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

Christoph Schinko, Ulrich Krispel, Eva Eggeling, and Torsten Ullrich
Institute of Computer Graphics and Knowledge Visualization, Graz University of Technology
& Visual Computing, Fraunhofer Austria Research GmbH, Austria
email: { christoph.schinko, ulrich.krispel, eva.eggeling, torsten.ullrich } @fraunhofer.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**

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 of a digital counterpart of a real object is called a shape description. At this point, it is important to emphasize that there are differences in the process of shape perception between human beings and computers. 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 [1]. 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 sections describe the model representations (Section 2), transformation (Section 3) and Level-of-Detail (Section 4) techniques, as well as semantic enrichment methods (Section 5).

II. Model Representations

In dictionaries, shapes are described by words forming a textual definition. 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. 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. 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 [2]. 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.

For rendering approaches of point sets, the literature survey by Markus Gross and Hanspeter Pfister offers in-depth explanation [3]. 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. 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 [4] [5].

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. 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. 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 [6].

C. Parametric Surface Representations

A parametric representation of a shape's surface is defined by a function $f : \Omega \to S$ mapping a 2D parameter domain $\Omega \subset \mathbb{R}^2$ to the surface $S = f(\Omega) \subset \mathbb{R}^3$. As any surface can be approximated by polynomials, the concept of polynomial surface patches has gained currency in the CAD domain [7] [8].

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 [9].

1. Bézier Surfaces

A Bézier surface is a three-dimensional surface generated from the Cartesian product of two Bézier curves [10]. A Bézier surface of degree (m, n) is defined as a parametric function $f(u, v) = \sum_{i=0}^{m} \sum_{j=0}^{n} b_{ij} B_i^m(u) B_j^n(v)$. It is evaluated over the unit square $(u, v) \in [0, 1] \times [0, 1]$ with the control points b_{ij} using the Bernstein polynomials $B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i}$ of degree n, for $t \in [0, 1]$.

In CAD, Bézier surfaces are often used in the form of bi-cubic Bézier patches, i.e., a set of 4×4 points represents the control mesh and is responsible for the shape of the surface. In all cases, Bézier curves and surfaces have important properties:

- A Bézier surface fulfils the partition of unity property, i.e., $\sum_{i=0}^{m} \sum_{j=0}^{n} B_i^m(u) B_j^n(v) = 1$, thus the relationship between a Bézier surface and its control mesh is invariant under affine transformations.
- A Bézier surface is contained within the convex hull of its control mesh and the four corner points of the control mesh are interpolated by the Bézier 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.

2. Rational Bézier Surfaces

The idea behind rational Bézier surfaces is to add adjustable weights to extend the design space of shapes [11]. 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 b_{ij}, the weights w_{ij}, and the Bernstein polynomials $B_i^m(u)$ and $B_j^n(v)$ as

$$f(u, v) = \frac{\sum_{i=0}^{m} \sum_{j=0}^{n} w_{ij} b_{ij} B_i^m(u) B_j^n(v)}{\sum_{i=0}^{m} \sum_{j=0}^{n} w_{ij} B_i^m(u) B_j^n(v)}.$$

3. B-spline Surfaces

B-spline surfaces exhibit advantages when joining patches under continuity requirements. Let $m, n, k, l \in \mathbb{N}$ with $n \geq k$ and $m \geq l$. Then, a B-spline surface of degree (l, k) is defined as $f(u, v) = \sum_{i=0}^{m} \sum_{j=0}^{n} d_{ij} N_i^l(u) N_j^k(v)$, with the basis functions $N_i^0(t) = 1, \text{if} \quad t_i \leq t < t_{i+1}; \text{and } 0, \text{otherwise}$ 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 nondecreasing 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 $(u, v) \in [u_l, u_{m+1}[\times [v_k, v_{n+1}[$ with the control points d_{ij} and the polynomials $N_i^l(u)$ and $N_j^k(v)$. The control points 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 with 4×4 control points per patch. B-splines with knots t_i satisfying the condition $t_0 = 0$ and $t_{i+1} = t_i$ or $t_{i+1} = t_1 + 1$, $(i = 0, \ldots, n + m)$ are called uniform B-splines.

B-spline surfaces satisfy properties similar to Bézier surfaces [10]. (1) The relationship between a B-spline 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. (3) In contrast to Bézier surfaces, the control points exert local control – if a control point is moved, only the local neighbourhood is affected and (4) by choosing appropriate knot vectors, a B-spline surface can become a Bézier surface.

4. NURBS Surfaces

The combination of rational Bézier techniques and B-Spline techniques leads to non-uniform, rational B-Splines, NURBS for short [12]:

Let $m, n, k, l \in \mathbb{N}$ with $n \geq k$ and $m \geq l$. Additionally, let $w_{00}, \ldots, w_{mn} \in \mathbb{R}$, u = $(u_0 \ldots u_{m+l+1})^T$ and v = $(v_0 \ldots v_{n+k+1})^T$ be two knot vectors and $d_{00}, \ldots, d_{mn} \in \mathbb{R}^3$. Then, a non-uniform, rational B-spline (NURBS) surface of degree (l, k) is defined as

$$f(u, v) = \frac{\sum_{i=0}^{m} \sum_{j=0}^{n} w_{ij} d_{ij} N_i^l(u) N_j^k(v)}{\sum_{i=0}^{m} \sum_{j=0}^{n} w_{ij} N_i^l(u) N_j^k(v)}.$$

over $(u, v) \in [u_l, u_{m+1}[\times [v_k, v_{n+1}[$ with the control points d_{ij}, the polynomials $N_i^l(u)$ and $N_j^k(v)$, the knot vectors u and v for the de Boor points $d_{00}, \ldots, d_{mn}$ and the 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 [13]. 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.

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. The resulting surfaces offer precise feature control at the cost of computational complexity due to trimming and stitching [14].

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 [15], Doo-Sabin [16], Loop [17], Kobbelt [18], etc.

The subdivision scheme presented by Edwin Catmull and Jim Clark is a generalization of bicubic B-Spline surfaces to arbitrary topology [15]. The set of 4×4 control points p_{ij}

forms the starting mesh for an iterative refinement process where each step results in a finer mesh.

Subdivision surfaces are invariant under affine transformations. They offer the benefit of being easy to implement and computationally efficient. Only the local neighbourhood 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.

E. Implicit Surface Representations

In contrast to the parametric surface representations described above, implicit surfaces, are defined as isosurfaces by a function $\mathbb{R}^3 \to \mathbb{R}$ [19]. 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

$$f(x, y, z) = (x^2 + y^2 + z^2 + R^2 - r2)^2 - 4R^2(x^2 + y^2) = 0.$$

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 [20].

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. Surface intersections, as well as boolean set operations can also be implemented efficiently. Since the surface is not represented explicitly, topology changes are easily possible.

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.

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 on the 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. 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.

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 $\mathbb{E}^d$. 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 $\mathbb{E}^n$ ($n \geq 0$) [21]. A set $S \subseteq \mathbb{E}^d$ is convex, if for any pair of points $x, y \in S$, the line segment $\lambda x + (1 - \lambda)y$ 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 [22]. 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. Open convex polytopes are defined similarly with strict inequalities instead of non-strict ones [23].

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.

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 [24].

However, CSG can also be performed on other shapes and shape representations. Two different approaches can be used to create 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. [25]. The algorithm is relying on accurate information of inside and outside of a shape (or, in case of planes, above and below).

G. Algorithmic/Generative Shape Descriptions

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 always terminate [26]. On the other hand, procedural descriptions offer additional features, like infinite loops (the functions it can compute are computable functions), 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. 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" [27] 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 [28].

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 [29]. 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 [30]. 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.

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 [31].

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 [32].

III. Model Transformation

In a product lifecycle, the digital counterpart of a future, real-world object has to pass several stages of a multistep pipeline. First sketches of a product are represented in a different representation than the final CAD production-ready dataset. 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 most important representations Table I lists the conversion methods and algorithms.

IV. 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 [53].

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 modelling software to run faster and reliably rendered across all the new and old hardware.

V. Semantic Enrichment

The problem of extracting semantic information from 3D data can be formulated simply as *What is the point?* [54] 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 [55]. To answer this question is called semantic enrichment and it is always an act of interpretation [1].

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 DL provides, namely markup, indexing, and retrieval. This defines a digital library in terms of the function it provides [56] [57]. Like any library, it contains meta-information for all data sets. In the simplest case, the metadata are of the Dublin Core type

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		Surface Reconstruction Library [33], Poison Reconstruction [34]	Surface Fitting and Regression [9]	Surface Fitting [35]	Surface Fitting [36]	Direct Evaluation [37]	Generative Fitting [32]
Polygonal Faces	Monte Carlo Sampling [38]		Surface Fitting [39]	Surface Fitting [40] [41]	Variational Interpolation [42]	Scan-line Filling [43]	Generative Fitting [32]
Parametric Surfaces	Monte Carlo Sampling [38]	Triangulation [10] [12]	Conversion [11]	NURBS-compatible Subdivision [44]	Spherical Coordinates [45]	Forward Differencing [46]	Inverse (Procedural) Modeling
Subdivision Surfaces	Point Sampling [38]	Evaluation [47]	NURBS-compatible Subdivision [44]			Evaluation [47] with Forward Differencing [46]	Inverse (Procedural) Modeling
Implicit Surfaces	Point Evaluation [48]	Marching Cubes [49]	Spherical coordinate representations [45]	Interpolation [50]		Voxelization [51]	Inverse (Procedural) Modeling
Volumetric Shapes	Point Sampling / Iso-Surface-Extraction	Marching Cubes [49]	via Marching Cubes	via Marching Cubes	via Marching Cubes		Inverse (Procedural) Modeling
Generative Shapes	Evaluation [28]	Evaluation [28]	Evaluation [28]	Evaluation [28]	Evaluation [28]	Evaluation [28]	Euclides [52]

(title, creator/author, and time of creation, etc.) [58]. 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 [59]. 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 [60] [61].

VI. 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.

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").

REFERENCES

[1] 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.

[2] 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.

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

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

[5] 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.

[6] 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.

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

[8] 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.

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

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

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

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

[13] 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.

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

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

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

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

[18] L. Kobbelt, "Interpolatory Subdivision on Open Quadrilateral Nets with

Arbitrary Topology," Computer Graphics Forum, vol. 15, no. 3, 1996, pp. 409–420.

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

[20] 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.

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

[22] 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.

[23] 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.

[24] 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.

[25] B. Naylor, J. Amanatides, and W. Thibault, "Merging bsp trees yields polyhedral set operations," SIGGRAPH Comput. Graph., vol. 24, no. 4, 1990, pp. 115–124.

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

[27] 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.

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

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China, May 9-13 2011.

[34] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson Surface Reconstruction," in Symposium on Geometry Processing, A. Sheffer and K. Polthier, Eds. The Eurographics Association, 2006.

[35] 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.

[36] 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," in Scientific Visualization: Interactions, Features, Metaphors, ser. Dagstuhl Follow-Ups, H. Hagen, Ed. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2011, vol. 2, pp. 129–143.

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

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

[39] 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.

[40] 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.

[41] 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.

[42] 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.

[43] A. Kaufman, "An Algorithm for 3D Scan-Conversion of Polygons," in EG 1987-Technical Papers, A. Kaufman, Ed. Eurographics Association, 1987.

[44] 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 Trans. Graph., vol. 28, no. 3, Jul. 2009, pp. 46:1–46:9.

[45] C. Ünsalan and A. Erçil, "Conversions between parametric and implicit forms using polar/spherical coordinate representations," Comput. Vis. Image Underst., vol. 81, no. 1, 2001, pp. 1–25.

[46] A. Kaufman, "Efficient algorithms for 3d scan-conversion of parametric curves, surfaces, and volumes," in Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, ser. SIGGRAPH '87, 1987, pp. 171–179.

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

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

[49] E. V. Chernyaev, "Marching Cubes 33: Construction of topologically correct isosurfaces," Technical Report CN/95-17, 1995.

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

[51] 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.

[52] 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.

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

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

[55] 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.

[56] 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.

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

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

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

[60] 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.

[61] 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.