specified accuracy requirements, for example, concerning placement. Furthermore quite often we must meet some topological conditions for the approximating surface not to have self intersections and singularities. Data corresponding to evaluation of continuous functions defined on geometric 2D or 3D objects may be obtained by measurements or by time consuming computational procedures such as those used in the area of differential equations. In all these cases one may encounter extremely large data sets that are far beyond the size that can be handled on current computers. In those situations one appreciates good approximation methods allowing an efficient approximation of the given data (or of that respective function). The description and evaluation of the approximation should need far less data than the original data set. Furthermore it should be possible to process the approximation data (substituting the original ones) efficiently on the computer for the particular computational purpose. This survey has been touching the basics of related topics for instance in the sections 4.4 and 4.7. Suffice to say that new concepts of wavelet and multiresolution theory appear to provide powerful tools that currently drive the progress in the respective fields that may be considered to belong to the subject of data compression (see [31] and [46]). It should be mentioned that very recent innovative efforts in the area of data compression employing new methods from a so called discrete Morse theory benefit from concepts that have been developed in the classical areas of modern global differential geometry and differential topology, see [15] and [35]. In the meanwhile there even exists a new field called ”computational topology” presenting fundamental research for geometric modeling that has been inspired strongly by methods and questions and ideas stemming from the classical area of topology and differential topology (for instance [2]).

Historically geometric modeling has been developed as a basic science for Computer Aided Design. In its early days the latter field has been employing descriptive geometry and Bezier geometry to design the shape of objects electronically instead of using blue prints created in technical drawings with the help of compasses and ruler. Meanwhile engineers want computer aided modeling systems whose capabilities go far beyond Computer Aided Design. Those systems shall not only describe the shape of objects but should allow also the simulation of various physical properties of the design object. This essentially implies that the computer system must be capable of solving partial differential equations (PDEs) defined on the geometry of the designed object. For this purpose we may need systems allowing very rapidly (ideally in real time) a good automated meshing procedure of the geometric design object. The resulting mesh must be appropriate for the approximate solution of the respective PDE used to analyze some properties of the designed object. Future geometric modeling and meshing systems will have to address those important needs. Those systems may therefore integrate the design and meshing functionalities in combined systems as it has been, for example, suggested in our medial modeler system described in section 7.4. In order to handle the combined needs of designing shape as well as designing the physics of objects it appears to make sense that the different engineering communities doing geometric modeling research, meshing research and computational engineering (PDE) research will cooperate more closely in the future. This collaboration should initiate learning processes where each community should profit from the knowledge available in the other communities.

Overall assessing future developments we think that new developments in geometric modeling and also in the aforementioned areas will increasingly employ concepts and insights from singularity theory, from local and global differential geometry and from advanced (singular) wavelet theory. The latter areas will help to provide mathematical concepts and tools being relevant to analyze and compute delicate singularities that may for instance be encountered analyzing dynamical processes related to various types of PDEs defined in the context of a physical analysis of the design object.

Finally we present a remark that corroborates our statement that new developments in geometric modeling and related fields benefit from using synergetically advanced concepts from global and local differential geometry. Geometric modeling is primarily involved with shape construction but it is also dealing with the area of shape interrogation and thus related to shape cognition of 3D and 2D objects. Shape cognition is concerned with methods identifying automatically the shape of an object in order to check if the shape design exists already in a data base containing shape design models being, for example, protected by some copyright.

We want to point out that recent advances on new strong methods concerning shape cognition benefit also from advanced concepts of global and local differential geometry such as singularities of principal curvature lines called umbilics, see [32], [23] and [24].

The authors like to thank Philipp Blanke and Patrick Klie being team members of the Welfenlab for carefully proof reading this paper.

References

[1] Allerkamp D., Böttcher G., Wolter F.-E., Brady A.C., Qu J., Summers I. R. A Vibrotactile Approach to Tactile Rendering, *The Visual Computer*, 23(2):97 - 108, 2007.

[2] Amenta N., Peters T.J., Russell A.C. Computational topology: ambient isotopic approximation of 2-manifolds, *Theoretical Computer Science*, Volume 305, Issues 1-3, Pages 3-15 , Aug 2003

[3] Arnold V.I., Gusein-Zade S.M. and Varchenko A.N. *Singularities of differentiable Maps*, Vol. I, II, Birkhauser, 1985.

[4] Arnold V.I. *Singularities of Caustics and Wave Fronts.* Kluwer, 1990

[5] Baumgart B. A polyhedron representation for computer vision. *National Computer Conference*, pp. 589–596, AFIPS Conf. Proc., 1975

[6] Biswas A., Shapiro V. and Tsukanov I. Heterogeneous Material Modeling with Distance Fields. *Technical Report, University of Wisconsin-Madison*, Mechanical Engineering Department, Spatial Automation Laboratory, SAL 2002-4, June 2002

[7] Blum H. Biological Shape and Visual Science. *Journal of Theoretical Biolgy*, 38, pp. 205-287, 1973

[8] Böttcher G. Medial Axis and Haptics. Master's Thesis, Leibniz Universität Hannover, Oktober 2004.

[9] Bremer P.T., Hamann B., Kreylos O., Wolter F.-E. Simplification of Closed Triangulated Surfaces Using Simulated Annealing, *Proceedings of the Fifth International Conference on Mathematical Methods for Curves and Surfaces*, Oslo, July, 2000, Mathematical Methods in CAGD, pp. 45-54, Vanderbilt University Press, Tennessee 2001.

[10] Brockman J.B., Cobourn T.F., Jacome M.F. and Director S.W. The Odyssey CAD Framework. *IEEE DATC Newsletter on Design Automation*, Spring 1992

[11] Bruce J.W. and Giblin P.J. *Curves and Singularities.* Cambridge University Press, Second Edition, 1992.

[12] Catmull E. and Clark J. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer Aided Design*, 10(6):350-355, 1978.

[13] Cobourn T.F. Resource Management for CAD Frameworks. *Dissertation* Carnegie Mellon University, May 1992, CMUCAD-92-39.

[14] De Bonet J.S. Multiresolution Sampling Procedure for Analysis and Synthesis of Texture Images. *ACM SIGGRAPH*, ACM Conf. Proc., 1997

[15] Edelsbrunner H., Harer J. and Zomorodian A. Hierarchical Morse Complexes for Piecewise Linear 2-Manifolds, *Symposium on Computational Geometry* (2001)

[16] Farin G. *Curves and Surfaces for Computer Aided Geometric Design: a practical Guide* (third edition). Academic Press, San Diego, 1993

[17] Foley J.D., van Dam A., Feiner S.K. and Hughes J.F. *Computer Graphics: Principles and Practice* (second edition in C). Addison-Wesley, Reading, 1996

[18] Hocking J.G. and Young G.S. *Topology* Dover Publications Inc., New York, 1988

[19] Hoffmann C.M. *Geometric and Solid Modeling: An Introduction.* Morgan Kaufmann, San Mateo, 1989

[20] Hoschek J. and Lasser D. *Fundamentals of Computer Aided Geometric Design.* A K Peters, 1993

[21] Hughes T. J. R. *Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement.* Comput. Methods Appl. Mech. Engrg., Vol. 194, pp. 4135–4195, 2005

[22] Jackson T. R., Cho W., Patrikalakis N. M. and Sachs E. M. Analysis of Solid Model Representations for Heterogeneous Objects. *JCISE: Journal of Computing and Information Science In Engineering*, Vol.2, No.1 Page1-10, March 2002, ASME Transactions.

[23] Ko K.H., Maekawa T., Patrikalakis N.M., Masuda H. and Wolter F.-E. Shape Intrinsic Properties for Free-Form Object Matching *ASME Journal of Computing and Information Science in Engineering (JCISE)* V(3)N(4), December, 2003, pp. 325-333

[24] Ko K.H., Maekawa T., Patrikalakis N.M., Masuda H. and Wolter F.-E. Shape Intrinsic Fingerprints for Free-Form Object Matching *Prodeedings of the Eighth ACM Symposium on Solid Modeling and Applications.* pp. 196-207. Seattle, WA, June 2003.

[25] Koenderink J.J. *Solid Shape.* MIT Press, Cambridge, 1990

[26] Kriezis G.A., Patrikalakis N. M. and Wolter F.-E. Topological and Differential-Equation Methods for Surface Intersections *Computer Aided Design*, vol.24 (1): 41-55, Jan. 1992.

[27] Leymarie F.F. Three-Dimensional Shape Representation via Shock Flows, Ph.D. thesis , *Brown University / Division of Engineering / Providence*, RI, USA May 2003

[28] Liu H., Maekawa T., Patrikalakis N. M., Sachs E. M. and Cho W. Methods for Feature-Based Design of Heterogeneous Solids. *Computer Aided Design*, vol. 36 (12): 1141–1159, Oct. 2003.

[29] Loop C. T. Smooth Subdivision Surfaces Based on Triangles. *Master's Thesis*, Department of Mathematics, University of Utah, August 1987

[30] Lorensen W. and Cline H. Marching cubes: a high resolution 3D surface construction algorithm. *ACM/SIGGRAPH Computer Graphics*, 21(4), pp. 163-169, (1987).

[31] Mallat S.G. *A Wavelet Tour of Signal Processing*, Academic Press, San Diego, 1998.

[32] Maekawa T., Wolter F.-E. and Patrikalakis N. M. Umbilics and Lines of Curvature for Shape Interrogation, *Computer Aided Geometric Design*, Volume 13, Issue 2, March 1996, Pages 133-161

[33] Mäntylä M. *An Introduction to Solid Modeling.* Computer Science Press, Rockville, 1988

[34] Metaxas D.N. *Physics-Based deformable Models: Applications to Computer Vision, Graphics and Medical Imaging.* Kluwer Academic Publishers, Boston, 1997

[35] Milnor J. *Morse Theory*, Princeton Univ. Press, Princeton, NJ, 1967.

[36] Nowacki H., Bloor M.I.G. and Oleksiewicz B. *Computational Geometry for Ships.* World Scientific, Singapore, 1995

[37] Patrikalakis N.M. and Maekawa T. *Shape Interrogation for Computer Aided Design and Manufacturing.* Springer, Berlin, 2003

[38] Peachey D.R. Solid texturing of complex surfaces. *SIGGRAPH 85.*, pp. 279-286.

[39] Peinecke N. *Eigenwertspektren des Laplaceoperators in der Bilderkennung.* Books on Demand GmbH, Norderstedt, Germany, 2006.

[40] Perlin K. An image synthesizer. *SIGGRAPH 85.*, pp. 287-296.

[41] Piegl L. and Tiller W. *The NURBS Book.* Springer, Berlin, 1995.

[42] Reuter M., Wolter F.-E. and Peinecke N. Laplace-Beltrami Spectra as Shape DNA of Surfaces and Solids. *Computer-Aided Design 38* (4), pp.342-366, April 2006.

[43] Reuter M. *Laplace Spectra for Shape Recognition.* ISBN 3-8334-5071-1, Books on Demand GmbH, Norderstedt, Germany, 2006.

[44] Samet H. The quadtree and related hierachical data structures. *ACM Comp. Surv.*, 16(2), June 1984, pp. 187-260.

[45] Sherbrooke E. C. and Patrikalakis N. M. Computation of the Solutions of Nonlinear Polynomial Systems *Computer Aided Geometric Design*, 10(5), 1993, pp. 379-405.

[46] Stollnitz E.J., DeRose T.D. and Salesin D.H. *Wavelets for Computer Graphics: Theory and Applications.* Morgan Kaufmann, 1996

[47] Thom R. *Structural Stability and Morphogenesis: An Outline of a General Theory of Models.* Benjamin-Cummings Publishing, Reading, Massachusetts, 1975.

[48] Warren J., Warren J. D. and Weimer H. *Subdivision Methods for Geometric Design: A Constructive Approach.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001.

[49] Wolter F.-E. Distance Function and Cut Loci on a Complete Riemannian Manifold. *ARCH MATH* 32 (1): 92-96, 1979.

[50] Wolter F.-E. Cut Loci in Bordered and Unbordered Riemannian Manifolds *Ph.D. Dissertation*, Technical University of Berlin, Department of Mathematics Berlin 1985

[51] Wolter F.-E. Cut locus and medial axis in global shape interrogation and representation. *MIT Seagrant Report*, 1992

[52] Wolter F.-E. and Tuohy S.-T. Approximation of High-Degree and Procedural Curves. *ENG COMPUT 8*, (2): 61-80 Sept. 1992.

[53] Wolter F.-E. and Friese K.-I. Local and global geometric methods for analysis interrogation, reconstruction, modification and design of shape. *Proc. CGI 2000*, June 2000, pp. 137-151.

[54] Yamaguchi F. *Curves and Surfaces in Computer Aided Geometric Design.* Springer, Berlin, 1988

[55] Zorin D. Subdivision and Multiresolution Surface Representations. PhD Thesis, Caltech, Pasadena, 1997

[56] Zorin D. and Schröder P. (Eds.) Subdivision for Modeling and Animation. ACM SIGGRAPH'2000 Course Notes, No. 23, July, 2000