Remote Sens. **2015**, *7*, 12763-12792; doi:10.3390/rs71012763

OPEN ACCESS

remote sensing

ISSN 2072-4292

www.mdpi.com/journal/remotesensing

Review

A Survey of Algorithmic Shapes [†]

Ulrich Krispel *, Christoph Schinko and Torsten Ullrich

Fraunhofer Austria Research GmbH, Visual Computing & Technische Universität Graz, Inffeldgasse 16c, 8010 Graz, Austria; E-Mails: christoph.schinko@fraunhofer.at (C.S.); torsten.ullrich@fraunhofer.at (T.U.)

[†] This paper is an extended version of our paper published in Proceeding of the International Workshop on 3D Virtual Reconstruction and Visualization of Complex Architectures (3D-ARCH), 2015, 6, 469–479: Built by Algorithms–State of the Art Report on Procedural Modeling.

* Author to whom correspondence should be addressed; E-Mail: ulrich.krispel@fraunhofer.at; Tel.: +43-316-873-5416.

Academic Editors: Diego Gonzalez-Aguilera and Prasad S. Thenkabai

Received: 29 June 2015 / Accepted: 15 September 2015 / Published: 28 September 2015

Abstract: In the context of computer-aided design, computer graphics and geometry processing, the idea of generative modeling is to allow the generation of highly complex objects based on a set of formal construction rules. Using these construction rules, a shape is described by a sequence of processing steps, rather than just by the result of all applied operations: shape design becomes rule design. Due to its very general nature, this approach can be applied to any domain and to any shape representation that provides a set of generating functions. The aim of this survey is to give an overview of the concepts and techniques of procedural and generative modeling, as well as their applications with a special focus on archeology and architecture.

Keywords: geometry processing; generative; procedural modeling; inverse modeling; modeling applications; shape description; language design

1. Introduction

In the context of computer-aided design (CAD) and shape description, the digital creation of a shape is called modeling. The most common representation of a shape is a composition of elementary objects.

However, a shape can also be described by its generating process. In this case, the description is called a generative model. A generative model does not describe a shape by the parts it consists of, but by the operations and steps needed to be performed in order to create it; *i.e.*, a generative model is an algorithm. Its implementation is an algorithmic description written in a programming language. Depending on the used software engineering paradigm, a generative model may also be called a procedural model or a functional model, if the algorithm is implemented in a procedural way, respectively functionally.

For many purposes in CAD, the mightiness of a Turing-complete programming language may lead to potential problems, such as the halting problem. In order to avoid these problems, CAD frameworks often offer a language that is not Turing-complete; *i.e.*, the set of language features is reduced to parametric modeling.

In generative modeling, the object is not just the end result of applied operations, as this paradigm describes a shape by a sequence of processing steps. The result is a paradigm shift from shape design to rule design. This general approach can be applied to many domains.

1.1. Ruler and Compass

Geometry from the days of the ancient Greeks placed great emphasis on problems of constructing various geometric figures using only a ruler without markings (to draw lines) and a compass (to draw circles). Ruler-and-compass constructions are based on EUCLID's axioms [1] using points, lines and circles that have already been constructed. The resulting geometric primitives together with the ruler-and-compass constructions are the first algorithmic descriptions of generative models. EUCLID's Elements is probably the most successful textbook ever written. It still influences modern curricula of mathematics [2]. As the history of geometry [3,4] is not within the scope of article, this article jumps directly to modern uses of generative modeling techniques.

The long history of geometric constructions [6] is also reflected in the history of civil engineering and architecture [7]. Gothic architecture, especially window tracery, exhibits a good example of these constructions. Their complexity is achieved by combining only a few basic geometric patterns. SVEN HAVEMANN and DIETER W. FELLNER show how constructions of prototypic Gothic windows can be formalized using generative modeling techniques [8]. By combining modular construction rules, it is possible that complex configurations can be obtained from elementary constructions. The different combinations of specific parametric features can be grouped together, leading to the concept of styles. A differentiation between basic shape and appearance allows, for example, the creation of ornamental decoration, as seen in Figure 1 [5]. This leads to an extremely compact representation for a whole class of shapes [9].

1.2. Natural Patterns

In today's procedural modeling systems, scripting languages and grammars are often used to create a set of rules to achieve a description of an object or pattern. Early systems based on grammars were Lindenmayer systems, short L-systems, named after ARISTID LINDENMAYER [10]. They were successfully used for modeling plants [11] or fractal structures [12]. Given a set of string rewriting rules, complex strings are created by applying these rules to simpler strings. Starting with an initial string, the predefined set of rules form a new, possibly larger string. In order to use L-systems to model geometry, an interpretation of the generated strings is necessary.

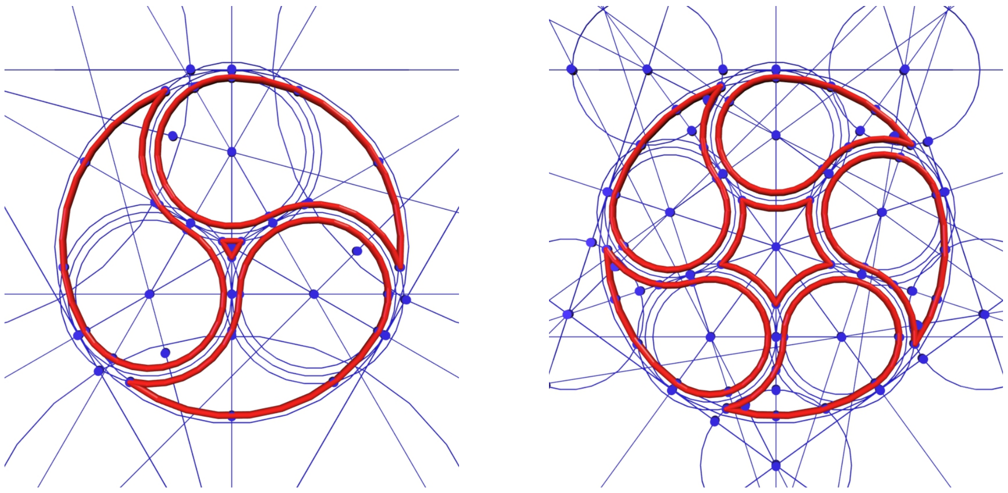

Figure 1. Compass-and-ruler operations have long been used in interactive procedural modeling. This Gothic window construction was created in the framework presented by WOLFGANG THALLER *et al.* using direct manipulation without any code or graph editing [5].

The modeling power of L-systems was limited to creating fractals and plant-like branching structures. This limitation led to the introduction of parametric L-systems. The idea is to associate numerical parameters with L-system symbols to address continuous phenomena, which were not covered satisfactorily by L-systems alone.

In combination with additional 3D modeling techniques, Lindenmayer systems can be used to generate complex geometry. ROBERT F. TOBLER *et al.* introduce a combination of subdivision surfaces, fractal surfaces and parametrized L-systems to create models of natural phenomena [13,14]. Different combinations can be used at each level of resolution. Since the whole description of such multi-resolution models is procedural, their representation is very compact and can be exploited by level-of-detai renderers.

This trade-off between data storage and computation time can be found in various fields of computer graphics, e.g., the tessellation of curved surfaces specified by a few control points directly on the GPU. The result is low storage costs, allowing the generation of complex models only when needed, while also reducing memory transfer overheads. Although L-systems are parallel rewriting systems, derivation through rewriting leads to very uneven workloads. Since the interpretation of an L-system is an inherently serial process, they are not straightforwardly applicable to parallel processing. In 2010, MARKUS LIPP *et al.* presented a solution to this algorithmic challenge [15].

2. Languages and Grammars

Scripting languages have been designed for a special purpose, e.g., for client-side scripting in a web browser. Nowadays, scripting languages are used for many different applications. JavaScript, for example, is used to animate 2D and 3D graphics in the Virtual Reality Modeling Language (VRML) [16] and Extensible 3D (X3D) [17] files. It checks user forms in PDF files [18], controls game engines [19], configures applications, defines 3D shapes [20] and performs many more tasks. According to JOHN K. OUSTERHOUT, scripting languages use a higher level of abstraction compared to system programming languages, as they are often typeless and interpreted to emphasize the rapid application development purpose [21]. System programming languages, on the other hand, are designed for creating algorithms and data structures based on low-level data types and memory operations. Consequently, graphics libraries [22], shaders [23] and scene graph systems [24,25] are usually written in C/C++ dialects [26], whereas procedural modeling frameworks incorporate scripting languages, such as Lua, JavaScript, *etc*.

2.1. Language Processing and Compiler Construction

For the evaluation of procedural descriptions, typically techniques used for the description of formal languages and compiler construction are used [27]. There is a wide range of different concepts of languages to describe a shape, including all kinds of linguistic concepts [28]. The main categories to describe a shape are:

- rule-based: using substitutions and substitution rules to generate complex structures out of simple starting structures [29–32];
- imperative and scripting-based: using a scripting engine and techniques from predominant programming languages [20,33,34] or;
- GUI and dataflow-based: using new graphical user interfaces (GUI) and intelligent GUIs to detect structures in modeling tasks, which can be mapped onto formal descriptions [35,36].

The general principles of formal descriptions and compiler construction are the same in all cases: independent of ahead-of-time compilation, just-in-time compilation or interpretation [37]. In the first stage of the compilation process, the input source code is passed to the lexer and parser. The first step here is to convert a sequence of characters into a sequence of tokens, which is done by special grammar rules forming the lexical analysis. Typically, only a limited number of characters is allowed for an identifier: all characters `A–Z`, `a–z`, digits `0–9` and the underscore _ are allowed with the condition that an identifier must not begin with a digit or an underscore. The lexer rules are embedded in another set of rules: the parser rules. They evaluate the resulting sequence of tokens to determine their grammatical structure. The complete grammar is of a hierarchical structure and consists of rules for analyzing all possible statements and expressions that can be formed in the language, thus forming the syntactic analysis.

For each available language construct, a set of rules ensures syntactic correctness and incorporates mechanisms to report possible syntactic errors and warnings. These rules are also used to create the intermediate abstract syntax tree (AST) structure that is a representation of the input source code to be used for the next stage: semantic analysis. Once all statements and expressions of the input source code are collected in the AST, a tree walker checks their semantic relationships for errors and warnings.

After performing all compile-time checks, a translator uses the AST to generate platform-specific files, possibly involving other intermediate structures.

As mentioned in the Introduction, the first procedural modeling systems were L-systems. Later on, L-systems were used in combination with shape grammars to model cities. YOGI PARISH and PASCAL MÜLLER presented a system that generates a street map enriched with geometry for buildings using a number of image maps as input [38]. The resulting framework called CityEngine is a modeling environment for the shape grammar Computational Geometry Algorithm (CGA) Shape. MARKUS LIPP *et al.* presented another modeling approach based on CGA Shape following the notation of PASCAL MÜLLER [31,35]. It enables more direct local control of the underlying grammar by introducing visual editing. Principles of semantic and geometric selection are combined, as well as functionality to store local changes persistently over global modifications.

SVEN HAVEMANN takes a different approach to generative modeling. He proposes a stack-based language called the Generative Modeling Language (GML) [33]. The postfix notation of the language is very similar to that of Adobe Postscript. High-level shape operations are created by using low-level shape functionality. A number of applications are based on the GML platform, because it is easily extensible and offers an integrated visualization engine. Current efforts in the context of the GML are devoted to directly creating interactive generative visualizations for the web.

Generative modeling inherits the methodologies of 3D modeling and programming [39], which lead to drawbacks in usability and productivity. The need to learn and use a programming language is a significant inhibition threshold, especially for non-computer scientists. The choice of the scripting language has a huge influence on the usability and effectiveness of procedural modeling. Processing is a good example of how an interactive, easy to use, yet powerful development environment can open up new user groups. It had been initially created to serve as a software sketchbook and to teach students fundamentals of computer programming. It quickly developed into a tool that is used for creating visual arts [40].

Processing is a Java-like interpreter offering new graphics and utility functions together with some usability simplifications. The large community behind the tool produced libraries to facilitate computer vision, data visualization, music, networking and electronics. The success of Processing is based on two factors: the simplicity of the programming language and the interactive experience. Instant feedback of the scripting environments allows the user to program via "trial and error".

2.2. Scripting Languages for Generative Modeling

There are many different programming paradigms in software development that are also used in the field of generative modeling, where some paradigms emerged to be useful for specific domains.

Imperative: Many generative models are described using classical programming paradigms: a programming language is used to generate a specific object, possibly using a library that utilizes some sort of geometry representation and operations to perform changes. Any modeling software that is scriptable by an imperative language or provides some sort of API falls into this category.

Dataflow based: A generative description can be represented by a directed graph of the data flowing between operations. This graph representation also allows for a graphical representation; visual programming languages (VPLs) allow one to create a program by linking and modifying visual elements. Many VPLs are based on the dataflow paradigm. Examples in the domain of generative modeling are the Grasshopper3D (online) plug-in for the Rhinoceros3D (online) modeling suite, or the work of GUSTOVA PATOW *et al.* built on top of the procedural modeler Houdini (online) [41].

Rule-based systems: Another different representation for generative modeling is rule-based systems. These systems provide a declarative description of the construction behavior of a model by a set of rules. An example are L-systems, as described in the Introduction. Furthermore, the seminal work of GEORGE STINY and JAMES GIPS introduced shape grammars, as a formal description of capturing the design of paintings and sculptures [42]. Similar to formal grammars, shape grammars are based on rule replacement.

2.3. Shape Grammars

A shape grammar consists of shape rules and a generation engine that selects and processes rules. A shape rule defines how an existing shape can be transformed. The work of PETER WONKA *et al.* applied the concepts of shape grammars to derive a system for generative modeling of architectural models [43]. This system uses a combination of a spatial grammar system (split grammar) to control the spatial design and a control grammar, which distributes the design ideas spatially (e.g., set different attributes for the first floor of a building). Both of these grammars consist of rules with attributes that steer the derivation process. The grammar consists of two types of rules: split and convert. The split rule is a partition operation, which replaces a shape by an arrangement of smaller shapes that fit in the boundary of the original shape. The convert rule replaces a shape by a different shape that also fits in the boundary of the original shape.

This system has further been extended by the work of PASCAL MÜLLER *et al.*, which introduced a component split to extend the split paradigm to arbitrary 3d meshes, as well as occlusion queries and snap lines to model non-local influences of rules [31]. For example, two wall segments that intersect each other should not produce windows, such that the window of one wall coincides with the other wall; therefore, occlusion queries are used to decide if a window should be placed or not.

JEAN-EUDES MARVIE *et al.* have shown that the derivation of a split grammar, starting from an initial shape, yields a tree structure, which suggests that the derivation can be sped up by a parallel implementation [44]. Parallel generation is especially useful in an urban context, with scenes with high complexity and detail. The work of LARS KRECKLAU *et al.* used GPU-accelerated generation in the context of generating and rendering highly detailed building facades [45]; the work of ZHENGZHENG KUANG *et al.* proposes a memory-efficient procedural representation of urban buildings for real-time visualization [46].

With more advanced shape grammar systems, the non-local influences are a problem because they introduce dependencies between arbitrary nodes of the derivation tree. Recent work by MARKUS STEINBERGER *et al.* shows how to overcome this problem in a GPU implementation [47]. Furthermore, the same authors presented methods to interactively generate and render only the visible part of a procedural scene using procedural occlusion culling and the level of detail [48].

2.4. Tools and Environments

A selection of commonly-used tools and programming environments for generative modeling is listed in the Tables 1 and 2.

Table 1. Overview of generative/procedural 3D modeling tools and approaches (Part 1).

Tool Name	Application Domain	Programming Category	Environment
Blender Scripting	general purpose modeling	python scripting	open source modeling software blender
CGAL, The Computational Geometry Algorithms Library[49]	general purpose modeling	C++	CGAL open source project
CityEngine [31]	urban modeling	CGA shape	commercial integrated development environment CityEngine
Generalized Grammar G^2 [30]	scientific	python scripting	commercial modeling software Houdini
Generative Modeling Language (GML) [33]	CAD	postscript dialect	proprietary, integrated development environment for polygonal and subdivision modeling
Grasshopper 3D	visual arts, rapid prototyping, architecture	visual programming based on dataflow graphs, Microsoft .NET family of languages	commercial modeling software Rhinoceros3D
HyperFun [50]	scientific	specialized high-level programming language	proprietary geometry kernel FRep (Function Representation)
Maya Scripting	general purpose modeling	Maya Embedded Language (MEL) and python scripting	commercial modeling software Autodesk Maya
OpenSCAD	CAD	OpenSCAD language	open source, based on CGAL geometry kernel
PLaSM	scientific	python scripting, Function Level scripting	integrated development environment Xplode
Processing	visual arts, rapid prototyping	Java dialect	open source, integrated development environment Processing

Table 2. Overview on generative/procedural 3D modeling tools and approaches (Part 2)

Tool Name	Application Domain	Programming Category	Environment
PythonOCC	general purpose modeling and CAD	python scripting	Open CASCADE Technology
Revit Scripting	architecture	Microsoft .NET family of languages	commercial modeling software Autodesk Revit
siteplan [51]	rapid prototyping, architecture	interactive GUI-based modeler	open source, integrated development environment siteplan
SketchUp Scripting	architecture, urban modeling and CAD	Ruby scripting	commercial modeling software SketchUp
Skyline Engine [41]	urban modeling	visual programming based on dataflow graphs, python scripting	commercial modeling software Houdini
speedtree	plants/trees	interactive GUI-based modeler, SDK for C++	standalone modeler and integration into various game engines
Terragen	landscape modeling	interactive GUI-based modeler	free and commercial, integrated development environment Terragen
XFrog [11]	plants/trees	interactive GUI-based modeler	integrated development environment, standalone and plugins for Maya and Cinema4D

3. Modeling by Programming

3D objects consisting of organized structures and repetitive forms are well suited for procedural descriptions, e.g., by the combination of building blocks or by using shape grammars.

3.1. Building Blocks and Elementary Data Structures

Creating shapes with elementary data structures requires the definition of modeling operations. Depending on the underlying representation, certain modeling operations are difficult or impossible to implement. The selection of operations for these data structures are manifold and can be grouped as follows:

- Instantiations are operations for creating new shapes.
- Binary creations are operations involving two shapes, such as constructive solid geometry (CSG) operations.
- Deformations and manipulations stand for all deforming and modifying operations, like morphing or displacing.

Building blocks can also be regarded as modeling operations. When creating an algorithmic description of a shape, an important task is to identify inherent properties and repetitive forms. These properties must be accounted for in the structure of the description. Identified subparts or repetitive forms are best mapped to functions in order to be reusable. However, the true power of an algorithmic description becomes obvious when parameters are introduced for these functions. Even if only used to position a subpart at a different location. From that point on, the algorithmic description no longer stands for a single object, but for a whole object family.

3.2. Architectural Modeling with Procedural Extrusions

This method utilizes the paradigm of footprint extrusion to automatically derive geometry from a coarse description. The inputs to this system are polygons whose segments can be associated with an extrusion profile polygon. The system utilizes the weighted straight skeleton method [52] to calculate the resulting geometry.

The growing demand for new building models for virtual worlds, games and movies, makes the easy and fast creation of modifiable models more and more important [53]. Nevertheless, 3D modeling of buildings can be a tedious task due to their sometimes complex geometry [54]. For historic buildings, especially the roofs can be challenging. JOHANNES EDELSBRUNNER *et al.* present a new method of combining simple building solids to form more complex buildings and give emphasis to the blending of roof faces [55]. Their method can be integrated in common pipelines for procedural modeling of buildings and extends their expressiveness compared to existing methods.

3.3. Deformation-Aware Shape Grammars

Generative models based on shape and split grammar systems often exhibit planar structures. This is the case because these systems are based on planar primitives and planar splits. There are many geometric tools available in modeling software to transform planar objects into curved ones, e.g., free-form deformation [56]. Applying such a transformation as a post-processing step might yield undesirable results. For example, if a planar facade of a building is bent into a curved shape, the windows inside the facade will have a curved surface, as well. Another possibly unwanted property arises when an object is deformed by scaling: the windows on a facade would have different appearances.

RENÉ ZMUGG *et al.* introduced deformation-aware shape grammars, which integrate deformation information into grammar rules [57]. The system still uses established methods utilizing planar primitives and splits; however, measurements that determine the available space for rules are performed in deformed space. In this way, deformed splits can be carried out, and the deformation can be baked at any point to allow for straight splits in deformed geometry.

3.4. Procedural Shape Modeling

The effectiveness of procedural shape modeling can be demonstrated with the mass customization of consumer products [58]. A generative description composed of a few well-defined procedures can generate a large variety of shapes. Furthermore, it covers most of the design space defined by an existing collection of designs; in this case, wedding rings.

The basic shape of most rings can be defined using a profile polygon, the angular step size defined by the number of supporting profiles to be placed around the ring's center, the radius and a vertex transformation function. A ring's design variations are decomposed into a set of transformation functions. Each function transforms selected parts of the profile in a certain way. Effects can be combined by calling a sequence of different transformations. The creation of the basic shape is separated from optional steps to create engravings, change materials or add gems. Engravings are implemented as per-vertex displacements (to maintain the option for 3D-printing) and can be applied on quadrilateral parts of the ring's mesh using half-edges to specify position and spatial extent.

Materials, like gold, silver and platinum, are used for wedding rings. Their surfaces can be treated with various finishing techniques, like polishing, brushing or hammering. In order to account for these effects, a per-pixel shading model is used featuring anisotropic highlights. By using a cube map, visually appealing reflections are created, and predefined surface finishes can be applied using normal mapping techniques. Procedural gem instances can also be placed on the ring.

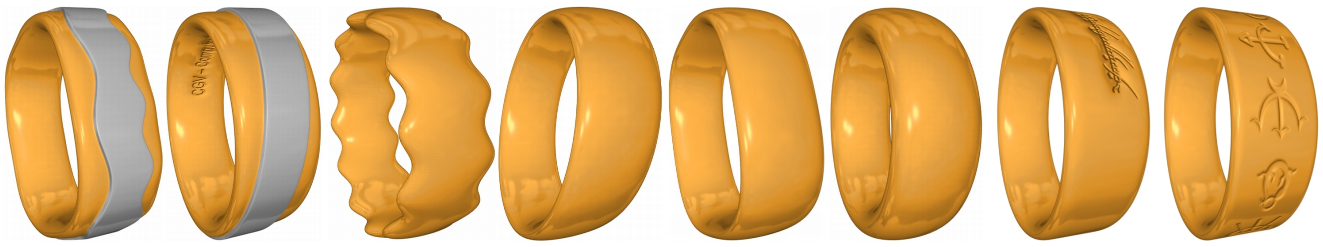

Figure 2. The presented generative description is able to produce a large variety of wedding rings. Features, like engravings, recesses, different materials, unusual forms and gems, can be created and customized.

The presented approach is used in a hardware-accelerated server-side rendering framework [59], which has been included in an online system called REx by Johann Kaiser. It offers an intuitive web interface for configuring and visualizing wedding rings.

This work demonstrates the efficiency of procedural shape modeling for the mass customization of wedding rings. The presented generative description is able to produce a large variety of wedding rings. Figure 2 shows a few results of the parametric toolkit.

3.5. Variance Analysis

The analysis and the visualization of the differences of similar objects is important in many research areas: scan alignment, nominal/actual value comparison and surface reconstruction, to name a few. In computer graphics, for example, differences of surfaces are used to validate reconstruction and fitting results of laser scanned surfaces. Scanned representations are used for documentation, as well as analysis of ancient objects, revealing the smallest changes and damage. Analyzing and documentation tasks are also important in the context of engineering and manufacturing to check the quality of productions.

CHRISTOPH SCHINKO *et al.* contribute a comparison of a reference/nominal surface with an actual, laser-scanned dataset [60]. The reference surface is a procedural model whose accuracy and systematics describe the semantic properties of an object, whereas the laser-scanned object is a real-world dataset without any additional semantic information. The first step of the process is to register a generative model (including its free parameters) to a laser scan. Then, the difference between the generative model and the laser scan is stored in a texture, which can be applied to all instances of the same shape family.

A generative model represents an ideal object rather than a real one. The combination of noisy 3D data with an ideal description enhances the range of potential applications. This bridge between both the generative and the explicit geometry description is very important: it combines the accuracy and systematics of generative models with the realism and the irregularity of real-world data, as pointed out by DAVID ARNOLD [61]. Once the procedural description is registered to a real-world artifact, we can use the fitted procedural model to modify a 3D shape. In this way, we can design both low-level details and high-level shape parameters at the same time.

3.6. Semantic Modeling

In the context of digital libraries, semantic metadata play an important role. They provide semantic information that is vital for digital library services: indexing, archival and retrieval. Depending on the field of application, metadata can be classified according to the following criteria [62]:

Data type: The data type of the object can be of any elementary data structure (e.g. polygons, non-uniform rational b-splines (NURBS), subdivision surfaces, *etc*.).

Scale of semantic information: This property describes whether metadata are added for the entire dataset or only for a sub-part of the object.

Type of semantic information: The type of metadata can be descriptive (describing the content), administrative (providing information regarding creation, storing, provenance, *etc*.) or structural (describing the hierarchical structure).

Type of creation: The creation of the semantic information for an object can be done manually (by a domain expert) or automatically (e.g., using a generative description).

Data organization: The two basic concepts of storing metadata are storing the information within the original object (e.g., Exchangeable Image File Format (Exif) data for images) or storing it separately (e.g., using a database).

Information comprehensiveness: The comprehensiveness of the semantic information can be declared varying from low to high in any gradation.

Many concepts for encoding semantic information can be applied to 3D data. Despite the large number of 3D data formats, only a few are standardized, non-proprietary and support semantic markup [63]:

Collada The XML-based Collada format is an ISO standard and allows storing metadata, like title, author, revision, *etc*., not only on a global scale, but also for parts of the scene [64]. This file format can be found in Google Warehouse where metadata are, for example, used for geo-referencing objects.

IGES Initial Graphics Exchange Specification (IGES), an American National Standards Institute (ANSI) standard since 1980, allows the definition of annotations, including dimensioning data, as well as labels and notes [65]. This file format is used as a vendor-neutral exchange format among CAD systems.

JT The Jupiter Tesselation (JT) file format has been an ISO standard since 2012 and is used for product visualization and data exchange in CAD systems [66]. Annotations in the form of attributes and properties, as well as filters are supported by this format. It is accompanied by the XML-based format for product lifecycle management (PLMXML) to represent product structure hierarchy.

PDF 3D PDF 3D is an ISO standard and allows one to store annotations separated from the 3D data, even allowing annotating of the annotations [67]. An advantage is that the viewer application is widely spread, and PDF documents are the quasi standard for textual documents.

STEP The standard for the exchange of product model data (STEP) has been an ISO standard since 1994 divided into different parts, data models and environments [68]. The current Application Protocol 242 supports product data and non-geometrical metadata.

X3D The X3D file format is an XML-based ISO standard for representing 3D computer graphics [69]. It supports a number of different metadata nodes, providing arrays of strongly typed data.

While a standard has advantages for accessibility, long-term archival and many other aspects, it does not solve inherent problems; *i.e.*, due to the persistent naming problem, a modification of the 3D model can break the integrity of the semantic information. Any change of the geometry can cause the referenced part of the model to no longer exist or be changed. Nevertheless, there are many examples for semantic modeling in various contexts [70–75].

4. Inverse Modeling

The full potential of the generative techniques is revealed when the inverse problem is solved; *i.e.*, what is the best generative description of one or several given instances of an object class? This problem can be interpreted in different ways. The simplest way is to create a generative model out of a given 3D object and to store it in a geometry definition file format. Obviously, this is not the desired result, as the generative model can only represent a single object, not a family of objects.

4.1. Parsing Shape Grammars

Shape grammars can be used to describe the design space of a class of buildings/facades. An interesting question in this context is: given a set of rules and measurements of a building, typically photographs or range scans, which application of rules yields the measurements? Here, the applied rules can also be seen as a parse tree of a given input.

The work of HAYKO RIEMENSCHNEIDER *et al.* [76] utilizes shape grammars to enhance the results of a machine learning classifier that is pre-trained to classify pixels of an orthophoto of a facade into categories, like windows, walls, doors and sky. The system applies techniques from formal language parsing to parse a two-dimensional split grammar consisting of horizontal and vertical splits, as well as repetition and symmetry operations. For the reduction of the search space, an irregular grid is derived from the classifications, and the parsing algorithm is applied to yield the most probable application of rules that yields a classification label per grid cell. These parse trees can easily be converted into procedural models.

FUZHANG WU *et al.* also address the problem of how to generate a meaningful split grammar explaining a given facade layout [77]. Given a segmented facade image, the system uses an approximate dynamic programming framework to evaluate if a grammar is a meaningful description. However, the work does not contribute to the problem of facade image segmentation.

4.2. Model Synthesis

PAUL MERELL and DINESH MANOCHA present an approach that, given an object (*i.e.*, a mesh) and constraints, derives a locally-similar object [78,79]. This method is related to texture synthesis. It computes a set of acceptable states, according to several types of constraints, and constructs parallel planes that correspond to the face orientations of the input model. The intersections of these planes yield possible vertex positions in the output model. Acceptable states are assigned to a vertex, while incompatible states are removed in its neighborhood. The system terminates if every vertex has been assigned a state.

4.3. Inverse Procedural Modeling of Trees

The method proposed by ONDREJ STAVA *et al.* estimates the parameters of a stochastic tree model, given polygonal input tree models [80]. This is done in such a way that the stochastic model produces trees similar to the input. The parameters are estimated using Markov chain Monte Carlo (MCMC) optimization techniques. A statistical growth model consisting of 24 geometrical and environmental parameters is used. The authors propose a similarity measure between the statistical model and a given input mesh that consists of three parts: shape distance, measuring the overall shape discrepancy, geometric distance, reflecting the statistics of the geometry of its branches, and structural distance, encoding the cost of transforming a graph representation of the statistical tree model into a graph representation of the input tree model. The MCMC method has also been applied by other methods to find parameters of a statistical generative model [81–83].

4.4. Parameter Fitting and Shape Recognition

TORSTEN ULLRICH and DIETER W. FELLNER presented an approach that uses generative modeling techniques to describe a class of objects and to identify objects in real-world data, e.g., laser scans [84]. A point cloud P and a generative model M are the input datasets of the algorithm. It answers the questions:

1. can the point cloud be described by the generative model, and if so,
2. what are the input parameters x_0, such that $M(x_0)$ is a good description of P.

A hierarchical optimization routine based on fuzzy geometry and a differentiating compiler are used. The complete generative model description $M(x_1, \ldots, x_k)$ (including all possibly called subroutines) is differentiated with respect to the input parameters. This differentiating compiler offers the possibility to use gradient-based optimization routines in the first place. Without partial derivatives, many numerical optimization routines cannot be used at all or in a limited way.

5. Architecture, Engineering and Design

5.1. Generative Architectural Design

The usage of generative modeling techniques in architecture is not limited to buildings of the past [85,86]. Over the last few decades, architects have used a new class of design tools that support generative design. Generative modeling software extends the design abilities of architects and may even help to reduce costs by harnessing computing power in new ways. Computers, of course, have long been used to capture and implement the design ideas of architects by means of CAD and 3D modeling. Generative design actually helps architects design by using computers to extend human abilities [87].

An impressive example is the Helix Bridge in Singapore (see Figure 3). The 280-m bridge is made up of three 65-m spans and two 45-m end spans. The major and minor helices, which spiral in opposite directions, have overall diameters of 10.8 m and 9.4 m, respectively. The outer helix is formed from six tubes, which are set equidistant from one another, whereas the inner helix consists of five tubes. The bridge design is the product of inseparable collaboration between architects (Cox Architecture and Architects 61) and civil engineers (Arup Consultant). For its 280-m length, the dual helix structure of the bridge utilizes five-times less steel than a conventional box girder bridge. This fact enabled the client to direct the structure to be constructed entirely of stainless steel for its longevity.

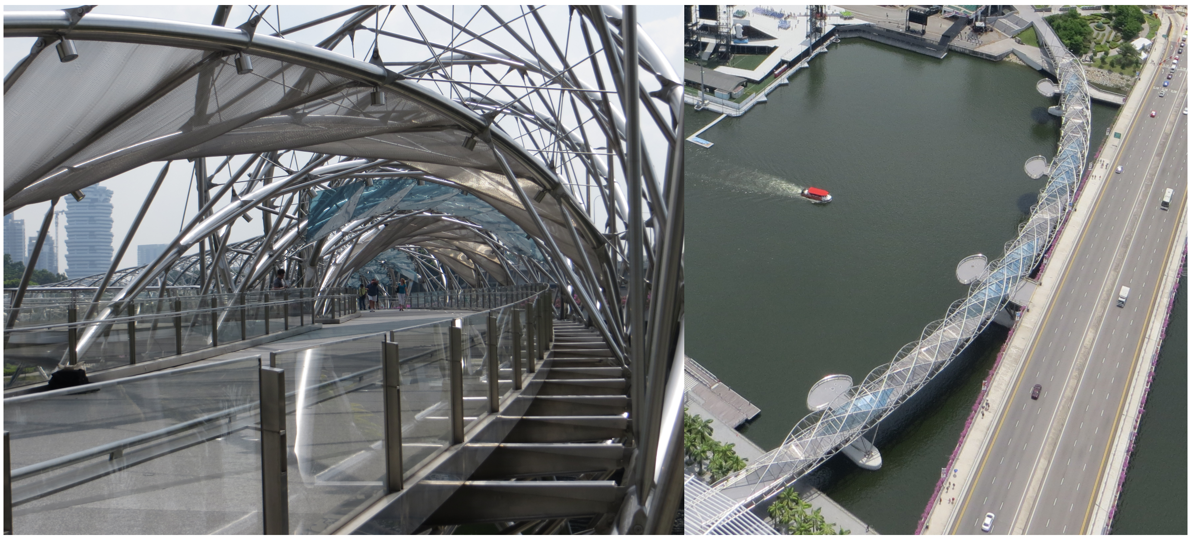

Figure 3. The Helix Bridge is a pedestrian bridge in the Marina Bay area in Singapore. Its generative design has been optimized numerically. Furthermore, the bridge was fully modeled in order to visualize its form and geometrical compatibility, as well as to visualize the pedestrian experience on the bridge.

Another example of generative, architectural design has been presented by TORSTEN ULLRICH *et al.* [88]. They interpret a generative script as a function, which is nested into an objective function. Thus, the script's parameters can be optimized according to an objective. They demonstrate this approach using architectural examples: each generative script creates a building with several free parameters. The objective function is an energy efficiency simulation that approximates a building's annual energy consumption. Consequently, the nested objective function reads a set of building parameters and returns the energy needs for the corresponding building. This nested function is passed to a minimization and optimization process. The outcome is the best building (within the family of buildings described by its script) concerning energy efficiency. The contribution is a new way of modeling: the generative approach separates design and engineering. The complete design is encoded in a script, and the script ensures that all parameter combinations (within a fixed range) generate a valid design. Then, the design can be optimized numerically.

The adjustment of architectural forms to local and specific conditions is a fundamental study. When discussing energy consumption and solar power harnessing in buildings, important aspects have to be taken into account, e.g., the relation between a building form and its energy behavior and the local weather conditions on an all-year basis. Several studies were published so far trying to answer these questions. "Form follows energy" has become an omnipresent dogma in architecture, but its realization is difficult. The manual analysis of the various relations between form, volume and energy consumption has to face many, not only numerical, problems.

The new approach by TORSTEN ULLRICH *et al.* [88] for architectural design is opening the door to new possibilities for the user. It relieves the user from additional, interdisciplinary burdens: the designer can concentrate on the design, while the civil engineer can focus on engineering aspects. This new approach based on procedural modeling can be used in many different fields of product design.

5.2. Engineering Design

The research area of computational design synthesis (CDS) is concerned with the automation of synthesis activities in design [89]. Computer systems are used to generate design candidates for a specific task. For example, the work of MICHAEL J. PUGLIESE *et al.* [90] investigated the possibility to capture brand specifics using a shape grammar. A more recent synthesis approach presented by FOREST FLAGER *et al.* is concerned with the sizing optimization of steel structures [91]. For an extensive overview of this topic, the authors refer to the work of AMARESH CHAKRABARTI *et al.* [92].

Each design process that involves repetitive tasks is perfectly suited for a generative approach. Engineering processes can be classified into repetitive and creative processes. In contrast to creative processes, repetitive ones consist of nearly identical tasks and are therefore independent of creative decisions. This is a precondition for modeling them in a system of rules, as is shown in this practical example [93]: Liebherr manufactures and sells an extensive range of products, including different kinds of cranes. Each crane has to be partially or fully engineered to the needs of the customer. Nevertheless, the design process of ascent assemblies is based on repetitive tasks that are described by a set of invariant rules. These rules have been modeled and stored by Liebherr. The integration into the existing CAD pipeline now allows a construction engineer to create ascent assemblies only by determining the defining parameters and filling out the corresponding input fields in a user interface. Using the procedural approach, the efforts of engineering ascent assemblies have been reduced to 10%.

5.3. Urban Modeling

In the context of urban modeling, procedural systems can be used to cover different levels of detail, as has been shown in a survey for urban reconstruction by PRZEMYSLAW MUSIALKSI *et al.* [94]. On a coarse scale, the procedural paradigm is applicable to the generation of terrain, e.g., using methods based on hydrology, as presented by JEAN-DAVID GÉNEVAUX [95], or for the inexpensive reconstruction of the landscape surrounding a road, as presented by CARLOS ANDÚJAR *et al.* [96].

Such systems have also been used for the generation of roads. ERIC GALIN *et al.* present an algorithm that generates roads on terrain [97]. JAN BENES *et al.* present a model for growing procedural road networks in and close to cities [98]. The work of MARKUS LIPP *et al.* is concerned with interactive modeling of entire city layouts [99] using procedural methods. An overview of modeling the appearance and behavior of urban spaces is given by CARLOS A. VANEGAS *et al.* [100].

Within the scale of a building, PAUL MERELL *et al.* present a method for automatic creation of residential building layouts; FAN BAO *et al.* formulate a constrained optimization to characterize good building layouts and a method to let a user explore the space of good building layouts [101]. Procedural systems are also used in the context of facade modeling: PRZEMYSLAW MUSIALSKI *et al.* present an interactive framework for modeling building facades from images [102]. FAN BAO *et al.* show a technique to create procedural facade variations from a single layout [103]. MICHAEL SCHWARZ *et al.* present an approach for designing exterior lighting for buildings with complex constraints [104]. When it comes down to the interior of a building, PAUL MERELL *et al.* present an automatic method for furniture placement following interior design guidelines [105].

5.4. Building Information Modeling

Procedural modeling can also be helpful in the context of building information modeling (BIM), the new paradigm of today's building industry [106]. The American National Building Information Model Standard (NBIMS-US) project committee defines BIM as "a digital representation of physical and functional characteristics of a facility. A BIM is a shared knowledge resource for information about a facility forming a reliable basis for decisions during its life cycle; defined as existing from earliest conception to demolition" [107]. Other definitions are summarized in a literature review by ABBASNEJAD and MOUD, who conclude that a generally accepted comprehensive definition of BIM has not been established yet, and different stakeholders (architects, builders, owners, *etc.*) have mixed expectations towards BIM [108]. In contrast to established computer-aided design (CAD), a building information model does not just store the geometry of a building, but includes semantic data about the functions of the buildings and its elements. Furthermore, BIM is intended to be used throughout the building's life cycle, containing information for planning, design, construction, operation and maintenance. That is, a model is not only used by architects, contractors and suppliers, but by all kinds other users, e.g., government agencies, owners, real estate agents, facility managers, *etc.*

EASTMAN *et al.* help to understand BIM by describing examples that are not BIM technology. As already mentioned, models without object attributes, but only 3D data, are not considered BIM. Furthermore, models composed of multiple 2D drawings that have to be combined or models that do not automatically reflect changes made in one view in other views are not building information models.

Moreover, EASTMAN *et al.* consider parametric object capabilities as essential for BIM. Parametric objects in BIM can include rules to automatically modify associated objects (e.g., a wall is changed when a door is placed in it) and for ensuring feasibility (e.g., regarding size and manufacturability) [109]. Such intelligent objects are similar to the idea of generative modeling, where a 3D object is described by the operations necessary to generate the object, rather than the result of these operations [110,111].

One use-case is documenting a building "as-built BIM" [112], to aid, amongst others, in the application scenarios of restoration, documentation and maintenance. Such a model is built from measured data, which is typically acquired by terrestrial laser scanning (TLS) or image-based approaches (photogrammetry or structure from motion techniques), which yields point positions in 3D. From these point clouds, a mesh can be created using 3D surface reconstruction techniques, e.g., Poisson surface reconstruction [113]. Furthermore, the surface appearance has to be acquired [114]. In the case of building interiors, the generation of orthographic images can be used for surface color representation and the retrieval of additional semantic information [115]. Such semantic relationships have to be acquired and represented within the model [116]; see also the foregoing section about semantic enrichment. A recent example of the usage of parametric elements for the reconstruction and documentation of complex architecture is the case of a reactor building, as shown by JEAN-FRANÇOIS HULLO *et al.* [117].

For historic building information modeling (HBIM), procedural methods have been used to aid the reconstruction and documentation process. CONOR DORE *et al.* applied a shape grammar approach to model classical building facades for HBIM [118] and reconstructed the Four Courts, a historic classical building in Dublin City [119] using rule-based modeling in ArchiCAD. Another recent example of creating a HBIM model with rich semantics from terrestrial laser scanning data has been shown by RAMONA QUATTRINI *et al.* in the case study of the Church of Santa Maria at Portonovo [120] using Autodesk Revit.

In the context of functional building information modeling (FBIM), generative techniques can be used to semantically filter a CAD dataset of a building. A major future challenge in the building industry is to reduce primary energy use of buildings. Hence, energy performance simulation becomes an increasingly important topic. Accurate, yet efficient simulation depends on simple building models. Most of the required data can be found in BIMs. However, typical BIM data contains a lot of irrelevant data, in particular geometric representations, which are too detailed for energy performance simulation. Using generative modeling techniques, DANIEL LADENHAUF *et al.* [121,122] show an approach of geometry simplification subject to semantic and functional groups. These simplified models are sufficiently accurate for energy calculations and small enough so that they do not flood simulation software with unnecessary details. As these semantically-filtered models are generated automatically, they simplify the design process significantly and offer an energy calculation, even at early design stages.

6. Archeology and Cultural Heritage

The increasing number of (3D) documents makes digital library services become more and more important. A digital library provides markup, indexing and retrieval services based on available metadata. In a simple case, metadata are of the Dublin Core type: title, creator/author, time of creation, *etc* [123]. This is insufficient for large collections of 3D objects, because of their versatility and rich structure.

6.1. Semantic Enrichment

Scanned models are used in raw data collections, for documentation archival, virtual reconstruction, historical data analysis and for high-quality visualization for dissemination purposes [124]. Navigation and browsing through the geometric models should be possible on the semantic level; this requires higher level semantic information. The need for semantic information becomes immediately clear in the context of electronic data exchange, storage and retrieval [125,126]. The problem of 3D semantic enrichment is closely related to the shape description problem [127]:

How to describe a shape and its structure
on a higher, more abstract level?

The traditional way of classifying objects, pursued both in mathematics and, in a less formal manner, in dictionaries, is to define a class of objects by listing their distinctive properties. This approach is hardly realizable, because of the fact that definitions cannot be self-contained. They depend on other definitions, which leads to circular dependencies that cannot be resolved automatically by strict reasoning, but rely on intuitive understanding at some point.

An alternative, non-recursive approach for describing shape uses examples. Each entry in a picture dictionary is illustrated with a photo or a drawing. This approach is widely used, for example, in biology for plant taxonomy. It avoids listing an exhaustive list of required properties for each entry. However, it requires some notion of similarity, simply because the decision of whether object x belongs to class A or B requires measuring the closeness of x to the exemplars $a \in A$, respectively $b \in B$. This decision can be reached by a classifier using statistics and machine learning [128,129]. A survey on content-based 3D object retrieval is provided by BENJAMIN BUSTOS *et al.* [130]. Statistical approaches clearly have their strength in discriminating object classes. However, feature-based object detection, e.g., of rectangular shapes, does not yield object parameters: the width and height of a detected rectangle must typically be computed separately.

To describe a shape and its construction process, its inner structure must be known. Structural decomposition is well in line with human perception. In general, shapes are recognized and coded mentally in terms of relevant parts and their spatial configuration or structure [131]. One idea to operationalize this concept was proposed, among others, by MASAKI HILAGA *et al.*, who introduce the Multiresolution Reeb Graph, to represent the skeletal and topological structure of a 3D shape at various levels of resolution [132]. Structure recognition is a very active branch in the field of geometry processing. The detection of shape regularities [133], self-similarities [134] and symmetries [135,136] is important to understand a 3D shape. To summarize, structural decomposition proceeds by postulating that a certain type of general regularity or structure exists in a class of shapes. This approach clearly comes to its limits when very specific structures are to be detected, *i.e.*, complicated constructions with many parameter interdependencies.

A possibility to describe a shape is realized by the generative modeling paradigm [29,137]. The key idea is to encode a shape with a sequence of shape-generating operations and not just with a list of low-level geometric primitives. In its practical consequence, every shape needs to be represented by a program, *i.e.*, encoded in some form of programming language, shape grammar [31], modeling language [33] or modeling script [138].

The implementation of the "definition by algorithm" approach is based on a scripting language [84]: Each class of objects is represented by one algorithm M. Furthermore, each described object is a set of high-level parameters x, which reproduces the object, if an interpreter evaluates $M(x)$. As this kind of modeling resembles programming rather than "designing", it is obvious to use software engineering techniques, such as versioning and annotations. In this way, model M may contain a human-readable description of the object class it represents.

In contrast to other related techniques using fitting algorithms, such as "Creating Generative Models from Range Images" by RAVI RAMAMOORTHI and JAMES ARVO, the approach by TORSTEN ULLRICH can classify data semantically [139,140]. Although RAVI RAMAMOORTHI and JAMES ARVO also use generative models to fit point clouds, they modify the generative description during the fitting process. As a consequence, the optimization can be performed locally with a computational complexity that is significantly reduced. However, starting with the same generative description to fit a spoon as well as a banana does not allow one to generate or preserve semantic data.

An example illustrates this process. The generative model to describe a vase takes 13 parameters: $R(r_x, r_y, r_z)$ is the base reference point of the vase in 3D and $T(t_x, t_y, t_z)$ is its top-most point. The points R and T define an axis of rotational symmetry. The remaining seven parameters define the distances $d_0, \dots, d_6$ of equally-distributed Bézier vertices to the axis of rotation (see Figure 4). The resulting 2D Bézier curve defines a surface of revolution: the generative vase.

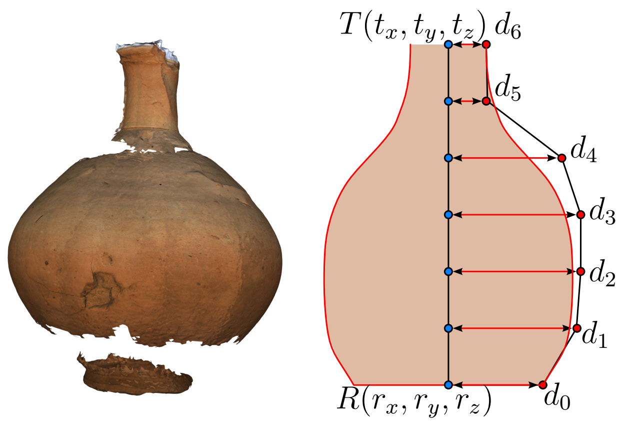

Figure 4. The vase on the left-hand side is a digitized artifact of the "Museum Eggenberg" collection. It consists of 364,774 vertices and 727,898 triangles. The example of a procedural shape on the right-hand side takes two points R and T in 3D and distance values, which define the control vertices of a Bézier curve.

6.2. Cultural Heritage

The huge volume of cultural objects is a challenge, even for the most ambitious plans for digitization campaigns [141]. The fact that probably 90 percent of museum collections are in storage and not accessible to the public is almost demanding for digitization and public accessibility. However, the digitization alone is only part of a larger process that begins at a field excavation and does not end with the presentation in museum exhibitions. Secondary exploitation, database access and sustainable long-time archiving of digitized artifacts is also part of the process [142]. A very important aspect is the choice of the 3D format used during this process [143,144]. However, the availability of large quantities of cultural heritage data will enable new methods for analysis and new applications [145].

The presented modeling system by CHRISTOPH SCHINKO *et al.* is restricted to techniques to meet sustainability conditions. By using JavaScript, the inhibition threshold to use a programming language is reduced, resulting in a beginner-friendly tool with a high degree of usability [111]. RENÉ BERNDT *et al.* present a system for the production of three-dimensional interactive illustrations in the domain of medieval castles [146]. A special focus is on creating generic modeling tools that increase the usability with a unified 3D user interface.

One of the advantages of procedural modeling techniques is the included expert knowledge within an object description [84]. Classification schemes used in architecture, archeology and other domains can be mapped to procedures [147]. When a procedural object description is available, only the type and instantiation parameters have to be identified in order to create an object [148] (see Figure 5). It is then also possible to use the fitted procedural model to modify existing 3D shapes [149].

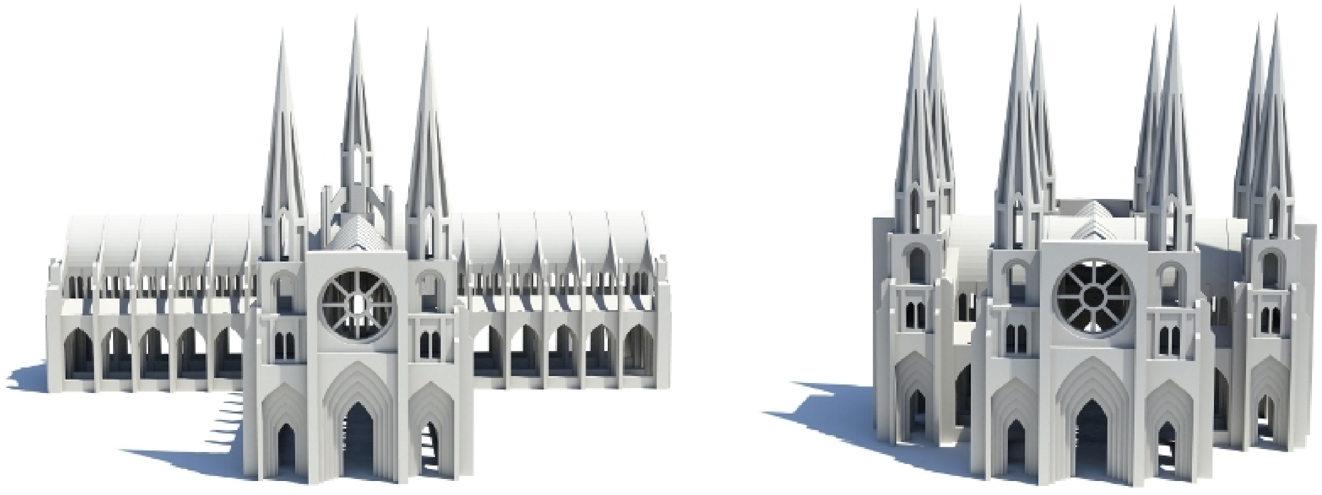
Figure 5. Gothic architecture is defined by strict rules with its characteristics. The generative description of Gothic cathedrals encodes these building blocks and the rules on how to combine them. These building blocks have been created by MICHAEL CURRY, http://www.thingiverse.com/thing:2030.

Another use-case is the creation of several building hypotheses in the context of historic analysis, as shown by ERICA CALOGERO *et al.* [150] in a case study that investigated different hypothesis for parts of the Louvre. Furthermore, MARIE SALDAÑA *et al.* carried out a similar approach for parts of the city of Rome [151]. Both works were carried out using the Esri CityEngine.

7. Open Research Questions

According to DIETER W. FELLNER and SVEN HAVEMANN, several research challenges have to be met: from the classification of shape representations via generic, stable and detailed 3D markup to 3D query operations [125,152–154].

A particularly important problem occurs in the context of internal structure organization and interfaces. Within a composition of modeling functions, where each function is attached via its parameters to topological entities defined in previous states of the model, referenced entities must be named in a persistent way in order to be able to reevaluate the model in a consistent manner. In particular, when a reevaluation leads to topological modifications, references between entities used during the design process are frequently reevaluated in an erroneous way, giving results different from those expected. This problem is known as the "persistent naming problem" [155].

Acknowledgments

The authors gratefully acknowledge the support of the European Commission's Seventh Framework Programme (FP7/2007-2013) under Grant Agreement No. 600908 (DURAARK: Durable Architectural Knowledge), founded by the program, ICT-2011-4.3-Digital Preservation, as well as the generous support of the Austrian Research Promotion Agency, the Forschungsförderungsgesellschaft (FFG), for the research project GINGER (Graphical Energy-Efficiency-Visualization in Architecture), Grant Number 840190, and the research project ProFitS (Procedural Fitting Service), Grant Number 840254.

Furthermore, the authors would like to thank Daniel Ladenhauf for his research on building information modeling, as well as Wolfgang Thaller and Volker Settgast for permitting use of their images in Figure 1 and Figure 3, respectively.

Author Contributions

This survey has been conducted by the stated authors, with Ulrich Krispel and Christoph Schinko contributing mainly to sections 2–5, and Torsten Ullrich mainly contributing to sections 1, 6–7.

Conflicts of Interest

The authors declare no conflict of interest.

References

1. Heiberg, J. *Euclid's Elements of Geometry*; Fitzpatrick Richard: Austin, TX, USA, 2007.
2. Hartshorne, R. Teaching geometry according to Euclid. *Not. AMS* **2000**, *47*, 460–465.
3. Maxfield, J.E.; Coolidge, J.L. *A History of Geometrical Methods*, 1st ed.; Dover Publications: New York, NY, USA, 2003.
4. Scriba, C.J.; Schreiber, P. *5000 Jahre Geometrie: Geschichte, Kulturen, Menschen (english: 5000 years of geometry: history, cultures, men)*; Springer: Berlin, Germany, 2004.
5. Thaller, W.; Krispel, U.; Zmugg, R.; Havemann, S.; Fellner, D.W. A Graph-Based Language for Direct Manipulation of Procedural Models. *Int. J. Adv. Softw* **2013**, *6*, 225–236.

6. Martin, G.E. *Geometric Constructions*; Springer: Berlin, Germany, 1998.
7. Mitchell, W.J. *The Logic of Architecture: Design, Computation, and Cognition*; MIT Press: Cambridge, UK, 1990.
8. Havemann, S.; Fellner, D.W. Generative parametric design of gothic window tracery. *IEEE Proc. Shape Model. Appl.* **2004**, doi:10.1109/SMI.2004.1314525.
9. Berndt, R.; Fellner, D.W.; Havemann, S. Generative 3D models: A key to more information within less bandwidth at higher quality. In Proceedings of the 10th International Conference on 3D Web Technology, Gwynedd, UK, 29 March–1 April 2005.
10. Prusinkiewicz, P.; Lindenmayer, A. *The Algorithmic Beauty of Plants*; Springer: Berlin, Germany, 1990.
11. Deussen, O.; Lintermann, B. *Digital Design of Nature: Computer Generated Plants and Organics*; Springer: Berlin, Germany, 2005.
12. Mandelbrot, B.B. *The Fractal Geometry of Nature*; W. H. Freeman and Co.: New York, NY, USA, 1982.
13. Tobler, R.F.; Maierhofer, S.; Wilkie, A. A multiresolution mesh generation approach for procedural definition of complex geometry. *Proc. Shape Model. Int.* **2002**, *6*, 35–44.
14. Tobler, R.F.; Maierhofer, S.; Wilkie, A. Mesh-based parametrized L-systems and generalized subdivision for generating complex geometry. *Int. J. Shape Model.* **2002**, *8*, 173–191.
15. Lipp, M.; Wonka, P.; Wimmer, M. Parallel generation of multiple L-systems. *Comput. Graph.* **2010**, *34*, 585–593.
16. Brutzman, D. The virtual reality modeling language and Java. *Commun. ACM* **1998**, *41*, 57–64.
17. Behr, J.; Dähne, P.; Jung, Y.; Webel, S. Beyond the web browser–X3D and immersive VR. *IEEE Virtual Real. Tutor. Workshop Proc.* **2007**, *28*, 5–9.
18. Breuel, F.; Bernd, R.; Ullrich, T.; Eggeling, E.; Fellner, D.W. Mate in 3D – publishing interactive content in PDF3D. In *Digital Publishing and Mobile Technologies*, Proceedings of the 15th International Conference on Electronic Publishing, Äřstanbul, Turke, 22–24 June 2011.
19. Di Benedetto, M.; Ponchio, F.; Ganovelli, F.; Scopigno, R. SpiderGL: A JavaScript 3D graphics library for next-generation WWW. In Proceedings of the 15th International Conference on Web 3D Technology, Los Angeles, CA, USA, 24–25 July 2010.
20. Schinko, C.; Strobl, M.; Ullrich, T.; Fellner, D.W. Scripting technology for generative modeling. *Int. J. Adv. Softw.* **2011**, *4*, 308–326.
21. Ousterhout, J.K. Scripting: Higher level pogramming for the 21st century. *IEEE Comput. Mag.* **1998**, *31*, 23–30.
22. OpenGL Architecture, R.B. *OpenGL Reference Manual*; Addison-Wesley: Boston, MA, USA, 1993.
23. NVidia. CUDA C Programming Guide. Available online: https://docs.nvidia.com/cuda/cuda-c-programming-guide/ (accessed on 29 June 2015).
24. Reiners, D.; Voss, G.; Behr, J. OpenSG: Basic concepts. *OpenSG Symp.* **2002**, *1*, 1–7.
25. Voß, G.; Behr, J.; Reiners, D.; Roth, M. A multi-thread safe foundation for scene graphs and its extension to clusters. *EGPGV* **2002**, *4*, 33–37.

26. Eckel, B. *Thinking in C++: Introduction to Standard C++, Practical Programming*; Prentice Hall: New Jersey, NJ, USA, 2003.
27. Parr, T. *Language Implementation Patterns: Create Your Own Domain-Specific and General Programming Languages*; Pragmatic Bookshelf: North Carolina, NC, USA, 2010.
28. Chomsky, N. Three models for the description of language. *IRE Trans. Inf. Theory* **1956**, *2*, 113–124.
29. Özkar, M.; Kotsopoulos, S. Introduction to shape grammars. In Proceedings of the International Conference on Computer Graphics and Interactive Techniques, Los Angeles, CA, USA, 11–15 August 2008.
30. Krecklau, L.; Pavic, D.; Kobbelt, L. Generalized use of non-terminal symbols for procedural modeling. *Comput. Graph. Forum* **2010**, *29*, 2291–2303.
31. Müller, P.; Wonka, P.; Haegler, S.; Andreas, U.; van Gool, L. Procedural modeling of buildings. *ACM Trans. Graph.* **2006**, *25*, 614–623.
32. Snyder, J.M.; Kajiya, J.T. Generative modeling: A symbolic system for geometric modeling. *ACM SIGGRAPH Comput. Graph.* **1992**, *26*, 369–378.
33. Havemann, S. Generative Mesh Modeling. Ph.D. Thesis, Technische Universit, Braunschweig, Germany, 2005.
34. Krecklau, L.; Kobbelt, L. Procedural modeling of interconnected structures. *Comput. Graph. Forum* **2011**, *30*, 335–344.
35. Lipp, M.; Wonka, P.; Wimmer, M. Interactive visual editing of grammars for procedural architecture. *ACM Trans. Graph.* **2008**, *27*, 1–10.
36. Thaller, W.; Krispel, U.; Havemann, S.; Fellner, D. Implicit nested repetition in dataflow for procedural modeling. In Proceedings of the International Conference on Computational Logics, Algebras, Programming, Tools, and Benchmarking (Computation Tools), Nice, France, 22–27 July 2012; pp. 45–50.
37. Schinko, C.; Ullrich, T.; Fellner, D.W. Minimally invasive interpreter construction—How to reuse a compiler to build an interpreter. In Proceedings of the International Conference on Computational Logics, Algebras, Programming, Tools, and Benchmarking (Computation Tools), Nice, France, 22–27 July 2012; pp. 38–44.
38. Parish, Y.; Müller, P. Procedural modeling of cities. In Proceedings of the 28th annual conference on Computer graphics and interactive techniques, Los Angeles, CA, USA, 1 August 2001; pp. 301–308.
39. Ullrich, T.; Krispel, U.; Fellner, D.W. Compilation of procedural models. In Proceeding of the 13th International Conference on 3D Web Technology, Los Angeles, CA, USA, 9–10 August 2008; pp. 75–81.
40. Reas, C.; Fry, B.; Maeda, J. *Processing: A Programming Handbook for Visual Designers and Artists*; The MIT Press: Cambridge, MA, USA, 2007.
41. Patow, G. User-friendly graph editing for procedural modeling of buildings. *IEEE Comput. Graph. Appl.* **2012**, *32*, 66–75.
42. Stiny, G.; Gips, J. Shape grammars and the generative specification of painting and sculpture. *IFIP Congr.* **1971**, *2*, 125–135.

43. Wonka, P.; Wimmer, M.; Sillion, F.; Ribarsky, W. Instant architecture. *IACM Trans. Graph.* **2003**, *22*, 669 – 677.
44. Marvie, J.E.; Buron, C.; Gautron, P.; Hirtzlin, P.; Sourimant, G. GPU Shape grammars. *Comput. Graph. Forum* **2012**, *31*, 2087–2095.
45. Krecklau, L.; Born, J.; Kobbelt, L. View-dependent realtime rendering of procedural facades with high geometric detail. *Comput. Graph. Forum* **2013**, *32*, 479–488.
46. Kuang, Z.; Chan, B.; Yu, Y.; Wang, W. A compact random-access representation for urban modeling and rendering. *ACM Trans. Graph.* **2013**, doi:10.1145/2508363.2508424.
47. Steinberger, M.; Kenzel, M.; Kainz, B.; Müller, J.; Peter, W.; Schmalstieg, D. Parallel generation of architecture on the GPU. *Comput. Graph. Forum* **2014**, *33*, 73–82.
48. Steinberger, M.; Kenzel, M.; Kainz, B.; Wonka, P.; Schmalstieg, D. On-the-fly generation and rendering of infinite cities on the GPU. *Comput. Graph. Forum* **2014**, *33*, 105–114.
49. The CGAL Project CGAL User and Reference Manual. Available online: http://doc.cgal.org/latest/Manual/ (accessed on 29 June 2015).
50. Pasko, A.; Adzhiev, V. Function-based shape modeling: Mathematical framework and specialized language. *Lect. Notes Comput. Sci.* **2004**, *2930*, 132–160.
51. Kelly, T.; Wonka, P. Interactive architectural modeling with procedural extrusions. *ACM Trans. Graph.* **2011**, doi:10.1145/1944846.1944854.
52. Aurenhammer, F. Weighted skeletons and fixed-share decomposition. *Comput. Geom.* **2008**, *40*, 93 – 101.
53. Watson, B.; Wonka, P. Procedural methods for urban modeling. *IEEE Comput. Graph. Appl.* **2008**, *28*, 16–17.
54. Whiting, E.; Ochsendorf, J.; Durand, F. Procedural modeling of structurally-sound masonry buildings. *ACM Trans. Graph.* **2009**, doi:10.1145/1618452.1618458.
55. Edelsbrunner, J.; Krispel, U.; Havemann, S.; Sourin, A.; Fellner, D.W. Constructive roof geometry. In Proceedings of the 2014 International Conference on Cyberworlds, Santander, Spain, 6–8 October 2014.
56. Sederberg, T.W.; Parry, S.R. Free-form deformation of solid geometric models. *ACM SIGGRAPH Comput. Graph.* **1986**, *13*, 151–160.
57. Zmugg, R.; Thaller, W.; Krispel, U.; Edelsbrunner, J.; Havemann, S.; Fellner, D.W. Procedural architecture using deformation-aware split grammars. *Visual Comput.* **2013**, *12*, 1–11.
58. Berndt, R.; Schinko, C.; Krispel, U.; Settgast, V.; Havemann, S.; Eggeling, E.; Fellner, D.W. Ring's anatomy—parametric design of wedding rings. *Content* **2012**, *4*, 72–78.
59. Schinko, C.; Berndt, R.; Eggeling, E.; Fellner, D. A scalable rendering framework for generative 3D content. In Proceedings of the 19th International ACM Conference on 3D Web Technologies, Vancouver, BC, Canada, 8–10 August 2014.
60. Schinko, C.; Ullrich, T.; Schiffer, T.; Fellner, D.W. Variance analysis and comparison in computer-aided design. In Proceedings of the International Workshop on 3D Virtual Reconstruction and Visualization of Complex Architectures, rento, Italy, 2–4 March 2011.
61. Arnold, D. Procedural methods for 3D reconstruction. *Rec. Model. Vis. Cult. Herit.* **2006**, *1*, 355–359.

62. Ullrich, T.; Settgast, V.; Berndt, R. Semantic enrichment for 3D documents: Techniques and open problems. In *the Networked World: Transforming the Nature of Communication*, Proceedings of the International Conference on Electronic Publishing, Helsinki, Finland, 16–18 June 2010.
63. Settgast, V. Processing Semantically Enriched Content for Interactive 3D Visualizations. Ph.D. Thesis, Technische Universität, Graz, Austria, 2013.
64. International Organization for Standardization (ISO) / Publicly Available Specification (PAS) 17506:2012 (Industrial Automation Systems and Integration–COLLADA Ddigital Asset Schema Specification for 3D Visualization of Industrial Data). Available online: http://www.iso.org/iso/catalogue_detail.htm?csnumber=59902 (accessed on 29 June 2015).
65. U.S. Product Data Association (US PRO), Formerly ANS US PRO/IPO-100-1996 (Initial Graphics Exchange Specification IGES 5.3). Available online: http://webstore.ansi.org/RecordDetail.aspx?sku=SAE+J+1881-2001+(SAE+J1881-2001) (accessed on 29 June 2015).
66. International Organization for Standardization (ISO) 14306:2012 (Industrial Automation Systems and Integratio–JT File Format Specification for 3D Visualization). Available online: http://www.iso.org/iso/catalogue_detail.htm?csnumber=60572 (accessed on 29 June 2015).
67. International Organization for Standardization (ISO) 32000-1:2008 (Document Management–Portable Document Format–Part 1: PDF 1.7). Available online: http://www.iso.org/iso/catalogue_detail.htm?csnumber=51502 (accessed on 29 June 2015).
68. International Organization for Standardization (ISO) 10303-1:1994 (Industrial Automation Systems and Integration–Product Data Representation and Exchange–Part 1: Overview and Fundamental Principles). Available online: http://www.iso.org/iso/catalogue_detail?csnumber=20579 (accessed on 29 June 2015).
69. International Organization for Standardization (ISO) / International Electrotechnical Commission (IEC) 19775-1:2013 (Information technology–Computer Graphics, Image Processing and Environmental Data Representation–Extensible 3D (X3D)–Part 1: Architecture and Base Components). Available online: http://www.iso.org/iso/catalogue_detail?csnumber=60760 (accessed on 29 June 2015).
70. Boulch, A.; Houllier, S.; Marlet, R.; Tournaire, O. Semantizing complex 3D scenes using constrained attribute grammars. *Proc. Eur. Symp. Geom. Proc.* **2013**, *32*, 33–42.
71. Haegeler, S.; Müller, P.; Van Gool, L. Procedural modeling for digital cultural heritage. *J. Image Video Process.* **2009**, *9*, 1–11.
72. Mendez, E.; Schall, G.; Havemann, S.; Fellner, D.W.; Schmalstieg, D.; Junghanns, S. Generating semantic 3D models of underground infrastructure. *IEEE Comput. Graph. Appl.* **2008**, *28*, 48–57.
73. Thaller, W.; Zmugg, R.; Krispel, U.; Posch, M.; Havemann, S.; Fellner Dieter, W. Creating procedural windowbuilding blocks using the generative fact labeling method. *Proc. ISPRS Int. Workshop 3D-ARCH* **2013**, *5*, 235–242.
74. Van Gool, L.; Martinovic, A.; Mathias, M. Towards semantic city models. *Proc. Photogramm. Week* **2013**, *1*, 217–232.

75. Yong, L.; Mingmin, Z.; Yunliang, J.; Haiying, Z. Improving procedural modeling with semantics in digital architectural heritage. *Comput. Graph.* **2012**, *36*, 178–184.
76. Riemenschneider, H.; Krispel, U.; Thaller, W.; Donoser, M.; Havemann, S.; Fellner, D.W.; Bischof, H. Irregular lattices for complex shape grammar facade parsing. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Providence, RI, USA, 16–21 June 2012.
77. Wu, F.; Yan, D.M.; Dong, W.; Zhang, X.; Wonka, P. Inverse procedural modeling of facade layouts. *ACM Trans. Graph.* **2014**, doi:10.1145/2601097.2601162.
78. Merrell, P.; Manocha, D. Continuous model synthesis. *ACM Trans. Graph.* **2008**, doi:10.1145/1409060.1409111.
79. Merrell, P.; Manocha, D. Model Synthesis: A general procedural modeling algorithm. *IEEE Trans. Vis. Comput. Graph.* **2010**, *17*, 715–728.
80. Stava, O.; Pirk, S.; Kratt, J.; Chen, B.; Měch, R.; Deussen, O.; Benes, B. Inverse procedural modelling of trees. *Comput. Graph. Forum*, **2014**, *33*, 118–131.
81. Talton, J.O.; Lou, Y.; Lesser, S.; Duke, J.; Mech, R.; Koltun, V. Metropolis procedural modeling. *ACM Trans. Graph.* **2011**, doi:10.1145/1944846.1944851.
82. Vanegas, C.A.; Garcia-Dorado, I.; Aliaga, D.G.; Benes, B.; Waddell, P. Inverse design of urban procedural models. *ACM Trans. Graph.* **2012**, doi:10.1145/2366145.2366187.
83. Yu, L.F.; Yeung, S.K.; Tang, C.K.; Terzopoulos, D.; Chan, T.F.; Osher, S. Make it home: Automatic optimization of furniture arrangement. *ACM Trans. Graph.* **2011**, doi:10.1145/1964921.1964981.
84. Ullrich, T.; Fellner, D.W. Generative object definition and semantic recognition. In Proceedings of the Eurographics Workshop on 3D Object Retrieval, Llandudno, UK, 10 April 2011.
85. Müller, P.; Vereenooghe, T.; Ulmer, A.; van Gool, L. Automatic reconstruction of Roman housing architecture. *Rec. Model. Vis. Cult. Heritage* **2006**, *1*, 287–298.
86. Müller, P.; Vereenooghe, T.; Wonka, P.; Paap, I.; Van Gool, L. Procedural 3D reconstruction of Puuc buildings in Xkipche. *Proc. Eur. Symp. Virtual Real. Archaeol. Cult. Heritage (VAST)* **2006**, *1*, 139–146.
87. Hohmann, B.; Krispel, U.; Havemann, S.; Fellner, D.W. Cityfit: High-quality urban reconstructions by fitting shape grammars to images and derived textured point clouds. *Proc. ISPRS Int. Workshop 3D-ARCH* **2009**, *3*, 61–68.
88. Ullrich, T.; Silva, N.; Eggeling, E.; Fellner, D.W. Generative Modeling and Numerical Optimization for Energy Efficient Buildings. In Proceedings of the IECON 2013-39th Annual Conference of the IEEE Industrial Electronics Society,Vienna, Austria, 10–13 November 2013.
89. Campbell, M.I.; Shea, K. Guest editorial: Computational design synthesis. *AI EDAM* **2014**, *28*, 207–208.
90. Pugliese, M.; Cagan, J. Capturing a rebel: Modeling the Harley-Davidson brand through a motorcycle shape grammar. *Res. Eng. Design* **2002**, *13*, 139–156.
91. Flager, F.; Soremekun, G.; Adya, A.; Shea, K.; Haymaker, J.; Fischer, M. Fully Constrained Design: A general and scalable method for discrete member sizing optimization of steel truss structures. *Comput. Struct.* **2014**, *140*, 55–65.

92. Chakrabarti, A.; Shea, K.; Stone, R.; Cagan, J.; Campbell, M.; Vargas-Hernandez, N.; Wood, K.L. Computer-based design synthesis research: An overview. *J. Comput. Inf. Sci. Eng.* **2011**, doi:10.1115/1.3593409.
93. Frank, G.; Hillbrand, C. Automatic support of standardization processes in design models. In Proceedings of the 2012 IEEE International Conference on Intelligent Engineering Systems (INES), Lisbon, Portugal, 13–15 June 2012.
94. Musialski, P.; Wonka, P.; Aliaga, D.G.; Wimmer, M.; van Gool, L.; Purgathofer, W. A survey of urban reconstruction. *Comput. Graph. Forum* **2012**, *31*, 1–28.
95. Génevaux, J.D.; Galin, E.; Guérin, E.; Peytavie, A.; Beneš, B. Terrain generation using procedural models based on hydrology. *ACM Trans. Graph.* **2013**, doi:10.1145/2461912.2461996.
96. Andújar, C.; Chica, A.; Vico, M.A.; Moya, S.; Brunet, P. Inexpensive reconstruction and rendering of realistic roadside landscapes. *Comput. Graph. Forum* **2014**, *33*, 101–117.
97. Galin, E.; Peytavie, A.; Marechal, N.; Guerin, E. Procedural generation of roads. *Comput. Graph. Forum* **2010**, *29*, 429–438.
98. Benes, J.; Wilkie, A.; Krivanek, J. Procedural modelling of urban road networks. *Comput. Graph. Forum* **2014**, *33*, 132–142.
99. Lipp, M.; Scherzer, D.; Wonka, P.; Wimmer, M. Interactive modeling of city layouts using layers of procedural content. *Comput. Graph. Forum* **2011**, *30*, 345–354.
100. Vanegas, C.A.; Aliaga, D.G.; Wonka, P.; Müller, P.; Waddell, P.; Watson, B. Modelling the appearance and behaviour of urban spaces. *Comput. Graph. Forum* **2010**, *29*, 25–42.
101. Bao, F.; Yan, D.M.; Mitra, N.J.; Wonka, P. Generating and exploring good building layouts. *ACM Trans. Graph.* **2013**, doi:10.1145/2461912.2461977.
102. Musialski, P.; Wimmer, M.; Wonka, P. Interactive coherence-based facade modeling. *Comput. Graph. Forum* **2012**, *31*, 661–670.
103. Bao, F.; Schwarz, M.; Wonka, P. Procedural facade variations from a single layout. *ACM Trans. Graph.* **2013**, doi:10.1145/2421636.2421644.
104. Schwarz, M.; Wonka, P. Procedural design of exterior lighting for buildings with complex constraints. *ACM Trans. Graph.* **2014**, doi:10.1145/2629573.
105. Merrell, P.; Schkufza, E.; Li, Z.; Agrawala, M.; Koltun, V. Interactive furniture layout using interior design guidelines. *ACM Trans. Graph.* **2011**, doi:10.1145/2010324.1964982.
106. Abrishami, S.; Goulding, J.S.; Rahimian, F.P.; Ganah, A. Integration of BIM and Generative Design to Exploit AEC Conceptual Design Innovation. Available online: http://clok.uclan.ac.uk/11420/ (accessed on 29 June 2015).
107. National Institute of Building Sciences. Frequently Asked Questions About the National BIM Standard. Available online: https://www.nationalbimstandard.org/faqs (accessed on 29 June 2015).
108. Abbasnejad, B.; Moud, H.I. BIM and basic challenges associated with its definitions, interpretations and expectations. *Int. J. Eng. Res. Appl.* **2013**, *3*, 287–294.
109. Eastman, C.; Teicholz, P.; Sacks, R.; Liston, K. *BIM Handbook*, 2nd ed.; John Wiley & Sons: New Jersey, NJ, USA, 2011.

110. Krispel, U.; Schinko, C.; Ullrich, T. The rules behind—Tutorial on generative modeling. *Proc. Symp. Geom. Process.* **2014**, *12*, 1–49.
111. Schinko, C.; Strobl, M.; Ullrich, T.; Fellner, D.W. Modeling procedural knowledge—A generative modeler for cultural heritage. In *Digital Heritage*; Springer: Berlin, Germany, 2010.
112. Hichri, N.; Stefani, C.; De Luca, L.; Veron, P. Review of the "as-built BIM" approaches. *ISPRSl Arch. Photogramm. Remote Sens. Spatial Inf. Sci.* **2013**, *XL-5/W1*, 107–112.
113. Kazhdan, M.; Bolitho, M.; Hoppe, H. Poisson surface reconstruction. In Proceedings of the Fourth Eurographics Symposium on Geometry Processing, Cagliari, Sardinia, 26–28 June 2006; pp. 61–70.
114. Weyrich, T.; Lawrence, J.; Lensch, H.P.A.; Rusinkiewicz, S.; Zickler, T. Principles of appearance acquisition and representation. *Found. Trends. Comput. Graph. Vis.* **2009**, *4*, 75–191.
115. Krispel, U.; Evers, H.L.; Tamke, M.; Viehauser, R.; Fellner, D.W. Automatic texture and orthophoto generation from registered panoramic views. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2015**, *XL-5/W4*, 131–137.
116. Tamke, M.; Blümel, I.; Ochmann, S.; Vock, R.; Wessel, R. From point clouds to definitions of architectural space-Potentials of automated extraction of semantic information from point clouds for the building profession. In Proceedings of the 32nd eCAADe Conference, Northumbria, UK, 10–12 September 2014; pp. 557–566.
117. Hullo, J.F.; Thibault, G.; Boucheny, C. Advances in Multi-sensor scanning and visualization of complex plants: The utmost case of a reactor building. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2015**, *XL-5/W4*, 163–169.
118. Dore, C.; Murphy, M. Semi-automatic modelling of building Faç with shape grammars using historic building information modelling. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2013**, *XL-5/W1*, 57–64.
119. Dore, C.; Murphy, M.; McCarthy, S.; Brechin, F.; Casidy, C.; Dirix, E. Structural simulations and conservation analysis-Historic building information model (HBIM). *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2015**, *XL-5/W4*, 351–357.
120. Quattrini, R.; Malinverni, E.S.; Clini, P.; Nespeca, R.; Orlietti, E. From TLS to HBIM. High quality semantically-aware 3D modeling of complex architecture. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2015**, *XL-5/W4*, 367–374.
121. Ladenhauf, D.; Berndt, R.; Eggeling, E.; Ullrich, T.; Battisti, K.; Gratzl-Michlmair, M. From building information models to simplified geometries for energy performance simulation. In Proceeding of the First International Academic Conference on Places and Technologies, Belgrade, Yugoslavia, 3–4 April 2014.
122. Ladenhauf, D.; Berndt, R.; Krispel, U.; Eggeling, E.; Ullrich, T.; Battisti, K.; Gratzl-Michlmair, M. Geometry simplification according to semantic constraints. *Comput. Sci. Res. Dev.* **2014**, *11*, 1–7.
123. Initiative, D.C.M. Dublin Core Metadata Initiative. Available online: http://dublincore.org/ (accessed on 29 June 2015).

124. Settgast, V.; Ullrich, T.; Fellner, D.W. Information technology for cultural heritage. *IEEE Potentials* **2007**, *26*, 38–43.
125. Fellner, D.W. Graphics content in digital libraries: Old problems, recent solutions, future demands. *J. Univers. Comput. Sci.* **2001**, *7*, 400–409.
126. Fellner, D.W.; Saupe, D.; Krottmaier, H. 3D documents. *IEEE Comput. Graph. Appl.* **2007**, *27*, 20–21.
127. Maybury, M.T. *Multimedia Information Extraction*; John Wiley & Sons: New Jersey, NJ, USA, 2012.
128. Bishop, C.M. *Pattern Recognition and Machine Learning*; Springer: Berlin, Germany, 2007.
129. Ulusoy, I.; Bishop, C.W. Generative versus discriminative methods for object recognition. In Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Diego, CA, USA, 20–26 June 2015.
130. Bustos, B.; Keim, D.; Saupe, D.; Schreck, T. Content-based 3D object retrieval. *IEEE Comput. Graph. Appl.* **2007**, *27*, 22–27.
131. King, B.D.; Wertheimer, M. *Max Wertheimer & Gestalt Theory*; Transaction Publishers: New Jersey, NJ, USA, 2005.
132. Hilaga, M.; Shinagawa, Y.; Kohmura, T.; Kunii, T.L. Topology matching for fully automatic similarity estimation of 3D shapes. In Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, Los Angeles, CA, USA, 12–17 August 2011; pp. 203–212.
133. Pauly, M.; Mitra, N.J.; Wallner, J.; Pottmann, H.; Guibas, L.J. Discovering structural regularity in 3D geometry. *ACM Trans. Graph.* **2008**, *27*, 1–11.
134. Bokeloh, M.; Wand, M.; Seidel, H.P. A connection between partial symmetry and inverse procedural modeling. In Proceedings of the ACM SIGGRAPH 2010, Los Angeles, CA, USA, 27–29 July 2010.
135. Mitra, N.J.; Guibas, L.J.; Pauly, M. Partial and approximate symmetry detection for 3D geometry. *ACM Trans. Graph.* **2006**, *25*, 560 – 568.
136. Mitra, N.J.; Guibas, L.J.; Pauly, M. Symmetrization. *Int. Conf. Comput. Graph. Interact. Tech.* **2007**, *26*, 1–8.
137. Ullrich, T.; Schinko, C.; Fellner, D.W. Procedural modeling in theory and practice. In Proceedings of the 18th WSCG International Conference on Computer Graphics, Visualization and Computer Vision, Plzen, Czech Republic, 27 March 2010.
138. Autodesk. Autodesk Maya API. Available online: http://docs.autodesk.com/MAYAUL/2014/ENU/Maya-API-Documentation/index.html (accessed on 29 June 2015).
139. Ramamoorthi, R.; Arvo, J. Creating generative models from range images. *Proc. ACM SIGGRAPH* **1999**, *1*, 195–204.
140. Ullrich, T. Reconstructive Geometry. Ph.D. Thesis, Technische Universität, Graz, Austria, 2011.
141. Arnold, D. Computer graphics and cultural heritage: From one-way inspiration to symbiosis. *Comput. Graph. Appl.* **2014**, *34*, 76–86.

142. Havemann, S.; Settgast, V.; Krottmaier, H.; Fellner, D.W. On the integration of 3D models into digital cultural heritage libraries. In Proceedings of the 7th International Symposium on Virtual Reality, Archaeology and Cultural Heritage (VAST), Nicosia, Cyprus, 30 October–4 November 2006.
143. Niccolucci, F. XML and the future of humanities computing. *ACM SIGAPP Appl. Comput. Rev.* **2002**, *10*, 43–47.
144. Niccolucci, F.; D'Andrea, A. An ontology for 3D cultural objects. In Proceedings of the 7th International Symposium on Virtual Reality, Archaeology and Cultural Heritage (VAST), Nicosia, Cyprus, 30 October–4 November 2006.
145. Arnold, D. Computer graphics and cultural heritage: Continuing inspiration for future tools. *Comput. Graph. Appl.* **2014**, *34*, 70–79.
146. Berndt, R.; Gerth, B.; Havemann, S.; Fellner, D.W. 3D modeling for non-expert users with the castle construction kit v0.5. In Proceedings of the 6th International Symposium on Virtual Reality, Archaeology and Cultural Heritage (VAST), Pisa, Italy, 8–11 Novermber 2005.
147. Ullrich, T.; Settgast, V.; Fellner, D.W. Semantic fitting and reconstruction. *J. Comput. Cult. Heritage* **2008**, *1*, 1201–1220.
148. Ullrich, T.; Schinko, C.; Schiffer, T.; Fellner, D.W. Procedural descriptions for analyzing digitized artifacts. *Appl. Geomat.* **2013**, *5*, 185–192.
149. Schinko, C.; Ullrich, T.; Fellner, D.W. Modeling with high-level descriptions and low-level details. In Proceeding of the International Conference on Computer Graphics, Visualization, Computer Vision and Image Processing, Lisbon, Portugal, 15–19 July 2014.
150. Calogero, E.; Arnold, D. Generating alternative proposals for the louvre using procedural modeling. In Proceedings of the 4th ISPRS International Workshop 3D-ARCH, Trento, Italy, 2–4 March 2011.
151. Saldana, M.; Johanson, C. Procedural modeling for rapid-prototyping of multiple building phases. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.*, **2013**, *1*, 205–210.
152. Fellner, D.W.; Havemann, S. Striving for an adequate vocabulary: Next generation metadata. In Proceedings of the 29th Annual Conference of the German Classification Society, Magdeburg, Germany, 9–11 March 2005.
153. Havemann, S.; Fellner, D.W. Seven research challenges of generalized 3D documents. *IEEE Comput. Graph. Appl.* **2007**, *3*, 70–76.
154. Havemann, S.; Ullrich, T.; Fellner, D.W. The meaning of shape and some techniques to extract it. *Multimed. Inf. Extr.* **2012**, *1*, 81–98.
155. Marcheix, D.; Pierra, G. A survey of the persistent naming problem. In Proceedings of the ACM Symposium on Solid Modeling and Applications, Saarbrucken, Germany, 17–21 June 2002.