State-of-the-Art Overview on 3D Model Representations and Transformations in the Context of Computer-Aided Design

Christoph Schinko[1,2], Andreas Riffnaller-Schiefer[1], Ulrich Krispel[1,2], Eva Eggeling[1,2], and Torsten Ullrich[1,2]

[1]Institute of Computer Graphics and Knowledge Visualization, Graz University of Technology &
[2]Fraunhofer Austria Research GmbH, Inffeldgasse 16c, 8010 Graz, Austria

{ christoph.schinko, ulrich.krispel, eva.eggeling, torsten.ullrich } @fraunhofer.at,
a.schiefer@cgv.tugraz.at

***Abstract*—Within a virtual world, either in virtual reality or in a simulation environment, the digital counterparts of real objects are described by mathematical and computational models. Depending on the purpose, the field of application, and the used toolchain a wide variety of model representations is established. As a consequence, conversion methods and transformation algorithms are becoming increasingly important. This article gives a state of the art overview on model representations and on the most important transformation techniques.**

Keywords–3D Model Representations; 3D Transformations; Computer-Aided Design

I. Introduction

Many different ways of model descriptions are available, tailored to the requirements in their respective areas of research. In the context of Computer-Aided Design (CAD), the model description and representation of a digital counterpart of a real object is called a shape description. An overview on shape descriptions and their transformations has been presented at the ninth International Conferences on Advances in Multimedia. This article is based on the corresponding conference contribution "3D Model Representations and Transformations in the Context of Computer-Aided Design: a State-of-the-Art Overview" [1] and gives a more detailed state of the art overview on shape representations and on the most important model transformation techniques.

At this point, it is important to emphasize that there are differences in the process of shape perception between human beings and computers. Two important aspects have to be mentioned in this context. On the one hand, there are sensory differences. In their natural surrounding, human beings can rely on their five senses to perceive a shape. Consequently, it is often a combination of these senses that makes up the sensation of a shape. While computers can be fitted with many different sensors, adding up to far more different senses compared to human beings, it usually boils down to a specific sensor being used to perceive a shape. The reason for that circumstance is directly related to the second aspect in this context – the reasoning itself. The human brain is yet to find a matching rival in the world of computer science. While computers are programmed to outperform the human brain in various, but rather specific tasks like number crunching, the computer is no thinking machine. Interdisciplinary developments in all fields of computer science over the recent years bring us ever closer to creating the thinking machine. However, especially for a computer, the task of shape classification heavily depends on the underlying description. Even after successfully classifying shapes, a computer is yet not aware of the meaning of shape, as discussed by Sven Havemann et al. in their work [2]. For the description of shape, it is important to be aware of these differences, even if shape classification is not in the context of this article.

The following two Sections describe the model representations (Section II) and the transformation techniques (Section III). The model representations include point sets (Section II-A), polygonal representations (Section II-B), parametric surfaces (Section II-C), subdivision surfaces (Section II-D), implicit surfaces (Section II-E), volumetric surfaces (Section II-F), and generative models (Section II-G). The description of transformation techniques gives a general overview and outlines important algorithms: level-of-detail techniques (Section III-A), marching cubes (Section III-B), random sample consensus (Section III-C), midsurfaces & isogeometry (Section III-D), parametric subdivision surfaces (Section III-E), and semantic enrichment (Section III-F). The final conclusion summarizes the state-of-art overview and shows open questions for future research directions (Section IV).

II. Model Representations

In dictionaries, shapes are described by words forming a textual definition:

> **bowl** a rather deep, round dish or basin, used chiefly for holding liquids, food, etc.
>
> Dictionary.com

From a computer science point of view, this definition is of a rather abstract nature representing a difficult basis for creating detectors. A computer program relies on more formal, mathematical definitions. For a human being, this description is sufficient enough to easily recognize the described shape when seeing it. The precondition for this accomplishment of the human brain is a basic understanding of the terms and definitions used in the description. Bootstrapping of the basic concepts on a textual basis alone is hardly possible. All available senses are used to create a mental image of the surrounding environment, making it possible to establish a connection between sensory input and concepts of shapes. With this connection available, a single sensory input is enough for the brain to be made aware of the related concepts. As an

example, the human visual system is capable of identifying and categorizing objects. Not only real-world objects, but also schematic drawings, pictures, paintings, etc. can serve as input. The description itself can also be available in the form of an image. This form of information is often found in biology, e.g., for describing animal or plant species.

Representation and description of shapes and objects in images is one of the basic methods to describe image content. However, similar to textual descriptions, image-based descriptions are a rather informal definition of shape, thus representing a difficult basis for creating algorithmic detectors. This is due to the loss of one dimension of object information when projecting a real-world object onto the 2D image plane. Shape information in images is often also affected by noise, distortion and occlusion. As a result, the shape extracted from the image only partially represents the real-world object.

In the context of CAD and Computer-Aided Manufacturing, a shape model has to be complete and has to comprehend all needed information. For these purposes, volumetric and boundary-/surface-based representations are used.

A. Point Sets

Points are a basic primitive to describe the surface of a shape [3]. A point set is a list of points defined in a coordinate system. While points are not the primitive of choice when using 3D modeling software to create shapes, they are widely used by 3D scanners due to the nature of their measurements. A point set is the outcome when measuring a large number of points on an object's surface.

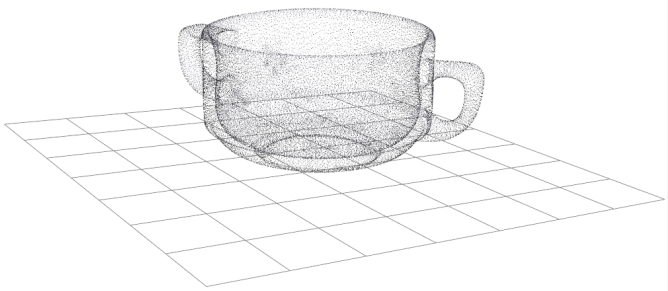

Figure 1. A laser scan of a soup bowl has been the basis of this point cloud. The data set has been processed (cleansed and resampled) and consists of 62 500 points.

The data set in Figure 1 shows a point cloud of a soup bowl consisting of 62 500 points. High resolution scans of larger objects require special techniques and/or out-of-core approaches due to the huge amount of data. For rendering approaches of point sets, the literature survey by Markus Gross and Hanspeter Pfister offers in-depth explanation [4]. The creation of another shape representation from point set data is called shape reconstruction.

B. Polygonal Faces

A very common representation to describe a shape's surface is to use a mesh of polygonal faces. The accuracy of the representation heavily depends on the shape's outline and is directly affected by the number of faces. A cylinder, for example, cannot be accurately represented by planar faces – it can only be approximated. Curved surfaces, in general, cannot be represented exactly, whereas objects having planar boundaries obviously can be. This limitation is often outweighed by its advantages in the field of CAD:

- Computer graphics hardware is tailored towards processing polygonal faces – especially triangles. This is the reason why many of the other shape representations are converted into polygonal meshes prior to rendering.
- A lot of tools and algorithms exist to create, process and display polygonal objects [5], [6].

The data structures for storing polygonal meshes are numerous. In a very simple form, a list of coordinates (x,y,z) representing the vertices of the polygons can be used. The de-facto standard data interface between CAD software and machines (e.g., milling machines, 3D printers, etc.) is the stereolithography file format (STL). It simply consists of a triangle list specifying its vertices. An illustrative STL-example is shown in Figure 2.

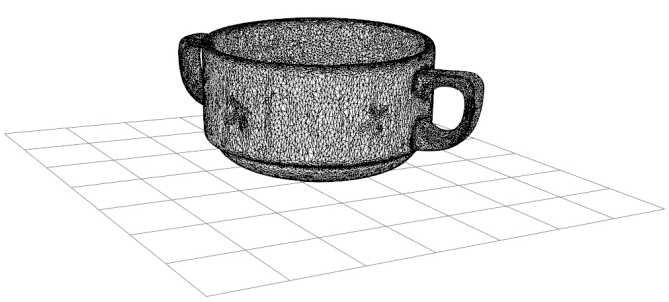

Figure 2. The laser scan of the "soup bowl" example (see Figure 1) is the input of a surface reconstruction algorithm, which returns a triangle mesh.

While this data structure is sufficient for some manufacturing purposes, it may not satisfy the needs of a 3D modeler for editing. More sophisticated data structures reproducing hierarchical structures (groups, edges, vertices) and adding additional attributes like normals, colors and texture coordinates provide a remedy. The problem of traversing a mesh can be tackled by introducing vertex-, face- and half-edge-iterators. They are typically, but not exclusively, used in combination with the concept of half-edges. The idea is to represent an edge between two vertices by two half-edges of opposite direction (see Figure 3).

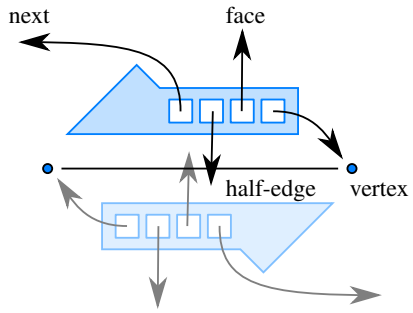

Figure 3. The half-edge data structure stores four references to (i) an associated vertex and (ii) an associated face. Furthermore, the half-edge data structure has (iii) a reference to its edge mate and (iv) to the next half-edge of the same face in counter-clockwise direction.

A half-edge is a directed edge with references to its opposite half-edge, its incident face, vertex and next half-edge. By defining operations using this data structure, it is possible to conveniently traverse a mesh [7].

C. Parametric Surface Representations

A parametric representation of a shape's surface is defined by a vector-valued parametrization function $f : \Omega \rightarrow S$ mapping a 2D parameter domain $\Omega \subset \mathbb{R}^2$ to the surface $S = f(\Omega) \subset \mathbb{R}^3$. This representation is a general way to specify a surface. The approximation theorem by Karl Weierstrass states, that finding an explicit formulation with a single function approximating a more complex function (shape) can be achieved by using polynomials.

Weierstrass Approximation Theorem Let f be a continuous real-valued function on the closed interval $[a, b]$. Then f can be uniformly approximated by polynomials.

A constructive proof of the theorem is given by Sergi Bernstein through his work on Bernstein polynomials. As any surface can be approximated by polynomials, the concept of polynomial surface patches has gained currency in the CAD domain [8], [9]. The idea is to split the function domain into smaller regions. Each surface patch, henceforth called patch, is described by a distinct parametric function approximating the local geometry of the patch. To obtain a good overall approximation of the surface, it is necessary to carefully chose the layout of the patches (form, size, number) and to deal with possible discontinuities on patch borders depending on the representation [10].

1. Bézier Surfaces

A Bézier surface is a two-dimensional surface in 3D generated from the Cartesian product of two Bézier curves [11]. A Bézier surface of degree (m, n) is defined as a parametric function

$$f(u, v) = \sum_{i=0}^{m} \sum_{j=0}^{n} \mathbf{b}_{ij} B_i^m(u) B_j^n(v).$$

It is evaluated over the unit square $(u, v) \in [0, 1] \times [0, 1]$ with the control points $\mathbf{b}_{ij} \in \mathbb{R}^3$ and two Bernstein polynomials $B_j^n(v)$. A Bernstein polynomial is defined by

$$B_i^n(t) = \binom{n}{i} t^i (1 - t)^{n-i}$$

of degree n for $t \in [0, 1]$.

In CAD, Bézier surfaces are often used in the form of bicubic Bézier patches, i.e., a set of 4×4 points represents the control mesh and is responsible for the shape of the surface as illustrated in Figure 4.

In all cases, Bézier curves and the corresponding Bézier surfaces have important properties:

- Bézier curves and surfaces fulfill the partition of unity property as
$$\sum_{i=0}^{n} B_i^n(u) = 1.$$
Thus the relationship between a Bézier curve / surface and its control mesh is invariant under affine transformations.
- A Bézier curve / surface is contained within the convex hull of its control mesh. Furthermore, the start and end points (of a curve) resp. the four corner points (of a surface) are interpolated by the Bézier curve / surface.
- A Bézier surface exhibits four boundary curves being Bézier curves themselves and their control points are the boundary points of the control mesh.
- The control points do not exert local control alone. Moving a single control point affects the whole surface. Geometric continuity (e.g., G^1, G^2) between patches can only be achieved by satisfying constraints on the control points' positions.

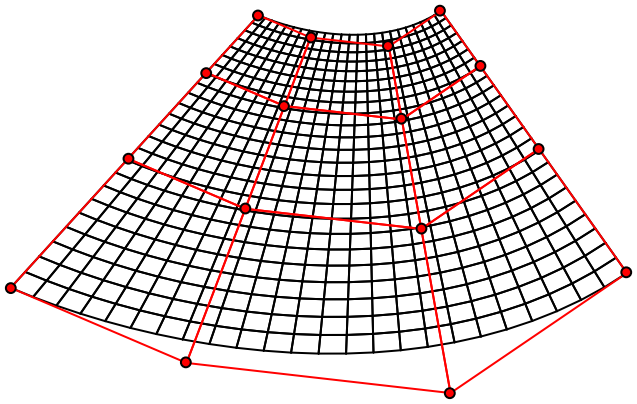

Figure 4. A tensor product surface is based on a curve representation. Consequently, this Bézier surface shares many properties (end point interpolation, convex hull property, etc.) with its corresponding curve type. A negative aspect of tensor product surfaces is the limitation to strictly rectangular topology (control mesh in red).

2. Rational Bézier Surfaces

The idea behind rational Bézier surfaces is to add adjustable weights to extend the design space of shapes [12]. In contrast to a Bézier surface, which can only approximate spheres and cylinders, the rational Bézier Surfaces can describe them exactly – a very important property in CAD. A rational Bézier surface of degree (m, n) is defined with the control points $\mathbf{b}_{ij} \in \mathbb{R}^3$, the weights $w_{ij} \in \mathbb{R}$, and the Bernstein polynomials $B_i^n(u)$ as

$$\begin{aligned} f(u, v) &= \sum_{i=0}^{m} \sum_{j=0}^{n} \begin{pmatrix} w_{ij} \mathbf{b}_{ij} \\ w_{ij} \end{pmatrix} B_i^m(u)\, B_j^n(v) \\ &= \frac{\sum_{i=0}^{m} \sum_{j=0}^{n} w_{ij}\, \mathbf{b}_{ij}\, B_i^m(u)\, B_j^n(v)}{\sum_{i=0}^{m} \sum_{j=0}^{n} w_{ij}\, B_i^m(u)\, B_j^n(v)}. \end{aligned}$$

Rational Bézier surfaces are a special case of non-uniform, ration B-spline (NURBS) surfaces, which are a generalization of B-spline Surfaces.

3. B-spline Surfaces

B-spline surfaces exhibit advantages when joining patches under continuity requirements. Let $m, n, k, l \in \mathbb{N}$ with $m \geq k$ and $n \geq l$. Then, a B-spline surface of degree (k, l) is defined as

$$f(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} \mathbf{d}_{ij}\, N_i^k(u)\, N_j^l(v),$$

with the basis functions

$$N_i^0(t) = \begin{cases} 1, & \text{if } \; t_i \leq t < t_{i+1} \\ 0, & \text{otherwise} \end{cases}$$

and

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

for $1 \leq r \leq n$ and a non-decreasing sequence of knots, a so-called knot vector,

$$T = \{t_0 \leq \cdots \leq t_n \leq \cdots \leq t_{n+m+1}\}.$$

It can be evaluated over $(u, v) \in [u_k, u_{m+1}[\times[v_l, v_{n+1}[$ with the control points $\mathbf{d}_{ij} \in \mathbb{R}^3$ and the polynomials $N_i^k(u)$ and $N_j^l(v)$. The control points $\mathbf{d}_{ij}$ forming the control polygon are called *de Boor* points.

In computer graphics, B-spline surfaces are typically used in the form of bi-cubic B-spline patches. A single cubic B-spline curve segment is defined by four control points, as a consequence, 4×4 control points define a bi-cubic B-spline patch segment. By choosing appropriate knot vectors, a B-spline surface can become a Bézier surface. B-splines with knots t_i satisfying the conditions

$$t_0 = 0$$

and

$$t_{i+1} = t_i \;\; \text{or} \;\; t_{i+1} = t_i + 1,$$

for $i = 0, \ldots, n + m$ are called uniform B-splines.

B-spline curves and surface satisfy properties similar to Bézier curves and surfaces [11]:

1) The relationship between a B-spline curve / surface and its control mesh is invariant under affine transformations.
2) A B-spline surface is contained within the convex hull of its control mesh: $f(u, v) \in$ convex hull $\{d_{kl} | i \leq k \leq i + m, j \leq l \leq j + n\}$.
3) In contrast to Bézier surfaces, the control points exert local control; i.e., if a control point is moved, only the local neighborhood is affected and
4) a B-spline surface can become a Bézier surface by choosing appropriate knot vectors.

Higher order geometric continuity (e.g., G^1, G^2) at borders of combined B-spline patches can be achieved by satisfying constraints on boundary control points and by appropriate choice of knot vectors.

4. NURBS Surfaces

The combination of rational Bézier techniques and B-spline techniques leads to non-uniform, rational B-splines, NURBS for short [13]:

Let $m, n, k, l \in \mathbb{N}$ with $m \geq l$ and $n \geq k$. Additionally, let

$$\mathbf{u} = (u_0, \ldots, u_{m+k+1})^T$$

and

$$\mathbf{v} = (v_0, \ldots, v_{n+l+1})^T$$

be two knot vectors and

$$w_{00}, \ldots, w_{mn} \in \mathbb{R},$$

$$\mathbf{d}_{00}, \ldots, \mathbf{d}_{mn} \in \mathbb{R}^3.$$

Then, a non-uniform, rational B-spline (NURBS) surface of degree (k, l) is defined as

$$\begin{aligned} f(u,v) &= \sum_{i=0}^{m} \sum_{j=0}^{n} \begin{pmatrix} w_{ij}\mathbf{b}_{ij} \\ w_{ij} \end{pmatrix} N_i^k(u)\, N_j^l(v) \\ &= \frac{\sum_{i=0}^{m} \sum_{j=0}^{n} w_{ij}\, \mathbf{b}_{ij}\, N_i^k(u)\, N_j^l(v)}{\sum_{i=0}^{m} \sum_{j=0}^{n} w_{ij}\, N_i^k(u)\, N_j^l(v)}. \end{aligned}$$

over $(u, v) \in [u_l, u_{m+1}[\times[v_k, v_{n+1}[$ with the polynomials $N_i^k(u)$ and $N_j^l(v)$, the knot vectors $\mathbf{u}$ and $\mathbf{v}$, the control points $\mathbf{d}_{ij}$ with weights $w_{00}, \ldots, w_{mn}$. Similar to B-spline patches, NURBS surfaces are commonly used in computer graphics in the form of bi-cubic NURBS patches.

B-spline surfaces and Bézier surfaces are special cases of NURBS surfaces [14]. If all weights are equal, a NURBS surface becomes a B-spline surface. Additionally, when all knot vectors are chosen appropriately, the B-spline surface becomes a Bézier surface.

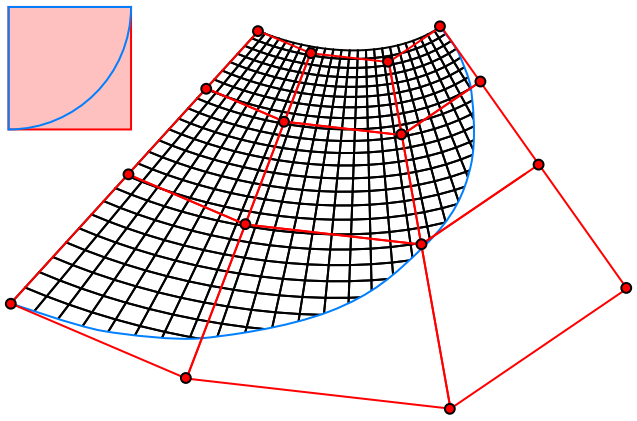

Figure 5. A trimmed Bézier / B-spline / NURBS surface consists of a parameter domain (upper left) and a set of control points. Furthermore, a closed curve (blue) within the parameter domain separates the domain into two parts: a valid part and an invalid part. The final surface in 3D only consists of those points whose parameters are valid [15].

A common way to model arbitrarily complex smooth surfaces is to use a mesh of bi-cubic NURBS patches. Regular meshes consisting of bi-cubic patches formed by vertices of valence four can be seen as connected planar graphs. A direct consequence of the Euler characteristic for connected planar graphs with the aforementioned properties is that such meshes must be topologically equivalent to an infinite plane, a torus, or an infinite cylinder – all other shapes cannot be constructed unless using trimming or stitching as illustrated in Figure 5. The resulting surfaces offer precise feature control at the cost of computational complexity due to trimming and stitching [16].

D. Subdivision Surfaces

Subdivision surfaces are the generalization of spline surfaces to arbitrary topology. Instead of evaluating the surface itself, the refinement of the control polygon represents the subdivision surface. There are many different subdivision schemes, e.g., Catmull-Clark [17], Doo-Sabin [18], Loop [19], Kobbelt [20], etc.

The subdivision scheme presented by Edwin Catmull and Jim Clark is a generalization of bicubic B-spline surfaces to arbitrary topology [17]. The set of 4×4 control points $\mathbf{p}_{ij}$ forms the starting mesh for an iterative refinement process where each step results in a finer mesh. One iteration of such a subdivision process is shown in Figure 6.

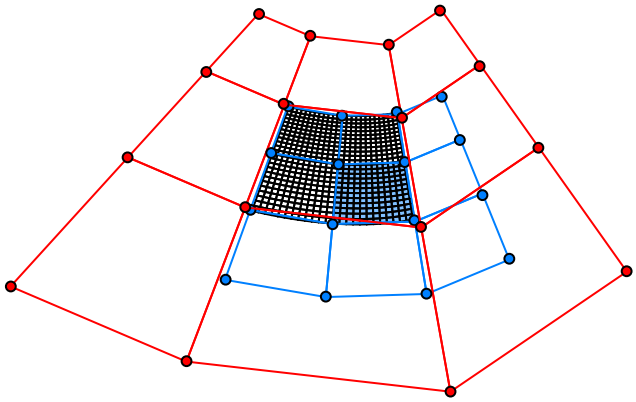

Figure 6. The Catmull-Clark subdivision scheme is based on the idea to describe a subpatch of a bicubic B-spline patch by a bicubic B-spline patch. The starting point is a bicubic patch (red), which generates a surface (wireframe in black), if evaluated over the domain $(u, v) \in [0, 1] \times [0, 1]$. Then the subsurface belonging to $[0, 1/2] \times [0, 1/2]$ is inspected and its corresponding control mesh (blue) is determined. The correspondences between the original mesh (red) and its subdivided version (blue) are the B-spline refinement rules. The Catmull-Clark subdivision scheme generalizes theses rules to arbitrary meshes.

Subdivision surfaces are invariant under affine transformations. They offer the benefit of being easy to implement and computationally efficient. Only the local neighborhood is used for the computation of new points. A major advantage of subdivision surfaces is their repeated refinement process – level-of-detail algorithms are always "included" by design.

Most commonly-used subdivision schemes, like those mentioned previously, only support non-rational surfaces of low degree, i.e., quadratic or cubic, with an uniform parameterization. This limits their use in the context of CAD. For example, due to the missing rational representation, conic sections like cylinders cannot be exactly represented by such surfaces. Also, a non-uniform parameterization is often required, e.g., to define Bézier-like interpolating boundaries, as is commonly done with NURBS surfaces. And higher degree surfaces provide additional advantages like higher continuity for smooth surfaces or improved convergence in the context of analysis.

Ideally, a CAD surface representation would provide the precise control of NURBS without the need for trimming and stitching. To achieve this, Cashman et al. [21] extended the Catmull-Clark subdivision scheme to non-uniform, higher degree, and rational surfaces. This subdivision scheme is compatible with odd degree NURBS in regular regions, away from extraordinary vertices with a valence other than four, but generalizes to arbitrary topology control meshes. It therefore combines the advantages of NURBS and subdivision surfaces in a single surface representation.

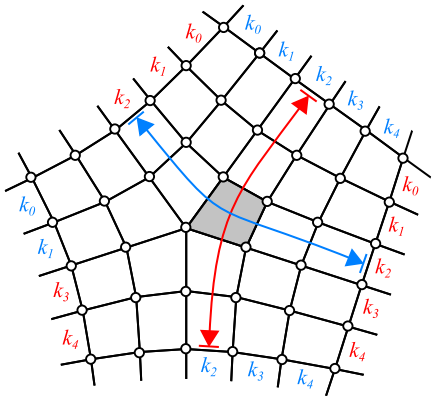

Figure 7. For the NURBS compatible subdivision scheme, the two local knot vectors, indicated by arrows, defining the parameterization of a face (shaded in gray) are derived from the knot spacings k associated with each edge of the control mesh.

Similar to rational Bézier surfaces or NURBS, each control point of a NURBS compatible subdivision surface gets an additional weight to provide a rational representation. To allow for a non-uniform parameterization, a knot spacing k is associated with each edge in the control mesh, defining the interval between two knot values in the knot vector. Knot spacings are required to be equal on opposite edges of a quadrilateral face. Therefore, a knot spacing is always defined for a strip of faces, similar to NURBS. This definition leads to each face of the control mesh having two associated local knot vectors $\mathbf{u}$ and $\mathbf{v}$, visualized as two colored arrows in Figure 7.

Subdivision is then performed in two stages. During the initial refine stage, edges and faces are split according to the subdivision rules [21], which depend on the valence of control points and the local knot vector. The refinement is followed by multiple smoothing steps, depending on the degree of the surface, in which all control points are moved to their new, updated position. This results in a smooth surface that is equivalent to the corresponding NURBS surface for regular

regions of the control mesh and is at least C^1 continuous in all other regions.

E. Implicit Surface Representations

In contrast to the parametric surface representations described above, implicit surfaces, are defined as isosurfaces by a function $\mathbb{R}^3 \rightarrow \mathbb{R}$ [22]. Therefore, similar to voxels, a surface is only indirectly specified. A simple 3D example of an implicit surface is the following definition of a torus with major radius R and minor radius r

$$f(x, y, z) = (x^2 + y^2 + z^2 + R^2 - r^2)^2 - 4R^2(x^2 + y^2) = 0.$$

Inside and outside of the surface is defined by $f(x, y, z) < 0$, respectively $f(x, y, z) > 0$. While a parametric description of the torus exists, many implicit surfaces do not have a closed, parametric form. In terms of expressiveness, implicit surfaces are more powerful than parametric surfaces [23].

Drawbacks of implicit surfaces are the inherent difficulty of describing sharp features (unless trimming is used) or finding points on the surface. However, this representation has several advantages. Efficient checks whether a point is inside a shape or not are possible. Since the surface is not represented explicitly, topology changes are easily possible. Surface intersections, as well as boolean set operations (the basis of constructive solid geometry) can also be implemented efficiently. Using implicit functions the operations of constructive solid geometry can be mapped to simple, mathematical terms:

If $g_1, g_2, \ldots, g_n$ are implicit functions, then the CSG operations are:

- **union**

$$\bigcup_{i=1,\ldots,n} g_i(p) = \min_{i=1,\ldots,n} g_i(p)$$

- **intersection**

$$\bigcap_{i=1,\ldots,n} g_i(p) = \max_{i=1,\ldots,n} g_i(p)$$

For a smooth blending Alexander A. Pasko and Vladimir V. Savchenko [24] suggest the blending function [25]

$$\bigcup_{i=1,\ldots,n} (g_1, g_2) = \frac{1}{1+\alpha}\left(g_1 + g_2 - \sqrt{g_1^2 + g_2^2 - 2\alpha g_1 g_2}\right).$$

Implicit surfaces can be described in algebraic form (see the example of the torus), as a sum of spherical basis functions (so called blobby models), as convolution surfaces (skeletons), procedurally, as variational functions, or by using samples. The latter approach directly relates to volumetric shape descriptions.

F. Volumetric Shape Descriptions

Volumetric approaches can be used to indirectly describe a shape's surface. In contrast to surface-based descriptions, they define the surface to be a boundary between the interior and the exterior of a shape. However, the idea behind these approaches is not so much a description of a shape's surface, but a description of the entire volume. Such representations are frequently used in visualization and analysis of medical and scientific data.

1. Voxels

Data sets originating from measurements do not have continuous values and are limited to the points in space where measurements have been collected. It is very common that data points form a uniform regular grid. Such data points in 3D are known as voxels, a name related to their 2D counterparts: the pixels. Since a voxel represents only a single point in a grid, the space between voxels is not represented. Depending on the area of application, the data point can be multi-dimensional, e.g., a vector of density and color. Due to the fact that position and size of a voxel are pre-defined, voxels are good at representing regularly sampled spaces. The approximation of free-form shapes suffers from this inherent property. Figure 8 illustrates the approximation artifacts of a free-form shape represented by voxels.

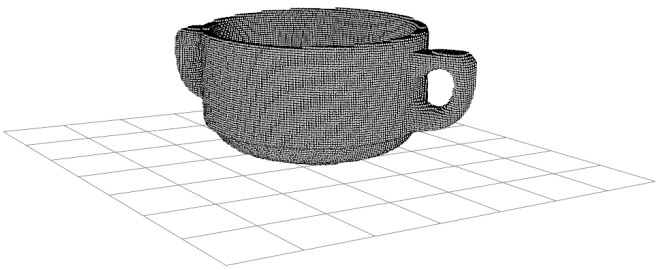

Figure 8. The "voxelized" soup bowl data set shows the typical approximation artifacts of a grid-based representation.

Voxel representations do not suffer from numerical instabilities as they are typically defined on an integer grid. A major drawback of voxel representations is the amount of data needed for storage. For example, a $512 \times 512 \times 512$ voxel grid storing 32-bit floating point values occupies 512MB of memory. Depending on the intended use, the memory footprint may be too high and it may appear rather coarse.

Typical use cases are the visualization and analysis of medical data (medical imaging) acquired from sources like computed tomography (CT), magnetic resonance imaging (MRI), or 3D ultrasonography.

2. Convex Polytopes

Shapes can be described as geometric objects with flat sides – so called polytopes. They are defined in any dimension as n-dimensional polytopes or n-polytopes. Two-dimensional polygons are called 2-polytopes and three-dimensional polytopes are called 3-polytopes. A special case of a polytope is a convex polytope having the additional property of being a convex set of points in n-dimensional space $\mathbb{R}^n$, respectively in n-dimensional Euclidean space $\mathbb{E}^n$. Convex polytopes can be defined over their convex hull, or by the intersection of half-spaces.

Branko Grünbaum and Geoffrey C. Shephard define a convex polytope as the convex hull of any finite set of points in Euclidean space $\mathbb{E}^n$ $(n \geq 1)$ [26]. A set $S \subseteq \mathbb{E}^n$ is convex, if for any pair of points $\mathbf{x}, \mathbf{y} \in S$, the line segment

$$\lambda \cdot \mathbf{x} + (1 - \lambda) \cdot \mathbf{y}$$

with $0 \leq \lambda \leq 1$, lies entirely in S. For any set S, the smallest convex set containing S is called the convex hull of S. A definition relying on the convex hull of a set of points is called a vertex representation.

Convex polytopes can also be defined as the intersection of a finite number of half-spaces [27]. Because of the fact that the intersection of arbitrary half-spaces need not be bounded, this property must be explicitly required. An algebraic formulation for convex polytopes consists of the set of bounded solutions to a system of linear inequalities. Hence, a closed convex polytope can be written as a system of linear inequalities.

$$\begin{aligned} a_{11}\mathbf{x}_1 + a_{12}\mathbf{x}_2 + \cdots + a_{1n}\mathbf{x}_n &\leq b_1 \\ a_{21}\mathbf{x}_1 + a_{22}\mathbf{x}_2 + \cdots + a_{2n}\mathbf{x}_n &\leq b_2 \\ \vdots \quad \vdots \quad \vdots \quad \vdots & \\ a_{m1}\mathbf{x}_1 + a_{m2}\mathbf{x}_2 + \cdots + a_{mn}\mathbf{x}_n &\leq b_m \end{aligned}$$

with m defining the number of half-spaces of the polytope. Open convex polytopes are defined similarly with strict inequalities instead of non-strict ones [28].

A limitation of convex polytopes is the inherent restriction to represent convex geometry only. The representation of non-convex geometry is possible through composition of convex polytopes. Topologically, convex polytopes are homeomorphic to a closed ball.

Convex polytopes are a subject of mathematical study since ancient Greek times. There is a number of special polytopes in three-dimensional space admitting a particularly high degree of symmetry – the so-called Platonic solids, tetrahedron, hexahedron, octahedron, dodecahedron and icosahedron. They are bounded by congruent regular polygonal faces exhibiting a consistent vertex valance over all vertices.

3. Constructive Solid Geometry

Constructive solid geometry (CSG) is a technique to create complex shapes out of primitive objects. These CSG primitives typically consist of cuboids, cylinders, prisms, pyramids, spheres and cones. Complex geometry is created by instantiation, transformation, and combination of the primitives. They are combined by using regularized boolean set operations like union, difference and intersection that are included in the representation. A CSG object is represented as a tree with inner nodes representing operators and primitives in the leaves.

In order to determine the shape described by a CSG tree, all operations have to be evaluated bottom-up until the root node is evaluated. Depending on the representation of the leaf geometry, this task can vary in complexity. Some implementations rely on representations that require the creation of a combined shape for the evaluation of the CSG tree, others do not create a combined representation. In that sense, CSG is not as much a representation as it is a set of operations that need to be implemented for the underlying shape representation [29].

As a consequence, CSG can also be performed on other shapes and shape representations. Two different approaches can be used to evaluate CSG objects: object-space approaches and image-space approaches. The main difference between the two approaches is that object-space approaches create shapes, while image-space approaches "only" create correct images.

Object-space CSG approaches using primitives described implicitly can be calculated accurately. Performing CSG on other shape representations (like polygonal meshes) typically introduces accuracy problems, due to the finite precision of floating-point numbers. A common representation used for CSG operations are binary space partitioning (BSP) trees. BSP is a method for subdividing a space into convex cells yielding a tree data structure. This data structure can be used to perform CSG operations using tree-merging as described by Bruce Naylor et al. [30]. The algorithm is relying on accurate information of inside and outside of a shape (or, in case of planes, above and below).

G. Generative Shape Descriptions & Design Automation

Algorithmic shape descriptions are also called generative, procedural, or parametric descriptions. However, there are differences between the three terms. Parametric descriptions are loop-computable programs (the functions it can compute are the primitive recursive functions), and therefore they always terminate [31]. On the other hand, procedural descriptions offer additional features (like arbitrary recursion), are structured in procedures, and are not guaranteed to terminate. Compared to procedural descriptions, generative descriptions are a more general term, including, for example, functional languages.

In this context, algorithmic descriptions are henceforth referred to as generative descriptions. The process of creating such descriptions is referred to as generative modeling and design automation. In contrast to many other descriptions, which are only describing a shape's appearance, generative shape descriptions represent inherent rules related to the structure of a shape. In simple terms, it is a computer program for the construction of the shape. It typically produces a surface-based or volumetric shape description for further use. In the article "Modeling Procedural Knowledge – A Generative Modeler for Cultural Heritage" [32] by Christoph Schinko et al., the authors state that all objects with well-organized structures and repetitive forms can be described in such a way. Many researchers enforce the creation of generative descriptions due to its many advantages [33].

Its strength lies in the compact description compared to conventional approaches, which does not depend on the counter of primitives but on the model's complexity itself [34]. Particularly large scale models and scenes – such as plants, buildings, cities, and landscapes – can be described efficiently. Generative descriptions make complex models manageable as they allow identifying a shape's high-level parameters.

Another advantage is the included expert knowledge within an object description, e.g., classification schemes used in architecture, archaeology, civil engineering, etc. can be mapped to procedures. For a specific object only its type and its instantiation parameters have to be identified. This identification is required by digital library services: markup, indexing, and retrieval [35]. The importance of semantic meta data becomes obvious in the context of electronic product data management, product lifecycle management, data exchange and storage or, more general, of digital libraries.

A disadvantage of generative shape descriptions is their dependency to (1.) a programming language, in which the shape is implemented and to (2.) a primitive geometry representation (point sets, meshes, etc.), which is the return type of the implemented shape function. As a consequence, generative descriptions can be realized in many different ways [33] and

the conversion between different kinds of generative descriptions relies on translation techniques developed in the field of compiler construction [36].

Generative descriptions have been developed in order to generate highly complex shapes based on a set of formal construction rules. They represent a whole family of shapes, not just a single shape. A specific exemplar is obtained by defining a set of parameters, or a sequence of processing steps: Shape design becomes rule design [37].

Because such descriptions already belong to a specific class of shapes, there is no need for detectors. However, with a generative description at hand, it is interesting to enrich other descriptions and representations. What is the best generative description of one or several given instances of an object class? This question is regarded as the inverse modeling problem [38].

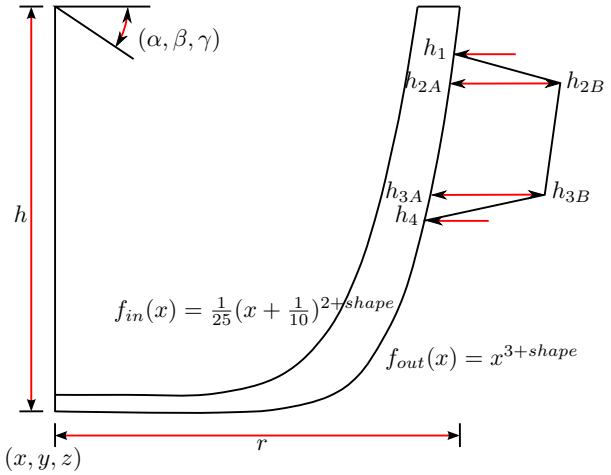

Figure 9. The generative model takes several parameters: (x, y, z) is the base point of the bowl and (α, β, γ) define its orientation. Its shape is defined by an inner f_{in} and outer f_{out} shape function with one free parameter $shape$. These functions are rotated around the cup's main axis and scaled with the parameters r and h. The handles are defined via control points of a Bézier curve.

In order to continue the example of the soup bowl mentioned before, Figure 9 sketches an automated design of a bowl. The generative model is implemented as a function M that takes several parameters $\mathbf{x} = (x_0, \ldots, x_n)$ and which returns a 3D model of a bowl. Automatic fitting routines [38] are able to register the generative design with the STL input data set (see Figure 2) and to determine the optimal input parameters (see Figure 10).

In detail, the registration algorithm converts the STL input data set into a point cloud P. For each instance of the generative model description $M(\mathbf{x})$; i.e., for each evaluation of a specific parameter set $\mathbf{x}$, the geometric distance $d(M(\mathbf{x}), P)$ between the two 3D models is calculated [39]. An optimization algorithm minimizes this distance d by evaluating different parameter sets $\mathbf{x}_1, \mathbf{x}_2, \ldots$ i.e., it minimizes the error function

$$f(\mathbf{x}) = d(M(\mathbf{x}), P) \stackrel{!}{=} \min_{\mathbf{x}} . \quad (1)$$

As the parameter domain may be a mixture of discrete and continuous spaces, the optimization routine should be able to handle both; for example using the combination of an evolution strategy with an integrated gradient approach [40].

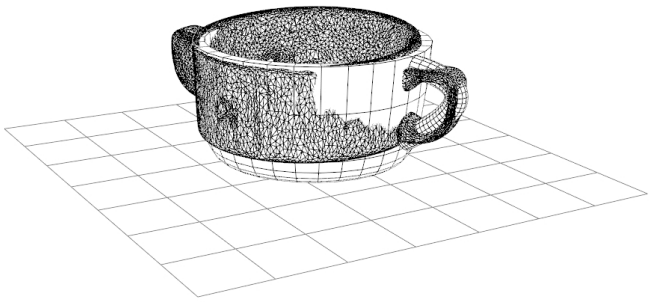

Figure 10. The "generative" soup bowl represents an ideal soup bowl. It is a "clean" quad mesh; in contrast to the "noisy" triangle mesh, which is the input of the generative fitting routine. The fitting calculates the optimal parameters so that the generative soup bowl fits its noisy counterpart. Showing input and output data in the same visualization outlines the quality of the fitting process.

III. Model Transformation

In a product life cycle, the digital counterpart of a future, real-world object has to pass several stages of a multi-step pipeline. First sketches of a product are represented in a different representation than the final CAD production-ready data set. Furthermore, virtual product tests and simulations require special purpose model representation as well. As a consequence, each transformation between two possible model representations has a field of application. For the presented representations Table I lists the conversion methods and algorithms. Furthermore, the following paragraphs describe the conversion ideas that have a wide-spread field of applications.

A. Level-of-Detail Techniques

Managing level of detail is at once a very current and a very old topic in computer graphics. As early as 1976 James Clark described the benefits of representing objects within a scene at several resolutions. Recent years have seen many algorithms, papers, and software tools devoted to generating and managing such multiresolution representations of objects automatically [70].

The idea of "Level of Detail", or LOD for short, is an important topic in computer graphics as it is one of the key optimization strategies that would help 3D graphical programs, such as modeling software to run faster and reliably rendered across all the new and old hardware.

B. Marching Cubes

Marching cubes is a computer graphics algorithm by William E. Lorensen and Harvey E. Cline for extracting a polygonal mesh of an isosurface from a three-dimensional discrete scalar field.

The algorithm proceeds through the scalar field, taking eight neighbor locations at a time (thus forming an imaginary cube), then determining the polygon(s) needed to represent the part of the isosurface that passes through this cube. The individual polygons are then fused into the desired surface.

This is done by creating an index to a precalculated array of 256 possible polygon configurations ($2^8 = 256$) within the cube, by treating each of the 8 scalar values as a bit in an 8-bit

integer. If the scalar's value is higher than the iso-value (i.e., it is inside the surface) then the appropriate bit is set to one, while if it is lower (outside), it is set to zero. The final value, after all eight scalars are checked, is the actual index to the polygon indices array.

Finally, each vertex of the generated polygons is placed on the appropriate position along the cube's edge by linearly interpolating the two scalar values that are connected by that edge.

The gradient of the scalar field at each grid point is also the normal vector of a hypothetical isosurface passing from that point. Therefore, it is reasonable to interpolate these normals along the edges of each cube to find the normals of the generated vertices, which are essential for shading the resulting mesh with some illumination model.

C. Random Sample Consensus

A simple and elegant conceptual framework to estimate parameters is the Random Sample Consensus (RANSAC) paradigm by Martin A. Fischler and Robert C. Bolles [71]. This technique is capable of extracting a variety of different models out of unstructured, noisy, sparse, and incomplete data. In the context of computer-aided design it is often used to fit geometric primitives such as planes, spheres, etc. to a point cloud [72].

RANSAC-based algorithms proceed by randomly taking (ideally few) samples, calculating the free parameters of a model (for example the four parameters of a plane). Then all samples of the input data set "vote", whether they agree with the hypothesis (if they are close enough to the suggested plane). This procedure is repeated a few times, and the hypothesis with the highest acceptance rate wins by "consensus". Samples, which agreed to a hypothesis, can be removed from the input data set and the process can be started again, basically until no samples remain.

The number of iterations, which are needed until a "good" hypothesis is found, can be determined stochastically. Let the input data set consist of $(r + s)$ elements, of which r belong to a model, which shall be identified. If n samples are needed to generate a model instance, the probability that k randomly chosen samples belong to this model is distributed hypergeometrically; that means $P(X = k)$ can be calculated via the formula

$$P(X = k) = \frac{\binom{r}{k}\binom{s}{n-k}}{\binom{r+s}{n}}.$$

Therefore, the probability that at least one sample does not belong to this model is $1-P(X = n)$. If the process of model generation and testing is done in j times, the probability that always at least one sample does not belong to the current model is $(1 - P(X = n))^j$. If p is the probability that the RANSAC algorithm returns the correct result, the probability of a failure is $1 - p$. The probability of such a failure is described by the term $(1 - P(X = n))^j$. Therefore, this term has to equal $1-p$. Solving the resulting equation for j returns the expected number of needed iterations:

$$j = \frac{\ln(1-p)}{\ln\left(1 - \frac{\binom{r}{n}}{\binom{r+s}{n}}\right)}$$

For example, if 20% of all points belong to a plane and the remaining points are distributed randomly, then the noise is at a level of 80%. In this case, the algorithm will need 373 iterations to detect this plane with a probability $p = 0.95$; respectively 574 iterations to ensure a probability of $p = 0.99$.

D. Midsurfaces & Isogeometry

Thin objects such as papers or metal sheets often appear in various contexts. Non-zero structural thickness is a factor that influences their dynamic movements. Nevertheless, those objects are often abstracted as two-dimensional entities, so-called thin shells. The simulation of 3D solids has been studied for a long time. Since thin shells are special cases of 3D solids, one may apply the techniques developed for 3D solids to the simulation of thin shells. Unfortunately, this approach does not produce satisfactory results; modeling thin shells as 3D elastic solids requires very fine FEM meshes to correctly capture the global bending behavior.

A thin shell is a 3D elastic solid, of which one dimension is small with respect to the others. This particular geometry covers a wide range of engineering designs common in the automotive and aerospace industries, but also in everyday life in the form of, e.g., objects made of metal sheet or thin plastic materials. For analysis this geometry is formulated in terms of the middle surface / midsurface of the shell.

The creation of a thin shell representation of an arbitrary object is an open question. Nevertheless, approaches to identify pairs of opposite patches, which can be merged to a single patch, and volume thinning techniques [73], [74] often lead to sufficient results.

As typical product development consists of several stages – mainly design in CAD system and analysis / simulation in computer aided engineering (CAE) systems – CAD models often have to be converted and transformed.

Isogeometric analysis (IGA) refers to analysis based directly on the geometry representation used in CAD. It therefore avoids transforming the designed geometry into an approximated simulation mesh for analysis. Isogeometric analysis was introduced by Hughes et al. [75] for NURBS based surfaces, but has also been successfully applied to other surface representations like subdivision surfaces [76], [77]. With IGA, the same basis functions used to define the CAD geometry are also used to represent the simulation geometry and the fields of unknowns for analysis. Because isogeometric analysis does not require the creation of a separate simulation mesh, it bridges the gap between CAD and CAE and facilitates an ideal integration of modeling and analysis.

While the main focus of IGA is to more closely link design and analysis in engineering, this technique is also used in other fields; for example, physics-based modeling tools based on IGA have been integrated into a modeling application [78]. The designed subdivision geometry is directly used to compute deformations based on an isogeometric thin shell analysis. The results are immediately available in the modeling application. Another example are soft-body deformations for virtual reality environments [79], which are computed interactively based on isogeometric analysis of the subdivision surfaces used for visualization.

E. Parametric Subdivision Surfaces

As subdivision surfaces are defined by an iterative refinement algorithm, they generally cannot be directly evaluated as a parametric surface. While regular regions often define a known parametric surface, like bi-cubic B-splines in the case of Catmull-Clark subdivision, this does not apply to irregular regions near extraordinary vertices.

How to exactly evaluate irregular regions of a subdivision surface at arbitrary parameter values has been shown by Stam, first for Catmull-Clark [59] and later also for Loop subdivision [60]. The evaluation is based on a set of eigenbasis functions, derived from the subdivision matrix of the particular subdivision scheme. This idea has also been extended to the higher degree NURBS compatible subdivision scheme [77]. These approaches therefore provide a parametric mapping of parameter values to points on the subdivision surface.

Another approach to convert a subdivision surface to an approximate parametric surface is to extract regular B-spline patches from the subdivision surface. For example, while regular regions of a Catmull-Clark surface can be directly mapped to bi-cubic B-spline patches, irregular regions need to be approximated to get C^1 continuous patches [61].

F. Semantic Enrichment

The problem of extracting semantic information from 3D data can be formulated simply as *What is the point?* [80] A-priori it is not clear whether a given point of a laser-scanned 3D scene, for example, belongs to a wall, to a door, or to the ground [81]. To answer this question is called semantic enrichment and it is always an act of interpretation [2].

The idea of generalized documents is to treat multimedia data, in particular 3D data sets, just like ordinary text documents, so that they can be inserted into a digital library. For any digital library to be able to handle a given media type, it must be integrated with the generic services that a digital library provides, namely markup, indexing, and retrieval. This defines a digital library in terms of the function it provides [82], [83]. Like any library, it contains meta-information for all data sets. In the simplest case, the metadata are of the Dublin Core type (title, creator/author, and time of creation, etc.) [84]. This is insufficient for large databases with a huge number of 3D objects, because of their versatility and rich structure. Scanned models are used in raw data collections, for documentation archival, virtual reconstruction, historical data analysis, and for high-quality visualization for dissemination purposes [85]. Navigating and browsing through the geometric models must be possible not only in 3D, but also on the semantic level. The need for higher-level semantic information becomes immediately clear when considering typical questions users might want to ask when a large database of 3D objects is available.

- How many different types of chairs are stored in the library?
- I want to compare the noses of all these statues, can you extract them?
- ...

These questions cannot be answered, if the library simply treats 3D objects as binary large objects (BLOB) as it is done quite often. For a heap of geometric primitives without semantics, it is hard – if not impossible – to realize the mandatory services required by a digital library, especially in the context of electronic data exchange, storage and retrieval.

In the context of CAD, the processes of markup, indexing, and retrieval are a challenge with many open problems [86], [87].

IV. Conclusion

Model representations and their transformation into each other have been a challenge in the past and will remain a future challenge as well. The search for a comprehensive model representation combining the advantages of the various, different approaches is still on-going.

Especially the semantic question remains unanswered. Adding semantics to shapes is an important, if not the vital, step towards the great vision of visual computing: To not only capture reality by sampling the world with 2D and 3D acquisition devices, but also to represent reality within a computer in a meaningful, ideally even in an editable form. Qualitative leaps can only be expected, if this open problem is solved, and the semantic gap is eventually closed in a reliable and sustainable way.

Acknowledgment

The authors gratefully acknowledge the support of the Austrian Research Promotion Agency, the Forschungsförderungsgesellschaft (FFG) for the research project AEDA (K-Projekt "Advanced Engineering Design Automation").

TABLE I. TRANSFORMATION BETWEEN MODEL REPRESENTATIONS.

Model Transformation from \ to	*Point Sets*	*Polygonal Faces*	*Parametric Surfaces*	*Subdivision Surfaces*	*Implicit Surfaces*	*Volumetric Shapes*	*Generative Shapes*
Point Sets		Poisson Reconstruction [41], [42]	Surface Fitting and Regression [10]	Surface Fitting [43]	Surface Fitting [44], Gaussian Density Computation [45]	Direct Evaluation [46], Gaussian Density Computation [45]	Generative Fitting [38]
Polygonal Faces	Monte Carlo Sampling [47], [48]	Mesh Processing [5]	Surface Fitting [49]	Surface Fitting [50], [51]	Variational Inter- polation [52]	Scan-line Filling [53]	Generative Fitting [38]
Parametric Surfaces	Monte Carlo Sampling [47], [48]	Triangulation [11], [13]	Conversion [12], [11]	Extended Subdivision Surfaces [54], NURBS-compatible Subdivision [21], Conversion [55]	Spherical Coordinates [56]	Forward Differencing [53]	Inverse (Procedural) Modeling [37]
Subdivision Surfaces	Monte Carlo Sampling [47], [48]	Evaluation [57], Tesselation [58]	Exact Evaluation [59], [60], Patching [61], Extended Subdivision Surfaces [54]			Evaluation [57] / Tesselation [58] with Forward Differencing [53]	Inverse (Procedural) Modeling [62]
Implicit Surfaces	Point Evaluation [63]	Marching Cubes [64], [65]	Spherical coordinate [56]	Interpolation [66]		Voxelization [67]	Inverse (Procedural) Modeling [68]
Volumetric Shapes	Point Sampling / Iso-Surface-Extraction [69]	Marching Cubes [64]	via polygonal faces representation (marching cubes [64])	via polygonal faces representation (marching cubes [64])	via polygonal faces representation (marching cubes [64])		Generative Fitting [38], Inverse (Procedural) Modeling [68]
Generative Shapes	Evaluation [33]	Evaluation [33]	Evaluation [33]	Evaluation [33]	Evaluation [33]	Evaluation [33]	Euclides [36]

REFERENCES

[1] C. Schinko, U. Krispel, E. Eggeling, and T. Ullrich, "3d model representations and transformations in the context of computer-aided design: a state-of-the-art overview," Proceedings of the International Conference on Advances in Multimedia (MMedia), vol. 9, 2017, pp. 10–15.

[2] S. Havemann, T. Ullrich, and D. W. Fellner, "The Meaning of Shape and some Techniques to Extract It," Multimedia Information Extraction, vol. 1, 2012, pp. 81–98.

[3] M. Zwicker, M. Pauly, O. Knoll, and M. Gross, "Pointshop 3D: an interactive system for point-based surface editing," Proceedings of 2002 ACM Siggraph, vol. 21, 2002, pp. 322–329.

[4] M. Gross and H. Pfister, Point-Based Graphics. San Francisco, California, USA: Morgan Kaufmann Publishers Inc., 2007.

[5] M. Botsch, L. Kobbelt, and M. Pauly, Polygon Mesh Processing. Natick, Massachusetts, USA: AK Peters, 2010.

[6] M. Attene, D. Giorgi, M. Ferri, and B. Falcidieno, "On converting sets of tetrahedra to combinatorial and pl manifolds," Computer Aided Geometric Design, vol. 26, 2009, pp. 850–864.

[7] M. Botsch, S. Steinberg, S. Bischoff, and L. Kobbelt, "Openmesh – a generic and efficient polygon mesh data structure," Proceedings of OpenSG Symposium, vol. 1, 2002, pp. 1–5.

[8] G. Farin, Curves and Surfaces for Computer Aided Geometric Design, G. Farin, Ed. Academic Press Professional, Inc., 1990.

[9] H. Pottmann and S. Leopoldseder, "Geometries for CAGD," Handbook of 3D Modeling, G. Farin, J. Hoschek, and M.-S. Kim (editors), vol. 1, 2002, pp. 43–73.

[10] J. Hoschek and D. Lasser, Grundlagen der Geometrischen Datenverarbeitung (english: Fundamentals of Computer Aided Geometric Design), J. Hoschek and D. Lasser, Eds. Teubner, 1989.

[11] H. Prautzsch, W. Boehm, and M. Paluszny, Bézier and B-Spline Techniques, H. Prautzsch, W. Boehm, and M. Paluszny, Eds. Springer, 2002.

[12] G. Aumann and K. Spitzmüller, Computerorientierte Geometrie (english: Computer-Oriented Geometry), G. Aumann and K. Spitzmüller, Eds. BI-Wissenschafts-Verlag, 1993.

[13] L. Piegl and W. Tiller, The NURBS book, L. Piegl and W. Tiller, Eds. Springer-Verlag New York, Inc., 1997.

[14] J. Fisher, J. Lowther, and C.-K. Shene, "If you know b-splines well, you also know NURBS!" Proceedings of the 35th SIGCSE technical symposium on Computer science education, vol. 35, 2004, pp. 343–347.

[15] B. Hamann and P.-Y. Tsai, "A tessellation algorithm for the representation of trimmed NURBS surfaces with arbitrary trimming curves." Computer Aided Design, vol. 28, 1996, pp. 461–472.

[16] G. Farin, NURBS for Curve and Surface Design from Projective Geometry to Practical Use, G. Farin, Ed. AK Peters, Ltd., 1999.

[17] E. Catmull and J. Clark, "Recursively generated B-spline surfaces on arbitrary topological meshes," Computer-Aided Design, vol. 10, 1978, pp. 350–355.

[18] D. Doo and M. Sabin, "Behavior of Recursive Division Surfaces near Extraordinary Points," Computer Aided Design, vol. 10, no. 6, 1978, pp. 356–360.

[19] C. Loop, "Smooth Subdivision Surfaces Based on Triangles," Master's Thesis, University of Utah, USA, vol. 1, 1987, pp. 1–74.

[20] L. Kobbelt, "Interpolatory Subdivision on Open Quadrilateral Nets with Arbitrary Topology," Computer Graphics Forum, vol. 15, no. 3, 1996, pp. 409–420.

[21] T. J. Cashman, U. H. Augsdörfer, N. A. Dodgson, and M. A. Sabin, "Nurbs with extraordinary points: High-degree, non-uniform, rational subdivision schemes," ACM Transactions on Graphics, vol. 28, no. 3, Jul. 2009, pp. 46:1–46:9.

[22] E. Sultanow, "Implizite Flächen (english: Implicit surfaces)," Technical Report at Hasso-Plattner-Institut, vol. 1, 2004, pp. 1–11.

[23] A. Knoll, Y. Hijazi, C. Hansen, I. Wald, and H. Hagen, "Interactive Ray Tracing of Arbitrary Implicits with SIMD Interval Arithmetic," Proceedings of IEEE Symposium on Interactive Ray Tracing, vol. 7, 2007, pp. 11–18.

[24] A. A. Pasko and V. V. Savchenko, "Blending Operations for the Functionally Based Constructive Geometry," Set-theoretic Solid Modeling: Techniques and Applications / Information Geometers, vol. 94, 1994, pp. 151–161.

[25] G. I. Pasko, A. A. Pasko, and T. L. Kunii, "Bounded Blending for Function-Based Shape Modeling," IEEE Computer Graphics and Applications, vol. 25, 2005, pp. 36–45.

[26] B. Grünbaum and G. C. Shephard, "Convex polytopes," Bull. Lond. Math. Soc., vol. 1, 1969, pp. 257–300.

[27] U. Krispel, T. Ullrich, and D. W. Fellner, "Fast and Exact Plane-Based Representation for Polygonal Meshes," Proceeding of the International Conference on Computer Graphics, Visualization, Computer Vision and Image Processing, vol. 8, 2014, pp. 189–196.

[28] W. Thaller, U. Krispel, R. Zmugg, S. Havemann, and D. W. Fellner, "Shape Grammars on Convex Polyhedra," Computers & Graphics, vol. 37, 2013, pp. 707–717.

[29] Y. Hijazi, A. Knoll, M. Schott, A. Kensler, C. Hansen, and H. Hagen, "CSG Operations of Arbitrary Primitives with Interval Arithmetic and Real-Time Ray Casting," Scientific Visualization: Advanced Concepts, vol. 978-3-939897-19-4, 2010, pp. 78–89.

[30] B. Naylor, J. Amanatides, and W. Thibault, "Merging bsp trees yields polyhedral set operations," ACM Transactions on Graphics, vol. 24, no. 4, 1990, pp. 115–124.

[31] U. Schöning, Theoretische Informatik - kurz gefasst, 5th ed. Heidelberg: Spektrum Akademischer Verlag, 2008.

[32] C. Schinko, M. Strobl, T. Ullrich, and D. W. Fellner, "Modeling Procedural Knowledge – a generative modeler for cultural heritage," Proceedings of EUROMED 2010 - Lecture Notes on Computer Science, vol. 6436, 2010, pp. 153–165.

[33] U. Krispel, C. Schinko, and T. Ullrich, "A Survey of Algorithmic Shapes," Remote Sensing, vol. 7, 2015, pp. 12 763–12 792.

[34] R. Berndt, D. W. Fellner, and S. Havemann, "Generative 3D Models: a Key to More Information within less Bandwidth at Higher Quality," Proceeding of the 10th International Conference on 3D Web Technology, vol. 1, 2005, pp. 111–121.

[35] D. W. Fellner and S. Havemann, "Striving for an adequate vocabulary: Next generation metadata," Proceedings of the 29th Annual Conference of the German Classification Society, vol. 29, 2005, pp. 13–20.

[36] C. Schinko, M. Strobl, T. Ullrich, and D. W. Fellner, "Scripting technology for generative modeling," International Journal on Advances in Software, vol. 4, no. 3-4, 2011, pp. 308–326.

[37] U. Krispel, C. Schinko, and T. Ullrich, "The Rules Behind – Tutorial on Generative Modeling," Proceedings of Symposium on Geometry Processing / Graduate School, vol. 12, 2014, pp. 21–249.

[38] T. Ullrich and D. W. Fellner, "Generative Object Definition and Semantic Recognition," Proceedings of the Eurographics Workshop on 3D Object Retrieval, vol. 4, 2011, pp. 1–8.

[39] T. Ullrich, V. Settgast, U. Krispel, C. Fünfzig, and D. W. Fellner, "Distance Calculation between a Point and a Subdivision Surface," Proceedings of 2007 Vision, Modeling and Visualization (VMV), vol. 1, 2007, pp. 161–169.

[40] T. Ullrich, V. Settgast, and D. W. Fellner, "Semantic Fitting and Reconstruction," Journal on Computing and Cultural Heritage, vol. 1, no. 2, 2008, pp. 1201–1220.

[41] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson surface reconstruction," Symposium on Geometry Processing, vol. 4, 2006, pp. 61–70.

[42] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," IEEE International Conference on Robotics and Automation (ICRA), vol. 28, May 9-13 2011, pp. 1–4.

[43] K.-S. D. Cheng, W. Wang, H. Qin, K.-Y. K. Wong, H. Yang, and Y. Liu, "Fitting Subdivision Surfaces to Unorganized Point Data using SDM," Proceedings of 12th Pacific Conference on Computer Graphics and Applications, vol. 1, 2004, pp. 16–24.

[44] P. Keller, O. Kreylos, E. S. Cowgill, L. H. Kellogg, and M. Hering-Bertram, "Construction of implicit surfaces from point clouds using a feature-based approach," Dagstuhl Publishing, vol. 2, 2011, pp. 129–143.

[45] R. Preiner, O. Mattausch, M. Arikan, R. Pajarola, and M. Wimmer,

"Continuous projection for fast l1 reconstruction," ACM Transactions on Graphics, vol. 20, no. 9, 2014, pp. 1280–1292.

[46] S. Muraki, "Volumetric shape description of range data using 'blobby model'," SIGGRAPH Comput. Graph., vol. 25, no. 4, Jul. 1991, pp. 227–235.

[47] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia, "Meshlab: an open-source mesh processing tool," Eurographics Italian Chapter Conference, vol. 3, 2008, pp. 129–136.

[48] D. P. Kroese, T. Brereton, T. Taimre, and Z. I. Botev, "Why the monte carlo method is so important today," Wires – Computational Statistics, vol. 6, 2014, pp. 386–392.

[49] W. Ma and J. P. Kruth, "Nurbs curve and surface fitting for reverse engineering," The International Journal of Advanced Manufacturing Technology, vol. 14, no. 12, 1998, pp. 918–927.

[50] X. Ma, S. Keates, Y. Jiang, and J. Kosinka, "Subdivision surface fitting to a dense mesh using ridges and umbilics," Computer Aided Geometric Design, vol. 32, 2015, pp. 5–21.

[51] D. Panozzo, M. Tarini, N. Pietroni, P. Cignoni, and E. Puppo, "Automatic construction of quad-based subdivision surfaces using fitmaps," IEEE Transactions on Visualization & Computer Graphics, vol. 17, no. undefined, 2011, pp. 1510–1520.

[52] G. Yngve and G. Turk, "Robust creation of implicit surfaces from polygonal meshes," IEEE Transactions on Visualization and Computer Graphics, vol. 8, no. 4, 2002, pp. 346–359.

[53] A. Kaufman, "Efficient algorithms for 3d scan-conversion of parametric curves, surfaces, and volumes," Proceedings of the annual conference on computer graphics and interactive techniques, vol. 14, 1987, pp. 171–179.

[54] K. Mueller, L. Reusche, and D. W. Fellner, "Acm transactions on graphics," Extended subdivision surfaces: Building a bridge between NURBS and Catmull-Clark surfaces, vol. 25, 2006, pp. 268–292.

[55] J. Shen, J. kosinka, M. Sabin, and N. Dodgson, "Computer aided geometric design," Converting a CAD model into a non-uniform subdivision surface, vol. 48, 2016, pp. 17–35.

[56] C. Ünsalan and A. Erçil, "Conversions between parametric and implicit forms using polar/spherical coordinate representations," Computer Vision and Image Understanding, vol. 81, no. 1, 2001, pp. 1–25.

[57] W. Ma, "Subdivision surfaces for cad: an overview," Computer-Aided Design, vol. 37, no. 7, 2005, pp. 693–709.

[58] K. Müller and S. Havemann, "Subdivision surface tesselation on the fly using a versatile mesh data structure," Computer Graphics Forum, vol. 19, 2000, pp. 151–159.

[59] J. Stam, "Exact evaluation of Catmull-Clark subdivision surfaces at arbitrary parameter values," Proceedings of the annual conference on computer graphics and interactive techniques, vol. 25, 1998, pp. 395–404.

[60] ——, "Evaluation of Loop subdivision surfaces," Proceedings of the annual conference on computer graphics and interactive techniques (Course Notes), vol. 26, 1999, pp. 1–15.

[61] J. Peters, "Patching Catmull-Clark meshes," Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques, vol. 27, 2000, pp. 255–258.

[62] C. Schinko, U. Krispel, T. Ullrich, and D. W. Fellner, "Built by Algorithms – State of the Art Report on Procedural Modeling," Proceeding of the International Workshop on 3D Virtual Reconstruction and Visualization of Complex Architectures (3D-ARCH), vol. 6, 2015, pp. 469–479.

[63] P. Ning and J. Bloomenthal, "An evaluation of implicit surface tilers," IEEE Computer Graphics and Applications, vol. 13, no. 6, 1993, pp. 33–41.

[64] T. S. Newman and H. Yi, "A survey of the marching cubes algorithm," Computers & Graphics, vol. 30, 2006, pp. 854–879.

[65] L. P. Kobbelt, M. Botsch, U. Schwanecke, and H.-P. Seidel, "Feature sensitive surface extraction from volume data," International Conference on Computer Graphics and Interactive Techniques, vol. 28, 2001, pp. 57–66.

[66] X. Jin, H. Sun, and Q. Peng, "Subdivision interpolating implicit surfaces," Computers & Graphics, vol. 27, no. 5, 2003, pp. 763–772.

[67] N. Stolte and A. Kaufman, "Novel techniques for robust voxelization and visualization of implicit surfaces," Graphical Models, vol. 63, no. 6, 2001, pp. 387–412.

[68] C. Schinko, U. Krispel, and T. Ullrich, "Know the Rules – Tutorial on Procedural Modeling," Proceedings of the International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (GRAPP Tutorial Notes), vol. 10, 2015, p. 27ff.

[69] R. U. Lobello, F. Dupont, and F. Denis, "Out-of-core adaptive isosurface extraction from binary volume data," Graphical Models, vol. 76, 2014, pp. 593–608.

[70] D. Luebke, M. Reddy, J. D. Cohen, A. Varshney, B. Watson, and R. Huebner, Level of Detail for 3D Graphics, 1st ed. Heidelberg, Germany: Morgan Kaufmann, 2002.

[71] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," Communications of the ACM, vol. 24, no. 6, 1981, pp. 381–395.

[72] R. Schnabel, R. Wahl, and R. Klein, "Efficient RANSAC for Point-Cloud Shape Detection," Computer Graphics Forum, vol. 26, no. 2, 2007, pp. 214–226.

[73] T. Itoh, Y. Yamaguchi, and K. Koyamada, "Fast isosurface generation using the volume thinning algorithm," IEEE Transactions on Visualization and Computer Graphics, vol. 7, 2001, pp. 32–46.

[74] T. Fujimori, Y. Kobayashi, and H. Suzuki, "Separated medial surface extraction from ct data of machine parts," Proceedings of the international conference on Geometric Modeling and Processing (GMP), vol. 4, 2006, pp. 313–324.

[75] T. J. R. Hughes, J. Cottrell, and Y. Bazilevs, "Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement," Computer Methods in Applied Mechanics and Engineering, vol. 194, 2005, pp. 4135–4195.

[76] F. Cirak, M. Ortiz, and P. Schröder, "Subdivision surfaces: A new paradigm for thin-shell finite element analysis," International Journal for Numerical Methods in Engineering, vol. 47, no. 12, 2000, pp. 2039–2072.

[77] A. Riffnaller-Schiefer, U. H. Augsdörfer, and D. W. Fellner, "Isogeometric shell analysis with NURBS compatible subdivision surfaces," Applied Mathematics and Computation, vol. 272, 2016, pp. 139–147.

[78] ——, "Isogeometric analysis for modelling and design," Proceedings of EUROGRAPHICS – Short Papers, vol. 34, 2015, pp. 17–20.

[79] ——, "Interactive physics-based deformation for virtual worlds," Proceedings of the International Conference on Cyberworlds, 2017 (to appear).

[80] S. Biasotti, B. Falcidieno, D. Giorgi, and M. Spagnuolo, Mathematical Tools for Shape Analysis and Description. Morgan & Claypool Publishers, 2014.

[81] M. Attene, F. Robbiano, M. Spagnuolo, and B. Falcidieno, "Characterization of 3d shape parts for semantic annotation," Computer-Aided Design, vol. 41, 2009, pp. 756–763.

[82] D. W. Fellner, "Graphics Content in Digital Libraries: Old Problems, Recent Solutions, Future Demands," Journal of Universal Computer Science, vol. 7, 2001, pp. 400–409.

[83] D. W. Fellner, D. Saupe, and H. Krottmaier, "3D Documents," IEEE Computer Graphics and Applications, vol. 27, no. 4, 2007, pp. 20–21.

[84] Dublin Core Metadata Initiative, "Dublin Core Metadata Initiative," http://dublincore.org/ [retrieved: Feb. 2017], 1995.

[85] V. Settgast, T. Ullrich, and D. W. Fellner, "Information Technology for Cultural Heritage," IEEE Potentials, vol. 26, no. 4, 2007, pp. 38–43.

[86] C. Schinko, T. Vosgien, T. Prante, T. Schreck, and T. Ullrich, "Search & retrieval in cad databases – a user-centric state-of-the-art overview," Proceedings of the International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (GRAPP 2017), vol. 12, 2017, pp. 306–313.

[87] H. Laga, M. Mortara, and M. Spagnuolo, "Geometry and context for semantic correspondences and functionality recognition in man-made 3d shapes," ACM Transactions on Graphics, vol. 32, 2013, p. 150ff.