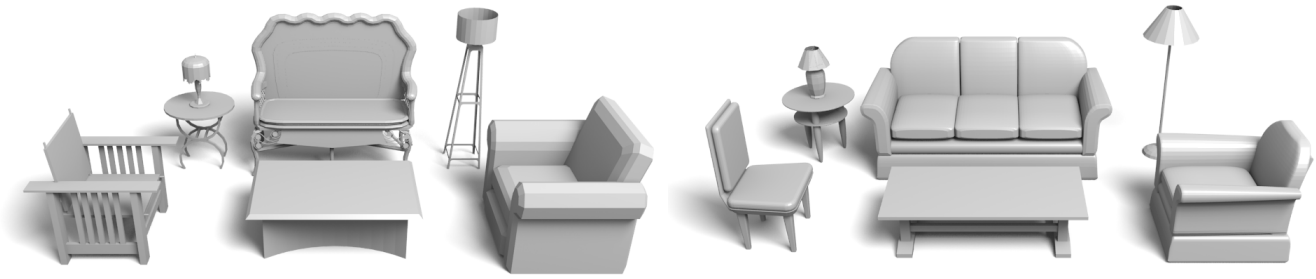
Style Compatibility for 3D Furniture Models

Tianqiang Liu[1] Aaron Hertzmann[2] Wilmot Li[2] Thomas Funkhouser[1]
[1]Princeton University [2]Adobe Research



(a) *Ignoring style compatibility* **(b)** *Optimizing style compatibility*

Figure 1: *This paper proposes a method to learn a metric for stylistic compatibility between furniture in a scene. (a) The image on the left shows a plausible furniture arrangement, but with a randomly chosen mix of clashing furniture styles. The scene on the right has the same arrangement but with furniture pieces chosen to optimize stylistic compatibility according to our metric.*

Abstract

This paper presents a method for learning to predict the stylistic compatibility between 3D furniture models from different object classes: e.g., how well does this chair go with that table? To do this, we collect relative assessments of style compatibility using crowdsourcing. We then compute geometric features for each 3D model and learn a mapping of them into a space where Euclidean distances represent style incompatibility. Motivated by the geometric subtleties of style, we introduce part-aware geometric feature vectors that characterize the shapes of different parts of an object separately. Motivated by the need to compute style compatibility between different object classes, we introduce a method to learn object class-specific mappings from geometric features to a shared feature space. During experiments with these methods, we find that they are effective at predicting style compatibility agreed upon by people. We find in user studies that the learned compatibility metric is useful for novel interactive tools that: 1) retrieve stylistically compatible models for a query, 2) suggest a piece of furniture for an existing scene, and 3) help guide an interactive 3D modeler towards scenes with compatible furniture.

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

Keywords: style, compatibility, crowdsourcing, scene synthesis

ACM Reference Format
Liu, T., Hertzmann, A., Li, W., Funkhouser, T. 2015. Style Compatibility For 3D Furniture Models. ACM Trans. Graph. 34, 4, Article 85 (August 2015), 9 pages. DOI = 10.1145/2766898 http://doi.acm.org/10.1145/2766898.

ACM 978-1-4503-3331-3/15/08 ... $15.00.
DOI: http://dx.doi.org/10.1145/2766898

1 Introduction

Modeling 3D scenes is one of the most common creative tasks in computer graphics. Large online repositories of 3D models make it possible for novice users and/or automatic programs to create new scenes by assembling models of objects found online. For example, the arrangement of furniture shown in Figure 1(a) was created with a tool that allows a person to select and place objects interactively.

While many existing tools help users select the appropriate categories and placements of objects when modeling a 3D scene [Merrell et al. 2011; Yu et al. 2011], they generally ignore style compatibility — the degree to which objects "exist together in harmony" [Merriam-Webster 2004]. For example, while the scene shown in Figure 1(a) has a plausible spatial arrangement of objects appropriate for a living room, it contains a mish-mash of different styles — e.g., a casual contemporary coffee table appears in front of a formal antique sofa. The jarring juxtaposition of incompatible styles diminishes the plausibility of this scene.

The goal of this paper is to develop a mathematical representation of style compatibility between objects that can be used to guide 3D scene modeling tools. More specifically, we consider the compatibility of furniture in indoor scenes, because furniture exhibits a diverse range of distinct styles, some of which are more compatible than others, and because indoor scenes account for a large fraction of scene modeling tasks. Our work focuses on understanding how the geometry of 3D models influences their stylistic compatibility. We leave the study of compatibility for other properties (materials, colors, etc.) for future work.

There are three main challenges in developing a style compatibility metric for furniture shapes. First, a person's notion of furniture style usually combines many subtle factors [Miller 2005] that would be hard to encode in a hand-tuned mathematical function. Instead, we learn a metric from examples. Second, furniture shapes are influenced by both function and style, with functional requirements reflected largely in the gross shapes and arrangements of parts, and styles reflected largely in the geometric details of parts (e.g., fluted legs, wing-tipped backs, etc.). Accordingly, we introduce part-aware shape features aimed at capturing the geometric details related to style. Third, style compatibility requires com-

parisons of models from different object classes, which may have different dimensionalities and distributions of features. We introduce a compatibility metric based on class-specific mappings that transform geometric features to a class-independent feature space.

In our approach, we first collect crowdsourced preference data about which furniture models people consider to be compatible (Section 3). Our system performs a consistent segmentation of models within the same object class and computes a part-aware geometric feature vector for each model (Section 4). We then learn a compatibility metric using the part-aware geometric features and crowdsourced preferences (Section 5); quantitative evaluation shows that this method gives more accurate predictions than existing methods (Section 6). The learned metric can then be used to reason about style compatibility in retrieval and interactive modeling applications (Section 7).

The main contribution of our work is in proposing the first method for learning style compatibility between 3D models from different object classes. In particular, our method introduces a part-aware geometric feature vector that encodes style-dependent information, and presents a new asymmetric embedding distance that is appropriate for estimating compatibility between objects of different classes. Furthermore, we demonstrate the utility of our learned metric in three novel style-aware scene modeling applications: retrieving furniture that is stylistically compatible with a query; suggesting a piece of furniture to include in an existing room; and helping people interactively build scenes with compatible furniture.

2 Related Work

Shape styles. Researchers have developed methods to model styles of 2D and 3D shapes [Xu et al. 2010; Li et al. 2013; Ma et al. 2014]. For example, Xu et al. [2010] investigate style variations that are caused by anisotropic part scaling. Li et al. [2013] propose a method to classify and synthesize 2D shapes styles according to curvature features. Rather than focusing on a specific source of shape style variation, our work develops a method to quantify style compatibility, which is usually determined by multiple aspects of shape styles.

Huang et al. [2013] propose a fine-grained classification approach by learning a distance metric from a 3D model collection with partial and noisy labels. Kalogerakis et al. [2012] develop a probabilistic model based on shape similarity and learned probability distributions for co-occurrences of discrete clusters of parts. Both of these approaches aim to capture style variations within a 3D model collection of a single object class and are not directly applicable to predicting style compatibility for different object classes within a scene.

Similarity metric learning. Our work is related to similarity metric learning in other domains. Researchers have used crowdsourced data to learn similarity metric for fonts [O'Donovan et al. 2014] and illustration styles [Garces et al. 2014]. Relative attributes have been learned from visual features for image analysis [Parikh and Grauman 2011] and from shapes for 3D model creation [Chaudhuri et al. 2013]. We build on this work, but focus on learning compatibility rather than similarity, which entails a different crowd study design, as well as a new distance function for heterogeneous data (e.g., comparing tables to chairs).

In concurrent work, Lun et al. [2015] measure 3D style similarity based on similarity of object elements. Their method, unlike ours, may generalize to object classes unseen in the training data. However, theirs may perform poorly for objects without corresponding elements (e.g., floor lamp and sofa). In contrast, our work aims to quantify style similarity across classes, and is not dependent on geometric similarity.

Figure 2: Style compatibility study. *In each task, we fix one furniture piece (e.g., the dining table), and show six different pieces of another object class (e.g., dining chair) with it. In each pair, the two furniture pieces are shown in the arrangement of a typical scene (e.g., chair next to table). We ask the rater to select the two pairs that are stylistically most compatible. In this example, most raters select the bottom-middle and the bottom-right pairs.*

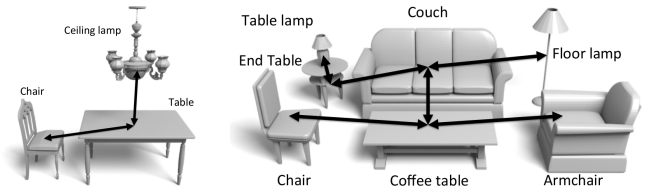
Shape-based retrieval. Our work is informed by prior work on shape-based retrieval of 3D models. Researchers previously have developed search algorithms for 3D models based on similarities in shapes [Funkhouser et al. 2003], symmetries [Kazhdan et al. 2004], part structures [Shapira et al. 2010], and other geometric cues [Tangelder and Veltkamp 2008]. However, these methods are generally geared toward retrieving similar shapes from the same object class. No previous system has considered stylistic compatibility between objects of different classes in a 3D model retrieval system.

Virtual world synthesis. Several systems have been developed to assist people in creating virtual worlds by combining 3D models from online repositories, including ones that suggest new objects for a scene based on spatial context [Fisher and Hanrahan 2010], probabilistic models [Chaudhuri et al. 2011; Kalogerakis et al. 2012], physical simulations [Umetani et al. 2012], and interior design guidelines [Merrell et al. 2011]. Other systems have aimed to create scenes completely automatically, for example learning object compatibilities based on substructure symmetries [Zheng et al. 2013], spatial contexts [Fisher et al. 2012; Fisher et al. 2011], and object contacts [Akazawa et al. 2005]. However, no prior system has explicitly considered stylistic compatibility when selecting objects to combine in a virtual world.

3 Crowdsourcing Compatibility Preferences

The first step in our process is to collect data for object compatibility using crowdsourcing. Our study is based on previous methods for crowdsourcing similarity. However, we focus on style compatibility rather than similarity, and modify the study questions appropriately. We gather compatibility preferences in the form of triplets (A, B, C). Each triplet represents a human evaluation whether reference object A is more compatible with object B or with object C. For example, given a sofa A, a human rater might be asked to judge whether chair B or chair C is more compatible with it. B and C are always from the same object class, and A is always from a different class.

To gather triplets efficiently, we use the grid technique proposed by Wilber et al. [2014]. Each task evaluates six target objects together with a reference object A. The worker is shown a grid of six images, each one pairing A together with a different target object (Figure 2). The rater must select the *two* target objects that are most compatible with A. Each response is then converted to 8 triplets: each of which consists of one object that is selected, one object that is not selected,



(a) *Dining room* **(b)** *Living room*

Figure 3: *Pairs of object classes for which style compatibility preferences are collected in our study. We chose pairs with close proximity and functional interactions.*

and the reference object. As demonstrated by Wilber et al. [2014], this format is more efficient than asking the participant to pick the best between two.

In our experiments, we first collected 3D furniture models for two types of scenes, dining rooms and living rooms, from the Digimation Archive model collection and Trimble 3D Warehouse. We collected 3 object classes for dining rooms: 50 dining chairs, 34 dining tables, and 21 ceiling lamps. We collected 7 object classes for living rooms: 49 coffee tables, 39 sofas, 37 chairs, 36 arm chairs, 42 end tables, 28 table lamps, and 23 floor lamps.

Then, we crowdsourced preference data for all pairs of classes linked in Figure 3 using Amazon Mechanical Turk (AMT). For each pair of object classes, we randomly generated 50 questions and picked 4 to use as control questions to test participant consistency; responses to the control questions were not used for learning. We split the questions into two Human Intelligence Tasks (HITs). Each HIT includes 25 questions and the 4 control questions, and each control question is asked twice with images in different orders. Each HIT was done by 50 different participants. To filter out lazy participants, we excluded a response if the participant (1) spent less than 15 seconds per question on average (filtering 57% of all responses) or (2) had less than 5 selections in common amongst the two sets of control questions (filtering 40% of the responses that are kept from (1)). This process yielded 20,200 responses (on 2956 unique triplets) for objects found in dining rooms, and 63,800 responses (on 8909 unique triplets) for objects in a living room.

Analyzing this data, we find that raters on AMT strongly agree on a minority, but significantly-larger-than-random, subset of the triplets. Among the 1,919 unique living room triplets for which at least 10 valid responses were collected, the number where 90% of raters agreed is 5 times larger than random (10% vs. 2%), and it is 7 times larger for the 598 such triplets in dining rooms (14% vs. 2%). This result is consistent with our subjective impression that each object in our data set contains just a few others for which it is strongly compatible (e.g., an IKEA table and an IKEA chair) or incompatible (e.g., an IKEA table with an ornate antique chair). People detect these important cases consistently, but there are many cases for which triplet comparisons are not meaningful, e.g., an IKEA table with a Queen Anne chair versus a Chippendale chair. We leave exploration of this particular hypothesis for future work.

4 Part-aware Geometric Features

Our next goal is to define a feature vector $\mathbf{x}$ of geometric properties indicative of an object's style. This problem is challenging because stylistic differences between objects in the same class are often due to subtle deviations from a common overall shape. Therefore, often-used shape descriptors geared for object classification, which aim to capture the overall shape of an object, are not appropriate for our task.

Our key observation is that furniture styles are often strongly connected to characteristic features of individual parts. For example, chairs of Queen Anne style often have cabriole legs and vase-shaped splats, while chairs of Gustavian style have fluted legs and oval-shaped backs [Miller 2005]. As a result, we are motivated to describe objects with part-aware geometric features.

Our approach is to compute a consistent segmentation of all objects within the same class, compute geometric features for each part separately, and then represent each object by the concatenation of feature vectors for all of its parts and its entire shape. This approach has the advantage that distinctive features of one part are not blended with features of another. For example, curvature histograms computed for the carved back of a Chippendale chair are kept separate from those of its smooth seat cushion. The result is a part-aware geometric feature vector that is better suited for characterizing styles. Unlike previous methods, our approach produces feature vectors with higher dimensionality for object classes with more parts.

Our implementation leverages the consistent segmentation algorithm of Kim et al. [2013]. Given a collection of models of the same object class and a single template, the algorithm produces a consistent segmentation and labeling for all models. For each labeled part and for the entire shape, we compute a geometric feature vector with 79 dimensions representing curvatures for different neighborhoods, shape diameter functions, bounding box dimensions, and surface areas, all computed with methods based on Kalogerakis et al. [2012]. Please refer to the supplemental materials for details.

5 Learning Compatibility

Given the crowdsourced triplet data and the part-aware geometric features, our next goal is to learn a measure of the compatibility between a pair of models from different object classes. In particular, let $\mathbf{x}_i$, $\mathbf{x}_j$ be feature vectors for models i and j, possibly with different dimensionalities. We want a function $d(\mathbf{x}_i, \mathbf{x}_j)$ that scores compatibility, with lower values being more compatible. A key challenge is that different object classes may have feature vectors with different elements, and so direct distance computation is not possible.

Previous work [Kulis 2012; O'Donovan et al. 2014; Garces et al. 2014] has employed distance functions of the form

$$d_{\text{symm}}(\mathbf{x}_i, \mathbf{x}_j) = ||\mathbf{W}(\mathbf{x}_i - \mathbf{x}_j)||_2 \quad (1)$$

In the simplest case, $\mathbf{W}$ may be a diagonal matrix, representing scaled Euclidean distance between feature vectors. Alternatively, it may be represented as a $K \times D$ *embedding* matrix that projects the input feature into a K-dimensional space for comparison [Kulis 2012]. The above distance assumes that all objects have the same type of feature vectors.

In order to handle heterogeneous furniture types, we propose to learn a separate embedding matrix $\mathbf{W}_c$ for each class c. The distance function is then:

$$d_{\text{asymm}}(\mathbf{x}_i, \mathbf{x}_j) = ||\mathbf{W}_{c(i)}\mathbf{x}_i - \mathbf{W}_{c(j)}\mathbf{x}_j||_2 \quad (2)$$

In other words, objects are compared by first projecting them into a shared, K-dimensional embedding space, but using a separate projection matrix for each class. For example, a table would be projected as $\mathbf{y}_1 = \mathbf{W}_{\text{table}}\mathbf{x}_1$, which could then be compared to a chair $\mathbf{y}_2 = \mathbf{W}_{\text{chair}}\mathbf{x}_2$ (Figure 4). We refer to this as the *asymmetric embedding distance*. This model is related to Canonical Correlation

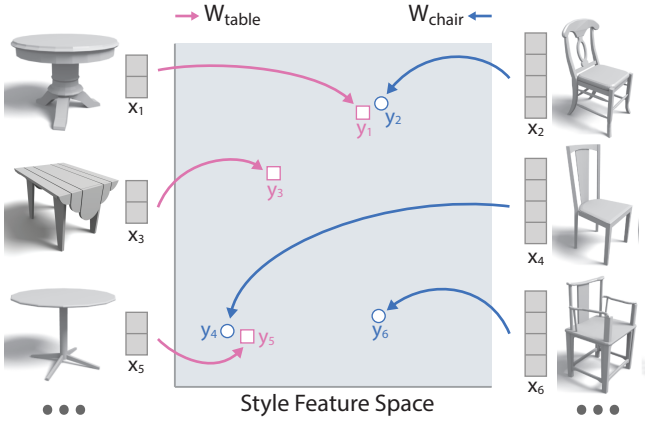


Figure 4: Mapping into shared feature space. *We learn separate embedding matrices for each object class that map shapes into a shared feature space where objects that are stylistically compatible are close to each other. Here, old-fashioned chairs and tables are clustered at the top, while modern objects are clustered in bottom left. Feature vectors for different classes may have different dimensionality based on the number of parts.*

Analysis [Hotelling 1936] and Neighborhood Components Analysis [Goldberger et al. 2004], but trained in a supervised manner in order to predict compatibility from triplets.

Note that this formulation does not require that each vector even have the same dimensionality; in principle, it could be used for compatibility of different types of entities, such as material and geometry, or images and colors.

Given a distance function, learning proceeds similar to previous work [O'Donovan et al. 2014; Garces et al. 2014]. The probability that a rater evaluates object A as more compatible to B than to C is modeled as a logistic function:

$$P^A_{B,C} = \frac{1}{1 + \exp(d(\mathbf{x}_A, \mathbf{x}_B) - d(\mathbf{x}_A, \mathbf{x}_C))} \tag{3}$$

Learning is performed by minimizing the negative log-likelihood of the training triplets $\mathcal{D}$ with regularization:

$$E(\mathbf{W}_{1:M}) = -\frac{1}{|\mathcal{D}|} \sum_{(A,B,C)\in\mathcal{D}} \log P^A_{B,C} + \frac{\lambda}{M} \sum_{1\le c\le M} R(\mathbf{W}_c) \tag{4}$$

where $|\mathcal{D}|$ is the number of triplets, M is the number of object classes, and R is the regularization term.

For learning, we represent each mapping as a product of two matrices: $\mathbf{W}_c = \mathbf{W}_c^{opt}\mathbf{W}_c^{PCA}$. The first matrix $\mathbf{W}_c^{PCA}$ is obtained by performing 21-dimensional Principal Components Analysis (PCA) on each object class separately. The second matrix is the obtained by optimizing $E(\mathbf{W}_{1:M})$ using BFGS [Zhu et al. 1997], for either the symmetric or asymmetric model. The PCA step is used for two reasons: first, it allows us to directly compare the symmetric and asymmetric models, since the symmetric model cannot be used on our heterogeneous input features; second, it makes optimization faster, since the input dimensionality is very large.

Regularization can be used to perform feature selection, i.e., to zero out the weights for dimensions after PCA analysis. Feature selection is used because we believe that some of the dimensions are not helpful for measuring compatibility, but it is difficult a priori to know which dimensions are necessary. Previous work has used the L1-norm to sparsify weight vectors [Garces et al. 2014]. However, applying the L1-norm to the entries of the embedding matrix ($R(\mathbf{W}_c) = |\mathbf{W}_c|_1$), would only have the effect of sparsifying individual matrix entries, rather than eliminating entire dimensions.

We instead use Group Sparsity for regularization:

$$R(\mathbf{W}_c) = \frac{1}{KD} \sum_{1\le\ell\le D} ||\mathbf{w}_\ell||_2 \tag{5}$$

which applies the L2-norm to each column $\mathbf{w}_\ell$ of the component $\mathbf{W}_c^{opt}$ of the embedding matrix $\mathbf{W}_c$. As shown by Bach et al. [2012], this has the effect of sparsifying entire columns of the matrix. It can be interpreted as applying the L1 norm to the magnitude of the column; it is a generalization of applying L1 to the individual matrix entries.

We use $K = 8$ and $\lambda = 2$ in all experiments, except where noted. Our algorithm was implemented in Python. We set the maximum number of iterations in BFGS to be 2000, and it takes up to 70 minutes to solve $\mathbf{W}_{1:M}$.

Method	Dining room	Living room
Chance	50%	50%
Euclidean	69%	58%
Ours	73%	72%
People	93%	99%

Table 1: *Accuracy of style compatibility rankings generated by random, Euclidean distance on non-part-aware features (with PCA), our method, and people for triplets of furniture models. The test set is filtered for consistency among human raters, hence the high scores among human raters.*

6 Results: Triplet Prediction

We ran a series of experiments to test how well our algorithm is able to predict the relative style compatibility between furniture of different object classes, using hold-out triplet data. We consider our basic algorithm, as well as variants with the novel aspects described in the previous sections disabled to investigate their impact on the results. This section presents a summary of the results – please refer to the supplemental materials to see the complete set of experimental data.

As discussed previously, many of the triplets in the crowdsourcing study described in Section 3 are inherently ambiguous, e.g., two furniture items are equally similar or dissimilar to the reference object. Thus, we formed the test set for our prediction experiments by only considering triplets where raters had strong agreement. Specifically, we include a crowdsourced triplet in the test set if: (1) at least 75% of raters agreed on the most-compatible object, and (2) there are sufficient numbers of triplets to estimate this percentage, evaluated by a Binomial Test and comparing the p-value to a threshold. Aiming for ~ 250 triplets, we set the threshold to 0.05 for the dining room test set, which yields 264 triplets; and 0.01 for the living room test set, which yields 229 triplets (the null hypothesis is that people do not have a preference in the triplet).

For each test triplet, we trained our algorithm using all crowdsourced triplets in the same scene (living room or dining room) that do not share any data with the test triplet. We then ran our algorithm to predict the relative compatibility of the two candidate objects and check to see whether it matches the object selected by the majority of people. Our overall *accuracy* measure is the percentage of triplets for which the prediction matches.

Method	Dining room	Living room
No part-aware, Symmetric	63%	55%
Part-aware, Symmetric	63%	65%
No part-aware, Asymmetric	68%	65%
Part-aware, Asymmetric (Ours)	73%	72%

Table 2: Impact of part-aware features and asymmetric embedding. *Accuracy of style compatibility rankings for our algorithm with and without part-aware features and asymmetric embedding enabled.*

Training set	Dining room	Living room
Same-task triplets	61%	65%
Same-pair triplets	73%	66%
All triplets (Ours)	73%	72%

Table 3: Impact of shared models. *Accuracy of style compatibility rankings for our algorithm using different training sets.*

Table 1 shows the overall accuracy of our algorithm (*Ours*) in comparison to a random prediction (*Random*) and to selections made by people (*People*). Since the test set only contains triplets for which people strongly agree, it is not surprising that they have high accuracy in this evaluation (93%, 99%). Euclidean distance on PCA-reduced feature vectors (*Euclidean*) performs above chance, though not much better for the more-complex living room arrangement. Our method does not achieve as high accuracy (73%, 72%) as people, but it does perform significantly better than random (50%, 50%).

By using Group Sparsity, we discard 0 to 9 of the 21 input PCA dimensions, depending on the object class experimented on.

Impact of part-aware geometric features. We ran a second experiment to test how part-aware geometric features help capture style characteristics. To test this, we compare our results to an alternative method, in which the same set of geometric features are computed without separating them according to the consistent part segmentation. Results are shown in Table 2. The accuracy of our method (73%, 72%) is clearly better than that of the method without part-aware features (68%, 65%).

Impact of asymmetric embedding. In a third experiment, we test whether asymmetric embedding outperforms symmetric embedding. Since asymmetric embedding has more free variables than symmetric embedding, it is not suitable to assume symmetric embedding works with the same set of parameters as our algorithm. To make a fair comparison, we ran the method of symmetric embedding multiple times with different combinations of parameters, and compared the best result to the one of our method with the default parameters ($K = 8, \lambda = 2$). Results are shown in Table 2. The performance of our algorithm (73%, 72%) is significantly better than the alternative method (63%, 65%). These results indicate that it is beneficial to learn different embeddings for different object classes, since the geometric features of different object classes are usually incomparable.

One key finding is the importance of using part-aware features together with the asymmetric model. For the dining room scenario, part-aware features provide no advantage when used with the symmetric model. This is likely because the symmetric model is not designed for heterogeneous features. However, when combined with the asymmetric model, they provide a significant boost over the non-part-aware features.

Impact of shared models. In a fourth experiment, we test whether it is better to learn a single model for all possible compatibility tasks, or to learn separate models for each type of object pairing. We will use the notation $X \rightarrow Y$ to denote the task of, given an object of class X, return compatible objects of type Y. For example, $chair \rightarrow table$ is the task of finding a table to go with a specific chair, and a triplet of this type would ask which of two tables better matches a given chair.

We compare the following alternatives:

- **Same-task triplets.** For each possible task ($X \rightarrow Y$), a separate model is learned from the $X \rightarrow Y$ subset of the training triplets. Depending on the number of tasks involved (Figure 3), an object class can have up to 6 different embedding matrices $\mathbf{W}_c$ associated with it, one for each of its tasks.
- **Same-pair triplets.** For each pair of objects ($X \rightarrow Y$ and $Y \rightarrow X$ tasks), a separate model is learned from the corresponding subset of the training triplets. An object class can have 1 to 3 different embedding matrices associated with it.
- **All triplets.** One embedding matrix is learned for each class, jointly optimized over all training triplets.

The same dimensionality is used in each case, and so far fewer parameters are learned in our method than in the alternative methods.

Results are shown in Table 3. The results of our method are better than or comparable to the method of training on a subset of the triplets, which indicates that more training examples are helpful, even when they are not examples of the same kind of task.

7 Applications

In this section, we investigate the utility of the proposed style compatibility metric in three applications: shape retrieval, furniture suggestion, and scene building. In each case, a classic application in computer graphics is extended to consider style compatibility.

7.1 Style-aware shape retrieval

In many cases, people want to retrieve models that are stylistically compatible to a query model of a different object class. For example, in an online furniture shopping system, a potential customer may look for dining chairs that are compatible to the dining table in his/her dining room.

To investigate this application, we have implemented a style-aware shape retrieval system. The system asks the user to give a query model and a target object class, and then it returns a ranked list of models in the target object class that are most compatible to the query according to the metric learned by our algorithm.

Figure 5 shows two examples. Given a dining table on the left, our system returns the 5 dining chairs shown on the left as the most compatible stylistically (for comparison, the 5 least compatible are shown on the right). We observe that these retrieval results are generally consistent with our expectations: the system returns heavy and ornamented chairs when the query table is heavy and ornamented, and it returns chairs with simple designs to match the more streamlined table. Though these are just two examples, retrieval results for all models are available in the supplemental material.

7.2 Style-aware furniture suggestion

When people create a virtual scene or furnish their homes, they may want to search for a model of a known object class that is compatible to the rest of the scene. For example, the user may want to find a coffee table compatible with a particular sofa, chair, and

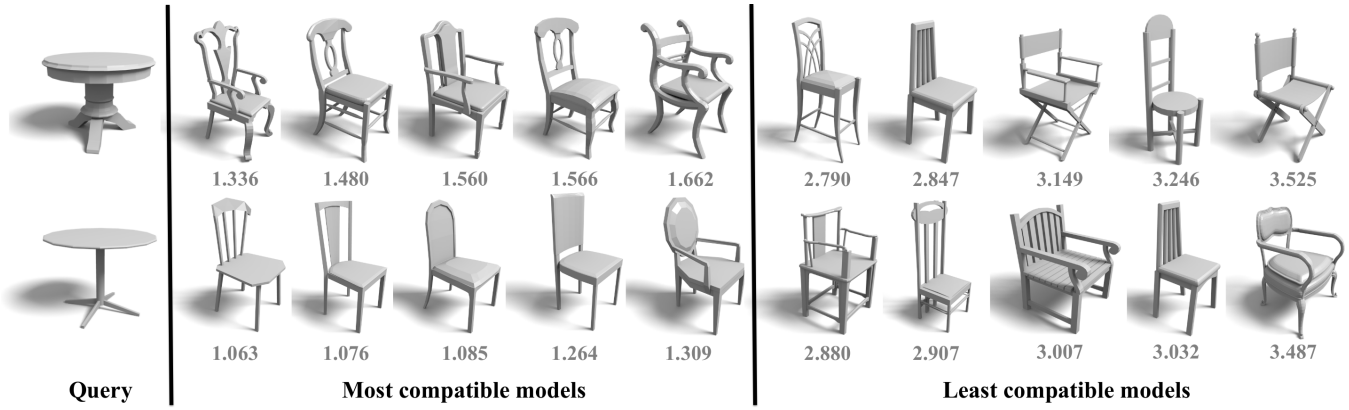


Figure 5: Style-aware shape retrieval. *Given a query (dining table), our system returns 5 dining chairs that are most stylistically compatible to the query as predicted by our learned metric. We also list the 5 most incompatible models for comparison. The numbers are the compatibility distance d between the chairs and the query, with lower values being more compatible.*

Figure 6: Style-aware furniture suggestion. *Both scenes are manually created by people except for the coffee tables. Our system suggests different coffee tables given different sets of furniture pieces in the scene to maximize style compatibility.*

end table that currently are in his living room. In contrast to the style-aware shape retrieval application, the suggestions should be compatible with multiple objects in a scene, rather than a single object.

We have implemented a style-aware furniture suggestion system to test this application. We define a compatibility energy for an entire scene as the sum of compatibility distances between all objects in the scene:

$$F(\{x_i\}) = \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{P}} d(\mathbf{x}_i,\mathbf{x}_j) \tag{6}$$

where $\{\mathbf{x}_i\}$ are the models in the scene, $\mathcal{P}$ is the set of linked model pairs shown in Figure 3, and $d(\mathbf{x}_i,\mathbf{x}_j)$ is the compatibility distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ (Equation 2). Then, given the query object class, we enumerate all candidate models of the object class, and return the one that minimizes the compatibility energy F.

Evaluation. We conducted user studies to evaluate the quality of suggestions made by our algorithm compared to random suggestions and people's selections. To generate test data, we asked people to create stylistically compatible living room scenes using the experimental setup described in the following subsection. In total, participants created 14 sets of scenes that correspond to the 14 different starting configurations of the room. From this data, we selected exactly 14 test scenes by selecting the most compatible scene for each starting configuration, as judged by workers on AMT. Then, for each test scene, we removed objects one at a time and used our system to automatically suggest a replacement. We also generated 10 suggestions by picking objects randomly from the same object class. In summary, we have a total of 98 test configurations (14 test scenes, 7 objects per scene) with three different types of resulting scenes per configuration: the original scenes where all objects are selected by the participant (*People*), scenes generated with our automatic suggestions (*Ours*), and scenes generated with random suggestions (*Random*).

We used AMT to obtain all pairwise comparisons between these three types of scenes for each test configuration. Since we generated 10 random suggestions per configuration, there are 980 unique comparisons for both *Random* vs. *Ours* and *Random* vs. *People*. In addition, there are 98 comparisons for *Ours* vs. *People*. In our study, each HIT includes between 14 and 20 comparisons, 8 of which are repeated to test participants' consistency. If a participant gives inconsistent answers for more than 2/8 of these questions, we exclude their responses from our analysis. Each unique comparison was done by 30 different participants, and in the end, we kept 48,766 responses for analysis, which is 55% of all responses.

Table 4 summarizes the results from this experiment. Overall, 57% of the participants preferred *Ours* to *Random*, 61% preferred *People* to *Ours*, and 65% preferred *People* to *Random*. The results indicate that the preference order is *Random* $<$ *Ours* $<$ *People*. It is not surprising that people's selections are most preferred, particularly since these selections are picked from the scenes with the highest overall compatibility among all the scenes that were created in the first step. However, our suggestions are still preferred to the people's selections in 39% of the time, which indicates even if the user has selected an object that is fairly compatible to the rest of the scene, our learned metric can still produce a better suggestion in many cases.

	Ours vs. Random	Ours vs. People
Table lamp	54%	41%
Arm chair	55%	33%
End table	55%	43%
Coffee table	56%	30%
Chair	57%	37%
Floor lamp	58%	45%
Sofa	63%	44%
Overall	**57%**	**39%**

Table 4: Style-aware furniture suggestion results. *Comparison of style compatibilities of furniture suggestion by our algorithm versus alternative methods. For each column titled "A vs. B," the table lists the percentage of tasks where the furniture suggested by A is preferred by AMT workers to the one suggested by B.*

7.3 Style-aware scene building

The metric learned by our method can also provide suggestions to help people create stylistically compatible scenes in interactive design tools. This feature could help designers of interior spaces, virtual worlds, and immersive games create more plausible scenes.

To investigate this application domain, we have implemented an interactive scene builder augmented with the compatibility metric learned by our algorithm. The input to the system is a set of prescribed object classes, an initial scene with a prescribed spatial layout, and a database of 3D models labeled by object class. During an interactive session, the user iteratively replaces models by other models from the same object class in an effort to make the furniture more compatible (Figure 7). At any time, the user is allowed to fix/free any number of objects in the scene, and our system suggests combinations of models from the database that are not currently fixed. Then, the user is able to choose one of the suggestions to perform the replacement. This procedure repeats until the user is satisfied with all models in the scene.

Our goal is to help people find compatible scenes efficiently, so the suggestion list should have two properties. First, the suggestion list should be ranked by the compatibility energy (Equation 6). Second, the suggestion list should be diverse, so that the user can navigate efficiently in the search space. In order to meet both properties, we take the following strategy in our system: we maintain a candidate set of models, which initially includes all the models. Each time we aim to pick the suggestion that leads to the scene with the lowest compatibility energy, and remove all the models in the suggestion from the candidate set. We repeat this until no new suggestion can be generated, i.e., models from one object class are used up. This strategy ensures that the suggestion list is ranked in an increasing order of the compatibility energy, and all suggestions are disjoint.

In contrast to style-aware furniture suggestion (Section 7.2), we aim to update multiple models at the same time in the interactive system, and thus the search space is prohibitively large. Fortunately, the acyclicity of the graph formed by pairs of object classes that are selected as semantically connected (Figure 3) allows us to use dynamic programming to quickly find the optimal solution. Specifically, we first pick one object class as the root of the graph to convert the graph to a tree T. Each node in T represents an object class, and has a set of models as candidate (only one candidate model if the object class is fixed). Then, we define a *state* as an object class C and a model m in the class (the state is denoted as (C, m)), and the energy of the state $E(C, m)$ as the lowest compatibility energy of the subtree rooted at C in T. We update the energies of all states in a bottom-up manner: starting with the leaf level of T, the energy is simply 0 for all states at leaves. Given the energies of the states of child nodes, we determine the energy of a state at a parent node (C_p, m_p) by picking a model m_c for each child node C_c such that $d(m_c, m_p) + E(C_c, m_c)$ is minimal. In our experiments with the living room database, with pre-computed geometric features of all the models, the suggestion list can update in the interactive rate (< 60 ms) on 2.3 GHz Intel Core i7.

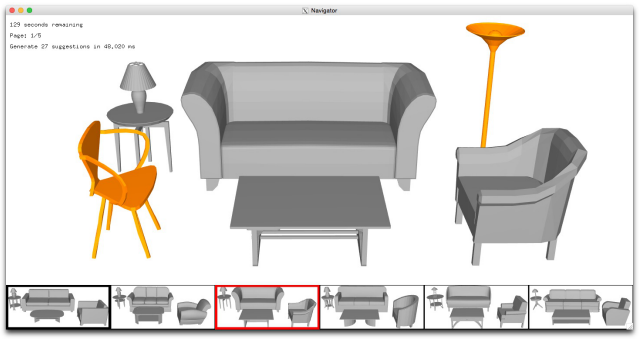


Figure 7: Interface of style-aware scene builder. *The user is allowed to free (in gray) or fix (in orange) any number objects in the scene, and our system suggests combinations of free objects in order of style compatibility (at the bottom of the window). When the user selects any suggestion from the list (red box), the relevant models are updated in place within the scene.*

Evaluation. We conducted a user study to evaluate the utility of the style-aware scene suggestions in our interactive scene builder system. We compare two conditions: the scene builder with suggestions ordered by our learned compatibility metric (*Ours*), and the same interface with randomly ordered suggestions (*Random*).

We recruited 12 participants (graduate students) who are not involved in the project and asked them to perform 16 different scene modeling tasks with the scene builder interface, half using *Ours* and half using *Random* (without being told which was which). We set up each task by selecting a single reference object to fix and then randomly generating the rest of the scene. From this starting configuration, participants were asked to improve the style compatibility of the scene while keeping the fixed object. Using the living room dataset, we generated two tasks per object class by selecting one "modern" and one "old-fashioned" reference object. In addition to these 14 tasks, we created 2 additional warm-up tasks whose results were excluded from our subsequent analyses. The tasks were presented in the same order for all participants, with the two warm-up tasks first. Participants were given 3 minutes for each task.

To evaluate the two conditions, we asked AMT workers to compare the results created using *Ours* vs. *Random*. For each scene modeling task, we obtained 6 scenes created using *Ours* and 6 using *Random*, resulting in a total of 504 unique comparisons (14 tasks, 36 comparisons per task). We designed each HIT to include 18 of these comparisons. For each, we asked workers to indicate which scene is more stylistically compatible with the option of specifying no preference. As with the furniture suggestion analysis, we repeated 8 questions and excluded responses with more than 2/8 inconsistent answers. Each comparison was done by 30 different participants. After filtering for consistency, we ended up with 10,322 answers to analyze, which is 47% of all responses.

Overall, we received 5,760 votes that favor the results created using *Ours*, and 4,135 votes that favor *Random*, and 427 "No preferences". This indicates that participants do have a preference in most of the cases. If we treat "No preference" as a half vote for each system, 58% of all the votes favor our system. While the

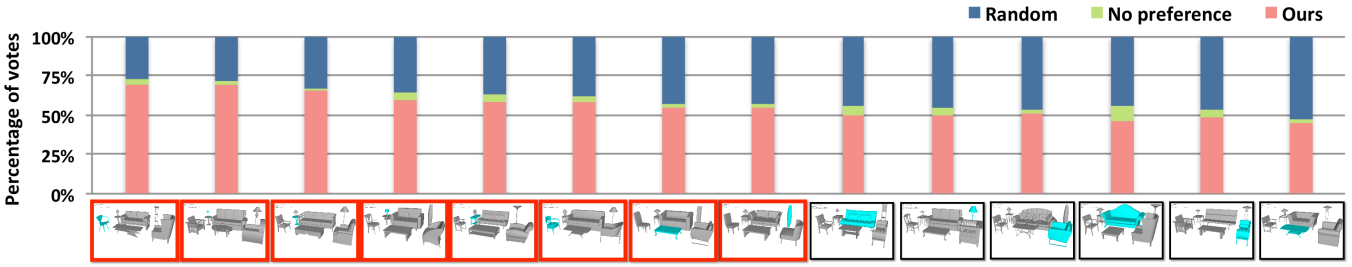


Figure 8: Style-aware scene building. *Preferences from AMT for the style compatibility of scenes created with our system's suggestions versus the alternative. The tasks are ranked by the descending order of the percentages of votes that favor the results creating by using our system (red bars). We show the initial scene of each task at the bottom, with the fixed object highlighted in cyan. The results of using our system are preferred in 13 out of 14 tasks, with statistical significance in 8 of them (in red boxes).*

numbers may seem to show a relatively small effect, we note that it is very difficult to demonstrate a significant difference in this experiment, because the suggestions have to be good enough to impact how quickly and effectively people can find compatible objects with a highly functional interface that allows scrolling through lists, iterative refinement, undo, etc. Nonetheless, our results are comparable to those obtained in other subjective evaluations for aesthetic suggestion interfaces [O'Donovan et al. 2014; Garces et al. 2014].

Moreover, if we look at the data for individual tasks (Figure 8), the results of using our system are preferred in 13 out of 14 tasks, and there are clearly some tasks where our system appears to have a larger impact. If we take the null hypothesis that people have no preference between the scenes created using *Ours* vs. *Random*, *Ours* is significantly preferred in 8 tasks (with $p < 0.05$), and the null hypothesis cannot be rejected in the remaining 6 tasks. There were no tasks where the *Random* scenes were significantly preferred. Note that we used the Binomial Test for significance and applied Holm-Bonferroni corrections for multiple comparisons. Based on these results, our system seems to be particularly helpful when the fixed object is small (e.g., table lamp, end table) and/or of modern style. This may be because the participants usually had few cues in these situations, and our system was able to give them some useful guidance. In general, we expect the impact of our system to be greatest when the task is challenging and randomly browsing through the database is unlikely to produce good results.

8 Conclusion

This paper presents a method for computing style compatibility between 3D furniture models using crowdsourced data and distance learning technique. The main conclusions are two-fold. First, our quantitative results show that it is possible to learn a compatibility metric for furniture of different classes from these triplets, with greater accuracy using part-aware geometric features and with joint embeddings of individual object classes. Second, our user studies show that the learned metric can be used effectively to achieve higher style compatibility in applications ranging from shape retrieval to scene building.

Our system is a first investigation of style compatibility for 3D models and thus suffers from several limitations. First, it considers only a simple set of geometric features and thus cannot detect fine-grained style variations, such as types of ornamentation. Second, it considers only geometric properties, and, in future work, it would be interesting to investigate how materials [Jain et al. 2012], colors, construction methods, affordances, and other properties of 3D models determine style compatibility. Third, it is targeted only at furniture within interior environments, whose styles have a rich history, but perhaps unique properties that do not extend to other object and scene types. Investigating how the proposed techniques could be used in systems for modeling avatars, garments, architecture, cities, or other domains where style is important could provide a fruitful line of research for future study.

Acknowledgments

We acknowledge Trimble and Digimation for providing 3D models, and Evangelos Kalogerakis for distributing code for computing geometric features. We thank Adam Finkelstein and Peter O'Donovan for helpful discussions. Finally, we acknowledge Adobe, Google, Intel, and the NSF (IIS-1251217) for funding the project.

References

AKAZAWA, Y., OKADA, Y., AND NIIJIMA, K. 2005. Automatic 3d scene generation based on contact constraints. In *Proc. Conf. on Computer Graphics and Artificial Intelligence*, 593–598.

BACH, F., JENATTON, R., MAIRAL, J., AND OBOZINSKI, G. 2012. Optimization with sparsity-inducing penalties. *Foundations and Trends in Machine Learning 4*, 1, 1–106.

CHAUDHURI, S., KALOGERAKIS, E., GUIBAS, L., AND KOLTUN, V. 2011. Probabilistic reasoning for assembly-based 3d modeling. *ACM Trans. Graph. 30*, 4, 35.

CHAUDHURI, S., KALOGERAKIS, E., GIGUERE, S., AND FUNKHOUSER, T. 2013. Attribit: content creation with semantic attributes. In *Proc. UIST*, ACM, 193–202.

FISHER, M., AND HANRAHAN, P. 2010. Context-based search for 3d models. *ACM Trans. Graph. 29*, 6, 182.

FISHER, M., SAVVA, M., AND HANRAHAN, P. 2011. Characterizing structural relationships in scenes using graph kernels. *ACM Trans. Graph. 30*, 4, 34.

FISHER, M., RITCHIE, D., SAVVA, M., FUNKHOUSER, T., AND HANRAHAN, P. 2012. Example-based synthesis of 3d object arrangements. *ACM Trans. Graph. 31*, 6, 135.

FUNKHOUSER, T., MIN, P., KAZHDAN, M., CHEN, J., HALDERMAN, A., DOBKIN, D., AND JACOBS, D. 2003. A search engine for 3d models. *ACM Trans. Graph. 22*, 1, 83–105.

GARCES, E., AGARWALA, A., GUTIERREZ, D., AND HERTZMANN, A. 2014. A similarity measure for illustration style. *ACM Trans. Graph. 33*, 4, 93.

GOLDBERGER, J., ROWEIS, S., HINTON, G., AND SALAKHUTDINOV, R. 2004. Neighbourhood components analysis. *Advances in Neural Information Processing Systems*.

HOTELLING, H. 1936. Relations between two sets of variates. *Biometrika 28*, 3-4, 321–377.

HUANG, Q.-X., SU, H., AND GUIBAS, L. 2013. Fine-grained semi-supervised labeling of large shape collections. *ACM Trans. Graph. 32*, 6, 190.

JAIN, A., THORMÄHLEN, T., RITSCHEL, T., AND SEIDEL, H.-P. 2012. Material memex: Automatic material suggestions for 3d objects. *ACM Trans. Graph. 31*, 5, 143.

KALOGERAKIS, E., CHAUDHURI, S., KOLLER, D., AND KOLTUN, V. 2012. A probabilistic model for component-based shape synthesis. *ACM Trans. Graph. 31*, 4, 55.

KAZHDAN, M., CHAZELLE, B., DOBKIN, D., FUNKHOUSER, T., AND RUSINKIEWICZ, S. 2004. A reflective symmetry descriptor for 3d models. *Algorithmica 38*, 1, 201–225.

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. Graph. 32*, 4, 70.

KULIS, B. 2012. Metric learning: A survey. *Foundations & Trends in Machine Learning 5*, 4, 287–364.

LI, H., ZHANG, H., WANG, Y., CAO, J., SHAMIR, A., AND COHEN-OR, D. 2013. Curve style analysis in a set of shapes. *Computer Graphics Forum 32*, 6, 77–88.

LUN, Z., KALOGERAKIS, E., AND SHEFFER, A. 2015. Elements of style: Learning structure-transcending perceptual shape style similarity. *ACM Trans. Graph. 34*, 4.

MA, C., HUANG, H., SHEFFER, A., KALOGERAKIS, E., AND WANG, R. 2014. Analogy-driven 3D style transfer. *Computer Graphics Forum 33*, 2, 175–184.

MERRELL, P., SCHKUFZA, E., LI, Z., AGRAWALA, M., AND KOLTUN, V. 2011. Interactive furniture layout using interior design guidelines. *ACM Trans. Graph. 30*, 4, 87.

MERRIAM-WEBSTER. 2004. *Merriam-Webster Dictionary*. Merriam-Webster Mass Market, July.

MILLER, J. 2005. *Furniture*. Penguin.

O'DONOVAN, P., LĪBEKS, J., AGARWALA, A., AND HERTZMANN, A. 2014. Exploratory font selection using crowdsourced attributes. *ACM Trans. Graph. 33*, 4, 92.

PARIKH, D., AND GRAUMAN, K. 2011. Relative attributes. In *Proc. ICCV*, 503–510.

SHAPIRA, L., SHALOM, S., SHAMIR, A., COHEN-OR, D., AND ZHANG, H. 2010. Contextual part analogies in 3d objects. *International Journal of Computer Vision 89*, 2-3, 309–326.

TANGELDER, J. W., AND VELTKAMP, R. C. 2008. A survey of content based 3d shape retrieval methods. *Multimedia tools and applications 39*, 3, 441–471.

UMETANI, N., IGARASHI, T., AND MITRA, N. J. 2012. Guided exploration of physically valid shapes for furniture design. *ACM Trans. Graph. 31*, 4, 86.

WILBER, M. J., KWAK, I. S., AND BELONGIE, S. J. 2014. Cost-effective hits for relative similarity comparisons. In *Proc. HCOMP*.

XU, K., LI, H., ZHANG, H., COHEN-OR, D., XIONG, Y., AND CHENG, Z.-Q. 2010. Style-content separation by anisotropic part scales. *ACM Trans. Graph. 29*, 6, 184.

YU, L.-F., YEUNG, S. K., TANG, C.-K., TERZOPOULOS, D., CHAN, T. F., AND OSHER, S. 2011. Make it home: automatic optimization of furniture arrangement. *ACM Trans. Graph. 30*, 4, 86.

ZHENG, Y., COHEN-OR, D., AND MITRA, N. J. 2013. Smart variations: Functional substructures for part compatibility. *Computer Graphics Forum 32*, 2pt2, 195–204.

ZHU, C., BYRD, R. H., LU, P., AND NOCEDAL, J. 1997. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM TOMS 23*, 4, 550–560.