Eurographics Workshop on 3D Object Retrieval (2011)
H. Laga, T. Schreck, A. Ferreira, A. Godil, I. Pratikakis, and R. Veltkamp (Editors)

# Generative Object Definition and Semantic Recognition

Torsten Ullrich[1] and Dieter W. Fellner[1,2]

[1]Fraunhofer Austria, Visual Computing, Austria
[2]TU Darmstadt & Fraunhofer IGD, Germany

**Abstract**

*"What is the difference between a cup and a door?" These kinds of questions have to be answered in the context of digital libraries. This semantic information, which describes an object on a high, abstract level, is needed in order to provide digital library services such as indexing, markup and retrieval. In this paper we present a new approach to encode and to extract such semantic information. We use generative modeling techniques to describe a class of objects: each class is represented by one algorithm; and each object is one set of high-level parameters, which reproduces the object if passed to the algorithm. Furthermore, the algorithm is annotated with semantic information, i.e. a human-readable description of the object class it represents.*
*We use such an object description to recognize objects in real-world data e.g. laser scans. Using an algorithmic object description, we are able to identify 3D subparts, which can be described and generated by the algorithm. Furthermore, we can determine the needed input parameters. In this way, we can classify objects, recognize them semantically and we can determine their parameters (cup's height, radius, etc.).*

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—H.3.1 [Information Storage and Retrieval]: Content Analysis and Indexing—G.1.6 [Numerical Analysis]: Optimization—

## 1. Introduction

With increasing number of (3D) documents, digital library services become more and more important. A digital library provides markup, indexing, and retrieval services based on metadata. In the simplest case, metadata is of the Dublin Core type [Ini95] (title, creator/author, time of creation, etc). 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 [SUF07]. Navigation and browsing through the geometric models should be possible not only in 3D, but also on the semantic level. This requires higher-level semantic information. Semantic questions ("How many windows does this facade have?", "How many steps do these stairs have?", etc.) cannot be answered, if the library simply treats 3D objects as binary large objects (BLOB). The need for semantic information becomes immediately clear in the context of electronic data exchange, storage and retrieval [Fel01, FSK07].

## 2. Object Descriptions

The problem of 3D semantic enrichment is closely related to the shape description problem [May11]: How to describe a shape and its structure on a higher, more abstract level?

### 2.1. Description by Definition

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:

> cup – a small, open container made of china, glass, metal, etc., usually having a handle and used chiefly as a receptable from which to drink tea, soup, etc.
>
> http://dictionary.reference.com

This approach is hardly realizable because of the fact that definitions cannot be self-contained. They depend on other definitions (e.g., container, handle, receptable, ...), which



DOI: 10.2312/3DOR/3DOR11/001-008

**Figure 1:** *The example data set consists of twelve laser-scanned cups made of porcelain. Each scanned cup comprehends between 15 573 (small espresso cup) and 130 973 triangles (big mug). As the real cups have clean and shiny surfaces, they are difficult to scan. The scan results are noisy and not-cleaned-up meshes with many holes. In this illustration the surfaces are rendered semi-transparent, so that incomplete parts (with missing inner / outer surface) appear brighter than complete parts. Please note, each subimage is scaled to a common bounding box.*

leads to circular dependencies that cannot be resolved automatically by strict reasoning, but rely on intuitive understanding at some point.

### 2.2. Description by Example

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 whether object $x$ belongs to class $A$ or $B$ requires measuring the closeness of $x$ to the exemplars $a \in A$ resp. $b \in B$. This decision can be reached by a classifier using statistics and machine learning [Bis07, UB05]. A good survey on content-based 3D object retrieval is provided by Benjamin Bustos et al. [BKSS07]. 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: width and height of a detected rectangle must typically be computed separately.

### 2.3. Shape Analysis

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 [KW05]. One idea to operationalize this concept was proposed, among others, by Masaki Hilaga [HSKK01], who introduces the Multiresolution Reeb Graph, to represent the skeletal and topological structure of a 3D shape at various levels of resolution. Structure recognition is a very active branch in the field of geometry processing. The detection of shape regularities [PMW*08], self-similarities [BWS10] and symmetries [MGP06, MGP07] 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.

### 2.4. Description by Algorithm

A possibility to describe a shape is realized by the generative modeling paradigm [ÖK08, USF10]. 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 [MWH*06], modeling language [Hav05] or modeling script [Aut07].

## 3. Generative Modeling and Semantic Enrichment

To encode a shape we use the "definition by algorithm" approach based on a scripting language: 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.

This encoding of semantic information can be used by our algorithm to enrich 3D objects semantically: the algorithm starts with a point cloud $P$ and a generative model $M$. Without user interaction it determines a parameter set $x_0$, which minimizes the geometrical distance between $P$ and $M(x_0)$. This distance $d$ can be interpreted as a multidimensional error function of a global optimization problem. Therefore, standard techniques of function minimization can be used. Having found the global minimum $x_0$, the geometric distance $d(P, M(x_0))$ can be interpreted. A low value corresponds to a perfect match; i.e. the point cloud $P$ is (at least partly) similar to $M(x)$, whereas a high value indicates no similarity. Consequently, the presented approach is able to semantically recognize instances of generative objects in real data sets.

As the computational complexity of global optimization depends on the dimensions of the error function, our approach uses a hierarchical optimization strategy with coarse model descriptions and few parameters at the beginning and detailed model descriptions at the end. This multi-step optimization determines free parameters successively, fixes them and introduces new parameters. This process stops, if the end of the hierarchy is reached, or if high error values indicate no object similarity.

In contrast to related work on fitting algorithms – such as "Creating Generative Models from Range Images" by Ravi Ramamoorthi and James Arvo [RA99] – our approach can classify data semantically. 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, which is significantly reduced. But starting with the same generative description to fit a spoon as well as a banana does not allow to generate or preserve semantic data.

## 4. Proof of Concept

The input data sets of our algorithm are a point cloud $P$ and a generative model $M$. Then, the algorithm answers the questions

1. whether the point cloud can 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 $P$.

To demonstrate this algorithm we laser-scanned twelve cups and scripted a generative model of a cup, which takes 15 input parameters: six parameters describe its position and orientation, nine parameters describe its attributes (radius, height, ...). The scans are visualized in Figure 1 whereas the generative model is sketched in Figure 2. The 15 parameters of the generative model are determined using three hierarchical levels.

1. The first level determines the cup's base point $(x, y, z)$ and its orientation $(\alpha, \beta)$, as well as its height $h$, radius $r$, and *shape*. At this level the cup is rotationally symmetric; therefore, position and orientation only need five instead of six parameters. The *shape* parameter is used in two

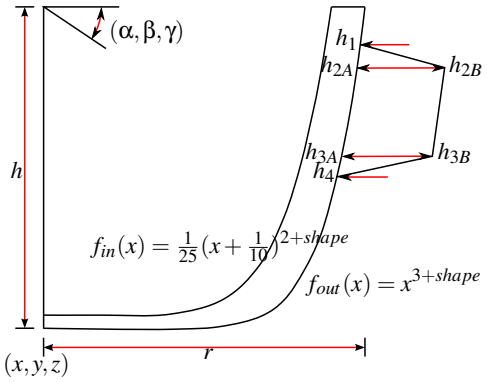


**Figure 2:** *The generative cup model takes 15 parameters: $(x, y, z)$ is the base point of the cup and $(\alpha, \beta, \gamma)$ define its orientation. Its shape is defined by an inner $f_{in}$ and outer $f_{out}$ shape function with one free parameter shape. These functions are rotated around the cup's main axis and scaled with the parameters r and h.*
*The handle is defined via six parameters, which form points in 2D (the plane of the handle); namely $(h_1, f_{out}(h1))$, $(h_{2A}, h_{2B})$, $(h_{3A}, h_{3B})$, and $(h_4, f_{out}(h_4))$. They are the control points of a Bézier curve. Its tube with a fixed diameter (10mm) defines the cup's handle.*

functions $f_{in}$, $f_{out}$ which define the cup's inner and outer shape (see Figure 2).
2. Afterwards, the algorithm determines the parameter $\gamma$ – the rotational position of the handle.
3. At the last level, the parameters $h_1$, $h_{2A}$, $h_{2B}$, $h_{3A}$, $h_{3B}$, $h_4$ are determined. These parameters define four points of a Bézier curve. As $h_1$ and $h_4$ are start and end point of the handle, they are located at $(h_1, f_{out}(h1))$ resp. $(h_4, f_{out}(h_4))$, whereas the second $(h_{2A}, h_{2B})$ and third point $(h_{3A}, h_{3B})$ may float freely within the plane with orientation $\gamma$. The resulting Bézier curve is expanded to a 3D tube with a fixed diameter of 10mm.

This generative model is able to describe the scanned cups and its parameters can be determined automatically by the proposed algorithm.

## 5. Algorithmic Details & Implementation

The scanned data set is a point cloud $P = \{p_1, \ldots, p_n\}$ and the generative model can be regarded as a function $M(x)$, $x \in G \subset \mathbb{R}^k$. The objective function of our algorithm minimizes the distance between the geometric objects $P$ and $M(x)$; i.e. $d(P, M(x))$. As the commonly-used least-squares approach weights outlying points, the chosen weighting function is $\psi(x) = 1 - e^{-x^2/\sigma^2}$. This function reduces the disproportional effect of outlying points [Zha97, USF08]. Consequently, the objective function is

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

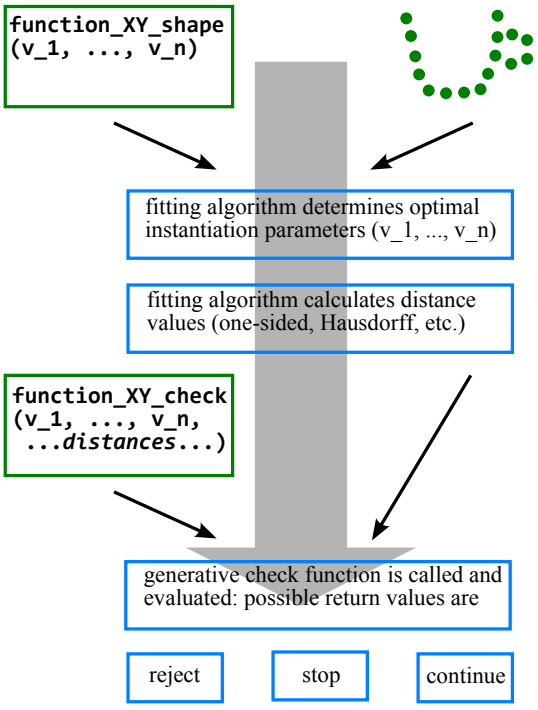


**Figure 3:** *This diagram illustrates the fitting process at one level within the fitting hierarchy. The input parameters (green) are a point cloud and a generative description of a shape.*

### 5.1. Hierarchical Models

In order to reduce the number of dimensions which are optimized at once, we use a hierarchy of model descriptions. Each level within this hierarchy reuses the parameters already defined in previous levels and refines the generative model. In this way the optimization problem is split up in several smaller problems. At each level of the hierarchy (as illustrated in Figure 3) a check function (a part of the generative description) is called and its returned value determines the following steps of the process. If it is too high, the fitting process is *rejected* or *stopped*. Otherwise the process is *continued* with the next level's shape function. The difference between *rejected* and *stopped* is subtle. In both cases the process has come to an end, but in one case the result is not acceptable (e.g. no cup found at all in hierarchy level #1) and in the other case the result is acceptable (e.g. no handle found in hierarchy level #2).

Using this hierarchy the generative model contains model descriptions at different levels of resolution. Especially at early stages within the hierarchy it may not be able to describe geometry precisely – due to missing parameters. We solved this problem by introducing fuzzy geometry.

### 5.2. Fuzzy Models

Fuzzy geometry is a point cloud, in which each point is extended by a probability value $\sigma$. This value defines a normal distribution in 3D. A fuzzy geometry model consists of all overlapping normal distributions; i.e. a blurred point cloud. This technique allows a modeler to describe diffuse geometry. For example, in hierarchy level #2 of the cup model, the orientation of the handle (parameter $\gamma$) can be determined

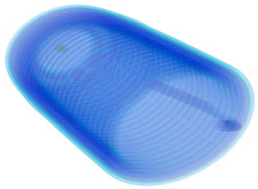

**Figure 4:** *Geometry at a low resolution can be described by a fuzzy point cloud; i.e. a point cloud in which each point is normally distributed in space. This illustration shows such a fuzzy point cloud. Each point is drawn as a semi-transparent sphere whose radius corresponds to a fixed probability – an isosurface in probability space.*

before its shape (parameters $h_1, \ldots, h_4$) is known. At hierarchy level #2 the handle is just a very fuzzy, fixed "standard" handle.

This model description is used for *all* hierarchy levels except those, which may stop the fitting process with an acceptable solution. These levels produce precise geometric models based on meshes.

### 5.3. Inverse Models

The fuzzy models can be interpreted as an energy field – each point is an energy source whose power is quantified by $\sigma$. According to this interpretation the objective function "tries" to place the generative model, so that the scan is in a high energy area. This attraction effect is inverted by a new geometry description: inverse models. In inverse models the values of $\sigma$ are negative and the used weighting function is $1 - \psi(x)$ (see Figure 5). As a consequence, negative points "try" to maximize the distance to the scan. Inverse models

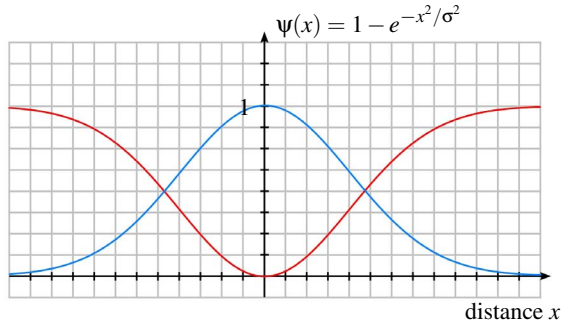


**Figure 5:** *The weighting function $\psi(x)$ (plotted in red) reduces the disproportionate effect of outlying points. Furthermore, its bounded codomain $0 \leq \psi(x) \leq 1$ can be used to introduce distance-based "penalty" terms $1 - \psi(x)$ (plotted in blue) in the objective function. These terms are the main idea of inverse models.*

are able to describe objects, which can hardly be described otherwise. For example, the main feature of a window is: being a hole in a wall. Using inverse models, this hole can be filled with negative points, which introduce penalty terms into the objective function, if placed within wall geometry.

### 5.4. Distance Calculation

The distance computation is by far the most time-consuming part of the algorithm. Due to the fact that the weighting function $\psi$ has an upper limit

$$\forall x \in \mathbb{R} : \psi(x) < 1 \tag{2}$$

and converges relatively fast to one (see Figure 5), the objective function can be evaluated very efficiently. The weighted distances of points "far away" (depending on a threshold calculated using $\sigma$) can be approximated by $\psi \approx 1$ and do not have to be calculated exactly. Bounding volumes, partitioning of space, and hashed grid structures speed up the evaluation of $f$. Only small distances are evaluated exactly [USK*07].

### 5.5. Script Compilation

The previous subsections describe the objective function $f$. This function is called by the numerical optimization routine, which consists of two parts. The first part uses a statistical optimization routine called Differential Evolution [SP97]. This algorithm is used to find a "good" starting point for the second optimization part – a Conjugate Gradients optimization according to Fletcher-Reeves [GMW82, Fle00, GJH95].

Both parts evaluate the generative script up to several thousand times. Therefore, we integrated a compiler which translates the script code to machine code. Furthermore, the compiler parses the script, creates an abstract syntax tree and differentiates it with respect to input parameters [UKF08, SSUF10a, SSUF10b]; i.e. the optimization can use both the objective function

$$f(x_1, \ldots, x_k) = \psi(d(P, M(x_1, \ldots, x_k))), \quad x_i \in \mathbb{R} \tag{3}$$

as well as its partial derivatives $\frac{\partial f}{\partial x_i}$. 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.

## 6. Results and Conclusions

### 6.1. Example

The example data set consists of twelve laser-scanned cups (see Figure 1) and a generative cup description (see Figure 2). The fitting results are visualized in Figure 6. In 11 of 12 cases (92%) the algorithm is able to detect an instance of the generative cup $M$. In these cases the cups' properties (position, orientation, radius, height, handle shape) are determined with only a small error.

In one case (cup #5) the global optimization routine is stuck in a local minimum. Despite the local minimum, the error values are too high, so that the algorithm rejects the hypothesis of a generative cup. For no apparent reason the algorithm failed in this case. Further investigations on this failure are a task of future work. For illustration purposes Figure 6 (top row, right) shows the last best-fit result of the optimization which would have been returned, if the algorithm had not stopped and rejected the fitting process.

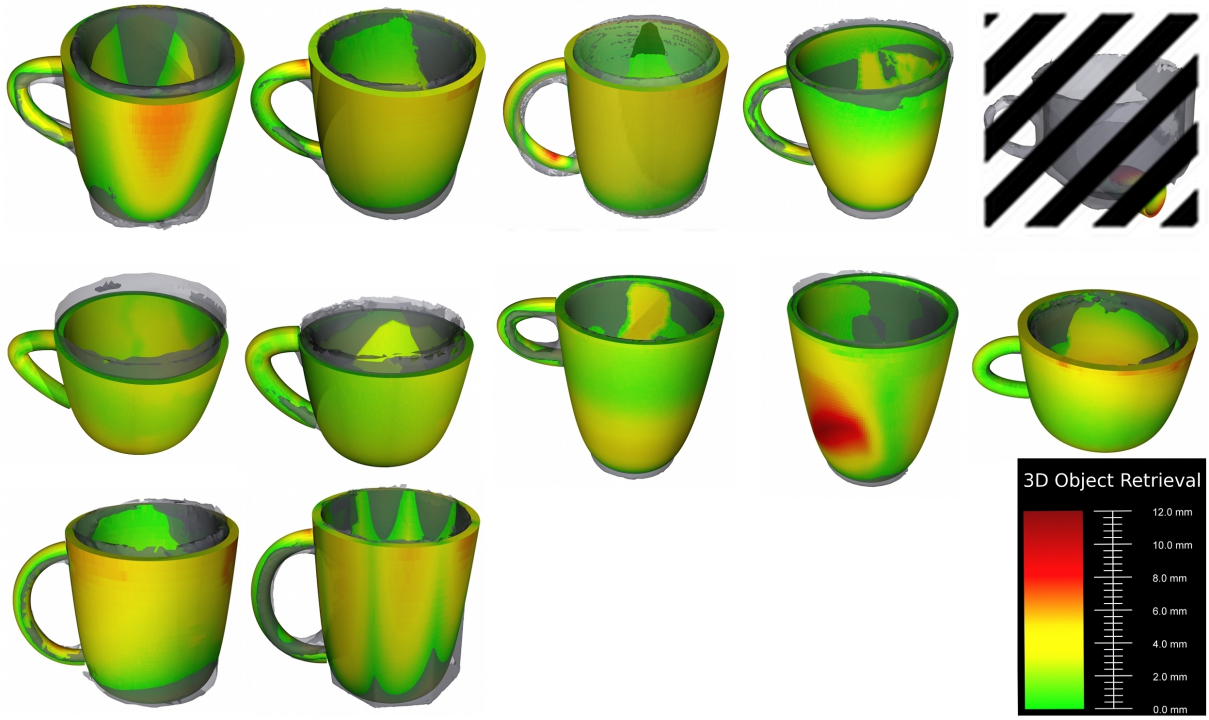


**Figure 6:** *The scanned cup (rendered in semi-transparent gray) have been identified as instances of the generative cup description in 11 of 12 cases. In these cases the cups' properties (position, orientation, radius, height, handle shape) are determined successfully. Detailed distance measurements are listed in Table 1.*
*Please note, each subimage is scaled to a common bounding box.*

In the other cases, the fitting process has been successful. Especially, the cups #1 and #12 have been fitted, although their shapes (the scan is rendered semi-transparent in gray) are not rotationally symmetric. Cup #1 (Figure 6, top row, left) has quadratic footprint with beveled edges; cup #12 (Figure 6, bottom row, middle-left) has an octagonal footprint. In both cases the generative cup is able to describe them within a tolerable rate of variance.

Also cup #9 (Figure 6, middle row, middle-right) has been detected to be a cup, although its shape looks like a busted paper cup. Its error value passed the threshold of rejection at the first level, but it did not pass the following ones. Therefore, the algorithm perfectly identifies a cup without a handle.

The distance values of all cups are listed in Table 1. During the fitting process mainly one-sided distances (from the generative model to the scan) and Hausdorff distances are used. While the Hausdorff distance gives an "overall impression", the one-sided distances can be used to measure local fits; e.g. if each point on the generated handle is (in average) half its diameter away from the scan, then the cup will most probably not have a handle.

### 6.2. Contribution & Benefit

In this article we present a shape description approach based on generative modeling techniques and an algorithm, which is able to identify instances of a generative description in real-world data sets. The algorithm demonstrates the proof of concept.

The main contributions and benefits of this article are an implementation of a generative shape description approach including the inverse recognition and indexing problem. Based on a generative description the algorithm is able to identify instances of a script and it can determine its calling parameters.

To implement this concept, we used a hierarchical model description with fuzzy geometry to represent "unknown" parts of a model (parts which are fitted at lower levels within the hierarchy). Furthermore, we used inverse models to describe the absence of geometry. This concept offers the possibility to formulate "missing" geometry (e.g. a window is a hole in a wall).

Including our generative compiler with automatic derivation, our algorithm can evaluate both the objective function $f(x_1,\ldots,x_k)$ as well as its partial derivatives $\frac{\partial f}{\partial x_i}$. This tech-

| | one-sided distance (generative model → scan) average | one-sided distance (generative model → scan) maximum | one-sided distance (generative model → scan) standard deviation | Hausdorff distance |
|---|---|---|---|---|
| scan #1 | 3.372281 mm | 8.048851 mm | 1.9997352 mm | 9.246573 mm |
| scan #2 | 3.162463 mm | 6.746494 mm | 1.8630469 mm | 6.800794 mm |
| scan #3 | 4.701682 mm | 8.877957 mm | 2.4739857 mm | 10.01246 mm |
| scan #4 | 2.355809 mm | 8.229928 mm | 1.5147543 mm | 8.229928 mm |
| scan #5 | – | – | – | 68.26691 mm |
| scan #6 | 2.098856 mm | 5.484409 mm | 1.1627039 mm | 14.90319 mm |
| scan #7 | 1.751712 mm | 5.558793 mm | 0.9201700 mm | 7.465546 mm |
| scan #8 | 2.239976 mm | 6.327161 mm | 1.3496941 mm | 6.327161 mm |
| scan #9 | 2.257052 mm | 10.77931 mm | 1.7870259 mm | 10.77931 mm |
| scan #10 | 2.258514 mm | 6.985035 mm | 1.3106395 mm | 6.985035 mm |
| scan #11 | 3.919888 mm | 7.815891 mm | 2.1347158 mm | 11.44331 mm |
| scan #12 | 2.961438 mm | 8.685675 mm | 1.9820354 mm | 8.685675 mm |

**Table 1:** *The detailed measurements of the fitted, generated cups. The scan numbers correspond to the visualizations shown in Figure 6. As cup #5 have been rejected, it does not have sensible distance values. In all other cases, the distance values have been calculated between a scan and its best-fit generative description.*

nique offers the possibility to use standard optimization algorithms to solve the inverse problem efficiently. Each fitting process (with one scan and one generative hierarchy) takes about 2-3 minutes one a single PC to finish. These timings just give an impression of the algorithm's performance. Due to open problems it is currently not sensible to run benchmarks at great length.

### 6.3. Open Problems and Future Work

An urgent open problem is the algorithm's failure in test case #5. As long as this problem is not solved, detailed benchmarks would not be reasonable. If the problem was caused by a premature termination of the routine to avoid local minima, the algorithm's timings would change signifcantly.

In the medium term we will use generative shape descriptions with human-readable annotations in order to index 3D data. Generative scripts will describe 3D subparts (stairs, windows, cups, etc.) and the presented algorithm processes them. It will search for them within the 3D data. Having found an instance of a generative description, this offline indexing step can copy the human-readable annotation ("This is a cup.") into its markup. Afterwards, a search query – performed on the copied textual annotations – can return all corresponding models. The challenge of this task is the generalizability of this approach, the concurrence of generative descriptions and their discriminatory power. We believe that our approach can be generalized to different kinds of objects. Nevertheless, the handling of concurrent generative descriptions will be interesting. They may be handled separately (in order to avoid interdependencies) or joined to one hierarchy (for performance). We will investigate these questions on a data base of 3D models in the future.

## 7. Acknowledgments

The authors would like to thank Christoph Schinko and Martin Strobl for their support on the scripting compiler *Euclides*.

Furthermore, we gratefully acknowledge the generous support from the European Commission for the integrated project 3D-COFORM (3D COllection FORMation, www.3D-coform.eu) under grant number FP7 ICT 231809.

We would also like to thank the Austrian Research Promotion Agency (FFG) for the research project METADESIGNER (Meta-Design Builder: A framework for the definition of end user interfaces for product mass-customization), grant number 820925 / 18236.

Additionally, the German Research Foundation DFG for the research project PROBADO (PROtotypischer Betrieb Allgemeiner DOkumente, http://www.probado.de) supports this work under grant INST 9055/1-1.

## References

[Aut07] AUTODESK: Autodesk Maya API. *White Paper 1* (2007), 1–30. 3

[Bis07] BISHOP C. M.: *Pattern Recognition and Machine Learning*. Springer, 2007. 2

[BKSS07] BUSTOS B., KEIM D., SAUPE D., SCHRECK T.: Content-based 3D Object Retrieval. *IEEE Computer Graphics and Applications 27*, 4 (2007), 22–27. 2

[BWS10] BOKELOH M., WAND M., SEIDEL H.-P.: A Connection between Partial Symmetry and Inverse Procedural Modeling. *Proccedings of ACM SIGGRAPH 2010 29* (2010), 104:1–104:10. 2

[Fel01] FELLNER D. W.: Graphics Content in Digital Libraries: Old Problems, Recent Solutions, Future Demands. *Journal of Universal Computer Science 7* (2001), 400–409. 1

[Fle00] FLETCHER R.: *Practical Methods of Optimization*. Wiley, 2000. 5

[FSK07] FELLNER D. W., SAUPE D., KROTTMAIER H.: 3D Documents. *IEEE Computer Graphics and Applications 27*, 4 (2007), 20–21. 1

[GJH95] GUANGHUI L., JIYE H., HONGXIA Y.: Global convergence of the fletcher-reeves algorithm with inexact linesearch. *Applied Mathematics - A Journal of Chinese Universities 10* (1995), 75–82. 5

[GMW82] GILL P. E., MURRAY W., WRIGHT M. H.: *Practical Optimization*. Academic Press, 1982. 5

[Hav05] HAVEMANN S.: Generative Mesh Modeling. *PhD-Thesis, Technische Universität Braunschweig, Germany 1* (2005), 1–303. 3

[HSKK01] HILAGA M., SHINAGAWA Y., KOHMURA T., KUNII T. L.: Topology Matching for Fully Automatic Similarity Estimation of 3D Shapes. *Proceedings of the 28th annual conference on Computer graphics and interactive techniques 28* (2001), 203–212. 2

[Ini95] INITIATIVE D. C. M.: Dublin Core Metadata Initiative. http://dublincore.org/, 1995. 1

[KW05] KING B. D., WERTHEIMER M.: *Max Wertheimer & Gestalt Theory*. Transaction Publishers, 2005. ISBN 0-7658-0258-9. 2

[May11] MAYBURY M. T. (Ed.): *Multimedia Information Extraction*. 2011. 1

[MGP06] MITRA N. J., GUIBAS L. J., PAULY M.: Partial and approximate symmetry detection for 3D geometry. *ACM Transactions on Graphics 25* (2006), 560 – 568. 2

[MGP07] MITRA N. J., GUIBAS L. J., PAULY M.: Symmetrization. *International Conference on Computer Graphics and Interactive Techniques 26* (2007), 1–8. 2

[MWH*06] MÜLLER P., WONKA P., HAEGLER S., ANDREAS U., VAN GOOL L.: Procedural Modeling of Buildings. *Proceedings of 2006 ACM Siggraph 25*, 3 (2006), 614–623. 3

[ÖK08] ÖZKAR M., KOTSOPOULOS S.: Introduction to shape grammars. *International Conference on Computer Graphics and Interactive Techniques ACM SIGGRAPH 2008 (course notes) 36* (2008), 1–175. 3

[PMW*08] PAULY M., MITRA N. J., WALLNER J., POTTMANN H., GUIBAS L. J.: Discovering structural regularity in 3D geometry. *ACM Transactions on Graphics 27* (2008), #43, 1–11. 2

[RA99] RAMAMOORTHI R., ARVO J.: Creating Generative Models from Range Images. *Proceedings of ACM Siggraph 1* (1999), 195–204. 3

[SP97] STORN R., PRICE K.: Differential Evolution: A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization 11* (1997), 341–359. 5

[SSUF10a] SCHINKO C., STROBL M., ULLRICH T., FELLNER D. W.: Modeling Procedural Knowledge – a generative modeler for cultural heritage. *Proccedings of EUROMED 2010 - Lecture Notes on Computer Science 6436* (2010), 153–165. 5

[SSUF10b] STROBL M., SCHINKO C., ULLRICH T., FELLNER D. W.: Euclides – A JavaScript to PostScript Translator. *Proccedings of the International Conference on Computational Logics, Algebras, Programming, Tools, and Benchmarking (Computation Tools) 1* (2010), 14–21. 5

[SUF07] SETTGAST V., ULLRICH T., FELLNER D. W.: Information Technology for Cultural Heritage. *IEEE Potentials 26*, 4 (2007), 38–43. 1

[UB05] ULUSOY I., BISHOP C. W.: Generative versus Discriminative Methods for Object Recognition. *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition 2* (2005), 258 – 265. 2

[UKF08] ULLRICH T., KRISPEL U., FELLNER D. W.: Compilation of Procedural Models. *Proceeding of the 13th International Conference on 3D Web Technology 13* (2008), 75–81. 5

[USF08] ULLRICH T., SETTGAST V., FELLNER D. W.: Semantic Fitting and Reconstruction. *Journal on Computing and Cultural Heritage 1*, 2 (2008), 1201–1220. 4

[USF10] ULLRICH T., SCHINKO C., FELLNER D. W.: Procedural Modeling in Theory and Practice. *Poster Proceedings of the 18th WSCG International Conference on Computer Graphics, Visualization and Computer Vision 18* (2010), 5–8. 3

[USK*07] ULLRICH T., SETTGAST V., KRISPEL U., FÜNFZIG C., FELLNER D. W.: Distance Calculation between a Point and a Subdivision Surface. *Proceedings of 2007 Vision, Modeling and Visualization (VMV) 1* (2007), 161–169. 5

[Zha97] ZHANG Z.: Parameter Estimation Techniques: A Tutorial with Application to Conic Fitting. *Image and Vision Computing Journal 15*, 1 (1997), 59–76. 4