Cupid: Cluster-Based Exploration of Geometry Generators with Parallel Coordinates and Radial Trees

Michael Beham, Wolfgang Herzner, M. Eduard Gröller, *Member, IEEE CS*, and Johannes Kehrer

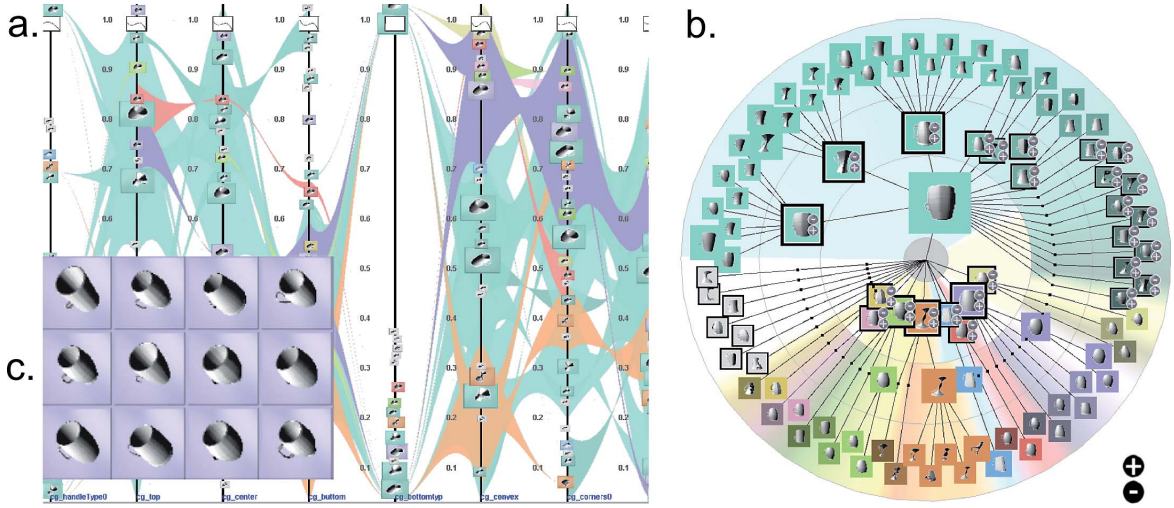

Fig. 1. Overview of *Cupid*: (a) Composite parallel coordinates relate the parameter space of a cup generator and clusters of similar 3D cups. (b) Linked radial tree depicting the hierarchy of the clusters. (c) Members of a cluster can be compared in a detail window.

Abstract—Geometry generators are commonly used in video games and evaluation systems for computer vision to create geometric shapes such as terrains, vegetation or airplanes. The parameters of the generator are often sampled automatically which can lead to many similar or unwanted geometric shapes. In this paper, we propose a novel visual exploration approach that combines the abstract parameter space of the geometry generator with the resulting 3D shapes in a composite visualization. Similar geometric shapes are first grouped using hierarchical clustering and then nested within an illustrative parallel coordinates visualization. This helps the user to study the sensitivity of the generator with respect to its parameter space and to identify invalid parameter settings. Starting from a compact overview representation, the user can iteratively drill-down into local shape differences by clicking on the respective clusters. Additionally, a linked radial tree gives an overview of the cluster hierarchy and enables the user to manually split or merge clusters. We evaluate our approach by exploring the parameter space of a cup generator and provide feedback from domain experts.

Index Terms—Composite visualization, hierarchical clustering, illustrative parallel coordinates, radial trees, 3D shape analysis

1 Introduction

Geometry generators create polygonal meshes that approximate a specific domain of geometric shapes such as cars, airplanes or vegetation. They are often used in video games to generate realistic worlds. For example, every tree in a 3D scene has a different shape by varying the parameters of the generator [32]. Evaluation systems for computer vision also use geometry generators to create virtual test cases, e.g., to evaluate the recognition of different shapes of an object [42].

Geometry generators usually have a set of parameters that determine the shape of the generated objects. A parameter can be categorical such as airplane type or continuous such as size of the wing. A geometry generator can be seen as a function $f : \mathbf{R}^n \rightarrow \mathbf{R}^m$, where n is the number of parameters, and m is the dimensionality of the resulting geometric shapes (in our case, we use 3D shapes, but geometry generators can also create 2D or time-varying shapes [32]). For creating shapes, the parameter space of the generator is usually sampled automatically. A general goal in this context is to understand the relations between the parameter space and the resulting geometric shapes [4]. We identify three tasks for analyzing geometry generators which are also representative for user tasks in other visual analytics applications:

Task 1: Finding similar 3D shapes and the corresponding parameter settings. The user wants to identify which regions of the parameter space create certain types of similar 3D shapes. In the case of airplanes, for example, different types of airplanes would be military jets, small airplanes, and airliners. Variations within the type military jet can be MIG-15, MIG-29, and Sukhoi Su-27. However, we are typically not interested in small variations of the geometric shape, for instance, whether an airplane has round or squared windows.

Task 2: Finding errors and implausible 3D shapes. Geometry generators can produce (physically) implausible or unwanted results. For example, the appropriate profile of a wing is important for the ability of the plane to fly. The generated meshes may also contain errors such as holes or self-intersections of the surface. However, not only one parameter region can produce unwanted results but a combination of parameters too. Our approach helps the user to identify

- *Michael Beham is with the Vienna University of Technology and the Austrian Institute of Technology. E-mail: e0726417@student.tuwien.ac.at.*
- *Wolfgang Herzner is with the Austrian Institute of Technology. E-mail: wolfgang.herzner@ait.ac.at.*
- *M. Eduard Gröller is with the Vienna University of Technology. E-mail: groeller@cg.tuwien.ac.at.*
- *Johannes Kehrer is with the Vienna University of Technology and the Technische Universität München. E-mail: johannes.kehrer@tum.de.*

Manuscript received 31 Mar. 2014; accepted 1 Aug. 2014. Date of publication 11 Aug. 2014; date of current version 9 Nov. 2014.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.
Digital Object Identifier 10.1109/TVCG.2014.2346626

such problematic regions in the parameter space by directly relating the parameters and the created 3D shapes in the visualization.

Task 3: Determine the sensitivity and influence of parameters on the result. It is often difficult to predict how changes in one or more of the parameter values influence the resulting geometric shapes. Changes may be global or local, depending on the parameter (e.g., changing the type of an airplane vs. changing the number of windows). Additionally, the sensitivity of a parameter may not be linear. That is, for certain regions of the parameter space even small variations may have a drastic effect on the resulting 3D shapes, while changes in less sensitive regions may have little to no effect on the created geometry.

In their previous workflow, our domain experts would manually study thumbnails of the generated 3D shapes in an image gallery and look up the corresponding parameter settings in a table. This is cumbersome and time consuming, especially in case of many generated 3D shapes. Motivated by the tasks described above, we introduce a new visualization approach called *Cupid* to efficiently explore the results of a geometry generator for cups. In this context, our paper represents a close collaboration between visualization researchers and domain experts (one of the paper authors has developed both the cup generator and *Cupid*). The contributions of this work are as follows:

- We propose a *composite visualization* that combines both the abstract parameter space of the geometry generator and the resulting 3D shapes in a single visualization. This enables the user to study relations between the parameter space and the created geometry.
- Adapting hierarchical clustering, illustrative parallel coordinates, and a linked radial tree, the user can interactively drill-down into groups of similar 3D shapes (compare to tasks 1 and 3).
- We evaluate our approach by exploring the results of a cup generator based on tasks 1–3 and provide feedback from domain experts.

2 Related Work

Visualization and exploration of geometry. The exploration of a set of geometric shapes is an active area of research. Smith et al. [38] and Ovsjanikov et al. [29] enable the exploration of a set of 3D shapes using template objects. The user can deform a template model and the corresponding object is shown. Coffey et al. [9] extend this technique to explore simulations or visual effects. By dragging parts of an object, the user can explore different simulations. Smith et al. [38] and Coffey et al. [9] also integrate abstract data into the visualization. Smith et al. add attributes to the generated cars such as weight and sportiness. The user can select the shape by changing the value of each attribute. Coffey et al. allow the user to select simulations using a wing plot. Talton et al. [39] present a collaborative tool to get high-quality and alternative models of a geometry generator depending on a given example. They use a linked representation of the modeling tool and a map that shows alternative 3D models. The 3D shapes are placed in a semantic map depending on their similarity. The map can be adjusted by an administrator. To detect high-quality models in the parameter space, the activity of users is tracked additionally.

Composite visualizations combine "two or more visual structures in the same view" [18]. DesignGalleries [26] use this concept for exploring multidimensional parameter spaces and (time-varying) spatial data. After the sampling, a compact representation of the parameter space is given using multidimensional scaling. Each data point is represented with thumbnails in the multidimensional layout. Additionally, some examples are visualized in detail in an image gallery. The idea of using abstract data to explore a set of shapes is also used by Buskin et al. [7]. They use a 2D scatterplot where the data points are replaced by 3D shapes. Additionally, linked views for the evaluation of a shape and comparison of different shapes are provided. Balabanian et al. [1] extends the idea of combining abstract and spatial data to hierarchical data. They nest spatial data from a computer tomography within a graph representing the hierarchy of the body parts. Kniss et al. [22] present direct manipulation widgets to specify transfer functions directly within a volume visualization. Schmidt et al. [35] perform hierarchical clustering in order to identify differences between sets of images. This approach also enables a drill-down into the hierarchy in order to evaluate the dataset. Our work differs from the approaches described above in that we focus on the exploration of the complex relations within the parameter space. We provide a new integration of spatial data into an abstract space using parallel coordinates. We do not only explore a set of 3D shapes but also provide techniques for evaluating the complex relationships between the parameter space and the generated 3D shapes.

Higher dimensional data. The visualization of higher dimensional data is a challenging problem. Such data are often analyzed using coordinated multiple views (see Roberts [33] for an overview). Different data dimensions are explored in linked views including scatterplot matrices [2] or parallel coordinates [15, 17]. Fua et al. [11] extend parallel coordinates for supporting hierarchical clustering. The user can interactively drill-down into a tree structure. McDonnell and Mueller [28] propose illustrative parallel coordinates for visualizing clusters. Their technique enables to define the level of detail of a cluster interactively. They use edge bundling to reduce visual clutter and to decrease the amount of covered screen space (compare to Holten [16]). Heinrich et al. [14] give an overview of edge bundling techniques. Lex et al. [25] propose a focus+context visualization for comparing separately clustered groups of variables of biomolecular data. Clustered records are connected across multiple groups of variables using bundled curves and ribbons. Clusters are a common technique for simplifying large multi-dimensional data. While Talton et al. [39] visualize clusters using an example, Lewis et al. [24] present with VisualIDs a technique to automatically generate icons for clusters of textual data.

Parameter studies. HyperSlice [41] is one of the earliest approaches for exploring higher dimensional parameter spaces. It represents a higher dimensional function as a matrix of orthogonal 2D slices around a user-controlled focal point. HyperMoVal [30] builds upon this concept and uses 2D and 3D projections of ensemble data around a focal point. Predictions of variations of one model parameter are represented as function graphs. Berger et al. [4] extend HyperMoVal for exploring the continuous space of input and output variables of a simulation. The local neighborhood around the focal point in the input parameters is mapped to the output domain. Additionally, the local area around the focal point is shown where variations of input parameters do not affect the predicted output by more than a certain threshold. Also other approaches use multiple linked views to study relations between the input and output of a simulation model [19, 20, 27].

Bruckner and Möller [6] present a result-driven exploration approach for physically-based ensemble simulations. Each volumetric time sequence is first split into similar segments over time and then grouped across different ensemble members using density-based clustering. Similar to *Cupid*, their work supports the user in identifying similar behavior in different ensemble members. However, the parameter space and the generated 3D volumes are not directly combined within the same visualization, but analyzed side-by-side using linked views. Piringer et al. [31] present an approach for analyzing 2D function ensembles. The 2D functions are represented as icons which are nested within scatterplots that depict the simulation parameters. The icons can be analyzed at different levels of detail using brushing. While their approach uses 2D scatterplots to represent the parameter space of the simulation, we deal with a higher dimensional parameter space using parallel coordinates. Additionally, we focus on the analysis of similar 3D shapes using hierarchical clustering.

Finally, Tory et al. [40] present an interface based on parallel coordinates for exploring volume visualizations. Visualization parameters such as view and transfer function settings are shown by small icons within the parallel coordinates. A history bar depicts the resulting images and connects them with a polyline to the corresponding visualization settings. Additionally, images resulting from different settings for two parameters can be compared in a table view. *Cupid* extends and improves this work in a different application area. While their parallel coordinates style interface shows only a few selected parameter combinations, our approach is designed to depict a larger number of parameter settings using hierarchical clustering and illustrative techniques. Moreover, we directly nest the resulting 3D shapes within

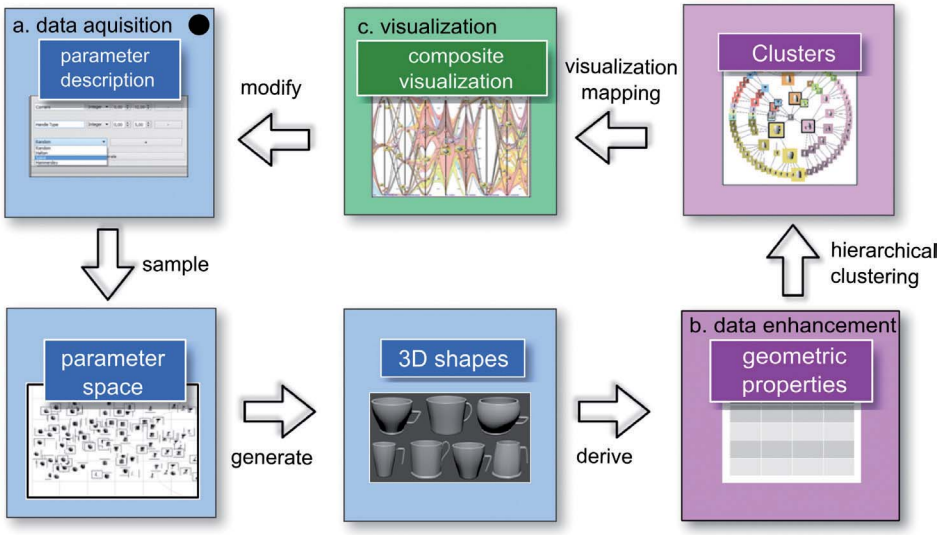

Fig. 2. Overview of our system for exploring the results of a cup generator: (a) The user determines the parameters of the geometry generator and the sampling strategy. The parameter space is then sampled and the corresponding 3D shapes are generated. (b) Measures describing geometric properties of the 3D shapes are derived and the 3D shapes are clustered based on their similarity. (c) The resulting clusters are visualized in a radial tree and a composite visualization that shows both the abstract parameter space of the geometry generator and the generated 3D shapes.

the parallel coordinates, instead of showing them in an additional history bar. Our approach is also more result-driven where we focus on finding similar 3D shapes and the corresponding parameter settings.

3 Overview of Cupid

Our main goal is to explore the relations between the abstract parameter space of a geometry generator and the generated 3D shapes. While other approaches show these two domains side-by-side using linked views and brushing, our idea is to combine both into a single *composite visualization* [18]. We use parallel coordinates for depicting the multivariate relations of the parameter space. The naïve approach of showing each parameter combination together with the resulting 3D shape in the parallel coordinates would lead to visual clutter. Also, it would be cumbersome for the user to manually search for similar 3D shapes and to examine their relations to the parameters (compare to task 1). Therefore, we first apply hierarchical clustering to group similar 3D shapes, which allows the user to study shape variations at different hierarchy levels.

According to the information seeking mantra [37], we first show the clusters at the top hierarchy level (overview first). The user can selectively drill-down into sub-clusters to study shape differences and inspect details-on-demand. The clusters are depicted using *illustrative parallel coordinates* [28]. We augment this abstract visualization of the parameter space by *nested icons* that represent the 3D shape or other properties of each cluster. This composite visualization enables the user to directly relate the parameter space to the corresponding clusters of similar 3D shapes, and vice versa (compare to tasks 1 and 3). For understanding the hierarchy of clusters, we adapt a linked *radial tree* to display the same icons representing the individual clusters. Here, the user can interactively select, split and merge clusters.

As an application example, we explore the results of a cup generator that is used for testing the object recognition capabilities of a domestic robot. The parameter space consists of eleven parameters (see Table 1). Certain parameters define parts of the cup such as handle type or number of corners, and other parameters represent global modifiers of the shape such as convexity.

A conceptional overview of *Cupid* is depicted in Fig. 2. The user starts with determining the parameters of the geometry generator and choses a sampling strategy for the parameter space. The parameter space is then sampled in order to generate the 3D shapes. We also derive certain measures that describe quantitative properties of the 3D shapes (e.g., surface area or convexity), which can be used in the visual exploration. The 3D shapes are hierarchically clustered based on their shape similarity. The resulting clusters are displayed in the radial tree as well as the composite parallel coordinates.

Table 1. Parameters of the cup generator.

Parameter name	Description	Type
number of corners	roundness of cup	discrete
convexity side	convexity from side-view	continuous
convexity top	convexity from top-view	continuous
bottom width	width of cup at bottom	continuous
center width	width of cup at center	continuous
top width	width of cup at top	continuous
bottom type	type of bottom	categorical
bulge at top	if true, then a bulge is created	categorical
size of bulge	size of the bulge	continuous
handle type	type of handle (e.g., round, open, closed)	categorical
handle size	size of handle	discrete

3.1 Data Generation and Pre-Processing

Initially, the user defines the parameters of the geometry generator and chooses a sampling algorithm (see Fig. 2a). Each parameter is described by a name, a type (categorical or continuous), and the range to be sampled. As shown in Table 1, the cup generator has attributes such as handle type, convexity, and parameters describing the overall shape (e.g., width of the cup at the top, middle and bottom). Additionally, the user defines the number of samples and selects the sampling method. We provide random sampling and low-discrepancy sampling. The advantage of using low-discrepancy sequences is a uniform sampling of the entire parameter space as opposed to random sampling [21].

3.2 Coregistration and Hierarchical Clustering

We use clustering to group the created 3D shapes according to their shape similarity. This helps to reduce visual clutter in the visualization and supports the user in finding regions in the parameter space that result in similar geometric shapes (compare to tasks 1 and 3). Since the 3D shapes are generated with different alignment, we have to coregister them first. We then use agglomerative hierarchical clustering to create a hierarchy of clusters, where clusters of similar 3D shapes are merged as one moves up the hierarchy. This has the advantage that initially only a few clusters need to be displayed, and the user can drill-down into selected sub-clusters.

To align the generated 3D shapes, we use the *iterative closest point* (ICP) algorithm which calculates a transformation T (translation and rotation) that minimizes the difference between the vertices of two

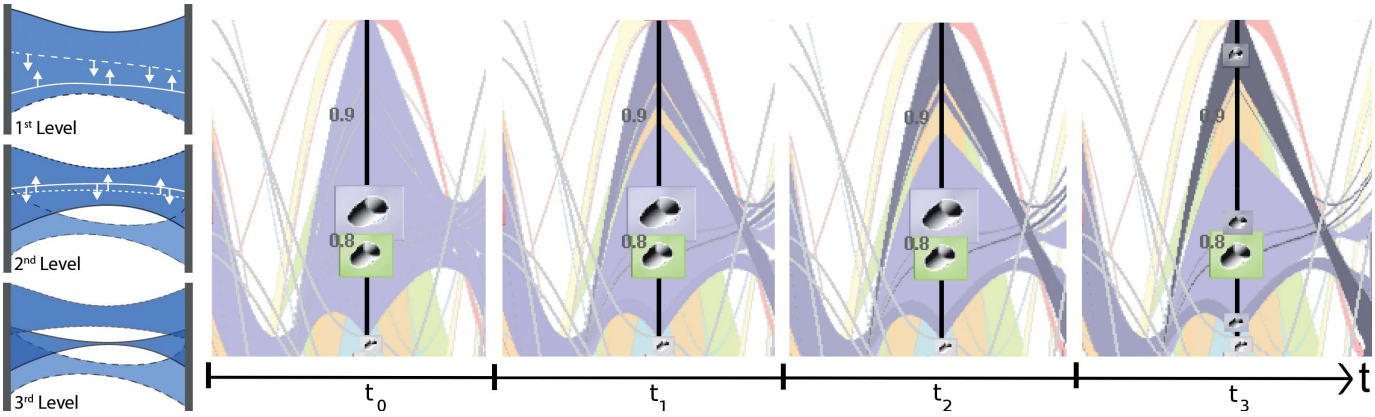

Fig. 3. Animated transition between two levels of the cluster hierarchy: The geometry and color from the parent are morphed to the child clusters. Coloring with different luminance enables the user to discriminate between child clusters and ensures consistency during interaction.

geometric meshes [43]. There are many approaches for computing mesh similarities. Due to its simplicity, we chose to use the sum of Euclidean distances between each vertex x_i in the mesh M_1 and its nearest neighbor in the mesh M_2, which can be computed rapidly using kD-trees:

$$d(M_1,M_2) = \frac{1}{|M_1|}\sum_{x_i \in M_1} \min_{x_j \in M_2} |x_i - x_j|$$

If smaller parts of the 3D shapes are not aligned (e.g., the handles of two similar cups have different positions), however, the measure can result in a large difference. Since our cup generator creates an identifier for each part of the cup, we align the individual parts first and compute the similarity based on the aligned parts of the 3D shapes:

$$d^* = \frac{1}{\sum_i w_i}\Big(\sum_i w_i \cdot \max(d(\text{part}_{i,1},\text{part}_{i,2}),\, d(\text{part}_{i,2},\text{part}_{i,1}))\Big),$$

where $\text{part}_{i,k}$ is the i^{th} part of mesh k and w_i is the corresponding weight. Accordingly, more important parts of the geometry can have higher influence on the similarity than less important ones. If a part of the shape is missing, we add a penalty value to the similarity value. Note that the idea of using parts of a 3D shape is not limited to our cup generator. For example, fractal vegetation generators often have symbols and predicates, which identify the parts of the geometry [32].

Our hierarchical clustering is based on DBSCAN [10] which is a popular density-based clustering algorithm. Areas with higher density (similarity) than the rest of the data form arbitrarily shaped clusters. Objects in sparse areas that separate the clusters are considered to be noise. Advantages of this algorithm are that it does not require a predefined number of clusters and it can identify arbitrarily shaped clusters of similar 3D geometries. Additionally, it can be easily adapted to create a hierarchy of clusters. DBSCAN has two parameters, namely a similarity threshold *st* and a minimum number of cluster members *minS* in order to be considered a dense area. Starting from an arbitrary 3D shape created by the geometry generator, the 3D shapes with a similarity d^* below *st* are queried. If the number of similar 3D shapes exceeds *minS*, the region is considered sufficiently dense and the similar 3D shapes are used as seeds for growing the cluster. 3D shapes located in non-dense regions are classified as noise.

Since the number of clusters resulting from DBSCAN can be very large—depending on the dataset and the parameter setting of the algorithm—, we extend our approach to hierarchical clustering (compare to Fua et al. [11]). The idea is to group similar clusters to reduce the number of clusters to be displayed. During the visual exploration, the user can then interactively drill-down into selected sub-clusters for a detailed inspection. The initial result of DBSCAN already provides us the leaf nodes in our hierarchical structure of clusters. In order to merge clusters, we iteratively apply DBSCAN where the similarity threshold *st* is increased by Δst (the resulting hierarchy is shown in the radial tree in Fig. 1b). In order to keep our approach simple and usable, we only display three hierarchy levels of the clustering. Since we precompute the similarities between each pair of 3D shapes, the user can interactively modify the parameters *st*, Δst and *minS*, and inspect the resulting hierarchy in the radial tree.

4 Visual Composition of Abstract & Spatial Data

After computing the hierarchical clustering, the data are visualized using an illustrative parallel coordinates approach and a linked radial tree. In the following, we describe a composite visualization that combines both abstract and spatial data. We then describe our linked radial tree, and finally the icons, which are the basic items for nesting the spatial information within the visualization and steering the views.

4.1 Composite Parallel Coordinates

For exploring the parameter space of the geometry generator, we use a novel cluster-based parallel coordinates approach with nested spatial information (compare to the *nest* design pattern by Javed and Elmqvist [18]). The basic items of the visualization are the clusters resulting from the hierarchical clustering, which are represented as nested icons. The icons show the 3D shapes or other properties of the cluster representatives. Additionally, we use illustrative techniques to enhance the visual representation of the clusters.

Illustrative techniques can enhance the readability of parallel coordinates [28]. Each cluster is represented by a colored polygon that shows the extend of the cluster. Rendering just the convex hull of the polygons would not provide much insight into the underlying data. Therefore, we use the branching technique from McDonnell and Mueller [28], which enables the user to control how densely or sparsely the data are distributed in the parallel coordinates (see Fig. 1a). We apply edge bundling to enhance the visual representation [16, 28]. The user can control the bundling factor, which reduces overlapping between clusters and decreases the amount of screen space occupied by a cluster. Moreover, individual 3D shapes that have been classified as noise are represented as B-splines with gray icons.

The cluster hierarchy is used for color assignment. Each cluster at the top level of the hierarchy gets a different color using a qualitative color map [13]. If the user drills-down into a cluster of interest, the sub-clusters get the same color but with a different luminance. This ensures consistency during the visual exploration, and the difference in luminance supports the discrimination between different sub-clusters. In order to preserve the mental map of the user, we perform an animated transition that morphs a selected parent cluster into its sub-clusters (see Fig. 3). For computing the animated transition, a unique ID is assigned to the vertices of each cluster. Using the ID, we can find the vertices of the sub-clusters that are located within their parent cluster. We use linear interpolation to change the position of the vertices during the animation as illustrated in Fig. 3 (left).

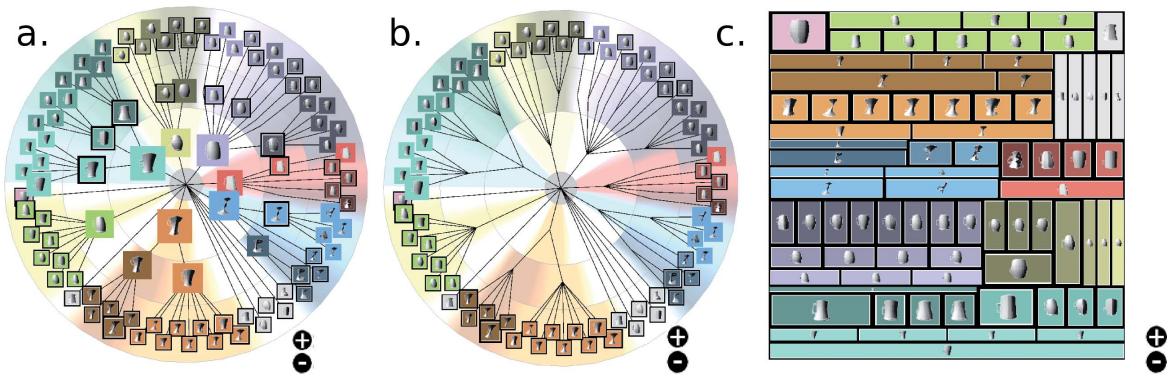

Fig. 4. Different representations of the cluster hierarchy: (a) radial tree, (b) circular dendrogram, (c) treemap.

In addition to the parameter values, each cluster is represented by nested icons that depict the 3D shape or other properties of the cluster (see Sec. 4.3). The size (area) of an icon represents the corresponding number of cluster members. The user can set a minimum and maximum size for the icons to avoid overlap or too small icons. For each parameter of the generator, the icons are vertically placed at the center of the corresponding clusters. If two icons overlap, they are moved away from each other. We place larger icons first, before placing smaller ones. The icons are used for steering the visualization. Mouse-over an icon highlights the corresponding cluster, which is then drawn in front of all other clusters and shown with a higher opacity (see the purple cluster in Fig. 1a).

Cupid also provides a *detail window* for comparing 3D shapes within and across clusters. For example, the user can click on the icon of a top-level cluster, which then opens a detail window and shows the 3D shapes of the corresponding sub-clusters (see Fig. 1c). Alternatively, a line brush [23] can be used to directly select currently invisible 3D shapes, where the corresponding B-splines (also not shown) intersect with a simple line segment drawn in the view (e.g., see Fig. 11c). Within the detail window, the selected 3D shapes are first sorted according to their cluster membership at the top level, before sorting them according to their similarity within the sub-clusters. The color of the clusters at the bottom level is thereby used as background color.

4.2 Composite Radial Tree

We apply hierarchical clustering to reduce visual clutter and to identify similar 3D shapes at multiple hierarchy levels. For exploring and modifying the results of the clustering, we adapt a radial tree which is a popular technique for representing hierarchical data (see Schultz et al. [36] for an overview). The radial tree is a common node-link tree layout with a transformation to polar coordinates. The advantage of radial transformations is a better usage of space for trees with few hierarchy levels but many nodes at the bottom. We replace the nodes of the radial tree with icons that represent the clusters (see Sec. 4.3). While drawing the radial tree, we check for overlapping of icons. If two icons overlap, one is shifted. Additionally, we draw circles in the background to discriminate between the different hierarchy levels. The user can compare the 3D shapes of different cluster members and manually split or merge clusters in the visualization.

We also evaluate other hierarchical representations like dendrograms and treemaps (see Fig. 4). Dendrograms are a typical technique for visualizing hierarchical clustering. A circular dendrogram is very similar to a radial tree but intermediate nodes are not drawn to avoid visual clutter (see Fig. 4b). However, we prefer the radial tree layout because the user can relate the intermediate nodes to the cluster representatives shown in the parallel coordinates, for example, using linking. Treemaps are another popular technique for representing hierarchical data. We implement a treemap using the strip tilling algorithm [3] which ensures good readability and preserves the ordering of the nodes. We replace the nested rectangles with icons (see Fig 4c). The advantage of a treemap is its compact layout, but the structure of the hierarchy is more difficult to read. In the evaluation, our domain experts rated the radial tree as the preferred choice for *Cupid*.

Our adapted radial tree can also be used to modify the result of the hierarchical clustering, where the user can manually split and merge clusters. In Fig. 5a, for example, the user clicks on the icons of two similar clusters (highlighted in red) and merges them by pressing the minus button (the result is shown in Fig. 5b). Clusters can be split as well by marking different 3D shapes in the radial tree, for example, using a detail window (see Fig. 5c) and by pressing the plus button. Since the ordering within a cluster does not provide information, the new cluster is simply added at the end of the sub-nodes in Fig. 5d. The user can also change the ordering of the sub-nodes by dragging.

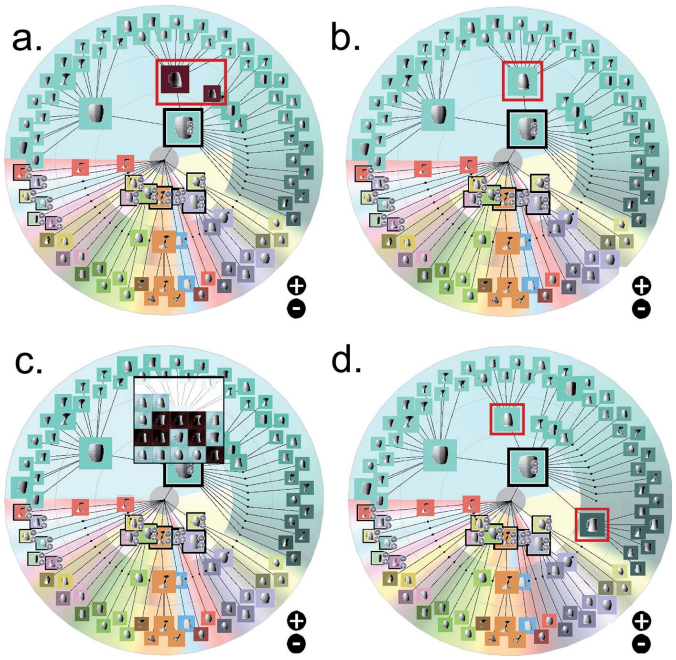

Fig. 5. Two similar clusters in (a) are merged in (b). Using a detail window (c), selected members of a cluster are split into a new cluster (d).

4.3 Nested Icons

We nest geometric representations within a view in *Cupid* using icons. The icons can either represent a cluster of similar 3D shapes or depict the individual cluster members at the bottom level of the hierarchy. The icon's background represents the color of the cluster, and the size/area indicates the number of cluster members. This consistent representation in both views ensures a good user experience. By default, the shaded geometry is shown from the same point of view set by the user (see Fig. 6a). For icons representing a cluster, we display the

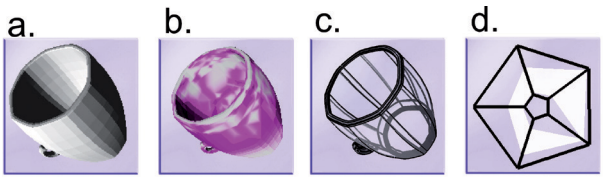

Fig. 6. Types of icons: (a) shaded geometry, (b) variability visualization, (c) sharp edges, and (d) starplots.

3D shape of the cluster member with the shortest Euclidean distance between its parameter values and the average values of the cluster. In the following, we describe three alternatives to the shaded geometry.

In some applications, the variability of the members within a cluster is in focus. For this purpose, we encode the variance between the nearest vertices of the cluster members to the cluster representative using a color map (see Fig. 6b). This enables the user to see constant and varying regions of the 3D shapes within a cluster. As another alternative representation, we offer an illustration of the object's silhouette which is very important for comparing 3D shapes. Following the idea of iWires [12], the "sharp edges" of the geometry are drawn to display just enough information to identify the 3D shape. Additionally, the shaded geometry is shown with low opacity (see Fig. 6c).

Finally, *Cupid* provides starplots [6, 8] as an additional type of icon that can depict derived geometric properties. The starplots are created analogous to parallel coordinates but with polar coordinates. To ensure good readability, we only show the convex hull of the line-paths for the cluster members. Since starplots are well suited for showing outliers, they enable the user to quickly find implausible cups (e.g., by looking at derived parameters such as shape stability, see Sec. 5).

Both the radial tree and the composite parallel coordinates are linked which enables the user to investigate clusters of similar 3D shapes across both views. The nested icons in both views provide the same operations which enables consistency during interaction. Clicking on an icon selects the related cluster, and mouse-over enlarges the related icons in both views and highlights the cluster in the composite parallel coordinates. Additionally, the user can drill-down into sub-clusters by clicking on the icon of an intermediate node. While the radial tree already displays the sub-clusters, an animated transition is performed in the composite parallel coordinates.

5 Geometric Properties

Cupid provides composite parallel coordinates and representations of the cluster hierarchy for visual exploration of similar 3D shapes and the corresponding parameters (compare to tasks 1 and 3). In addition, we provide derived geometric properties that help the user to identify interesting 3D shapes such as implausible or deformed cups (compare to task 2). The properties can be mapped to an axis in the parallel coordinates or shown in the nested icons using starplots.

5.1 Derived Geometric Properties

In the pre-processing stage (see Fig. 2b), we calculate the following geometric properties in addition to shape similarity. We use these properties for exploring the geometry generator's parameter space:

- Surface area $= \sum_{i=1}^{N_T} A(t_i)$, where $A(t_i)$ is the area of a triangle t_i and N_T is the number of triangles of a cup.
- Volume enclosed by the mesh V of a cup.
- Convexity $= V/C$, where V is the volume enclosed by the mesh and C is the volume enclosed by the convex hull.
- Shape stability $= \frac{\|d-b\|}{r}$, where d is the origin of the coordinate system of the 3D shape, b is the center of gravity of the shape, and r is the radius of the bounding sphere.
- Number of crossing faces.

We have chosen these properties because they enable the user to detect invalid 3D shapes (e.g., number of crossing faces larger than zero) and describe characteristics of cups such as shape stability or convexity. Finding good properties that describe 3D shapes is still an open problem. In order to deal with arbitrary geometries, we will include further properties such as the fractal dimension of shapes (used to evaluate plants) in future work.

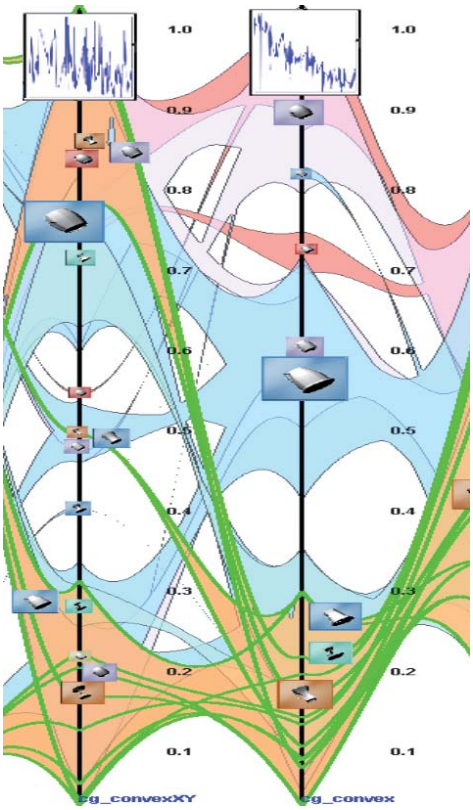

Fig. 7. A function plot in the parallel coordinates shows the derived geometric property "surface area" with respect to two parameters in order to identify possible correlations (like in the right case).

5.2 Visualization of Geometric Properties

The derived geometric properties can be represented by starplots or mapped to the axes of the parallel coordinates. The starplots can be enlarged in additional windows that are linked with the parallel coordinates. If the user selects a region of interest, the corresponding B-splines are shown in the parallel coordinates. This enables a result-driven exploration of the parameter space using linking and brushing.

Additionally, we add a function plot to each parameter (see Fig. 7). These plots show the function $f : x \rightarrow g$, where x is a parameter value of the geometry generator and g is a user-selected geometric property. In the function plot, we show the linear regression between the parameter value and the geometric property. This gives the user a hint about a possible correlation between the parameter and the geometric property and hides noise resulting from other parameters. Our function plots are similar to those in HyperSlice [41] at the diagonal positions of the matrix. Additionally, they added 2D slices for the visualization of pairs of parameters. Our approach provides a similar visualization using the B-splines of the parallel coordinates. The geometric properties can be mapped to coloring and opacity, which are specified with a user-defined transfer function (e.g, B-splines representing cups with small surface area are shown in green in Fig. 7). The user can also fade-in (or hide) the B-splines in order to prevent visual clutter.

6 Implementation

We implement *Cupid* in VolumeShop [5], which is a 3D visualization framework and rapid-prototyping toolkit. A key feature of VolumeShop is its modular design. All components like renderings and

Fig. 8. Some examples of detected categories: The cups vary in profile, roundness, handle, width and even more.

object loading can be extended by a plug-in mechanism. The plug-ins enable to adapt VolumeShop to the needs of the visualization. Additionally, the modular design of VolumeShop ensures a good separation between data, logic and rendering, according to the Model-View-Controller (MVC) pattern. A great support for implementing composite and linked visualization techniques makes VolumeShop an ideal visualization framework for *Cupid*.

The visualizations are implemented with C++, OpenGL and GLSL as a shader-language. This combination gives enough computational power to reduce the latency of our visualizations. Individual tasks can be outsourced into shader programs, which enables a better separation between logic and viewing code. The pre-processing operations are outsourced to external applications. This design enables us to easily replace pre-processing operations and test different preprocessing pipelines. Moreover, we use the ICP algorithm from the Point Cloud Library (PCL) for aligning shapes [34].

7 Results

In the introduction, we discuss several tasks for analyzing geometry generators. According to these tasks, we present results and the used workflow to solve the tasks with *Cupid* in this section. The first task is to find categories of similar 3D shapes and the corresponding parameter values (see task 1). We then present our workflow to find unwanted cups (see task 2). According to the third task, the last two sub-sections show how to detect parameters that strongly influence the shapes and how to find sensitive ranges of a parameter (see task 3).

7.1 Finding Similar 3D Shapes and Corresponding Parameter Values

The first task is to characterize the generated 3D shapes. We want to detect all variations of the shapes and to identify the corresponding parameter values. Some example 3D shapes are shown in Fig. 8.

Initially, the results from the hierarchical clustering are inspected in a radial tree. We can interactively adjust the parameters of the clustering to improve the results (e.g., having between five and nine clusters at the top hierarchy level). After this step, clusters can be split or merged manually. In Fig. 9a, for example, the 3D shapes within the top cluster look rather different from their parent and are thus split into separate clusters. In contrast, the sub-clusters in Fig. 9b look very similar to their parent. At the bottom hierarchy level, only very similar 3D shapes are grouped together using a low similarity threshold st. Furthermore, the user can quickly detect variations of shapes within a cluster using the starplot icon (not shown here).

After clustering the 3D shapes, the corresponding parameter values can be studied easily using the composite parallel coordinates. Since both views are linked, we only need to highlight (mouse-over) the selected icon. The corresponding cluster is then highlighted in the parallel coordinates. In the following, we use the hierarchical clustering to analyze the parameter space.

7.2 Finding Implausible or Unwanted 3D Shapes

The next task is to find unwanted or (physically) implausible cups. Such shapes would affect computer games or evaluation systems for computer vision and thus need to be identified. *Cupid* derives geometric properties to detect unwanted 3D shapes (see Sec. 5). Such cups can, for instance, have a high value for shape stability or a very small volume of the enclosing shape. Moreover, erroneous meshes can be detected using the number of crossing faces. By mapping the derived properties to the axes in the parallel coordinates or by using the starplot icon, the user can quickly identify such shapes.

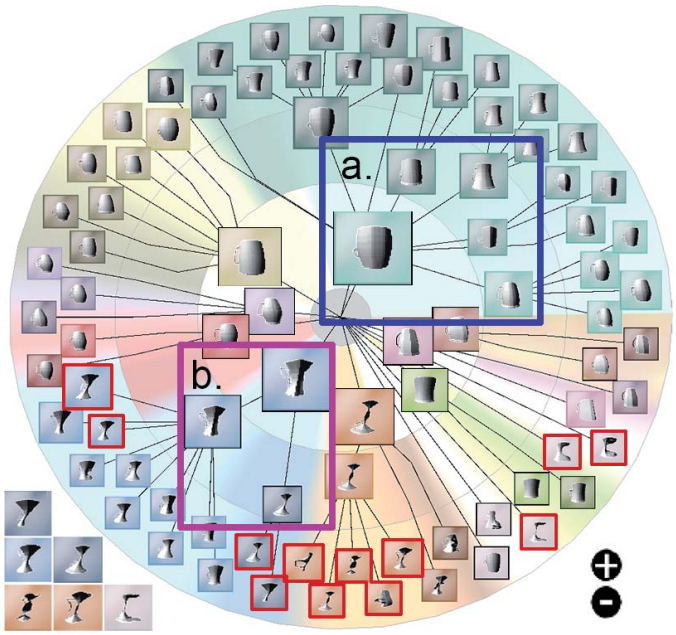

Fig. 9. Parent and child icons within the blue rectangle look different and are thus split into separate clusters. In contrast, the icons within the pink rectangle belong to the same category. Icons highlighted with a red rectangle are labeled as unwanted cups by the user (selected examples are enlarged).

While invalid cups can often be detected automatically, we are also interested in cases that are difficult to describe by derived properties (e.g., a cup that is deformed or looks like a beer mug). In such cases, the validity of a cup depends on semantic knowledge. For example, if a cup has a very small handle, some users classify it as unwanted and others as a cup with novel design. To handle such ambiguous cases, the user has to manually determine the validity of the cups. In the radial tree, we first inspect the icons located at the bottom level of the hierarchy. Since the icons are grouped by similarity, it is easy for the user to spot outliers. In Fig. 9, the cups with a red border have been labeled as unwanted by the user. The corresponding B-spline curves are drawn in the linked parallel coordinates, where we can investigate the respective parameter values. We can also change the view-point for the cups or use a detail window for further inspection.

7.3 Evaluating Parameter Influence

According to task 3, we study the influence of parameters in this subsection and identify sensitive regions in the parameter space in the next sub-section. The clustering provides information about the influence and sensitivity of the generator with respect to its parameters.

To determine the influence of parameters, we use the composite parallel coordinates and analyze the distribution of clusters of similar 3D shapes. If the polygon of a cluster covers only small portions of a parameter range, the parameter has high influence because small changes have a large effect on the resulting geometry (e.g., see the orange and turquoise clusters for “convexity side” in Fig. 10). In contrast, if large changes in the parameter value result in similar 3D shapes, the parameter has low influence. The individual polygons (or branches) of a cluster then cover large portions of the parameter range (see the orange and turquoise clusters for “center width” in Fig. 10).

Next, we investigate 3D shapes that have been classified as noise by the clustering. Many such shapes have been generated by small values for the parameter “number of corners.” We select those shapes using a line brush and inspect the cups in a detail window. As shown in Fig. 11c, the corresponding cups look very angular. Moreover, the clusters in this parameter range are very narrow, while the clusters for larger values of “number of corners” are larger. This is not surprising because changing from three to four corners has a higher influence on

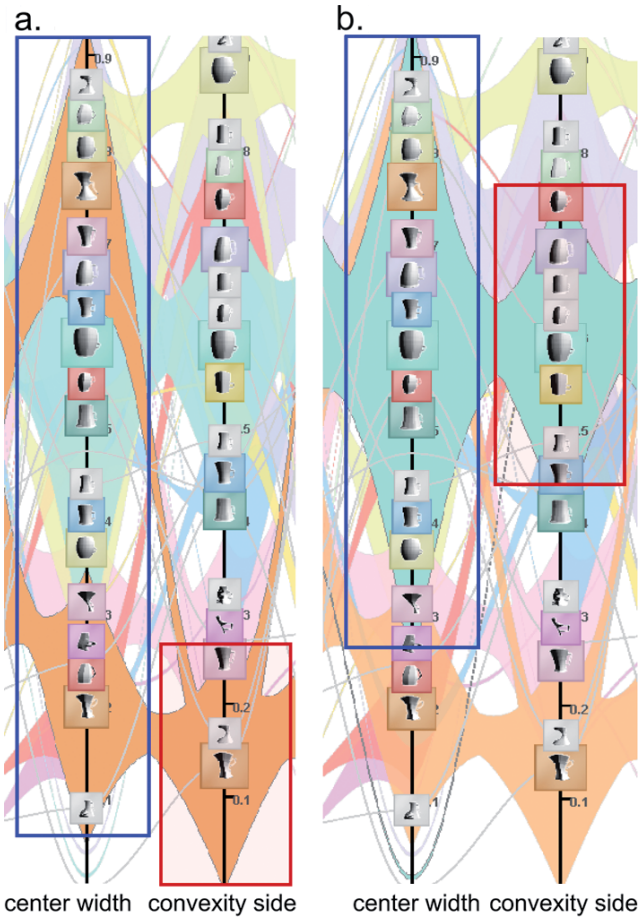

Fig. 10. Exploring the influence of parameters on the created cups: The orange (a) and turquoise (b) clusters cover large regions of the parameter "center width" but only a small region of "convexity side."

the generated 3D shape than changing from 30 to 31, for example. Using additional detail windows, we can investigate the influence of "number of corners" on the roundness of the generated shapes.

We analyze other parameters in a similar way. For example, the parameter "convexity side" also has a high influence on the resulting cups because the individual clusters cover only small parameter ranges and modify the profile of the shape (see Fig. 11b). In contrast, the parameter "center width" has low influence because it influences the shape in combination with "top width" and "bottom width." The respective clusters in Fig. 11a cover larger portions of the parameter range.

The derived geometric properties can also be used to evaluate the influence of a parameter. The idea is to find correlations between the parameter of the generator and the derived geometric properties using the function plots (e.g., surface area or shape stability). If such a correlation is found, the parameter has a high influence on the selected geometric property (and on the result). Such an example is shown in Fig. 7, where the parameter "surface area" influences the derived convexity of the shape. The function includes some noise because also other parameters affect the convexity.

7.4 Sensitivity Analysis of Parameter Regions

As discussed in the introduction, a sensitive region of a parameter affects the resulting 3D shapes more than other regions of the same parameter. A good starting point for finding sensitive regions is to look for narrow clusters or noise. The "number of corners" is such an example that has been discussed already (see Fig. 11c).

The main challenge of finding a sensitive region is to identify whether the number of clusters within a (small) region depends on the current parameter or its interplay with other parameters. In Fig. 11c,

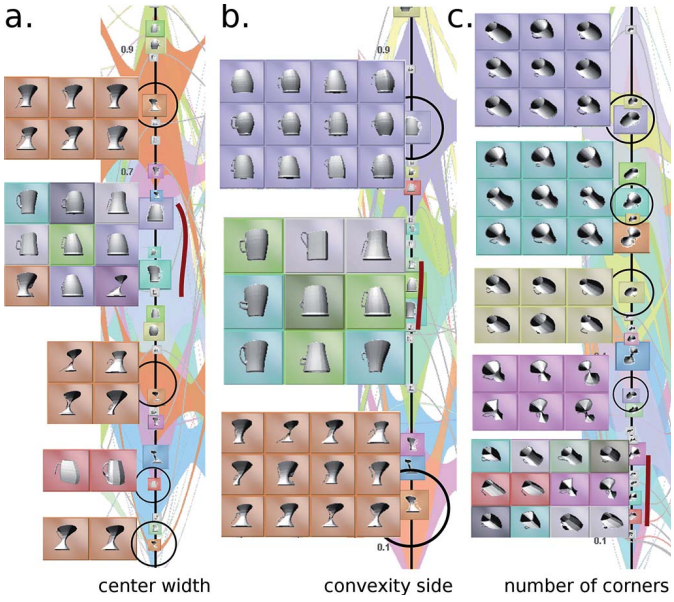

Fig. 11. Comparison of selected 3D shapes using a detail window: (a) The orange cluster covers a large portion of the parameter value with similar 3D shapes. (b) The parameter has a high influence on the resulting shapes, since clusters are rather narrow. (c) Changes in low values of "number of corners" already create different 3D shapes, while changes in large values produce rather similar cups.

for example, several clusters with round cups but different silhouettes can be seen. The difference results from other parameters such as "handle type" and "convexity side." In contrast, the clusters with lower values for "number of corners" have different roundness, where variations stem mainly from the parameter itself. The corresponding region has a high sensitivity.

Another example of a parameter with high influence is "convexity side." To identify sensitive regions, we analyze the parameter in the same way as the "number of corners." The parameter varies the profile of the 3D shape (see Fig. 11b). Low convexity values result in concave cups and high convexity values result in convex cups. While the cups at the top and bottom (mainly) differ according to the amount of convexity/concavity, a significant change is found in the middle. The clusters in the middle contain straight cups (neither convex nor concave), the clusters above and below differ according to the convexity.

8 Evaluation

Our approach is evaluated by three domain experts for geometry generators and evaluation systems for computer vision. The domain experts have to solve tasks similar to those described in the introduction. For the evaluation, we sample the cup generator with 100 cups using random sampling.

At the beginning of the evaluation, the main functionality is presented to the domain experts, followed by a live presentation. Then, the domain experts have to solve different user tasks. They can ask questions if something is ambiguous or not well traceable. After solving a task, the domain experts report the degree of difficulty and guidance provided by *Cupid*. If the task is not solved successfully, potential problems of the workflow are discussed to identify the weaknesses of our system. After solving all tasks, the domain experts grade the used techniques according to usefulness and usability.

8.1 User Tasks

Our evaluation focuses primarily on the workflows and not the used visualization techniques. For example, we are interested if the coloring of the clusters is sufficient to solve our tasks. The domain experts have to solve the following tasks:

User task 1: Evaluation of the parameter space and geometric result. The first task is to give an overview which groups of cups are generated. Then, the domain experts have to find the associated parameter regions. We test several tree layouts. Additionally, the domain expert have to explore the hierarchical clustering and rate the similarity of objects within a cluster.

User task 2: Finding implausible cups. At first, the user needs to find examples of physically implausible or unwanted cups. Physical implausible cups result from big handles or very small bodies, for example. Also meshes with errors occur (e.g., self-intersections of the surface). The domain experts have to find examples for both errors and find the corresponding parameter values.

User task 3: Determine the influence and sensitivity of parameters. In tasks 1 and 2 the generated 3D shapes are in focus. Task 3 requires the analysis of the parameter space to determine the influence and sensitivity of a parameter. Three different parameters with different influences are selected. The domain experts have to determine the effect of each parameter on the geometry and sort the parameters according to their influence. The parameter with the largest influence is then used for a sensitivity analysis, i.e., the experts have to look for regions where the resulting shapes have a high or a low variation.

Finally, we asked the experts about the difficulty to solve the tasks and about the usefulness of the visualization and interaction methods.

8.2 Feedback from Domain Experts

The domain experts like the combination of the parallel coordinates with the radial tree. One of them especially values that *Cupid* allows to deal with a larger number of parameters at a time. Due to our approach, they can now detect interrelationships between parameters of the geometry generator rather quickly and identify sensitive parameter ranges (i.e., where a slight change has a large influence on the created 3D shape). During the evaluation, the domain experts use the radial tree to get an overview of the clusters, while the parallel coordinates are used for inspecting details. They see this combination as an intuitive way of exploring the geometry generator and rate this combination as excellent. Including the spatial information into the abstract representation of the parameter space supports the easy exploration of the generated shapes. All domain experts were able to assign the parameters to the resulting 3D shapes, and vice versa.

The first task of finding a group of similar shapes is easily solved by all domain experts. They explore the data using the radial tree as described in Sec. 7.1. The domain experts detect several categories of 3D shapes and inspect the corresponding parameter values with linking between the radial tree and the parallel coordinates. While the initial clustering represents a good starting point for the exploration, some cups were wrongly assigned. This is because our similarity metric works well for very similar shapes, but it has problems with less similar shapes. In such cases, the domain experts interactively modify the clustering using the merging and splitting tool. They rate the task with a low degree of difficulty and a good support by *Cupid*.

While solving task 1, the domain experts find some examples of implausible cups (see task 2). For finding more implausible cups, they navigate through all clusters at the top or middle hierarchy level of the radial plot and inspect their members. This can be done fast because the cups within a cluster have similar characteristics. In contrast, the starplots and function plots depicting derived geometric properties are not used primarily. One domain expert explains that the properties are not intuitive. After a discussion, the domain experts think that they are a useful extension for specific tasks. For example, finding physical implausible cups can be done with the shape stability property.

For evaluating the influence of a parameter on the result, the domain experts use the parallel coordinates. They look for clusters that only cover a small portion of the parameter range (see Sec. 7.3). The effect of each parameter is explored using the line brush. The domain experts test different regions of the parameter space and analyze the variation of shapes in a detail window. They like this integration of geometry into the parallel coordinates and rate it as good. Furthermore, they use the function plots and test several derived geometric properties. Since the domain experts do not find reasonable geometric properties, they select the properties by guessing. They rate the function plots as satisfactory. In summary, the domain experts rate the task of evaluating the parameter influence with an average degree of difficulty.

The domain experts denote the analysis of the sensitivity as the most difficult task. This task requires to analyze a combination of parameters because clusters can be affected by several parameters (see Sec. 7.4). The domain experts explore various regions and study the variations using line brushes. Additionally, the clustering is also used for a sensitivity analysis. The domain experts identify several regions with different sensitivity. They rate this task with a high degree of difficulty because the relationships between parameters and 3D shapes are difficult to understand. Furthermore, the domain experts ask for methods to test own parameter combinations and to vary the sampling rate interactively. Testing own parameter combinations could enable a new workflow to vary only one parameter and set other parameters to a default value. Varying the sampling interactively can be used to exclude uninteresting regions of the parameter space. We will investigate this further in our future work.

At the end of the evaluation, an interview about the techniques provided by *Cupid* is done. The domain experts appreciate the interaction techniques like linking and brushing. The integration of clustering into the parallel coordinates using illustrative methods is also rated between good and excellent. The domain experts appreciate the possibilities provided by *Cupid*. They rate the integration of geometry between good and excellent and like the different variations of icons. The methods using derived geometric properties are rated lower because the workflow is not so intuitive.

9 Conclusion and Future Work

We present a novel visualization approach for exploring the parameter space of a geometry generator. Our goal is to support a quick exploration of the generated 3D shapes within their parameter space (task 1). The parameter regions of implausible or unwanted objects are easily detected by the user (task 2). Moreover, the sensitivity analysis of the parameter space is an important and difficult task (task 3). Our solution is a new composite parallel coordinates visualization which combines both the abstract parameter space of the generator and the resulting cups into the same visualization. This combination allows the user to study complex relations between both domains. To reduce visual clutter and to find similar 3D shapes, we use hierarchical clustering and an illustrative approach. Our technique supports level-of-details by controlling the similarity of the objects and the details of the clustering. For reducing visual clutter, we use several techniques like edge bundling, hierarchical clustering, and different cluster styles. Additionally, a linked radial tree layout is used to analyze or modify the hierarchy of the clustering. For quickly finding implausible objects, we derive geometric properties which are also helpful for a sensitivity analysis of the parameter regions and their degree of influence.

The feedback from the domain experts is positive. They see the demand for this tool and want to use it in their current workflow. However, the sensitivity task could not be solved sufficiently because some features like visual steering were missing. Therefore, future work will focus on these features and on the exploration of other geometry generators like an airplane generator. For this application, the preprocessing steps need to be evaluated to address the new domain areas and to ensure a quick response of the steering. Finally, we want to further evaluate the different components and possibilities of our approach in a detailed user study.

Acknowledgments

The authors wish to thank Oliver Zendel and Markus Murschitz from the Austrian Institute of Technology (AIT) for evaluating our application and Prof. Jan Koenderink for his expertise on the visual perception of 3D shapes. We also thank the 2Vis1Cup infrastructure at the ICGA, Vienna University of Technology. Parts of this work were supported by the Austrian Science Fund (FWF) in scope of the projects Nr. P21695 (ViMaL) and Nr. P24597-N23 (VISAR) as well as the European Union under the ERC Advanced Grant 291372: SaferVis – Uncertainty Visualization for Reliable Data Discovery.

REFERENCES

[1] J.-P. Balabanian, I. Viola, and M. E. Gröller. Interactive illustrative visualization of hierarchical volume data. In *Graphics Interface*, pages 137–144, 2010.

[2] R. Becker and W. Cleveland. Brushing scatterplots. *Technometrics*, 29(2):127–142, 1987.

[3] B. B. Bederson, B. Shneiderman, and M. Wattenberg. Ordered and quantum treemaps: Making effective use of 2D space to display hierarchies. *ACM Trans. Graph.*, 21(4):833–854, 2002.

[4] W. Berger, H. Piringer, P. Filzmoser, and M. E. Gröller. Uncertainty-aware exploration of continuous parameter spaces using multivariate prediction. *Comput. Graph. Forum*, 30(3):911–920, 2011.

[5] S. Bruckner and M. E. Gröller. VolumeShop: An interactive system for direct volume illustration. In *Proc. IEEE Conf. Visualization*, pages 671–678, 2005.

[6] S. Bruckner and T. Möller. Result-driven exploration of simulation parameter spaces for visual effects design. *IEEE Trans. Vis. Comput. Graph.*, 16(6):1468–1476, 2010.

[7] S. Busking, C. P. Botha, and F. H. Post. Dynamic multi-view exploration of shape spaces. *Comput. Graph. Forum*, 29(3):973–982, 2010.

[8] J. Chambers, W. Cleveland, B. Kleiner, and P. Tukey. *Graphical Methods for Data Analysis*. Chapman and Hall, 1983.

[9] D. Coffey, C.-L. Lin, A. G. Erdman, and D. F. Keefe. Design by dragging: An interface for creative forward and inverse design with simulation ensembles. *IEEE Trans. Vis. Comput. Graph.*, 19(12):2783–2791, 2013.

[10] M. Ester, H. P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. Knowledge Discovery and Data Mining*, pages 226–231, 1996.

[11] Y.-H. Fua, M. O. Ward, and E. A. Rundensteiner. Hierarchical parallel coordinates for exploration of large datasets. In *Proc. IEEE Conf. Visualization*, pages 43–50, 1999.

[12] R. Gal, O. Sorkine, N. J. Mitra, and D. Cohen-Or. iWIRES: An analyze-and-edit approach to shape manipulation. *ACM Trans. Graph.*, 28(3):33:1–33:10, 2009.

[13] M. Harrower and C. A. Brewer. ColorBrewer.org: An online tool for selecting colour schemes for maps. *The Cartographic Journal*, 40(1):27–37, 2003.

[14] J. Heinrich, Y. Luo, A. E. Kirkpatrick, H. Zhang, and D. Weiskopf. Evaluation of a bundling technique for parallel coordinates. *Proc. Int'l. Conf. Information Visualization Theory and Applications (IVAPP)*, pages 594–602, 2011.

[15] J. Heinrich and D. Weiskopf. State of the art of parallel coordinates. In *Eurographics 2013 State of the Art Reports*, pages 95–116, 2013.

[16] D. Holten. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE Trans. Vis. Comput. Graph.*, 12(5):741–748, 2006.

[17] A. Inselberg. The plane with parallel coordinates. *The Visual Computer*, 1(2):69–91, 1985.

[18] W. Javed and N. Elmqvist. Exploring the design space of composite visualization. In *IEEE Pacific Visualization Symp.*, pages 1–8, 2012.

[19] J. Kehrer and H. Hauser. Visualization and visual analysis of multifaceted scientific data: A survey. *IEEE Trans. Vis. Comput. Graph.*, 19(3):495–513, 2013.

[20] J. Kehrer, P. Muigg, H. Doleisch, and H. Hauser. Interactive visual analysis of heterogeneous scientific data across an interface. *IEEE Trans. Vis. Comput. Graph.*, 17(7):934–946, 2011.

[21] A. Keller. The fast calculation of form factors using low discrepancy sequences. In *Proc. Spring Conf. Comput. Graph. (SCCG)*, pages 195–204, 1996.

[22] J. Kniss, G. Kindlmann, and C.Hansen. Multi-dimensional transfer functions for interactive volume rendering. *IEEE Trans. Vis. Comput. Graph.*, 8(3):270–285, 2002.

[23] Z. Konyha, K. Matković, D. Gračanin, M. Jelović, and H. Hauser. Interactive visual analysis of families of function graphs. *IEEE Trans. Vis. Comput. Graph.*, 12(6):1373–1385, 2006.

[24] J. P. Lewis, R. Rosenholtz, N. Fong, and U. Neumann. VisualIDs: Automatic distinctive icons for desktop interfaces. *ACM Trans. Graph.*, 23(3):416–423, 2004.

[25] A. Lex, M. Streit, C. Partl, K. Kashofer, and D. Schmalstieg. Comparative analysis of multidimensional, quantitative data. *IEEE Trans. Vis. Comput. Graph.*, 16(6):1027–1035, 2010.

[26] J. Marks et al. Design galleries: A general approach to setting parameters for computer graphics and animation. In *Proc. SIGGRAPH*, pages 389–400, 1997.

[27] K. Matković, D. Gračanin, B. Klarin, and H. Hauser. Interactive visual analysis of complex scientific data as families of data surfaces. *IEEE Trans. Vis. Comput. Graph.*, 15(6):1351–1358, 2009.

[28] K. T. McDonnell and K. Mueller. Illustrative parallel coordinates. *Comput. Graph. Forum*, 27(3):1031–1038, 2008.

[29] M. Ovsjanikov, W. Li, L. Guibas, and N. J. Mitra. Exploration of continuous variability in collections of 3D shapes. *ACM Trans. Graph.*, 30(4):33:1–33:10, 2011.

[30] H. Piringer, W. Berger, and J. Krasser. HyperMoVal: Interactive visual validation of regression models for real-time simulation. *Comput. Graph. Forum*, 29(3):983–992, 2010.

[31] H. Piringer, S. Pajer, W. Berger, and H. Teichmann. Comparative visual analysis of 2D function ensembles. *Comput. Graph. Forum*, 31(3):1195–1204, 2012.

[32] P. Prusinkiewicz and A. Lindenmayer. *The Algorithmic Beauty of Plants*. Springer, 1996.

[33] J. C. Roberts. State of the art: Coordinated & multiple views in exploratory visualization. In *Proc. Coordinated & Multiple Views in Exploratory Visualization (CMV)*, pages 61–71, 2007.

[34] R. Rusu and S. Cousins. 3D is here: Point cloud library (PCL). In *Proc. IEEE Int'l. Conf. Robotics and Automation (ICRA)*, pages 1–4, 2011.

[35] J. Schmidt, M. E. Gröller, and S. Bruckner. VAICo: Visual analysis for image comparison. *IEEE Trans. Vis. Comput. Graph.*, 19(12):2090–2099, 2013.

[36] H. Schulz, S. Hadlak, and H. Schumann. The design space of implicit hierarchy visualization: A survey. *IEEE Trans. Vis. Comput. Graph.*, 17(4):393–411, 2011.

[37] B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Proc. IEEE Symp. Visual Languages*, pages 336–343, 1996.

[38] R. C. Smith, R. Pawlicki, I. Kókai, J. Finger, and T. Vetter. Navigating in a shape space of registered models. *IEEE Trans. Vis. Comput. Graph.*, 13(6):1552–1559, 2007.

[39] J. O. Talton, D. Gibson, L. Yang, P. Hanrahan, and V. Koltun. Exploratory modeling with collaborative design spaces. *ACM Trans. Graph.*, 28(5):167:1–167:10, 2009.

[40] M. Tory, S. Potts, and T. Möller. A parallel coordinates style interface for exploratory volume visualization. *IEEE Trans. Vis. Comput. Graph.*, 11(1):71–80, 2005.

[41] J. J. van Wijk and R. van Liere. HyperSlice: Visualization of scalar functions of many variables. In *Proc. IEEE Conf. Visualization*, pages 119–125, 1993.

[42] O. Zendel, W. Herzner, and M. Murschitz. VITRO – model based vision testing for robustness. *Proc. Int'l. Symp. Robotics (ISR)*, pages 24–26, 2013.

[43] Z. Zhang. Iterative point matching for registration of free-form curves and surfaces. *Int. J. Comput. Vision*, 13(2):119–152, 1994.