SHED: Shape Edit Distance for Fine-grained Shape Similarity

Yanir Kleiman[1] Oliver van Kaick[1,2] Olga Sorkine-Hornung[3] Daniel Cohen-Or[1]
[1]Tel Aviv University [2]Carleton University [3]ETH Zurich

Abstract

Computing similarities or distances between 3D shapes is a crucial building block for numerous tasks, including shape retrieval, exploration and classification. Current state-of-the-art distance measures mostly consider the overall appearance of the shapes and are less sensitive to fine changes in shape structure or geometry. We present *shape edit distance* (SHED) that measures the amount of effort needed to transform one shape into the other, in terms of rearranging the parts of one shape to match the parts of the other shape, as well as possibly adding and removing parts. The shape edit distance takes into account both the similarity of the overall shape structure and the similarity of individual parts of the shapes. We show that SHED is favorable to state-of-the-art distance measures in a variety of applications and datasets, and is especially successful in scenarios where detecting fine details of the shapes is important, such as shape retrieval and exploration.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Geometric algorithms, languages, and systems.

Keywords: Shape similarity, intra-class retrieval, edit distance.

1 Introduction

With the growth of on-line shape repositories in recent years, there is an increasing need to organize and efficiently explore large collections of 3D shapes [Ovsjanikov et al. 2011; Huang et al. 2013b; Kim et al. 2013; Kleiman et al. 2013; Averkiou et al. 2014]. The question underlying this task, and a fundamental question in shape analysis, is how to compare shapes and measure their similarity. In fact, the choice of an appropriate similarity measure is in the core of many algorithms that handle 3D shapes. In this regard, previous work has focused on the development of similarity measures for classification of shapes into broad sets of categories [Tangelder and Veltkamp 2008], but little attention has been given to estimating the similarity between shapes that belong to the *same* class. That is, detection of inter-class differences has been emphasized over quantification of intra-class differences. With the large repositories available today, organization and exploration of families of shapes has become as important as categorizing shapes into different classes, and such tasks require an estimation of fine-grained shape similarities.

In this work, we introduce *shape edit distance* to measure similarities between shapes. Intuitively, the shape edit distance (SHED) measures the amount of effort needed to transform one shape into the other, in terms of rearranging the parts of one shape so that they closely match the parts of the other shape, or by adding and deleting parts (Figure 1). SHED takes into account both the similarity of shape structure and the similarity of individual shape parts. Here, we follow the recent trend of representing shapes as graphs of parts [Kalogerakis et al. 2012; Laga et al. 2013; Mitra et al. 2013; Zheng et al. 2014]; however, we use the matching between graphs to extract a *global measure* of shape similarity.

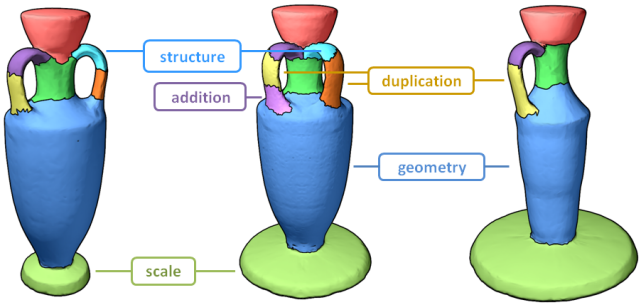


Figure 1: *The distance between shapes is measured by edit operations that transform the parts of one shape into corresponding parts in the other shape.*

The strength of the shape edit distance is its tolerance to part rearrangements, additions and deletions. Thus, SHED is flexible in quantifying the similarity between shapes that have partial similarities, articulated parts or repositioned parts. This leads to a similarity measure that accurately captures finer shape differences, enabling a finer-grade organization of shapes. In contrast, other traditional similarity measures are oblivious to the shape structure: for example, the light field descriptor, popular in shape retrieval [Chen et al. 2003], is highly sensitive to any type of shape difference or deformation, while in bag-of-feature approaches, the similarity is invariant to the arrangement of shape components [Bronstein et al. 2011; Litman et al. 2014].

We do not explicitly find a sequence of editing operations that transforms one shape into the other, but indirectly estimate the edit distance by finding a part correspondence and using it to extract a measure of similarity. First, shapes are segmented into parts and an approximate correspondence is computed between the parts of each shape. To find this correspondence, we develop an adaptive spectral matching technique. Our method incorporates complex constraints into an iterative optimization whereas previous solutions solved the approximation in a one-shot manner. We do not enforce a strict one-to-one correspondence, since a part in one shape may be duplicated or missing from the second shape. Instead, we apply constraints to the matching by associating additional costs when parts change their context. Then, each match between two parts is associated with a transformation cost: a weighted sum of terms that relate to the differences in part geometry, scaling and position of the parts in the shape. Finally, the edit distance of the shape is the aggregated cost of transforming all parts in the correspondence. We also present supervised learning for automatic computation of the weights from examples, as opposed to manual tuning, which could be unintuitive.

We demonstrate the advantage of using SHED with a series of experiments. First, we evaluate SHED in a quantitative manner by constructing categorization trees that can be used for shape exploration. We compare these trees to the trees generated using other state-of-the-art similarity measures, as well as ground truth trees created by expert users. In addition, we cluster shapes into a predefined number of clusters and compare the results to clusters generated from the ground truth trees. These evaluations demonstrate that the similarity estimated by SHED is preferable to other distance measures and leads to a more intuitive shape organization in the intra-class context. In the inter-class context, we perform shape retrieval according to SHED and show that it yields comparable results to state-of-the-art similarity measures. Finally, in settings where ground truth data is not well defined, we show qualitative results of nearest neighbors queries and embeddings of sets of shapes.

2 Related work

Our work comprises ideas such as shape comparison, graph edit distances and part-based matching, which we discuss as follows.

Shape comparison, retrieval and exploration. There has been much work on the development of shape similarity measures that can be used for retrieval, exploration, or any type of shape comparison [Tangelder and Veltkamp 2008]. In terms of shape retrieval and categorization, state-of-the-art approaches that currently give the best performance are a combination of the light field descriptor with bag-of-features and metric learning approaches [Li et al. 2012]. For intra-class organization, Xu et al. [2010] cluster a set of shapes into different groups by factoring out the effect of non-homogeneous part scaling and then establishing a correspondence between shape parts. Huang et al. [2013a] present an approach for fine-grained labeling of shape collections. Similarly to our work, their goal is to learn a distance metric within a class of shapes to capture finer shape differences. However, their method follows a different paradigm than our work: the shapes are globally aligned with an affine transformation followed by local deformations, and the metric is learned on the aligned shapes. Individual parts obtained from segmentation and their transformation are not considered as in our approach. In the more restricted context of isometric matching, there has been much activity in deriving signatures for shape comparison, such as GPS embedding [Rustamov 2007] or heat kernel signature [Ovsjanikov et al. 2010]. Kurtek et al. [2013] define a shape space and metric that capture more comprehensive deformations than nearly isometric, but require surfaces of the same topology. Bag-of-feature approaches [Bronstein et al. 2011; Litman et al. 2014] are considered state of the art for retrieval of non-rigid isometric shapes. The goal of these methods is to retrieve shapes with similar topology from a collection of shapes in the same class, such as human models in different poses. Hence, these methods are not suitable for comparison of shapes with different part composition, structure or topology, which is the focus of our work.

Shape exploration necessitates not only the estimation of the similarity of shapes to a query shape, but also a way of organizing the shapes. Thus, different strategies have been proposed for exploration, such as the use of a deformable template [Ovsjanikov et al. 2011], region selection [Kim et al. 2012], dynamically adapted views of close neighborhoods [Kleiman et al. 2013], or parameterization of the template space [Averkiou et al. 2014]. In the work of Huang et al. [2013b], the goal is to obtain a qualitative organization of a collection of shapes, since an organization based on a single similarity measure is not always meaningful when comparing both similar and dissimilar shapes. Likewise, our goal is to properly capture both inter- and intra-class differences. However, instead of aggregating the scores of several similarity measures, we develop an edit distance to estimate the shape similarity.

Graphs of parts for shape analysis. The idea of describing 2D shapes and images as graphs of parts has appeared prominently in the field of computer vision. A few representative works include matching shapes according to *shock graphs* [Sebastian et al. 2004] and skeletons [Sundar et al. 2003], and matching images according to graphs that represent their segmentations [Harchaoui and Bach 2007]. In the graphics literature, comparing shapes by matching graphs was utilized for consistent joint segmentation [Huang et al. 2011] and co-segmentation [Sidi et al. 2011] of a set of shapes. A group of works has estimated the similarity between shapes by matching Reeb graphs, which are constructed from functions defined on manifold shapes [Hilaga et al. 2001; Barra and Biasotti 2013]. Other works have explicitly segmented shapes and created graphs of segments, with applications in shape synthesis [Kalogerakis et al. 2012] and semantic correspondence [Laga et al. 2013]. These works are directly related to the idea of modeling shapes by combining parts from different models [Funkhouser et al. 2004]. Templates or part arrangements have also been learned from collections, although these do not explicitly represent the connectivity between parts [Kim et al. 2013; Zheng et al. 2014]. The fundamental difference of our approach to these representative works is that we use the matching between two graphs of parts as input to estimate the overall similarity between two shapes; the correspondence between the graphs is the base for a distance measure that enables us to quantify finer shape differences.

Graph matching and integer programming. The graph matching problem is commonly posed as an integer quadratic programming problem, which is NP-hard. There is a large body of work regarding the relaxation of such problems to a tractable convex quadratic programming optimization. Two prominent works in this area are the spectral correspondence presented by Leordeanu and Hebert [2005] and a relaxation of the quadratic optimization by using bounding linear integer optimizations, proposed by Berg et al. [2005]. These relaxations often yield good results in practice in the one-to-one matching scenario. However, performing gradient descent from a continuous relaxation of the integer problem has been shown to yield non-optimal permutations in most cases [Lyzinski et al. 2015]. Indeed, the above methods perform poorly in our one-to-many scenario where a part can correspond to several parts in the other shape. Recently, Kezurer et al. [2015] suggested lifting the problem to a higher dimension, followed by a linear semi-definite relaxation. However, their method is computationally expensive and does not extend easily to one-to-many scenarios. Bommes et al. [2012] perform iterative relaxation of the problem where in each iteration a single integer constraint is added to the optimization. We follow a similar approach, but instead of adding hard constraints in each step, we adjust the objective function to give precedence to solutions which are compatible with previously selected matches.

Graph edit distance. The graph edit distance has been used to find a correspondence between graphs in several areas of visual computing, such as computer vision and medical imaging [Gao et al. 2010]. The idea of an edit distance is attractive because it poses the problem of matching two graphs as finding a sequence of operations that transforms one graph into the other. The edit distance can consider not only the matching of similar nodes and edges, but also their addition, duplication and deletion. However, finding the minimal edit distance is NP-hard, so different heuristics have been proposed to compute it. A common technique is to use a graph kernel that estimates the similarity between two nodes according to their attributes and their neighborhoods in the graphs [Neuhaus and Bunke 2007]. Our shape edit distance does not require an explicit sequence of operations, but an aggregation of all the changes necessary to transform one shape into the other.

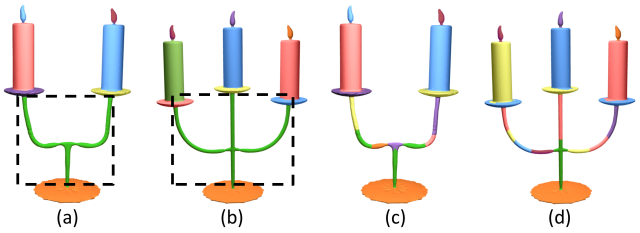


Figure 2: *Difference between the semantic segmentation of two shapes in (a) and (b), and their nearly convex decomposition in (c) and (d). Note how, in (a) and (b), the bounding boxes of the parts corresponding to the candle supports have considerably different sizes. In (c) and (d), both supports are composed of small nearly convex segments with similar sizes.*

In the context of computer graphics, Fisher et al. [2011] used graph kernels to estimate the similarity between graphs representing scenes composed of multiple objects. In addition, Denning and Pellacini [2013] proposed a technique based on the edit distance to quantify *localized* differences between two models. Their method is better suited for comparing models generated by editing the same source shape. On the other hand, our work is aimed at computing the similarity between any pair of shapes. We derive the edit distance directly from a correspondence between graph nodes, as opposed to the methods above based on graph kernels. In addition, we do not require a one-to-one correspondence between the shape parts, but find a one-to-many correspondence and quantify the edit distance without explicitly searching for a sequence of editing operations. We explain the details in the next section.

3 Shape edit distance

Input, output, and shape representation. The edit distance measure takes as input two shapes and returns a real number representing the distance (dissimilarity) between the shapes. The distance is lower for shapes that have similar part geometry and structure, taking into account part rearrangements and partial correspondence, and higher for shapes that differ in these aspects.

We represent each shape as a collection of parts and connections among these parts, i.e., a graph of parts. Our method is generic and can take as input different shape representations, although in this work we represent the shapes as triangle meshes. The first step in our method is the partitioning of input meshes into parts. One possibility is to use semantic segmentation techniques [Shapira et al. 2008; Shamir 2008]. However, semantic parts do not have a clear definition and can greatly vary among different shapes. Moreover, a semantic part can have a complex geometry, making its comparison to other semantic parts non-trivial (Figure 2). In a sense, the problem of comparing two complex segments can be as involved as that of comparing two shapes. Instead, we segment the shapes into simpler primitives that can be more easily analyzed. For this task, we use the recent weakly-convex decomposition technique of van Kaick et al. [2014], which partitions the input shapes into nearly convex parts. Nearly convex parts are easier to analyze, since they have a simpler geometry and can be approximated well by their bounding boxes (Figure 2). In addition, the convex decomposition of a shape is robust to small changes in the shape.

Our method also supports using a manual segmentation of the shapes into parts, if such data is available. However, the results in this paper were produced using the automatic weakly-convex decomposition to provide a complete solution. The part graph is defined by creating a node for each nearly convex part of the shape, and an edge between adjacent parts in the shape segmentation.

Part similarities and matching. Given two shapes represented as graphs of parts, our goal is to find a set of editing operations that transform the parts of one shape into the parts of the other. Possible editing operations include deforming, displacing, duplicating, adding, or removing parts. Then, a cost is associated with each editing operation based on the extent of the transformation. The editing costs are aggregated to produce the final shape edit distance between the two input shapes.

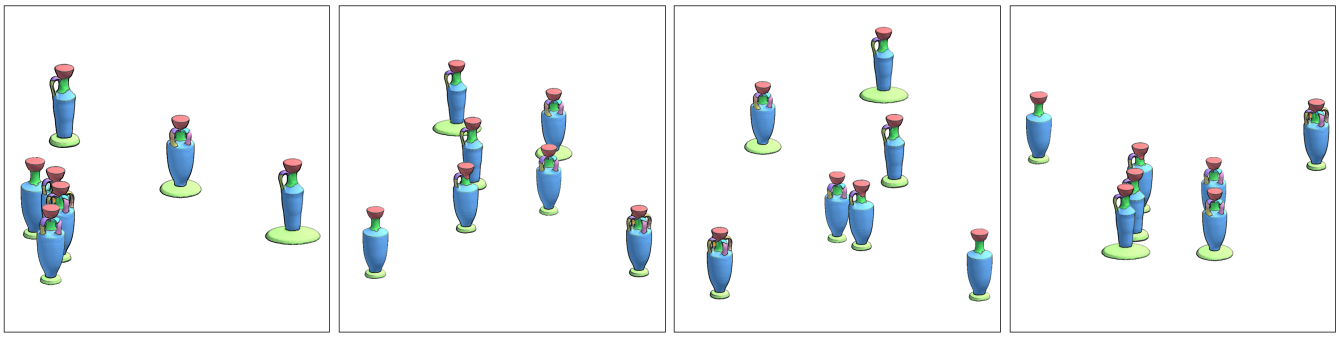
In SHED, we derive the set of editing operations from a mapping between the parts, since we can associate each pairwise match with a single operation. This mapping depends on the similarity of parts to each other as well as their context and the structure of the shape. For example, two parts with different geometry can be matched if their neighborhood is similar. On the other hand, two parts in different locations in the shape can be matched if their geometry is similar. Thus, the mapping of each part depends not only on the part properties, but on the mapping of all other parts of the shape. This makes the problem of finding the correct matching intractable, so an approximate solution is necessary. To this end, we formulate our objective in a quadratic form by constructing unary terms for each match between two parts, and binary terms for pairs of matches, representing only pairwise dependencies between matches. Then, we develop a novel adaptive spectral matching technique to find an approximate solution for this formulation. Our technique uses similar principles as the method of Leordeanu and Hebert [2005], but instead of solving the optimization once and applying constraints in a greedy manner, we iteratively improve the optimization by incorporating the constraints that arise in previous steps. We explain the computation of the matching in detail in Section 4.

Given the mapping between two shapes, a cost can be computed for each edit operation. The costs reflect the following aspects of shape similarity:

- **Similarity of the geometry of the parts.** For example, morphing a cylindrical part into another cylindrical part is less costly than morphing a cube into a cylinder, as the former pair is geometrically more similar than the latter.
- **Similarity of the structure of the part graphs.** We allow nodes to move in the part graph, with a cost proportional to the magnitude of the structural change. Duplicated parts and additional parts also incur additional costs as the structure of the shape changes.
- **Scaling of the parts.** The scale of each part plays a critical role in the global similarity of a shape; different shapes can have similar graphs of parts where each part is scaled differently relative to its neighborhood. Thus, we introduce scale-specific terms in our formulation.

To produce a scalar similarity measure between two shapes, the terms described above need to be weighted and aggregated. A question arises of how to determine the weights for each term. Shape similarity is a subjective measure, so different users might have different views on which shapes are more similar, which implies that different weights are necessary. Moreover, while a set of manually selected weights can provide a reasonable similarity measure for all shapes, it is clearly beneficial to fine-tune the weights to better reflect the variation in a specific set. Therefore, we employ a weight learning scheme that finds the optimal weights to match a set of given distances. We elaborate on the details of the distance formulation and the weight learning scheme in Section 5.

In Figure 3, we show the effect of considering these different factors in the edit distance. We compare a 2D embedding created with multi-dimensional scaling, according to the similarities given by SHED and the light field descriptor (LFD). For SHED, we show



(a) (b) (c) (d)

Figure 3: *Embeddings obtained with multi-dimensional scaling on a small set of vases, based on the following distance measures: (a) LFD, (b) SHED with default weights, (c) SHED with high weight for scaling changes, and (d) SHED with high weight for structural changes.*

the results of using three configurations of weights: equal weights for each term (b), weights learned from user input giving high priority to the scaling of parts (c), and weights learned from user input giving high priority to the structural difference between parts (d). The example set contains vases ranging from zero to four handles, some with a slightly thinner body and some with a bigger base. The consistency of distances provided by SHED yields an intuitive embedding that is true to the observed properties of the shapes, namely the number of handles and size of the parts. On the other hand, the embedding generated by LFD groups shapes according to their overall appearance, and does not take into account the finer details of the shapes. Thus, LFD is not able to distinguish well between the vases that differ by the number of handles, as their projected views are very similar.

4 Part matching

The correspondence between two shapes can be represented as a list of *matches* or pairings between two parts, one from each shape. The mapping does not have to be one-to-one; a part in one shape can be duplicated and have several matches in the other shape. However, we constrain the mappings so that if a part is duplicated, then its matching parts in the other shape are not duplicated, to ensure consistency in the editing operations. In other words, for each edge in the matching graph, the degree of at least one of its vertices is one. The dependencies between different possible matches are complex and can involve more than two matches. We approximate such dependencies by using pairwise constraints only, so the problem becomes tractable. We formulate the correspondence problem using unary terms that depend on a single match, and binary terms involving a pair of matches. Unary terms represent the likelihood of a match, or the affinity between a part in one shape and a part in the other shape. Binary terms represent the compatibility of two matches, i.e. the likelihood that both matches will be a part of the same mapping.

Unary term. The unary term represents the amount of effort necessary to morph the geometry of a part into another part. One of the advantages of segmenting the shape into nearly convex parts is the simplicity of each part, which allows us to use efficient descriptors to effectively distinguish between part geometries. We use the shape distribution signatures to represent the geometry of the parts [Osada et al. 2002]. Specifically, we use the D1 descriptor (also called *shell histogram* [Ankerst et al. 1999]), which computes a histogram of the distance between uniformly sampled points on the surface and the center of mass of the part, and the D2 descriptor, which computes a histogram of the distance between pairs of uniformly sampled points on the surface. These descriptors are relatively simple and fast to compute, yet they are able to distinguish well between parts with simple geometry such as nearly convex parts. The D1 and D2 histograms are computed for each part, and compared using χ^2 distance, which is defined as

$$d_{\chi^2}(H_i, H_j) = \sum_{k=1}^{K} \frac{(H_i(k) - H_j(k))^2}{H_i(k) + H_j(k)}, \quad (1)$$

where H_i, H_j are the input histograms, and K is the number of bins in each histogram. The geometry cost is thus

$$C(i,j) = \alpha \cdot d_{\chi^2}(D1_i, D1_j) + (1-\alpha) \cdot d_{\chi^2}(D2_i, D2_j) \quad (2)$$

where $D1_i$ and $D2_i$ are respectively the D1 and D2 histograms for part i, and α controls the balance between the D1 and D2 descriptors. In our implementation $\alpha = 0.5$ (equal weights). The cost is transformed into an affinity using the natural exponent:

$$U(i,j) = exp(-C(i,j)/\sigma), \quad (3)$$

where σ is chosen such that the affinity values have a wide spread between 0 and 1. In our implementation $\sigma = 0.5$.

Binary term. The binary term represents the compatibility of one match (i,j) to another match (k,l). When two shapes are similar, adjacent parts in one shape are expected to be mapped to adjacent parts in the other shape. In addition, the scaling factor of all matches is expected to be similar, since a match that has significantly different scale than other matches in the mapping is less likely to be correct. Therefore, we define a graph distance cost and a scaling factor cost for each possible match.

The graph distance is defined for each pair of parts on the same shape as the shortest path on the shape graph. We use the ratio between the graph distances of each match to measure the compatibility between matches:

$$G(i,j,k,l) = \frac{\max(g(i,k), g(j,l))}{\min(g(i,k), g(j,l))} - 1, \quad (4)$$

where $g(i,k)$ is the graph distance between parts i and k on the same shape. This term is zero when the matches are fully compatible, i.e. both pairs of parts have the same graph distance in their respective shape, and is highest when one pair of parts is adjacent and the other is not. Note that the cost is low when the graph distances between both pairs are high, so adjacent parts have more weight in the total cost.

We define the scaling factor of each match as the ratio between the volumes of the source and target part: $s(i,j) = \frac{VOL(i)}{VOL(j)}$. Similarly

to the graph distance cost, we use the ratio between the scaling factors of two matches as the scaling factor cost:

$$S(i,j,k,l) = \frac{\max(s(i,j), s(k,l))}{\min(s(i,j), s(k,l))} - 1. \quad (5)$$

The binary term is defined as the affinity between two matches, which is computed from the above costs as follows:

$$B(i,j,k,l) = \beta \cdot \exp(-(G(i,j,k,l) + S(i,j,k,l))/2). \quad (6)$$

The parameter β controls the weight of the binary term compared to the unary term. If β is large, the structure of the shape takes precedence over the geometry of parts, and if β is small, the geometry of the parts is more important than the shape structure. If $\beta = 0$, the only consideration is the part geometry and the correspondence resembles a bag-of-features approach. In our implementation, $\beta = 0.3$.

Matching technique. There are several matching techniques in the literature that find an approximate solution to pairwise-constrained correspondence problems, such as the spectral matching technique of Leordeanu and Hebert [2005], or the integer quadratic programming relaxation proposed by Berg et al. [2005]. The main idea of these methods is that the pairwise constraints can be presented in a quadratic form by constructing a matrix M of $n \cdot m$ rows and $n \cdot m$ columns, where n and m are the numbers of parts in the first and second shape, respectively. The diagonal of M contains the values of the unary term $U(i,j)$, and the values outside of the diagonal of M are the binary terms $B(i,j,k,l)$. The best correspondence is then represented by the binary vector x that maximizes the product $x^T Mx$ and does not break additional constraints, such as the requirement for one-to-one mapping, etc. This poses an integer quadratic programming problem, which is NP-hard, therefore different approximation methods are suggested in the above methods.

Leordeanu and Hebert [2005] propose to first solve an unconstrained assignment problem in the continuous setting, where x is allowed to have values in the range $[0, 1]$. This can be solved easily by setting x to the normalized principal eigenvector of M. Then, the result vector x is binarized in a greedy manner, taking into consideration additional constraints in the process. In each step, the match with the highest value in x is marked, and the values of the match and all conflicting matches in x are reduced to zero. This process continues until all values in x are zeros, and the final mapping is returned as the collection of marked matches. Since the constraints are not incorporated into the cost matrix, the greedy binarization process is less successful when several conflicting mappings are possible. While strictly conflicting matches are filtered out, matches which are compatible with those conflicting matches might still be selected since their score is computed before the conflicting matches are discarded. This effect is most prominent in less constrained scenarios such as ours. For example, we allow duplications of parts, but a matching in which almost all parts are matched to the same part is valid but not desirable in most cases.

To address these issues, we introduce *adaptive spectral matching*, which incorporates the desired constraints directly into the objective function, leading to a more consistent global solution to the correspondence problem. We iteratively adjust the affinity matrix M according to the constraints and re-run the eigenvector decomposition. In this way, not only conflicting matches are excluded from the solution, but matches that are compatible with conflicting matches are also less likely to be selected in subsequent steps. The iterative method starts by setting x to the principal eigenvector of M, and then performs the following steps:

- Mark the match with the highest value in x.
- Set the affinity of the match in M to 1.
- Incorporate constraints into M, by setting the affinities of each conflicting match or pair of matches to zero. In our case, once a match (i,j) is selected, the compatibility of matches that contain part i to matches that contain part j becomes zero (i.e. the binary scores $B(i,j',i',j) = 0$ for each i' and j'), since having both of these matches would mean that there is a many-to-many relation between parts i and j.
- Set x to the principal eigenvector of the adjusted M, and ignore all matches that are conflicting or were already selected.
- Repeat until there are no more valid matches.

A few examples of matchings between segmented shapes using the above algorithm are shown in Figure 4. In each sub-figure, the parts are color coded according to their matching to the shape on the left. Parts that are matched to the same part in the source shape have the same color. Parts that are matched to more than one part in the source shape have the colors of all matching parts mixed in a random pattern. For example, in the bottom left of (a), indicated by cyan and orange lines, both the top and the base of the source vase were matched to the top of the target vase, since it has no base. Similarly, for vases with one handle, both handles of the source vase are matched to the parts of a single handle.

Minor differences in the segmentation of similar shapes do not typically cause significant changes in the matching. For example, as can be seen by the red line in (a), two of the nearest neighbors of the shape have an extra part in the handle. The extra part is matched to a similar part, and the rest of the matching remains correct. Since the duplicated part is small, the similarity between these shapes according to SHED remains high. Similarly, most of the shapes on the top rows of (c), (d) and (f) have minor differences in their segmentation, yet they are considered similar by SHED. On the other hand, significant differences in the segmentation may lead to incorrect matchings, as can be seen in (b) and (e). The vase in (b) is only segmented into four parts while similar vases are segmented into seven parts. Thus the matching between these vases is weak, and matched parts are not similar in their geometry, scale and structure. This causes SHED to assign low similarity score to similar shapes.

5 Distance formulation

The matching algorithm output is a list of matches $(i,j) \in \mathcal{M}$. The transformation of each part in the shape is directly defined by the matches it belongs to. Each transformation is associated with a cost which is determined by the magnitude of change and the relative volume of parts in the shape. Below we describe the four types of transformations and how their associated costs are computed.

Change of geometry. For each match (i,j) in the mapping, the cost of deforming the geometry of one part into the other is computed using the same formula for $C(i,j)$ in Equation 2. Each term is weighted according to the volume of the parts associated with it. For this, we define a match volume $m(i,j) = VOL(i) + VOL(j)$, and normalize it using the sum of volumes of all matches $\hat{m}(i,j) = m(i,j) / \sum_{(i,j)\in\mathcal{M}} m(i,j)$. The geometry cost $C(i,j)$ is then weighted by the normalized match volumes $\hat{m}(i,j)$.

Change of scale. Since the global scale of two shapes can be different, the change of scale between parts must be measured compared to the change of scale in other matches in the mapping. Thus, the scaling costs are computed for each pair of matches (i,j) and (k,l). The scale term is similar to the formula in Equation 5 and

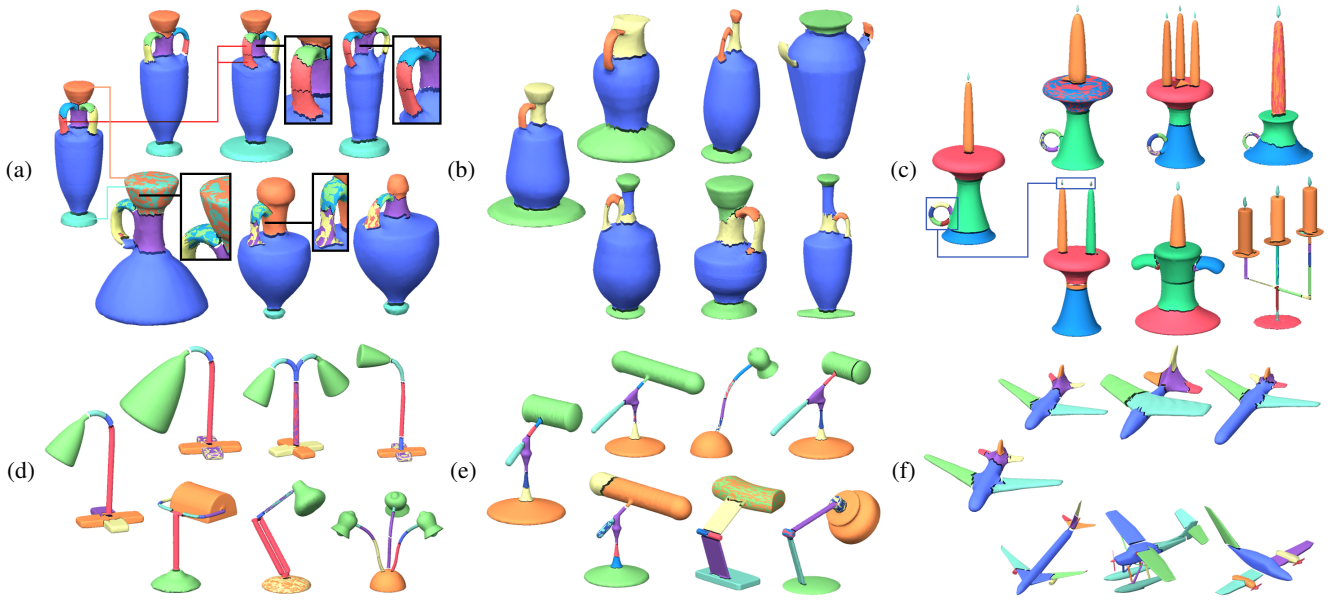


Figure 4: *Matching between shapes. In each set, the source shape (left) is matched with three nearest neighbors according to SHED (top), and three additional shapes which are not neighbors (bottom). Multiple target parts that match the same part in the source shape are marked with the same color (see red line, top insets in (a)). A single target part that is matched with multiple parts in the source shape is marked with mixed colors (see orange and cyan lines, bottom insets in (a)). Note that minor differences in the segmentation do not affect the matching or nearest neighbors computation (a, d, f). On the other hand, significant differences in the segmentation may lead to incorrect matching (b, e).*

measures the difference between the change of scale in the two matches:

$$C_s(i,j,k,l) = \frac{\max(s(i,j), s(k,l))}{\min(s(i,j), s(k,l))} - 1. \tag{7}$$

Note that $C_s = 0$ when the scale change of the two matches is exactly the same, and $C_s = 1$ when the magnitude of change in one match is exactly twice than the other match. The scale costs are weighted by $\hat{m}(i,j) \cdot \hat{m}(k,l)$, such that the total weights of all the pairs which contain match (i,j) is $\hat{m}(i,j)$.

Change of position. To detect a part that changed position, it must be compared with its environment, so the position costs are also computed for each pair of matches (i,j) and (k,l). We compare the graph distance of parts i and k in the first shape $g(i,k)$ and the graph distance of parts j and l in the second shape $g(j,l)$:

$$C_p(i,j,k,l) = abs(g(i,k) - g(j,l)). \tag{8}$$

Note that if a part is duplicated, we compare the adjacency with the most similar instance, such that if several parts are duplicated together as a group they will only be compared to parts in the same group. The position costs are also weighted by $\hat{m}(i,j) \cdot \hat{m}(k,l)$.

Duplication costs. When a part is duplicated, there are two or more matches with the same part. Each of the matches incurs the above costs if applicable. In addition, we aggregate the volume of the shape that is being duplicated, by summing the volume of all parts in all matches and subtracting the total volume of the shapes. The remainder is the volume of all parts (in both shapes) that appear twice or more in the matches. The duplication cost is normalized by the total volume of the matches, so it represents the percent of matches that have duplicated parts. It is formulated as:

$$C_d = \frac{\sum_{(i,j)\in\mathcal{M}} m(i,j) - \sum_{i\in\mathcal{S}} VOL(i) - \sum_{j\in\mathcal{T}} VOL(j)}{\sum_{(i,j)\in\mathcal{M}} m(i,j)}, \tag{9}$$

where $\mathcal{S}$ and $\mathcal{T}$ are the shapes being compared. Note that we do not define a cost for parts that were added, since adding a new part can be thought of as duplicating the most similar part and morphing it to the desired shape.

Aggregation and weight learning. The shape edit distance is formulated as a weighted sum of the above costs:

$$\begin{aligned}
\text{SHED}(\mathcal{S},\mathcal{T}) = {} & w_g \cdot \sum_{(i,j)\in\mathcal{M}} \hat{m}(i,j) \cdot C(i,j) \\
& + w_s \cdot \sum_{(i,j)\in\mathcal{M},(k,l)\in\mathcal{M}} \hat{m}(i,j) \cdot \hat{m}(k,l) \cdot C_s(i,j,k,l) \\
& + w_p \cdot \sum_{(i,j)\in\mathcal{M},(k,l)\in\mathcal{M}} \hat{m}(i,j) \cdot \hat{m}(k,l) \cdot C_p(i,j,k,l) \\
& + w_d \cdot C_d,
\end{aligned} \tag{10}$$

where w_g, w_s, w_p and w_d are the respective weights of the geometry term, scale term, position term and duplication term. Since semantic similarity between shapes is a subjective matter, it makes sense to learn the values of these weights from user input. However, similarity or semantic distance between two shapes cannot be quantified numerically by the user. Instead, we ask users to indirectly provide the semantic similarity of a set of shapes by generating categorization trees, which group together similar shapes in several levels of hierarchy. For more details see Section 6. To learn the weights from the categorization trees, we extract trios of shapes, where in each trio two shapes are similar (i.e. they belong to the same subtree of depth two), and the third shape is semantically far (i.e. it belongs to a different subtree). Each trio of shapes defines a relative relation of the form *"shape A is closer to shape B than to shape C"*. Each categorization tree provides many thousands of trios, from which we randomly select 1000 trios as a training set. To learn the weights from such relations, we employ a weight learning scheme suggested by Schultz and Joachims [2004]. Each relation between shapes A, B, and C, is transformed into a constraint of the form: $D(A,C) - D(A,B) \geq 1$ where $D(A,B)$ is the weighted

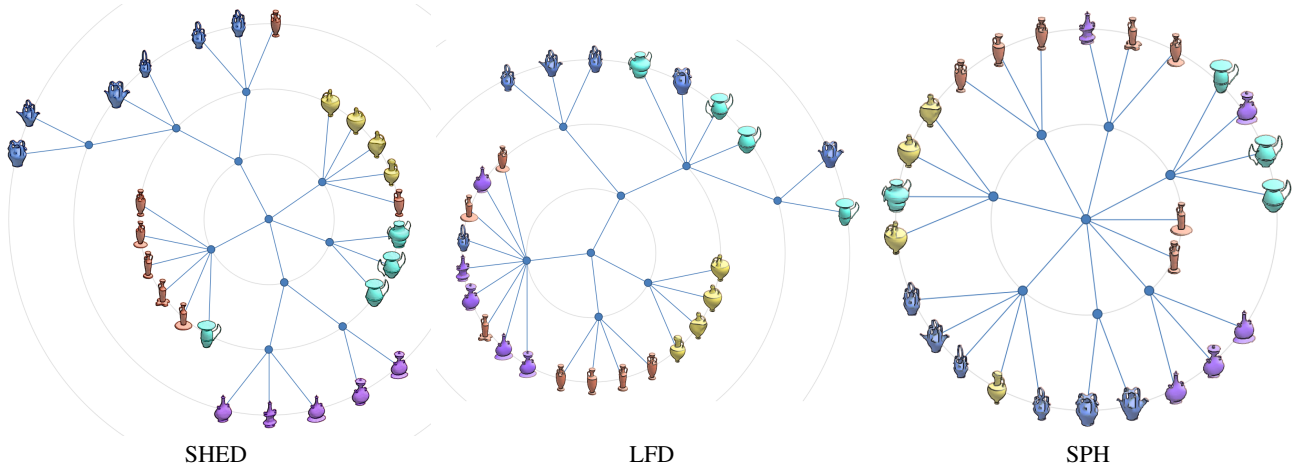


Figure 5: *Categorization trees automatically generated for a set of vases according to SHED, LFD and SPH. The vases are colored according to their shape style. Note that the organization of shapes is more consistent when using SHED (3 categorization errors) than when using LFD or SPH (6 categorization errors each), as seen by the number of shapes with a different color than their lowest level neighbors in the tree.*

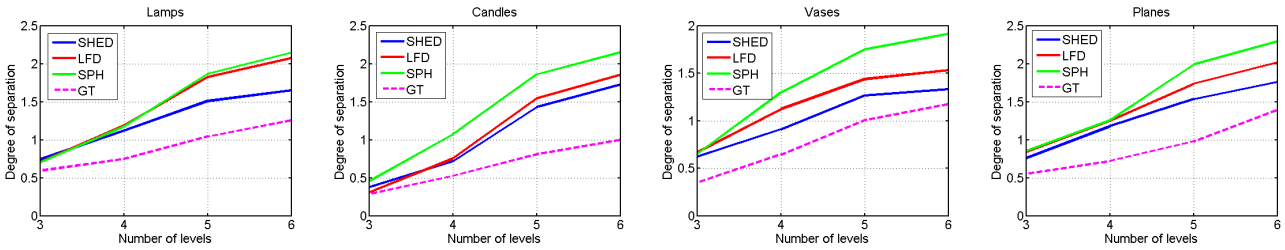


Figure 6: *Comparison of automatically generated trees to ground truth trees, according to the average difference in the degree of separation.*

distance between shapes A and B. Then, a convex quadratic optimization is formulated and solved similarly to a support vector machine. For more details see [Schultz and Joachims 2004].

Using this method, we can fine tune the weights for a specific set of shapes such as lamps or vases. For example, the scaling differences between parts affects the semantic distance between lamps more than it affects the semantic distance between vases. Alternatively, we can use trios from categorization trees of several sets of shapes to learn a global set of weights. Using this method, we propose a set of default weights (see Table 1) that would approximate well the semantic similarity of any set of shapes. Note that these weights also reflect the relations between the different units in which the different costs are measured.

6 Evaluation

The distance between two shapes cannot be directly measured or estimated numerically by a human observer, hence evaluating the accuracy of a similarity measure is somewhat challenging. Still, we are able to compare SHED with state-of-the-art distance measures, namely the light field descriptor (LFD) [Chen et al. 2003] and the spherical harmonic descriptor (SPH) [Kazhdan et al. 2003], and demonstrate its success in various applications. We evaluate the results quantitatively using ground truth data for shape exploration and clustering, and qualitatively for nearest neighbors queries and embedding, where ground truth data is not well defined.

Datasets. We evaluate SHED using three sets of shapes from the COSEG dataset [Wang et al. 2012] and three sets from the Princeton Segmentation Benchmark (PSB) [Chen et al. 2009]. In addition, we collected a set of airplanes from Google Warehouse and other online resources. The set of airplanes and the sets in the COSEG dataset were enriched by introducing finer intra-class variation. The enriched sets include 100 lamps, 80 vases, 70 airplanes, and 40 candelabra, and contain shapes that vary in their part composition, geometry, and articulation. The PSB sets include 20 humans, 20 hands and 20 Teddy bears, which vary mostly in articulation.

Categorization trees. We present an application where categorization trees of shapes are automatically generated for each enriched set. The resulting trees hierarchically organize the shapes in a set and can be used for exploration. The trees are created using Self-Tuning Spectral Clustering [Zelnik-Manor and Perona 2004], which is a non-parametric clustering method, i.e., the number of clusters in each set is selected automatically. We used this method recursively to build a categorization tree for each distance measure (SHED, LFD, and SPH). An example of the generated trees on a subset of shapes is presented in Figure 5, where the shapes are colored according to their shape style. Note that the tree generated using SHED has fewer categorization errors. The generated trees for the full sets can be found in the supplementary material.

To evaluate the quality of the generated trees in a quantitative manner, we use multiple ground truth categorization trees. Since creating a single categorization tree of a set may be subjective, we asked three expert users to independently create a tree for each enriched set. All of the ground truth trees can be found in the supplementary material. The ground truth trees are compared to the generated trees by averaging the difference in the *degree of separation* (DoS) between each pair of shapes in the trees. The DoS is defined as the length of the path between two shapes in the tree [Huang et al. 2013b]. The average difference of DoS measures whether shapes are organized in a similar manner in two trees, without being influ-

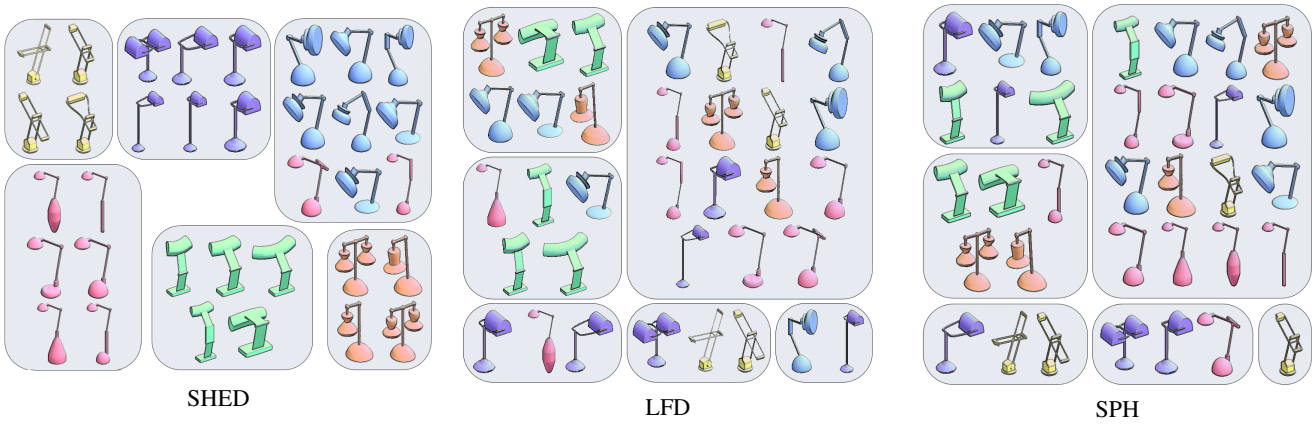


Figure 7: *Comparison of clustering results according to SHED, LFD and SPH on a set of lamps. The shapes are clustered into six groups and colored according to their ground-truth clusters.*

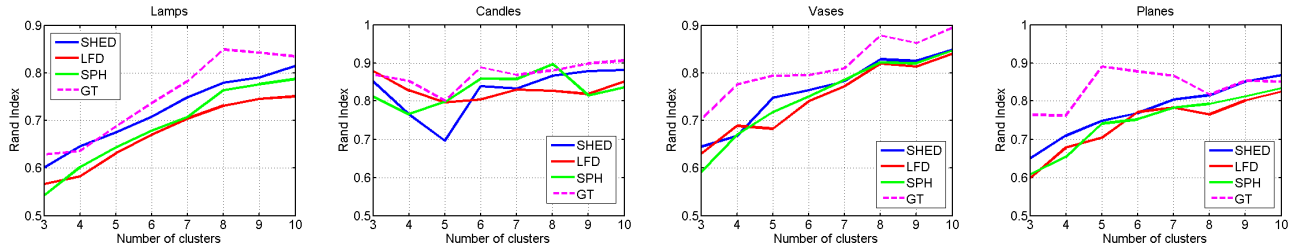


Figure 8: *Comparison of autamatically generated clusterings to ground truth, according to the Rand Index (see text for details).*

enced by the specific structure of each tree. To compare the trees at different levels of granularity, we truncate the trees up to a given number of levels by connecting all the shapes in lower levels directly to their ancestor at the lowest allowed level. The results for a level are given by averaging the difference in DoS over all pairs of shapes and all ground truth trees. The results are shown in Figure 6 (lower values imply trees closer to the ground truth). The curve labeled GT denotes the average difference in DoS between the ground truths. It indicates how much variation exists among the different ground truths and establishes a bound for the accuracy. Note that trimming a tree after two levels effectively provides a quantitative comparison of the first level of clustering. Similarly, trimming the tree after three levels provides a comparison of the clustering generated in the second level, and so on for other levels.

Clustering. In addition to the hierarchical clustering, we also experiment with clustering when the number of clusters is known in advance. We cluster each set of shapes using the self-tuning spectral clustering method mentioned above [Zelnik-Manor and Perona 2004], this time providing the number of clusters as a parameter. We compute ground truth clusterings from each ground truth tree by measuring the degree of separation between every two shapes, and then using the computed DoS as a measure of shape similarity to cluster the shapes with the same clustering method. We generate clusterings according to SHED, LFD, and SPH and measure the difference between the generated clusters and the ground truth using the Rand Index [Chen et al. 2009]. Figure 8 shows the average Rand Index over all ground truths for each set and measure (higher values imply clusters closer to the ground truth). The curve labeled GT denotes the average Rand Index between the ground truth clusterings. It indicates the level of agreement between clusterings generated from different ground truth trees. Figure 7 shows visual results for a subset of lamps.

Shape retrieval. As a shape retrieval experiment, a nearest neighbors search was performed for each shape according to SHED and LFD. Figure 9 shows a selection of shapes from four different sets along with the retrieved nearest neighbors. The full results containing each of the shapes as a query are available in the supplementary material. The distances measured by SHED reflect changes in part composition such as parts that change position on the graph or parts that exist in one shape and not the other, as well as changes in geometry. Therefore, shapes retrieved using SHED tend to have similar part composition. For example, vases tend to have the same number of handles as the query shape (g, i), and candelabra tend to have a similar number of candles (e). In contrast, some of the shapes retrieved by LFD have a different shape structure (c, i). Additionally, SHED retrieves shapes whose parts have a similar geometry to the parts of the query shape (b, f, g, k), whereas shapes retrieved by LFD are more varied. Moreover, SHED deals particularly well with articulations (a), added parts (b), and partial shape matching (h), which pose a challenge to existing methods.

Ground truth data is not well defined for such tasks in intra-class scenarios, where all the shapes belong to the same class. For such scenarios we show qualitative results only. However, for inter-class scenarios, we can quantify how many of the retrieved shapes belong to the same class as the query shape. Figure 11 shows the precision recall curves obtained for all shapes from the PSB sets using SHED, LFD and SPH. The curve labeled "SHED Equal Weights" shows the results when all weights are set to 1. The curve labeled "SHED" shows the results when using the default weights suggested in Section 3. Note that these weights were learned using a different sets of shapes, and the results could potentially be improved further by finetuning the weights specifically for the PSB sets.

Embedding to a lower dimension. Another important application that benefits from defining a more accurate distance measure between shapes is mapping a set of shapes onto a low dimensional

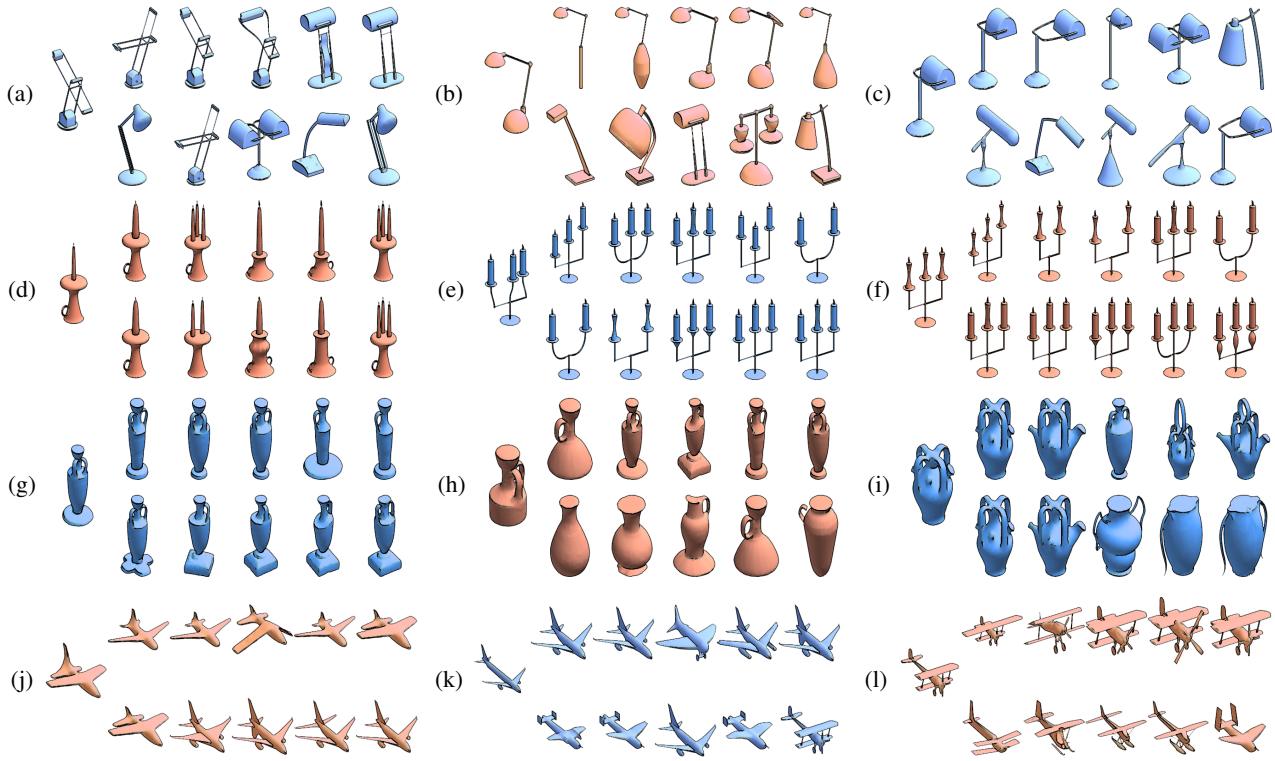


Figure 9: *Nearest neighbors for four sets, ordered by similarity to a query. In each example, the shape on the center left is the query, the first row are the 5 nearest neighbors ordered according to SHED, and the second row are the neighbors ordered according to LFD.*

Set Name	Default	Lamps	Candles	Vases	Airplanes
Geometry	0.4795	0.4376	0.2779	0.4788	0.4285
Scale	0.1258	0.1921	0.1794	0.0256	0.0206
Position	0.0034	0.1216	0.1203	0.1697	0.0047
Duplication	0.3914	0.2486	0.4224	0.5396	0.5462

Table 1: *Learned weights for different sets of shapes. Each column is normalized such that its sum is one.*

manifold. We use standard multi-dimensional scaling (MDS) to generate an embedding of a set of shapes in two dimensions. In Figure 3 we show a toy example comparing the embedding generated by SHED and LFD for a small set of vases. For inter-class similarity estimation, we show in Figure 10 the MDS embedding of shapes from the PSB sets using SHED, LFD, and SPH. The figure clearly shows that SHED produces an intuitive map with a significant distinction between different sets, while LFD and SPH tend to produce less organized maps where shapes of different sets are mixed together. This experiment and the quantitative evaluation in Figure 11 allow us to conclude that SHED is effective when used to separate shapes into different classes (inter-class context), while the previous experiments show that SHED is able to appropriately quantify finer shape differences, which is of importance in an intra-class context.

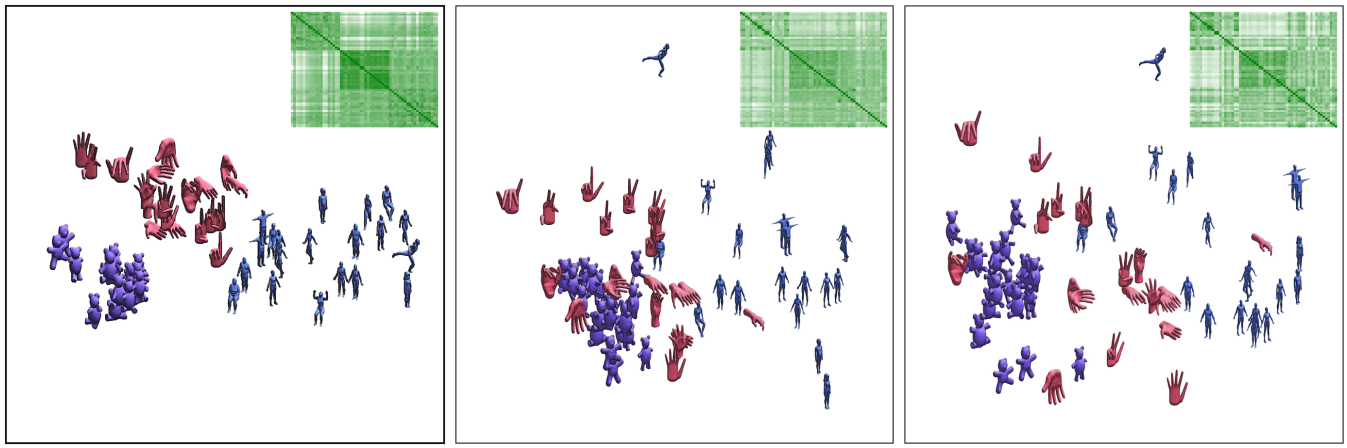
Weights. The weights for the sets of lamps, candles, vases, and airplanes were learned from training sets of 1000 trios each, obtained from the ground truth of each set separately. In addition, default weights were learned using a training set of 1000 trios, obtained from the ground truth of all four sets collectively. The default weights were used to produce the results for the PSB sets in Figures 10 and 11. The weights for each set are given in Table 1.

Timing. Our method can be decomposed into two parts: finding the matching between two shapes and computing the SHED according to a given matching and weights. The computation time of the matching algorithm described in Section 4 depends on the number of parts in each shape, and takes up to 5 seconds for shapes with up to 20 parts. Given the matching and weights, computing the SHED takes a fraction of a second, and the computation of the entire set of 100 lamps, or 4950 pairs of shapes, takes a total of 9 seconds. Segmenting the shapes using [van Kaick et al. 2014] takes up to 5 minutes per shape. Note that the segmentation method can be easily replaced. In some cases the segmentation of shapes can be given as input, in which case the method is very fast to compute.

7 Conclusion

We introduce SHED, an edit distance that quantifies shape similarity based on a structural comparison of shapes. The shape edit distance captures re-arrangements, additions, and removals of parts. We show a variety of applications which benefit from an accurate distance measure between shapes. Finally, we demonstrate that SHED leads to a more intuitive estimation of the similarity between two shapes than state-of-the-art methods, when comparing shapes within the same class as well as shapes from different classes.

Future work and limitations. The current formulation of SHED takes into account the similarity of the shape parts and the shape structure in terms of connectivity of the parts. Additional relationships between parts can be considered, for example, the difference in rotation of pose after an alignment of matched parts. Incorporating pose considerations may constitute an advantage on sets where the pose of the shape parts is one of the main dissimilarity factors, while it may be less suitable for more general sets where pose-invariance is sought.



(a) SHED (b) LFD (c) SPH

Figure 10: *Embedding obtained with multi-dimensional scaling on a set of articulated shapes with three classes. The insets show the distance matrix for each method, where dark green is low distance and white or light green is high distance. Note how SHED groups the shapes into their respective classes, while the distance matrices and embeddings given by LFD and SPH are less organized.*

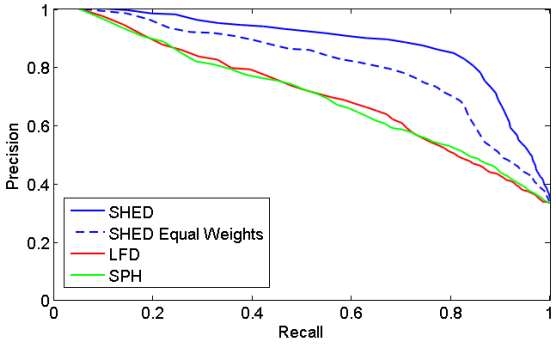


Figure 11: *Precision-recall on sets of articulated shapes.*

An adequate segmentation of the shapes is required for the computation of SHED. In general, segmentation is an ill-posed problem. As a practical solution, we opted to use a segmentation into approximately convex parts, although other segmentation methods can be used. For example, methods that aim at obtaining a close-to-semantic segmentation of the shape are possible, although their usage would require the introduction of more sophisticated measures to compare the geometry of parts.

Finally, distances between shapes are subject to interpretation and are dependent on the semantics of the shapes. Thus, we would like to conduct an investigation to gain insight on how humans perceive finer shape differences, to enhance our edit distance. Quantification of intra-class distances is still an open avenue for further research.

Acknowledgments

We thank the anonymous reviewers for their suggestions. We would like to acknowledge the support of Google Focused Research Award. This work was supported in part by the ERC grant iModel (StG-2012-306877). Oliver van Kaick is grateful to the Azrieli Foundation for the award of an Azrieli Fellowship.

References

ANKERST, M., KASTENMÜLLER, G., KRIEGEL, H.-P., AND SEIDL, T. 1999. 3D shape histograms for similarity search and classification in spatial databases. In *Proc. Int. Symp. Advances in Spatial Databases*, 207–226.

AVERKIOU, M., KIM, V. G., ZHENG, Y., AND MITRA, N. J. 2014. ShapeSynth: Parameterizing model collections for coupled shape exploration and synthesis. *Computer Graphics Forum (Eurographics) 33*.

BARRA, V., AND BIASOTTI, S. 2013. 3D shape retrieval using kernels on extended Reeb graphs. *Pattern Recognition 46*, 11.

BERG, A. C., BERG, T. L., AND MALIK, J. 2005. Shape matching and object recognition using low distortion correspondences. In *Proc. IEEE Conf. on CVPR*, 26–33.

BOMMES, D., ZIMMER, H., AND KOBBELT, L. 2012. Practical mixed-integer optimization for geometry processing. In *Curves and Surfaces*. Springer, 193–206.

BRONSTEIN, A. M., BRONSTEIN, M. M., GUIBAS, L. J., AND OVSJANIKOV, M. 2011. Shape Google: Geometric words and expressions for invariant shape retrieval. *ACM Trans. on Graph 30*, 1, 1:1–20.

CHEN, D.-Y., TIAN, X.-P., SHEN, Y.-T., AND OUHYOUNG, M. 2003. On visual similarity based 3D model retrieval. *Computer Graphics Forum 22*, 3, 223–232.

CHEN, X., GOLOVINSKIY, A., AND FUNKHOUSER, T. 2009. A benchmark for 3D mesh segmentation. *ACM Trans. on Graph (SIGGRAPH) 28*, 3, 73:1–12.

DENNING, J. D., AND PELLACINI, F. 2013. MeshGit: Diffing and merging meshes for polygonal modeling. *ACM Trans. on Graph (SIGGRAPH) 32*, 4, 35:1–10.

FISHER, M., SAVVA, M., AND HANRAHAN, P. 2011. Characterizing structural relationships in scenes using graph kernels. *ACM Trans. on Graph (SIGGRAPH) 30*, 4, 34:1–12.

FUNKHOUSER, T., KAZHDAN, M., SHILANE, P., MIN, P., KIEFER, W., TAL, A., RUSINKIEWICZ, S., AND DOBKIN, D. 2004. Modeling by example. *ACM Trans. on Graph (SIGGRAPH) 23*, 3, 652–663.

GAO, X., XIAO, B., TAO, D., AND LI, X. 2010. A survey of graph edit distance. *Pattern Anal. Appl. 13*, 1, 113–129.

HARCHAOUI, Z., AND BACH, F. 2007. Image classification with segmentation graph kernels. In *Proc. IEEE Conf. on CVPR*, 1–8.

HILAGA, M., SHINAGAWA, Y., KOHMURA, T., AND KUNII, T. L. 2001. Topology matching for fully automatic similarity estimation of 3D shapes. In *Proc.SIGGRAPH*, 203–212.

HUANG, Q., KOLTUN, V., AND GUIBAS, L. 2011. Joint shape segmentation with linear programming. *ACM Trans. on Graph (SIGGRAPH Asia) 30*, 6, 125:1–12.

HUANG, Q.-X., SU, H., AND GUIBAS, L. 2013. Fine-grained semi-supervised labeling of large shape collections. *ACM Transactions on Graphics (TOG) 32*, 6, 190.

HUANG, S.-S., SHAMIR, A., SHEN, C.-H., ZHANG, H., SHEFFER, A., HU, S.-M., AND COHEN-OR, D. 2013. Qualitative organization of collections of shapes via quartet analysis. *ACM Trans. on Graph (SIGGRAPH) 32*, 4, 71:1–10.

KALOGERAKIS, E., CHAUDHURI, S., KOLLER, D., AND KOLTUN, V. 2012. A probabilistic model of component-based shape synthesis. *ACM Trans. on Graph (SIGGRAPH) 31*, 4, 55:1–11.

KAZHDAN, M., FUNKHOUSER, T., AND RUSINKIEWICZ, S. 2003. Rotation invariant spherical harmonic representation of 3D shape descriptors. In *Symp. on Geom. Proc.*, 156–164.

KEZURER, I., KOVALSKY, S. Z., BASRI, R., AND LIPMAN, Y. 2015. Tight relaxation of quadratic matching. *Computer Graphics Forum 24*, 5.

KIM, V. G., LI, W., MITRA, N. J., DIVERDI, S., AND FUNKHOUSER, T. 2012. Exploring collections of 3D models using fuzzy correspondences. *ACM Trans. on Graph (SIGGRAPH) 31*, 4, 54:1–11.

KIM, V. G., LI, W., MITRA, N. J., CHAUDHURI, S., DIVERDI, S., AND FUNKHOUSER, T. 2013. Learning part-based templates from large collections of 3D shapes. *ACM Trans. on Graph (SIGGRAPH) 32*, 4, 70:1–12.

KLEIMAN, Y., FISH, N., LANIR, J., AND COHEN-OR, D. 2013. Dynamic maps for exploring and browsing shapes. *Computer Graphics Forum (SGP) 32*, 5, 187–196.

KURTEK, S., SRIVASTAVA, A., KLASSEN, E., AND LAGA, H. 2013. Landmark-guided elastic shape analysis of spherically-parameterized surfaces. *Computer Graphics Forum 32*, 2pt4, 429–438.

LAGA, H., MORTARA, M., AND SPAGNUOLO, M. 2013. Geometry and context for semantic correspondences and functionality recognition in man-made 3D shapes. *ACM Trans. on Graph 32*, 5, 150:1–16.

LEORDEANU, M., AND HEBERT, M. 2005. A spectral technique for correspondence problems using pairwise constraints. In *Proc. Int. Conf. on Comp. Vis.*, 1482–1489.

LI, B., GODIL, A., AONO, M., BAI, X., FURUYA, T., LI, L., LOPEZ-SASTRE, R., JOHAN, H., OHBUCHI, R., REDONDO-CABRERA, C., TATSUMA, A., YANAGIMACHI, T., AND ZHANG, S. 2012. SHREC'12 track: Generic 3D shape retrieval. In *Eurographics Workshop on 3D Object Retrieval (3DOR)*, M. Spagnuolo, M. Bronstein, A. Bronstein, and A. Ferreira, Eds.

LITMAN, R., BRONSTEIN, A., BRONSTEIN, M., AND CASTELLANI, U. 2014. Supervised learning of bag-of-features shape descriptors using sparse coding. *Computer Graphics Forum 33*, 5, 127–136.

LYZINSKI, V., FISHKIND, D., FIORI, M., VOGELSTEIN, J., PRIEBE, C., AND SAPIRO, G. 2015. Graph matching: relax at your own risk. *IEEE TPAMI*.

MITRA, N. J., WAND, M., ZHANG, H., COHEN-OR, D., AND BOKELOH, M. 2013. Structure-aware shape processing. In *Proc. Eurographics State-of-the-art Reports*.

NEUHAUS, M., AND BUNKE, H. 2007. *Bridging the Gap Between Graph Edit Distance and Kernel Machines*. World Scientific, River Edge, NJ, USA.

OSADA, R., FUNKHOUSER, T., CHAZELLE, B., AND DOBKIN, D. 2002. Shape distributions. *ACM Trans. on Graph 21*, 4, 807–832.

OVSJANIKOV, M., MÉRIGOT, Q., MÉMOLI, F., AND GUIBAS, L. 2010. One point isometric matching with the heat kernel. *Computer Graphics Forum (SGP) 29*, 5, 1555–1564.

OVSJANIKOV, M., LI, W., GUIBAS, L., AND MITRA, N. J. 2011. Exploration of continuous variability in collections of 3D shapes. *ACM Trans. on Graph (SIGGRAPH) 30*, 4, 33:1–10.

RUSTAMOV, R. M. 2007. Laplace-Beltrami eigenfunctions for deformation invariant shape representation. In *Symp. on Geom. Proc.*, 225–233.

SCHULTZ, M., AND JOACHIMS, T. 2004. Learning a distance metric from relative comparisons. *Advances in neural information processing systems (NIPS)*, 41.

SEBASTIAN, T., KLEIN, P., AND KIMIA, B. 2004. Recognition of shapes by editing their shock graphs. *IEEE Trans. Pat. Ana. & Mach. Int. 26*, 5, 550–571.

SHAMIR, A. 2008. A survey on mesh segmentation techniques. *Computer Graphics Forum 27*, 6, 1539–1556.

SHAPIRA, L., SHAMIR, A., AND COHEN-OR, D. 2008. Consistent mesh partitioning and skeletonisation using the shape diameter function. *The Visual Computer 24*, 4, 249–259.

SIDI, O., VAN KAICK, O., KLEIMAN, Y., ZHANG, H., AND COHEN-OR, D. 2011. Unsupervised co-segmentation of a set of shapes via descriptor-space spectral clustering. *ACM Trans. on Graph (SIGGRAPH Asia) 30*, 6, 126:1–10.

SUNDAR, H., SILVER, D., GAGVANI, N., AND DICKINSON, S. 2003. Skeleton based shape matching and retrieval. In *Shape Modeling International*, 130–139.

TANGELDER, J. W. H., AND VELTKAMP, R. C. 2008. A survey of content based 3D shape retrieval methods. *Multimedia Tools and Applications 39*, 3, 441–471.

VAN KAICK, O., FISH, N., KLEIMAN, Y., ASAFI, S., AND COHEN-OR, D. 2014. Shape segmentation by approximate convexity analysis. *ACM Trans. Graph. 34*, 1, 4:1–11.

WANG, Y., ASAFI, S., VAN KAICK, O., ZHANG, H., COHEN-OR, D., AND CHEN, B. 2012. Active co-analysis of a set of shapes. *ACM Trans. on Graph (SIGGRAPH Asia) 31*, 6, 157:1–10.

XU, K., LI, H., ZHANG, H., COHEN-OR, D., XIONG, Y., AND CHENG, Z. 2010. Style-content separation by anisotropic part scales. *ACM Trans. on Graph (SIGGRAPH Asia) 29*, 6.

ZELNIK-MANOR, L., AND PERONA, P. 2004. Self-tuning spectral clustering. In *NIPS*, vol. 17, 1601–1608.

ZHENG, Y., COHEN-OR, D., AVERKIOU, M., AND MITRA, N. J. 2014. Recurring part arrangements in shape collections. *Computer Graphics Forum (Eurographics) 33*.