Relation-Based Parametrization and Exploration of Shape Collections

Kurt Leimer, Lukas Gersthofer, Michael Wimmer, Przemyslaw Musialski

TU Wien

Abstract

With online repositories for 3D models like 3D Warehouse becoming more prevalent and growing ever larger, new possibilities have emerged for both experienced and inexperienced users. These large collections of shapes can provide inspiration for designers or make it possible to synthesize new shapes by combining different parts from already existing shapes, which can be both easy to learn and a fast way of creating new shapes. But exploring large shape collections or searching for particular kinds of shapes can be difficult and time-consuming tasks as well, especially considering that online repositories are often disorganized. In our work, we propose a relation-based way to parametrize shape collections, allowing the user to explore the entire set of shapes based on the variability of spatial arrangements between pairs of parts. The way in which shapes differ from each other is captured automatically, resulting in a small number of exploration parameters. Furthermore, a copy-and-paste system for parts allows the user to change the structure of a shape, making it possible to explore the entire collection from any initial shape.

Keywords: 3D database exploration, model variability, shape analysis, shape collections

1. Introduction

High-quality 3D geometric models are needed in many industries, such as entertainment, computer-aided design, urban planning, the fabrication of physical objects, as well as research areas like medical or scientific visualizations and simulations. Manually creating a 3D model not only requires a lot of artistic, but also technical skills, as traditional modeling software often contain hundreds of different tools that all take significant effort to learn how to be used properly.

However, with online repositories for 3D models like 3D Warehouse or Turbosquid becoming more prevalent and growing ever larger, new possibilities have emerged for both experienced and inexperienced users. The availability of large collections of shapes allows users to quickly populate virtual scenes with a multitude of different objects. Expert designers can draw inspiration from the large variety of different shapes and creating novel shapes can be made easier through shape synthesis, where parts of existing models are combined to create a new one, for example by interactive approaches like modeling-by-example [1] or structural blending [2], or even by algorithms that automatically generate novel shapes that in turn can also serve as additional inspiration for the user [3]. On the other hand, these online repositories are often disorganized, with thousands of shapes listed under the same category and no meaningful way to distinguish between differently looking shapes of the same family. This makes it a difficult task to look for specific shapes or explore the collection in a meaningful manner.

Shape retrieval is one proposed solution to this problem [4, 5, 6, 7]. From a high-level point of view, given an existing shape or a sketch provided by the user, a number of shapes that are similar to the input are retrieved from the collection based on some distance measure. Such a method can be useful when the user already has a specific shape in mind, but is less suited for exploration where one might not even know what kind of shapes are available. A more general idea of how to do this would be to find some kind of ordering of the shapes so that similar shapes are close and dissimilar ones are far apart.

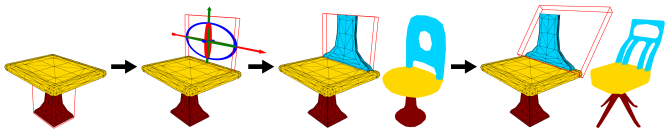


Figure 1: Exploring the shape collection by changing the structure of an initial shape. Existing parts can be copied and assigned a different label. Such a new part can then be used as a proxy to explore the collection.

In this paper we address this problem, denoted as *shape exploration*. In order to provide such an ordering, we propose a relation-based way to parametrize shape collections, allowing the user to explore the entire set of shapes by controlling a small number of parameters. Given a co-segmented set of shapes from the same family, we look at pairs of adjacent parts and how their spatial arrangements vary in regards to other part pairs of the same category.

To make the exploration process more intuitive, we visualize the parameter changes in the shape viewer. The currently selected part is represented by its bounding box which is then transformed relatively to its adjacent part based on the current setting of the exploration parameters. This gives a rough first look at the kind of shapes that can be found near the current point in the exploration space and also provides an understand-

ing of how the shapes change when moving in a specific direction of the exploration space.

Since our method only captures local variability, we also consider the possibility of not using just one such relation to differentiate between shapes, but multiple relations. As an example, one could adjust the spatial arrangements of a chair seat compared to both the legs and backrest to find more specific shapes. Finally, we also look at the problem of shapes containing different numbers of parts. This is something that is often ignored in existing approaches or dealt with by clustering shapes according to their part numbers and only allowing exploration within each such cluster. In our method, we provide a simple tool to allow the copying and pasting of existing parts to add parts not present in the current shape, making it possible to explore the entire shape collection from any initial shape. This can be seen in Figure 1.
In summary, our contributions are as follows:

- A parametrization of shape collections based on relations between shape parts that automatically captures the way in which the shapes vary.
- A simple exploration process that allows browsing the collection based on local variability (spatial arrangements between a pair of adjacent parts) or searching for more specific shapes by considering multiple such relations at the same time.
- A visualization of the exploration parameters that can aid the user in understanding the effects of altering the parameters, making exploration more intuitive.
- A way to change the structure of the initial shape, allowing the user to find shapes with different structures and explore the entire collection from every initial shape.

Our approach can be separated into two stages. The first stage is the parametrization stage described in Section 3. In this stage, the collection of shapes is analyzed to find relations between shape parts that are useful in exploring the collection. This allows us to compute exploration parameters that can be used to explore the collection in the exploration stage which is described in Section 4. We also consider the possibility of using nonlinear methods described in Section 5.

We perform a number of experiments with our system, results of which are presented in Section 6. Furthermore, in Section 7 we conduct a user study to show that the ordering of shapes produced by our approach is not less intuitive than ordering by the individual spatial features like distance, angle or scale between parts. The discussion of our results, as well as limitations of our approach can be found in Section 8. Finally, we conclude this paper with a short summary in Section 9.

2. Related Work

With online model repositories growing larger, a need for automatically organizing these huge collections of shapes has arisen. As such, the problem of how to organize and explore large collections of shapes has been a popular topic of research in recent years. In this section we give an overview of related work on the topic.

Template deformations: Ovsjanikov et al. [8] construct a template made out of boxes from an initial shape and compute a shape descriptor for each shape based on how the template has to be deformed to fit the shape. These descriptors are projected to a low dimensional manifold from which a deformation model is then learned, encoding the way in which the shapes vary the most. The user can then deform the template to explore the collection, with deformations with high variability being suggested by arrows. However, this method only works for shapes that are sufficiently close to the template and is thus less suited for shapes with varying topology. Similarly, Averkiou et al. [9] also abstract each shape by a box template and then encode the spatial arrangements of the boxes as a vector. Based on a distance measure between such configuration vectors, the shapes are embedded into a two-dimensional space that can be explored by the user by clicking on a position inside this space.

Semantic attributes: Another possibility is to use semantic attributes to encode the variations of shapes inside a collection. Chaudhuri et al. [10] conduct a user study to obtain semantic attributes that describe shape parts. To explore the collection of shape parts, the user can manipulate a slider for each attribute to find parts that posses more or less of that certain attribute. Yumer et al. [11] also perform a user study to gather semantic attributes for a shape category, but additionally connect those attributes to shape geometry to allow shape deformation by altering the semantic attributes of a shape. For exploration, the shapes are embedded into a two-dimensional space based on their attributes and a color map is used to visualize high and low values of a chosen attribute in the 2D space.

Images: More recently, the possibility of incorporating 2D images in the exploration of shape collections has also been considered. Hueting et al. [12] train classifiers on both images and shapes to align the image viewpoints with views of the 3D shape and then in turn estimate geometric properties of the shapes shown in the images based on the properties of the 3D shapes. By computation of the differences between their geometric properties, the user can then find 3D shapes that are similar to the shapes displayed in images or find images by setting the orientation of a 3D shape.

Shape descriptors: Kleiman et al. [13] use a nearest neighbor graph of the input shapes, computed using a combination of similarity measures, to generate a dynamic map that can be explored. A subset of shapes of the collection is arranged on a grid of limited size, with similar shapes close to each other. When the user scrolls in any direction or changes the zoom level, the new cells of the grid that enter the screen space are dynamically filled with shapes based on the similarity graph. Huang et al. [14] also combine multiple shape descriptors for their similarity measure and construct a category tree based on quartets of shapes, consisting of two pairs of shapes with high intra-pair similarity and low inter-pair similarity. Based on an initial shape, the other shapes are then arranged in a circular chart with the selected shape in the center and similar shapes in the circles close to the center.

Local features: The methods mentioned above mostly capture the global variability of the shapes, but it is also possible to concentrate on local features. Kim et al. [15] compute fuzzy correspondences between sample points on each shape. These correspondences are then used to define the similarity between shapes. The user can select one or more regions on an initial shape (regions with high variability are highlighted with a color map) to find other shapes with corresponding regions that are either very similar or dissimilar to the selected regions. Rustamov et al. [16] use a shape difference operator to extend this approach by allowing magnification or interpolation of shape differences, meaning that other shapes where a certain difference is more or less pronounced can also be found. Further extension of this method is described by Huang et al. [17] who use functional map networks to even allow interpolation and magnification of discrete differences in topology.

Combined approach: Gao et al. [18] combine both global shape descriptors and local similarity based on fuzzy correspondences in their approach. Given a user-provided sketch, the system retrieves a number of representative models that are similar to the sketch. The user can then mark shapes or even regions of shapes as preferred or disliked to refine the search using an active learning scheme. Fish et al. [19] use global shape descriptors to first find a small number of similar shapes for each shape and then analyse local differences between similar shapes to construct a network of shape clusters where edges between clusters signify a change in shape structure.

Pairwise relations: The meta-representation method of Fish et al. [20] allows exploration of shape collections based on spatial arrangements of a part to other parts or the shape as a whole. For each type of parts, a probability distribution is computed that encodes some kind of spatial feature, like the relative scale compared to the whole shape or the distance to another type of part. This probability distribution is then used as a map of the exploration space that gives the user an overview of the kinds of settings that exist in the shape collection. Our method is inspired by this approach, but we additionally consider correlations between spatial features and filter out features that do not contribute to the variability within the shape collection.

A more comprehensive survey on data-driven shape analysis, including organization and exploration of large shape collections, can be found in the works of Mitra et al. [21] and Xu et al. [22].

3. Parametrization Stage

The aim of the parametrization stage is to find a small number of parameters that allow the user to explore the shape collection. The idea is to use spatial features, such as vertical and horizontal distance, angle and relative scale between adjacent parts as a guide, similar to the method by Fish et al. [20]. However, using every spatial feature as a parameter would result in a large number of parameters, and some of them might not vary much across the collection, making them a bad choice for exploring the collection. Furthermore, correlations might exist between different features, resulting in a lot of empty regions in the exploration space if these features are used as separate parameters. Thus we use Principal Component Analysis (PCA) to reduce the dimensionality of the parameter space, which results in a small number of exploration parameters.

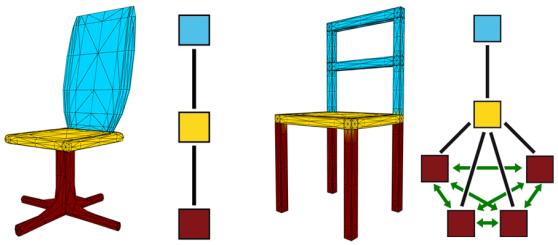


Figure 2: Two co-segmented meshes and their corresponding shape graphs. Each node corresponds to a part of the shape, visualized by its color. Black edges represent adjacencies between parts, while green edges represent symmetries between parts.

The whole stage can be divided into three steps. First we create a graph abstraction of each shape, followed by computing the spatial features between each pair of adjacent parts. In the last step, the exploration parameters are found by analyzing how these spatial features vary for similar pairs of parts.

3.1. Shape Graph

The input is a set of shapes, each given as a single or multi-component 2-manifold polygonal mesh with an existing co-segmentation, meaning that every face of a shape has been assigned a label l from a set of labels $\mathcal{L}$ that is the same for the entire collection. If such a co-segmentation is not available, it can be computed with a variety of co-analysis methods (see [21, 22] for an overview). We furthermore assume that each shape has an upright orientation, which can be assured using for instance the method of Fu et al. [23].

We compute a structural graph for each input shape. Nodes correspond to parts, meaning a single or multiple intersecting components of the mesh where every face has the same label, thus fulfilling a certain function of the shape. The edges of the graph correspond to relationships between the parts, specifically adjacency and symmetry. An example of such a graph can be seen in Figure 2. To construct the graph we first abstract each shape part by its Oriented Bounding Box (OBB).

To compute the adjacency edges, we test for adjacency between parts with different labels. This is done either by iterating over all mesh edges to find adjacent faces with different labels, or by running intersection tests between differently labeled components using a hierarchy of OBBs [24]. We also create an approximation of the contact area between the parts by taking their boundary vertices (or triangle-level intersections in the latter case) and computing the OBB enclosing them. We will refer to the center of this OBB as the *contact center*. We also take the axis of the OBB corresponding to the smallest eigenvalue which we will refer to as the *contact normal* since it serves as the normal vector of the plane that best approximates the contact area between the parts. The contact center and normal are stored with the adjacency edge and will be useful later when computing the spatial features between the two parts.

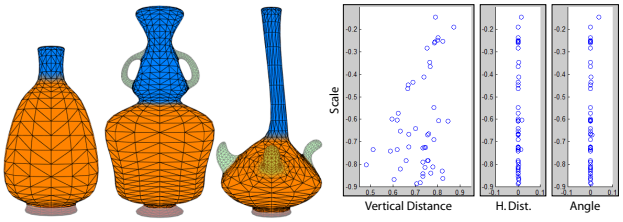


Figure 3: The distribution of the shapes in the vase dataset based on the spatial features between the neck and body of the shapes.

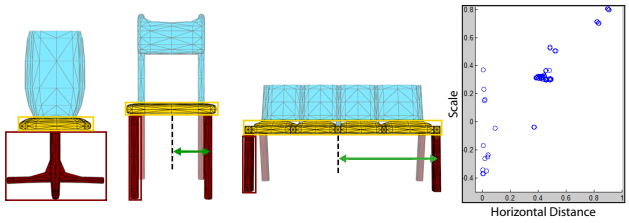


Figure 4: The spatial features between adjacent parts can be correlated. In this case there is a correlation between the horizontal distance and scale between seat and legs of the chair.

To complete the shape graph, symmetry relations are computed between nodes with the same label. Symmetries are detected using an approach similar to the one in Jain et al. [25]. In particular, we create candidate translational, reflectional or rotational symmetry transforms between parts and check if their OBB centers and axes are close enough after applying the transform, in which case a symmetry edge is created.

3.2. Spatial Features

Next we compute a feature vector $\mathbf{x}_{pq}$ for each adjacency edge between parts p and q. We have chosen four features to capture the spatial arrangement between adjacent parts: the vertical distance d_y and horizontal distance d_{xz} between part centers, the angle θ between parts and the bounded relative scale s between parts.

Distance: Let $\mathbf{d} = (d_x, d_y, d_z)^T$ be the distance vector between part centers $\mathbf{p}_c$ and $\mathbf{q}_c$. Then, the vertical distance d_y is simply the y-coordinate of $\mathbf{d}$. The horizontal distance is the Euclidean distance between the part centers in the xz-plane: $d_{xz} = \sqrt{d_x^2 + d_z^2}$. Both d_y and d_{xz} are normalized by dividing by the diameter of the bounding box obtained by merging the bounding boxes of the two parts. We have chosen not to use the distance in x- and z-direction as separate features because it would necessitate the assumption that the orientation of all shapes is globally aligned, which is not always the case with models taken from an online database. We do, however, assume that the shapes are in upright position.

Angle: To compute the angle θ between parts, we select the OBB axis of each part that is best aligned with the contact normal of the adjacency edge and compute the angle between them.

Scale: The relative scale $s_r(p, q)$ between two parts p and q is obtained by dividing the OBB diameter of p by the OBB diameter of q. However, the scale is not symmetric since $s_r(p, q) = 1/s_r(q, p)$, so we define a consistent ordering based on the part labels. Let L_p and L_q be the labels of parts p and q, then the ordered relative scale s_o is given by

$$s_o(p, q) = \begin{cases} s_r(p, q), & \text{if } L_p < L_q \\ s_r(q, p), & \text{otherwise} \end{cases}.$$

Furthermore, we want to limit the range of the feature. We define a mapping s with

$$s(p, q) = \begin{cases} 1 - \dfrac{1}{s_o(p, q)}, & \text{if } s_o(p, q) > 1 \\ s_o(p, q) - 1, & \text{otherwise} \end{cases}.$$

This mapping limits the range of the feature to the interval $[-1, 1]$ such that $s(p, q) = 0$ means that the OBB diameter of the two parts have equal length, positive values mean that p is larger than q and negative values mean that p is smaller than q.

3.3. Exploration Parameters

For the last parametrization step, we define a set of relations $\mathcal{R}$ for the shape collection. A relation R_{lk} is contained in the set only if there exists at least one shape in the collection whose graph contains an adjacency edge between nodes labeled l and k. Since the adjacency edges are undirected, the relations are symmetrical, i.e. $R_{lk} = R_{kl}$. For each relation, we want to compute one or two exploration parameters, depending on how much the spatial features are correlated. This is done by using PCA to reduce the dimensionality of the feature space spanned by the spatial features. For each relation R_{lk}, all corresponding adjacency edges are embedded into a feature space based on the centered feature vectors $\hat{\mathbf{x}}_{pq} = \mathbf{x}_{pq} - \bar{\mathbf{x}}_{lk}$, where $\mathbf{x}_{pq}$ is the feature vector of the adjacency edge between parts p and q, and $\bar{\mathbf{x}}_{lk}$ is the mean of the feature vectors of all adjacency edges between parts labeled l and k.

Then we compute each feature's variance for every relation and remove the features with a variance smaller than a threshold. The reasoning is that for any given relation there might be some features that do not significantly contribute to the variability within the shape collection. An example of this can be seen in Figure 3, where the horizontal distance and angle between the neck and body of the shapes in the vase dataset are the same for most shapes. Thus they are not useful parameters for browsing the collection. We set the threshold to 0.005 which was chosen empirically.

Next we perform PCA for each relation since there is the possibility of correlation between features. If two features are strongly correlated, it is possible to use a single parameter for these features instead of using one parameter for each. Figure 4 shows an example in which the horizontal distance and relative scale between the legs and seat of the shapes in the chair dataset are correlated. Since the legs of a chair are usually placed at the corners of the seat, their horizontal distance to the center of the

seat depends on the size of the seat. On the other hand, for chairs that only have a single leg, the leg is usually placed at the center for balance and needs to be larger to carry the weight of the seat.

Performing the PCA yields the principal component coefficient matrix $\mathbf{V}_{lk}$ and the variances $\boldsymbol{\lambda}$ for each relation, as well as a principal component score $\mathbf{y}_{pq}$ for each adjacency edge between parts p and q. The rows of $\mathbf{V}_{lk}$ can be seen as the axes of the principal component space expressed as vectors in the feature space. The scores $\mathbf{y}_{pq}$ express the coordinates of the adjacency edge in the principal component space and are used as our parameters for the exploration of the shape collection. They are obtained by transforming the centered feature space coordinates $\hat{\mathbf{x}}_{pq}$ with the corresponding transposed coefficient matrix $\mathbf{V}^T_{L_pL_q}$:

$$\mathbf{y}_{pq} = \mathbf{V}^T_{L_pL_q}\hat{\mathbf{x}}_{pq}\,. \tag{1}$$

The principal component variances $\boldsymbol{\lambda}$ are simply the variances of the data for each axis of the principal component space. They are used to decide the number of parameters used for the exploration. We choose to use the two principal components with the largest variances if the variance of the second component is larger than half the variance of the first component. Otherwise we choose only the first principal component since the second component does not significantly contribute to the variation of the relation. In cases where we need two principal components but two features have been discarded because their variance is too small, we do not use the principal components as parameters. Instead we use the two remaining features directly since it is more intuitive to control them separately. Since we only need one or two principal components, we only store at most two columns of $\mathbf{V}_{lk}$ and $\mathbf{y}_{pq}$ with each relation and adjacency edge respectively. Finally, we also store the mean feature vector $\bar{\mathbf{x}}_{lk}$, as well as the minimum and maximum of the features and scores of each relation for later use.

4. Exploration Stage

Using the coefficient scores $\mathbf{y}$ as parameters, it is now possible to explore the shape collection. Starting with an existing shape, the user selects an adjacency edge a_{pq} by picking a pair of adjacent parts p and q. It is then possible to alter the parameters $\mathbf{y}_{pq}$ of the chosen edge a_{pq}. This can be done in two ways. First, it is possible to change the parameters directly by interacting with one or two sliders, each corresponding to an entry of $\mathbf{y}_{pq}$. The second way of altering the parameters is by interacting with the shape itself, using a manipulator tool to transform the selected part and then recomputing the parameters from the new transformation. A coupling between the parameter sliders and the visual representation ensures that any changes are always reflected on both sides. This is explained in Section 4.1. Once the altered parameters are computed, we search for the shape in the collection that best fits these parameters, which is described in Section 4.2.

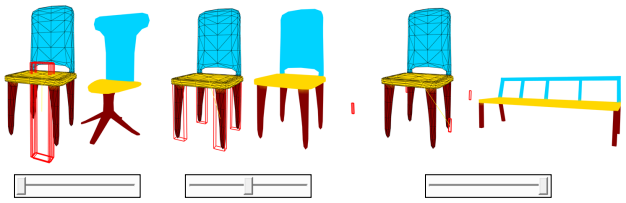


Figure 5: Exploration of the shape collection by directly changing the parameters of the chosen adjacency edge. In each image, the original shape is shown on the left, while the shape that best fits the altered parameters is shown on the right. The bounding boxes of the chair's legs are updated according to the parameter changes.

4.1. Visual Representation

Interacting with a slider is a simple and quick way of changing a parameter value. The problem with this kind of interaction is that the effect of increasing or decreasing the parameters is not immediately apparent to the user. Because of that, we have designed a visual aid in the model viewer that is coupled to the parameters. The selected part p as well as all parts that are connected to p by a symmetry edge are visually represented by their OBBs. To show the effect of altering the parameters, the position, rotation and scale of the OBB of p is updated accordingly. An example of this can be seen in Figure 5.

To apply the parameter changes to the visual representation, it is first necessary to compute the altered spatial features from the altered parameters. Let $\mathbf{y}'_{pq}$ denote the altered parameters of the adjacency edge a_{pq}, and let $\Delta\mathbf{y}_{pq} = \mathbf{y}'_{pq} - \mathbf{y}_{pq}$ be the difference between the altered and original parameters. Using $\Delta\mathbf{y}_{pq}$ and the coefficient matrix $\mathbf{V}_{L_pL_q}$ we can compute the change in terms of the original spatial features:

$$\Delta\mathbf{x}_{pq} = \mathbf{V}_{L_pL_q}\Delta\mathbf{y}_{pq},$$

with $\Delta\mathbf{x}_{pq} = (\Delta d_y, \Delta d_{xz}, \Delta\theta, \Delta s)^T$. We can apply these changes to the original arrangement between p and q to represent the change visually.

Distance: We translate the OBB of p along the vector between part centers $\mathbf{p}_c$ and $\mathbf{q}_c$, first by Δd_y in y-direction, then by Δd_{xz} in the xz-plane.

Angle: To apply the change of the angle between the parts, it is first necessary to determine the axis of rotation. Let $\mathbf{u}_p$ and $\mathbf{u}_q$ denote the OBB axes of p and q that are best aligned with the contact normal of a_{pq}. Since we want to transform the OBB of p in relation to the OBB of q, the axis of rotation can be obtained by the cross product $\mathbf{u}_q \times \mathbf{u}_p$. Of course this is only possible when $\mathbf{u}_q$ and $\mathbf{u}_p$ are linearly independent, in which case the OBB of p is rotated by $\Delta\theta$ around the computed axis of rotation using the contact center as the pivot. Otherwise we do not visualize the rotation, since we have no means to determine the axis of rotation.

Scale: To apply the change in scale, we first compute the new relative scale s'_r from $s' = s + \Delta s$:

$$s'_o = \begin{cases} \dfrac{1}{1-s'}, & \text{if } s' > 0. \\ 1 + s', & \text{otherwise} \end{cases}, \qquad s'_r = \begin{cases} s'_o, & \text{if } L_p < L_q. \\ \dfrac{1}{s'_o}, & \text{otherwise} \end{cases}.$$

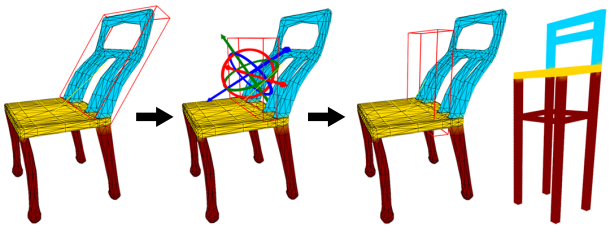


Figure 6: It is also possible to change the parameters by transforming the bounding box of the selected part using a manipulator tool.

Since s'_r denotes the new relative scale of p in relation to q, it is first necessary to scale p to be of the same size as q before applying the new scaling s'_r. Thus the OBB of p is scaled by $s'_r(p,q)/s_r(p,q)$.

To keep the symmetry between symmetric parts, we also apply any parameter change $\mathbf{\Delta y}$ to the parameters of relevant adjacency edges of symmetric parts and transform their OBBs accordingly.

It is important to note that the altered features can possibly take on values that exceed the minimum or maximum features present in the shape collections, or even result in invalid values such as a negative distance or scale. This can happen because at most the first two principal component scores are changed, while the others remain the same. For this reason we store the minimum and maximum features for each relation during the parametrization so we can restrict the visual representation to those values.

The other method of changing the parameters is by interacting with the shape in the model viewer. After selecting an adjacency edge by picking a pair of adjacent parts p and q, the user can activate edit mode to freely translate, rotate and scale the OBB of p. This can be done with a manipulator tool similar to those in conventional modeling programs (cf. Fig. 6). Once the user accepts the transformation of the OBB, a feature vector $\mathbf{x}_{p'q}$ is computed, with p' denoting the transformed OBB. We center the feature vector by subtracting the mean feature vector $\bar{\mathbf{x}}_{L_{p'}L_q}$ of the corresponding relation $R_{L_{p'}L_q}$ and compute the new principal component score $\mathbf{y}_{p'q}$ using Equation 1. If the new score falls outside the range of scores of the relation, the values are clamped so they do not exceed the minimum and maximum values. Finally, the OBB is transformed once more based on $\mathbf{y}_{p'q}$ to ensure that the visual representation stays in line with the parameters. Figure 6 shows an example of this process.

4.2. Finding The Best Fit

The altered parameters are then used to find shapes in the collection that best fit these parameters. There are two options for this search: finding the pair of parts in the collection that best satisfy the parameters of the currently selected adjacency edge, or incorporating multiple adjacency edges of the current shape in the search of the best fit. The first case is simple. Let a_{pq} be the currently selected adjacency edge with altered parameters $\mathbf{y}'_{pq}$, and let $A_{L_pL_q}$ denote the set of adjacency edges between parts labeled L_p and L_q (note that a_{pq} is also included in this set). Then we can find the best fitting adjacency edge a^* by

$$a^* = \underset{a \in A_{L_pL_q}}{\arg\min} \|\mathbf{y}'_{pq} - \mathbf{y}_a\| .$$

In order to take multiple adjacency edges into account during the search, we must first consider how to deal with cases where the cardinality of parts varies across the collection. For every shape M_i we first compute the average parameter vector $\bar{\mathbf{y}}^i_{lk}$ for each relation R_{lk} given by

$$\bar{\mathbf{y}}^i_{lk} = \begin{cases} \dfrac{\sum_{a \in A^i_{lk}} \mathbf{y}_a}{|A^i_{lk}|}, & \text{if } |A^i_{lk}| > 0 \\ \dfrac{\mathbf{y}^{max}_{lk} - \mathbf{y}^{min}_{lk}}{2}, & \text{otherwise} \end{cases},$$

with A^i_{lk} being the set of adjacency edges of shape M_i between parts labeled l and k, and $\mathbf{y}^{max}_{lk}$ and $\mathbf{y}^{min}_{lk}$ being the maximum and minimum values for $\mathbf{y}_{lk}$. Furthermore, we extend each vector $\bar{\mathbf{y}}^i_{lk}$ by adding another entry z^i_{lk}, which is the weighted number of adjacency edges between parts labeled l and k given by

$$z^i_{lk} = \begin{cases} \omega \dfrac{|A^i_{lk}| - A^{min}_{lk}}{A^{max}_{lk} - A^{min}_{lk}}, & \text{if } A^{max}_{lk} - A^{min}_{lk} > 0 \\ 0, & \text{otherwise} \end{cases},$$

with $A^{min}_{lk} = \min_{i=1\ldots n} |A^i_{lk}|$, $A^{max}_{lk} = \max_{i=1\ldots n} |A^i_{lk}|$ and $\omega \geq 0$ being a weight parameter that decides how important the number of adjacency edges is in finding similar shapes. The average of the altered parameters $\bar{\mathbf{y}}'_{lk}$ is computed in the same manner. The shape M^* that is closest to the altered parameters can then be found by

$$M^* = \underset{M_i \in \mathcal{M}}{\arg\min} \sum_{R_{lk} \in \mathcal{R}} \|\bar{\mathbf{y}}'_{lk} - \bar{\mathbf{y}}^i_{lk}\| .$$

4.3. Structural Changes

Since we can also take the number of adjacency edges into account when searching for the best fit, it is necessary to provide the user with a way to add or remove adjacency edges of the current shape. Either operation is only allowed when the number of adjacency edges $|A^i_{lk}|$ remains within the interval $[A^{min}_{lk}, A^{max}_{lk}]$. The first step is to select an adjacency edge by picking two adjacent parts p and q. An adjacency edge can be deleted by pressing the corresponding button. Creating a new adjacency edge is done by creating a new part p' that is then connected to q. This can be achieved by copying the OBB of p and using the manipulator tool to move it to a new location. Once confirmed, the vertices and faces of p are copied and placed at the new location using the transformation of the OBB. To obtain an approximation for the contact center and contact normal of the new adjacency edge, we intersect the edges of the OBB of p' with the planes of the OBB of q and vice versa. The points of intersection are then treated as the boundary vertices between the two parts to compute the contact center and normal. If the OBB of p' is placed so that there are less than 3 intersections, the new location is not accepted and the user can

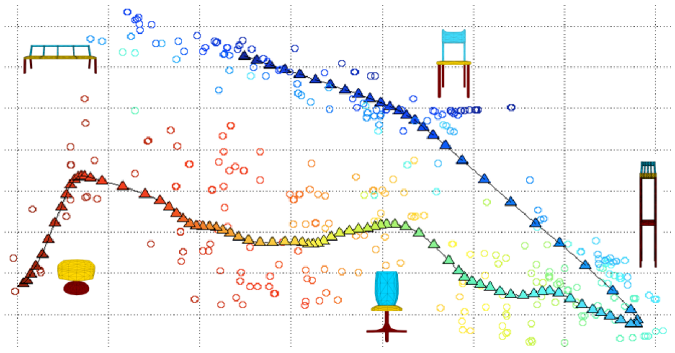


Figure 7: Nonlinear dimensionality reduction for a dataset in a three-dimensional space. The data points, represented by circles, are colored depending on their position in the lower dimensional space. The triangles are points corresponding to uniformly sampled points from the lower dimensional space.

either alter the transformation of the OBB or cancel the creation of the new part.

Finally, it is also possible to change the label of a copied part, thus allowing the creation of adjacency edges that do not occur in the current shape. While the new part still has the geometry of the part that was copied, it can be used as a proxy to find shapes with similar part arrangements and to swap in parts with the same label from other shapes. Figure 1 shows an example of this process. The shown shape from the chair dataset only consists of a seat and a leg. The leg part is copied, transformed, placed on top of the seat and assigned the label of a back part. It is then used as a proxy to find a shape with a different backrest.

5. Nonlinear Parametrization

In our proposed method, we use PCA to reduce the dimensionality of the feature space in order to obtain our exploration parameter space. Using PCA has the advantage that we get a mapping between the feature and parameter space, allowing us to transform any point from one space to the other, which is necessary for our visualization. However, PCA is a linear method and thus might not provide a good mapping if the data points are distributed in a way that cannot be well approximated with a linear function. In this case it might be a good idea to consider nonlinear dimensionality reduction techniques to compute our parameter space. Figure 7 shows an example of this for the relation between the legs and seat of the chair dataset. The circles represent shapes in a three-dimensional feature space and are colored based on their position in the one-dimensional exploration parameter space. The triangles represent uniformly sampled points from the exploration parameter space to illustrate their spacing in the feature space, which corresponds to the visualization computed from our exploration parameter values.

We use Locally Linear Embedding (LLE) [26] for this purpose, although many other nonlinear dimensionality reduction methods would work as well. In this method, each data point is represented by a linear combination of its nearest neighbors, thus assuming that the neighborhood is locally linear. This neighborhood information is used to compute a lower dimensional embedding of the data under the constraint that the representation of a point by its neighbors should be the same in both spaces.

In contrast to PCA, the computation of the lower dimensional space with LLE does not yield a mapping between the two spaces that would allow us to transform any arbitrary point from one space to the other. However, we require this information for our visualization, so we need an alternative. To do this we follow the suggestion of Saul and Roweis [27] to compute a statistical model that allows us to estimate the corresponding point based on the distribution of the existing data points. In short, we compute a joint distribution by first using a mixture-model of Gaussians to model the data distribution in the lower dimensional space and then compute a distribution of these Gaussians in the higher-dimensional space.

In total, this process requires three parameters: the desired dimensionality of the lower dimensional space, the number of neighbors for the neighborhood construction and the number of Gaussians to model the joint distribution. Ideally, we want to use just a single exploration parameter for each relation, so we set the dimensionality of the exploration parameter space to 1. The other two parameters are more difficult to determine. A low number of neighbors can lead to bad results when the data is heavily clustered, with large distances between cluster boundaries. On the other hand, if the number is too large, the result will be similar to linear methods since the neighborhoods are assumed to be locally linear. The number of Gaussians used for the mixture-model depends on the number of data points and their distribution. Ideally, we want a large number of samples for each Gaussian to improve the accuracy of our model. Inferring the optimal parameter values automatically for a given dataset is one of the open questions regarding this method.

6. Results

We performed a number of tests on a machine with an Intel Core i5 processor with four 3.40GHz cores, 8 GB RAM and an AMD Radeon HD 6950 graphics card. For our test data we used the chair, vase and candelabra datasets from the Shape COSEG Dataset [28] with 400, 300 and 28 shapes respectively, and a plane dataset consisting of 15 shapes taken from the Princeton Shape Benchmark [29] and 3D Warehouse. For most datasets, the labeling was provided together with the dataset. The shapes of the plane dataset do not come with an existing labeling, so we created our own manual segmentation. We perform tests for both single-relation and multi-relation exploration.

The parametrization of the shape collection only needs to be done once before exploration becomes possible. The performance depends on the number of shapes and the number of faces for each shape. On the test system, the parametrization step takes 2 minutes and 32.48 seconds for the chair dataset and 1 minute and 14.48 seconds for the vases dataset. Parametrization of the candelabra dataset takes 26.79 seconds, while computations for the plane dataset take 11.14 seconds.

Since the computation of the shape that best fits the altered parameters is fast, exploration of the shape collection can be

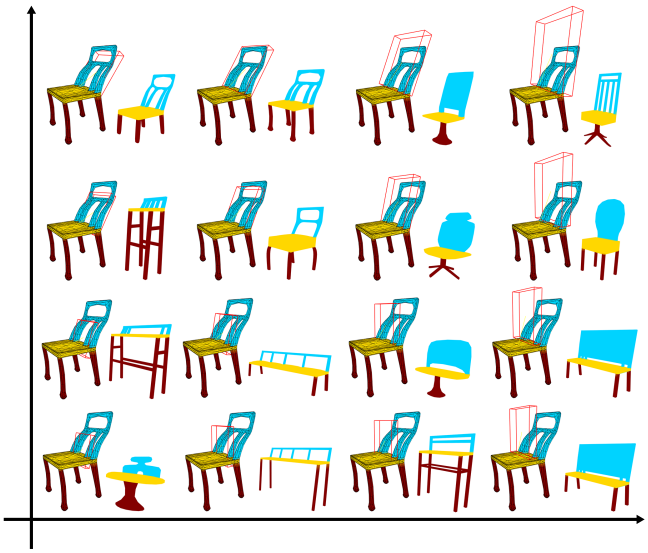


Figure 8: Exploration of the chair dataset using the backrest-seat relation. For each pair of shapes, the initial shape is shown on the left, while the preview of the shape that best fits the altered exploration parameters is shown on the right. The altered parameters are represented visually by the bounding box of the backrest.

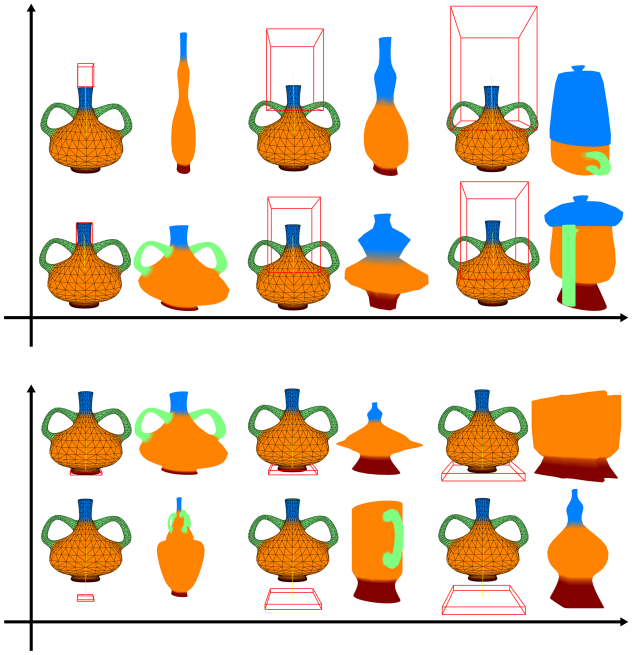


Figure 9: Exploration of the vase dataset using the relation between neck and body (top), and base and body (bottom). Since only two features show a high variance, but with low correlation, the individual features are used directly for exploration.

done in real-time even for large sets. As a performance stress test, we combined the vases and chair datasets (for a total of 700 shapes) where exploration results in a framerate of 30 – 40 frames per second on the test system even when considering all relations in the computation of the best fit.

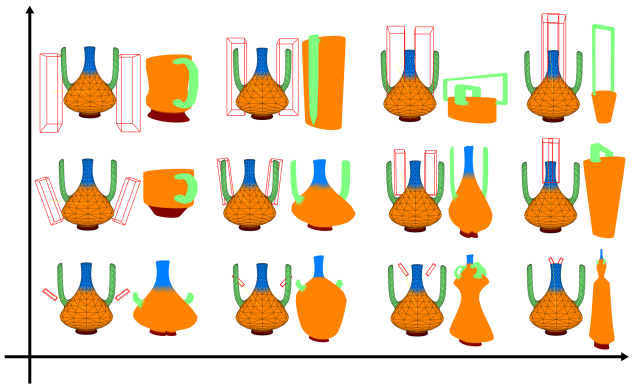


Figure 10: Exploration of the vase dataset using the relation between handle and body. Changes to the first principal component (x-axis) result in a change in angle and vertical distance, while changes to the second principal component (y-axis) result in a change in scale and horizontal distance.

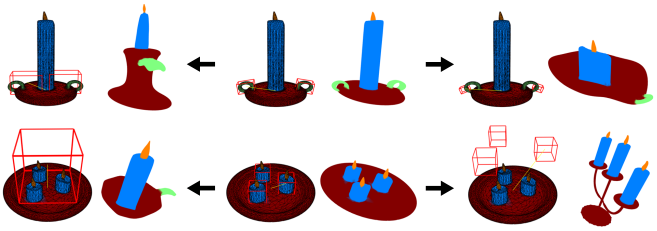


Figure 11: Exploring the candelabra dataset using different relations. In the first row, the relation between the handle and the base of the candelabra is changed. In the second row, the dataset is explored by changing the relation between the candles and the base.

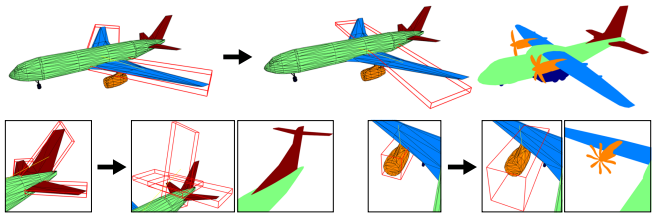


Figure 12: Exploring the plane dataset using different relations. We use the relations between wing and fuselage (top), stabilizer and fuselage (bottom left), and engine and wing (bottom right) to explore the collection.

6.1. Single-Relation Exploration

In our tests, we explore the chair and vase datasets using different relations. Figure 8 shows the exploration of the chair dataset using the backrest-leg relation, while exploration of the vases dataset can be seen in Figures 9 (using the relations of neck or base to the body) and 10 (using the handle-body relation). The shapes are ordered roughly according to their position in the 2-dimensional exploration parameter space. As can be seen, similar shapes are closer to each other than dissimilar shapes.

Finally, we also test two small datasets – the candelabra set from the COSEG benchmark and the plane dataset consisting of shapes that have been put together from shapes found on 3D Warehouse and the Princeton Shape Benchmark. The exploration of these datasets is depicted in Figures 11 and 12.

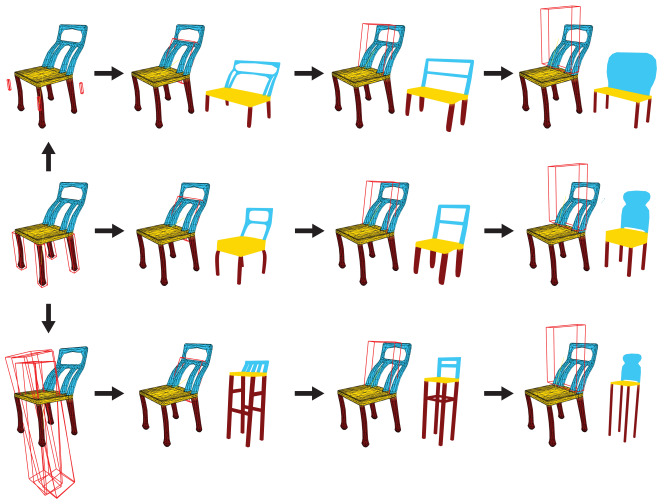


Figure 13: Exploration of the chair dataset using multiple relations. After first adjusting the parameters of the leg-seat relation, changing the parameters of the backrest-seat relation yields different results because both relations are taken into account when computing the best fit. The backrests of the chairs in each column are similar, while the legs are different.

6.2. *Multi-Relation Exploration*

In this section we explore the shape collections by taking multiple relations into account. By changing the parameters of multiple relations it is possible to find more specific results. The tests are performed using the same datasets as before.

For the first test, we use the chair dataset, with the following testing procedure: We first set the parameter for the leg-seat relation of the chair. Then we alter the parameters for the relation between the backrest and the seat. The results can be seen in Figure 13. In each column, the shown shapes use the same parameter values for the backrest-seat relation. This results in the shapes of each column having a similar backrest, but the type of shape is different since the leg-seat relation is also taken into account.

For the next test we also consider the number of parts as a parameter. Finding a good weight ω to determine how much influence the difference in part numbers has depends on the use case. For this test we intend to only search for shapes that have the exact same number of parts. For that purpose, ω is set to a large value, in this case 100. We choose a shape from the vase dataset containing a neck, a base, a body and two handles as the starting shape to search for other shapes in the dataset. Then we explore the collection by alternately changing the parameters of the neck and the base of the shape. We perform the same exploration process three times, each time with a different number of handles. In the first test, the number of handles is unchanged. In the second test, both handles are removed. Then in the third test the two handles are duplicated using our copy-and-paste operation.

The results are shown in Figure 14. As can be seen, the shapes depicted in each column have similar characteristics, with the main difference being the number of handles.

In the last exploration test, we show how to deal with cases where the starting shape does not contain all possible relations.

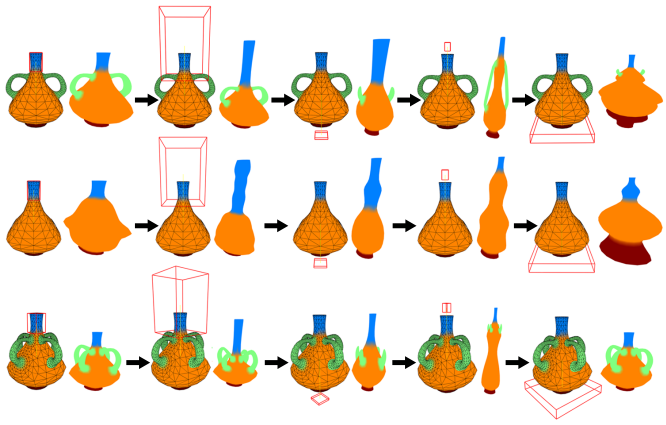


Figure 14: Exploration of the vases dataset using multiple relations and different numbers of parts. The shapes in each column have similar characteristics, but a different number of handles.

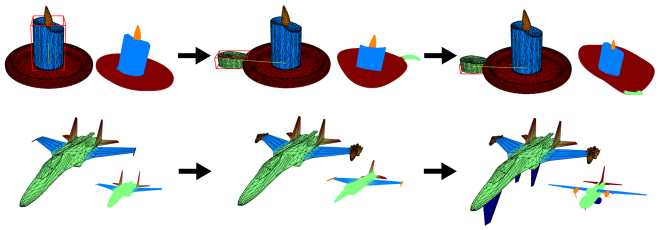


Figure 15: Exploration of the candelabra and plane datasets by adding parts that do not exist in the starting shapes shown on the left.

For this we use the candelabra and plane datasets. Figure 15 shows how we can add a handle to the base of a candle or engines and wheels to a plane in order to find other shapes that do contain these parts.

6.3. *Exploration Using Nonlinear Parametrization*

We performed a number of tests with nonlinear parametrization for the two large chair and vase datasets. Choosing appropriate parameter values proved to be a difficult task. Empirically, we have found that a number of 24 to 32 neighbors and 2 to 4 Gaussian led to the best results for our data. Using a lower number of neighbors mostly causes the data in the lower dimensional space to become more clustered with large inter-cluster distances, resulting in discontinuities when transitioning from one cluster to another. A high number of clusters can have a similar effect since there are less samples per cluster and thus the model becomes less accurate.

The chair to leg relation of the chair dataset is good example for a dataset which can benefit from using nonlinear parametrization, which is illustrated in Figure 7. The data forms two clusters, the top cluster containing benches and conventional chairs, while the bottom cluster contains office chairs and stools with just one leg. This clustering originates from the difference in horizontal distance between legs and seat. The triangles show the uniformly sampled exploration parameter values in feature space. As can be seen, the path traced by the parameter values follows a curve. Moving along the curve starting

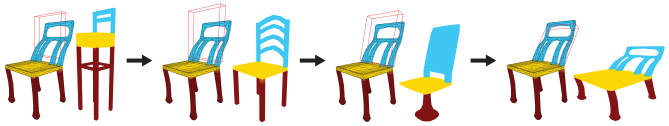


Figure 16: Exploration of the chair dataset using the backrest-seat relation after nonlinear parametrization. Shapes with a small straight backrest can be found on the left side of the parameter space. Moving towards the right side, the backrest becomes larger while still remaining straight, then smaller again as the angle also increases.

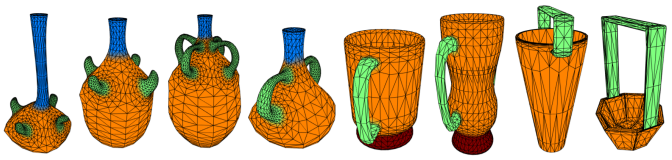


Figure 17: The order of shapes obtained with nonlinear parametrization of the vase dataset using the handle-body relation.

from the top left, this corresponds to the size of the legs becoming larger within the first cluster, and then becoming smaller again within the second cluster.

Using nonlinear parametrization for the backrest to seat relation results in a similar curve for the sampled exploration parameter values. Beginning with shapes containing a small straight backrest, changing the parameters causes the backrest to become larger while still remaining straight, and finally smaller again as the angle between backrest and seat increases at the same time. Example shapes found in this exploration can be seen in Figure 16.

Finally, we also test nonlinear parametrization for the handle-body relation of the vase dataset, which results in an ordering of shapes demonstrated in Figure 17.

7. Evaluation

We performed a user study to evaluate our method and compare our orderings of shapes to the orderings produced by each individual spatial feature, as would be the case using the method by Fish et al. [20]. Informally, the goal of the user study was to analyze if the orderings provided by our methods (PCA and LLE) would match with an ordering given by human intuition.

We surveyed 16 participants, where each was given a number of shapes printed on paper cards (cf. supplementary material) and was then instructed to order the shapes by a specified relation. 3 different relations, each consisting of 8 shapes, were used as test cases. The shapes were randomly sampled from the COSEG chair and vase datasets, but to ensure that no shapes were visually too similar, some of them were replaced manually. While we did explain to the participants which spatial features we used to compute the orderings, we did not tell them which features are more useful to differentiate the shapes of the different datasets.

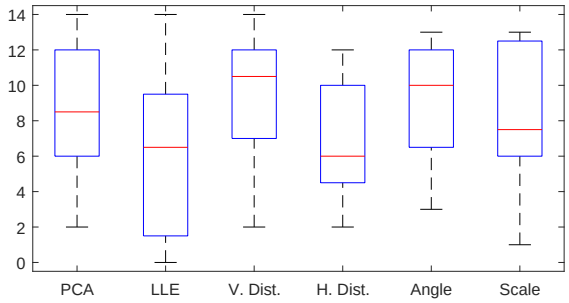
To allow for a 3-dimensional perception of each shape, a perspective view and orthogonal front and side views were presented on every card, depictions of which can be found in the supplementary material. Every user received the datasets in a randomized order to reduce potential bias caused by any previous orderings. The shapes for the first and second test cases were picked out of the COSEG chair dataset. Whereas in the first case the users had to sort by the relation between legs and seat (test case T1), in the second case the relation between seat and backrest was used (test case T2). The third test case (T3) consisted of vases where the relation between handles and body was important. Note that we reduced the dimensionality of the feature space to a single dimension for both PCA and LLE for this comparison. Both parametrizations were computed using the entire dataset, after which the relative order of the test samples was determined. For LLE we used 24 neighbors in the parametrization.

Kendall Tau metric [30] was used to compute the distance between pairs of orderings. Since the global direction of an ordering does not matter in our case, we used the reversed order of the second ordering in the pair if its distance to the first ordering was smaller than its original distance. Table 1 lists the average distances of the user orderings to the orderings obtained using PCA, LLE and the four spatial features, while Figure 18 shows the boxplots of the distances between the individual user orderings and the 6 computed orderings for each dataset. A visual comparison between different computed and user orderings for the leg-seat relation of the chair dataset is illustrated in Figure 19.

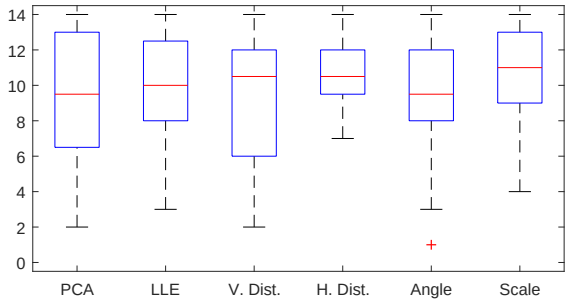
The results show that, for both test cases T1 and T3, our nonlinear dimensionality reduction method produced the ordering that is closest to the user orderings, suggesting that it corresponds well to the intuition of the users. The ordering obtained by PCA did not perform as well, but is in both cases still better than 2 of the 4 orderings by individual features as used by Fish et al. If the user does not have any prior knowledge about the variability of the shape collection, it can prove to be an advantage over having to pick one of the features at random.

For the second test case T2, none of the computed orderings produced a result that aligned well with the orderings provided by the users. According to the feedback gathered after performing the tests with each user, most users found it difficult to order the shown chairs based on the spatial arrangements between backrest and seat, and instead found it more intuitive to order them based on visual features of the backrests such as the structural complexity, topology, or roundness. In the future, it would certainly be interesting to include such features in the parametrization and also conduct further studies to determine the types of features that users find most intuitive to sort by.

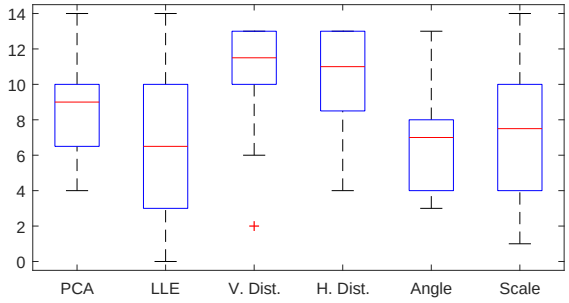
We perform a right-tailed Student's t-test to validate the results of our user study. Since our method has the advantage of not requiring any prior knowledge about the variability of shapes in the dataset to allow efficient exploration, we want to show that our method does not perform worse than using an ordering based on the individual features. Our null hypothesis states that the mean distance of our parametrized orderings to the user orderings is not larger than the mean distance between the individual features and the user orderings. The resulting p-values can be found in Table 2. These values show the probability of obtaining a result like in our user study under the assumption that the null hypothesis is true.



(a) Chair dataset, leg-seat relation



(b) Chair dataset, backrest-seat relation



(c) Vase dataset, handle-body relation

Figure 18: These boxplots show the distances of the user orderings to the orderings obtained using PCA, LLE, vertical distance, horizontal distance, angle and scale.

The results for test cases T1 and T3 confirm our observations that PCA yields an ordering that is at least as intuitive as half of the individual features, but less intuitive compared to the other half. For LLE we get p-values above 0.95 for 2 of the 4 features, and above 0.9 for the third feature. Only the comparison to the individual feature with the smallest mean distance to the user orderings – horizontal distance for T1 and angle for T3 – results in p-values lower than 0.9, but even in the case that LLE performs worse in this case, the difference is likely to be small.

Further data of our user study can be found in the supplementary material, including depictions of all sample shapes, orderings provided by the users and more visual comparisons between different orderings.

	PCA	LLE	V. Dist.	H. Dist.	Angle	Scale
T1	8.563	6.063	9.188	6.938	9.188	8.063
T2	9.375	9.625	9.125	10.5	9.25	10.38
T3	8.813	6.688	10.688	10.438	7.188	7.313

Table 1: The average distance of the user orderings to the computed orderings. T1: Chair dataset, leg-seat relation. T2: Chair dataset, backrest-seat relation. T3: Vase dataset, handle-body relation.

	V. Dist.	H. Dist.	Angle	Scale
T1 - PCA	0.9519	0.0244	0.9418	0.0205
T1 - LLE	0.9674	0.8755	0.9734	0.9364
T2 - PCA	0.2722	0.7875	0.4464	0.9481
T2 - LLE	0.2613	0.7786	0.1914	0.7680
T3 - PCA	0.9621	0.9876	0.0362	0.0657
T3 - LLE	0.9910	0.9952	0.8334	0.9226

Table 2: P-values obtained in a right-tailed Student's t-test under the null hypothesis that the mean distances of our PCA and LLE methods are not larger than the mean distances of the individual features.

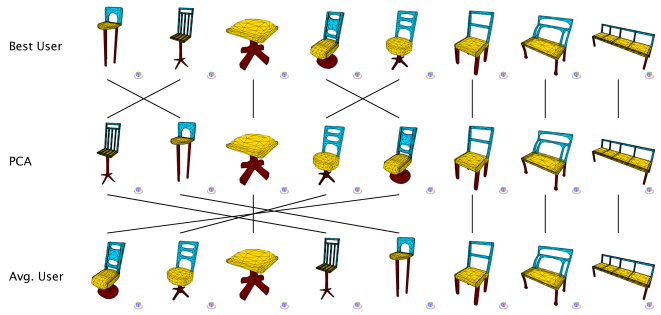
8. Discussion and Limitations

The results of our tests for the parametrization and exploration stages show that the shapes found while browsing the collection are mostly in line with our expectations. Spatial features with low variation within the collection – which are not very useful for distinguishing between shapes – are eliminated and correlations between features are detected successfully.

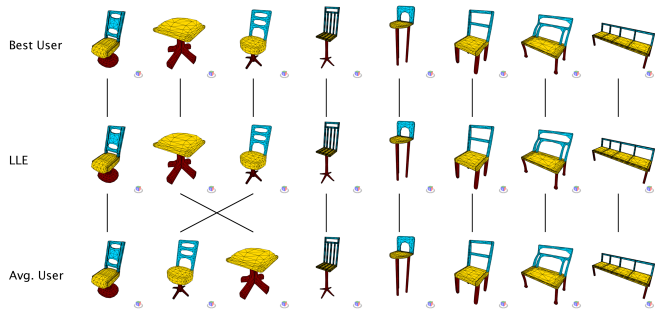
As an example, consider the leg-seat relation depicted in Figure 19. Our statistical evaluation shows that, considering all orderings based on individual features, the ordering based on horizontal distance is the one closest to the user orderings. While it is easy to differentiate one-legged chairs from four-legged chairs or benches using just the horizontal distance between the legs and the seat, differentiating between different one-legged chairs requires the vertical distance or relative scale as a second feature, since the horizontal distance is zero for each shape.

This is an advantage over the method of Fish et al. [20] where each spatial feature is treated as a separate parameter that can be used to explore the collection. Our visual representation generally provides a good idea of the changes caused by altering the exploration parameter values. However, since it is just an approximation, the results can sometimes differ and the translation and rotation of the bounding boxes can only be computed based on the current transformation.

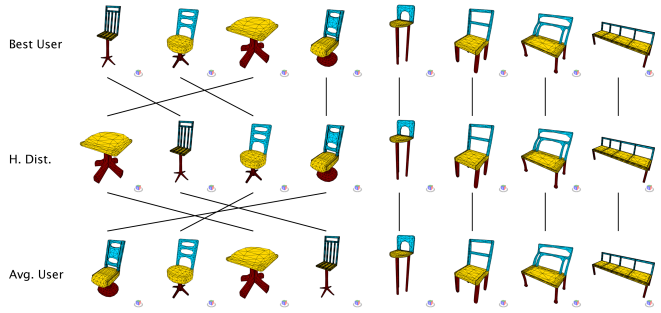
Exploration using a single relation can give a quick overview of the shapes in the collection, while using multi-relation exploration allows the user to find more specific shapes. This is somewhat similar to the method of Kim et al. [15] who allow the user to mark one or more regions to find shapes with similar or dissimilar features in the marked regions. However, while their measure of similarity can be used to order the shapes in the collection based on their similarity to the selected shape, there is no distinction between two shapes that are not only dis-



(a) Parametrization with PCA.



(b) Nonlinear parametrization with LLE.



(c) Horizontal distance (best-fitting individual feature).

Figure 19: A visual comparison between computed and user orderings for the leg-seat relation of the chair dataset. The middle row in each image shows the ordering produced by PCA, LLE and horizontal distance respectively, while the top row shows the user ordering with the smallest distance to the computed ordering, and the bottom row shows the user ordering with the smallest average distance to all other user orderings.

similar to the selected shape, but also dissimilar to each other. Our approach on the other hand provides a global ordering of the shapes for each relation, making it easier to get an understanding of the shape distribution in the exploration space.

The delete and copy-and-paste operations allow the user to alter the structure of the initial shape, making it possible to find other shapes with different structures. This addresses a problem that is common with part-based methods that consider global variability, namely the question of how discrete differences in topology should be handled. For example, template deformation methods ([8, 9]) can only be applied to shapes that fit a given template reasonably well. While our system does not incorporate discrete topology differences into the parametrization, we provide the user with the possibility to find shapes with a desired structure by altering the structure of any initial shape. However, we realize that our implementation could still be improved considering that manual transformation and placement of the parts can be a bit tedious. An interesting idea would be to extend this approach by finding suggested placements of the copied parts based on the other shapes in the collection.

While we use only four spatial features in the computation of our exploration parameters, it would be theoretically possible to include many more, even hundreds of features. However, depending on the variability of each individual feature and their correlations, a lot of information might be lost when mapping hundreds of dimensions down to just one or two using PCA. Furthermore, in the case of LLE, a lot more samples will be required for the parametrization since distances between samples will be quite large. There is also the tradeoff that the exploration process will likely become less intuitive, as a lot of features change at the same time when changing the parameter values. The simplest solution to this problem would be to only incorporate a small number of features with the largest variance in the computation. As an alternative, it would also be possible to ask the user to order a number of shapes in a similar way to our user study to determine the features that seem most important to the user.

In the future, our method could be extended to also incorporate other types of geometric features that are not necessarily based on a relation between two different parts. Unary features that are only related to the part itself (like roundness or topology) could be modeled in the shape graph as a loop edge, connecting the part node with itself, while features that are measured in relation to the entire shape could be accommodated by adding a virtual node to the graph that represents the entire shape.

One of the most important outcomes of our research is the observation that using nonlinear parametrization allows us to find a better low-dimensional embedding. This fact appears clear, since the distribution of the data in the feature space is in the majority of the cases also nonlinear. While in this case we need to specify the number of additional parameters, e.g., number of neighbors for the LLE and the number of Gaussians for the computation of the statistical model, finding optimal values can be approached with a number of empirical tests. However, once suitable values for a given dataset are found, nonlinear parametrization turned out to be a very useful tool for the exploration of shape collections. This is especially evident by the data obtained in our user study which shows that a nonlinear parametrization of the shapes corresponds very well with the intuitive orderings given by the users.

9. Conclusions

We introduced a new method of parametrization and exploration of large shape collections obtained from online repositories. In our approach to this problem, we analyze relations between parts to find out how their spatial arrangements vary across the collection, yielding a small number of parameters that can be used to explore the collection. A visual representation of how the spatial arrangements change when altering the parameters provides a better understanding of the kind of shapes that can be found in the exploration process, making

it more intuitive to search for specific shapes. To account for shapes containing a varying number of parts, the system includes a copy-and-paste operation for parts that also allows for the addition of parts that are not present in the currently displayed shape.

Acknowledgements

We thank all the participants of the user study for their time and patience, and the reviewers for their valuable comments and suggestions. This work was partially funded by the FWF, project no. P27972-N31.

References

[1] Funkhouser T, Kazhdan M, Shilane P, Min P, Kiefer W, Tal A, et al. Modeling by example. ACM Transactions on Graphics 2004;23(3):652–63.

[2] Alhashim I, Li H, Xu K, Cao J, Ma R, Zhang H. Topology-varying 3D shape creation via structural blending. ACM Transactions on Graphics 2014;33(4):158:1–158:10.

[3] Cohen-Or D, Zhang H. From inspired modeling to creative modeling. The Visual Computer 2016;1(32):7–14.

[4] Bronstein AM, Bronstein MM, Guibas LJ, Ovsjanikov M. Shape google: Geometric words and expressions for invariant shape retrieval. ACM Trans Graph 2011;30(1):1:1–1:20.

[5] Liu Z, Xie C, Bu S, Wang X, Han J, Lin H, et al. Indirect shape analysis for 3d shape retrieval. Comput Graph 2015;46(C):110–6.

[6] Savelonas MA, Pratikakis I, Sfikas K. Partial 3d object retrieval combining local shape descriptors with global fisher vectors. In: Proceedings of the 2015 Eurographics Workshop on 3D Object Retrieval. 3DOR; Aire-la-Ville, Switzerland, Switzerland: Eurographics Association. ISBN 978-3-905674-78-1; 2015, p. 23–30.

[7] Kleiman Y, van Kaick O, Sorkine-Hornung O, Cohen-Or D. Shed: Shape edit distance for fine-grained shape similarity. ACM Trans Graph 2015;34(6):235:1–235:11.

[8] Ovsjanikov M, Li W, Guibas L, Mitra NJ. Exploration of continuous variability in collections of 3d shapes. ACM Trans Graph 2011;30(4):33:1–33:10.

[9] Averkiou M, Kim VG, Zheng Y, Mitra NJ. ShapeSynth: Parameterizing model collections for coupled shape exploration and synthesis. Computer Graphics Forum 2014;33(2):125–34.

[10] Chaudhuri S, Kalogerakis E, Giguere S, Funkhouser T. Attribit: Content creation with semantic attributes. In: Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology. UIST '13; New York, NY, USA: ACM; 2013, p. 193–202.

[11] Yumer ME, Chaudhuri S, Hodgins JK, Kara LB. Semantic Shape Editing Using Deformation Handles. ACM Transactions on Graphics 2015;34(4):86:1–86:12.

[12] Hueting M, Ovsjanikov M, Mitra NJ. Crosslink: Joint understanding of image and 3d model collections through shape and camera pose variations. ACM Trans Graph 2015;34(6):233:1–233:13.

[13] Kleiman Y, Fish N, Lanir J, Cohen-Or D. Dynamic maps for exploring and browsing shapes. In: Proceedings of the Eleventh Eurographics/ACMSIGGRAPH Symposium on Geometry Processing. SGP '13; Aire-la-Ville, Switzerland, Switzerland: Eurographics Association; 2013, p. 187–96.

[14] Huang SS, Shamir A, Shen CH, Zhang H, Sheffer A, Hu SM, et al. Qualitative organization of collections of shapes via quartet analysis. ACM Trans Graph 2013;32(4):71:1–71:10.

[15] Kim VG, Li W, Mitra NJ, DiVerdi S, Funkhouser T. Exploring collections of 3d models using fuzzy correspondences. ACM Trans Graph 2012;31(4):54:1–54:11.

[16] Rustamov RM, Ovsjanikov M, Azencot O, Ben-Chen M, Chazal F, Guibas L. Map-based exploration of intrinsic shape differences and variability. ACM Trans Graph 2013;32(4):72:1–72:12.

[17] Huang Q, Wang F, Guibas L. Functional map networks for analyzing and exploring large shape collections. ACM Trans Graph 2014;33(4):36:1–36:11.

[18] Gao L, Cao YP, Lai YK, Huang HZ, Kobbelt L, Hu SM. Active exploration of large 3d model repositories. IEEE transactions on visualization and computer graphics 2015;21(12):1390–402.

[19] Fish N, van Kaick O, Bermano A, Cohen-Or D. Structure-oriented networks of shape collections. ACM Trans Graph 2016;35(6):171:1–171:14.

[20] Fish N, Averkiou M, van Kaick O, Sorkine-Hornung O, Cohen-Or D, Mitra NJ. Meta-representation of shape families. ACM Transactions on Graphics 2014;33(4):34:1–34:11.

[21] Mitra N, Wand M, Zhang HR, Cohen-Or D, Kim V, Huang QX. Structure-aware shape processing. In: SIGGRAPH Asia 2013 Courses. SA '13; New York, NY, USA: ACM. ISBN 978-1-4503-2631-5; 2013, p. 1:1–1:20.

[22] Xu K, Kim VG, Huang Q, Kalogerakis E. Data-driven shape analysis and processing. In: Computer Graphics Forum. Wiley Online Library; 2016,.

[23] Fu H, Cohen-Or D, Dror G, Sheffer A. Upright orientation of man-made objects. ACM Trans Graph 2008;27(3).

[24] Gottschalk S, Lin MC, Manocha D. Obbtree: A hierarchical structure for rapid interference detection. In: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques. SIGGRAPH '96; New York, NY, USA: ACM; 1996, p. 171–80.

[25] Jain A, Thormählen T, Ritschel T, Seidel HP. Exploring Shape Variations by 3D-Model Decomposition and Part-based Recombination. Computer Graphics Forum 2012;31(2):631–40.

[26] Roweis ST, Saul LK. Nonlinear dimensionality reduction by locally linear embedding. Science 2000;290(5500):2323–6.

[27] Saul LK, Roweis ST. Think globally, fit locally: Unsupervised learning of low dimensional manifolds. J Mach Learn Res 2003;4:119–55.

[28] Wang Y. The Shape COSEG Dataset. 2012. URL: http://irc.cs.sdu.edu.cn/~yunhai/public_html/ssl/ssd.htm; (accessed September 13, 2016).

[29] Shilane P, Min P, Kazhdan M, Funkhouser T. The Princeton Shape Benchmark. In: Shape Modeling International. 2004,.

[30] Kendall MG. A new measure of rank correlation. Biometrika 1938;30(1/2):81–93.