Relationship Templates for Creating Scene Variations

Xi Zhao
Xi'an Jiaotong University

Ruizhen Hu
Shenzhen University

Paul Guerrero   Niloy Mitra
University College London

Taku Komura
Edinburgh University

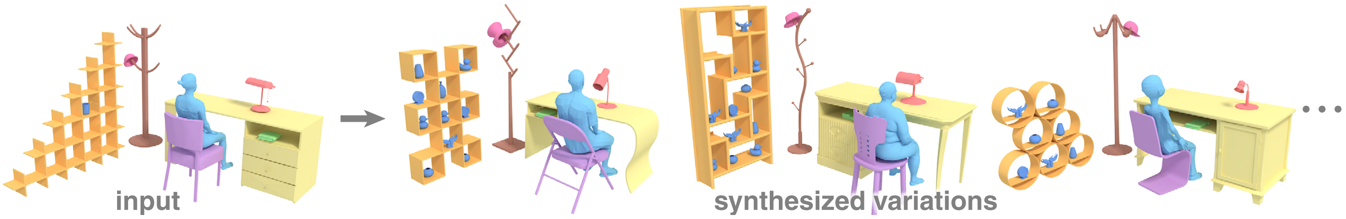

Figure 1: *Our system synthesizes new scenes based on how objects interact with each other in the example scene. The method handles complex relationships such as 'hooked-on,' 'tucked-under,' etc. and does not rely on classification or labeling of input objects.*

Abstract

We propose a novel example-based approach to synthesize scenes with complex relations, e.g., when one object is 'hooked', 'surrounded', 'contained' or 'tucked into' another object. Existing relationship descriptors used in automatic scene synthesis methods are based on contacts or relative vectors connecting the object centers. Such descriptors do not fully capture the geometry of spatial interactions, and therefore cannot describe complex relationships. Our idea is to enrich the description of spatial relations between object surfaces by encoding the geometry of the open space around objects, and use this as a template for fitting novel objects. To this end, we introduce *relationship templates* as descriptors of complex relationships; they are computed from an example scene and combine the interaction bisector surface (IBS) with a novel feature called the space coverage feature (SCF), which encodes the open space in the frequency domain. New variations of a scene can be synthesized efficiently by fitting novel objects to the template. Our method greatly enhances existing automatic scene synthesis approaches by allowing them to handle complex relationships, as validated by our user studies. The proposed method generalizes well, as it can form complex relationships with objects that have a topology and geometry very different from the example scene.

Keywords: Spatial Relationships, Scene Synthesis, Relationship Templates

Concepts: •**Computing methodologies** → **Shape analysis;**

1 Introduction

The ability to synthetically generate plausible 3D scene arrangements is critical for many content-hungry applications like games, movies and virtual worlds. Generated realistic 3D scenes can also be used as training data for various scene understanding tasks like classification and segmentation. A significant amount of time and manual labor can be saved if production of these complex scenes can be automated, starting from only a handful of example scenes.

Automatic scene synthesis, however, remains an elusive task. The challenge lies in creating synthesized scenes that are *realistic*, i.e., ensuring that complex inter-object relations along with their spatial configurations follow those observed in the real world. For example, when creating a scene where a hat is hooked on a rack, position and orientation of the hat must be set correctly such that it is physically hooked, or when creating a person at a writing desk, the relative arrangement of desk and chair with respect to the person should be realistic.

Existing scene synthesis methods mainly rely on approaches where objects in an example scene are replaced based on compatible keywords along with a similarity based on simple geometric descriptors. Inter-object relations are typically restricted to simple geometric contacts [Fisher et al. 2011] and/or relative displacement vectors [Fisher et al. 2012; Chen et al. 2014] between object centroids. Although such descriptors can effectively capture simple spatial relations such as an object on top of another, or objects next to each other, they are not designed to capture complex relationships such as an object 'hooked on', 'surrounded by', or 'tucked into' another object. As a result, objects in these relationships cannot be synthesized, limiting the range of scenes that can be produced.

In complex relations, the geometry of objects and the spatial relations between objects are often tightly coupled. Take the example in Figure 2, where a chair is tucked into a desk. The chair must be positioned to fit the empty space under the desk and leave enough space for a person to sit. A simple geometric match of individual objects is not sufficient for finding a good fit. With this insight, descriptors such as Interaction Bisector Surface (IBS) [Zhao et al. 2014] and Interaction Context (ICON) [Hu et al. 2015] have recently been proposed that analyze the space in-between two objects to support relation-based retrieval.

In this paper, we go beyond retrieval to directly synthesize novel scenes with objects interacting according to specified (complex) relations, as shown in Figure 1. More specifically, we address a subproblem of scene synthesis: placing novel objects in scenes such that they form given complex relations. This requires addressing two related key challenges: (i) retrieving object pairs that can interact similar to an example relationship; and (ii) arranging the retrieved objects according to the example relations. To address these problems, we introduce *relationship templates* that are computed from an example scene and describe the relation between two objects.

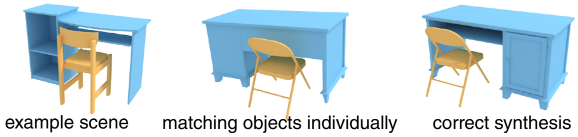

Figure 2: *Object relations and geometry are tightly coupled. When synthesizing scenes from a given example (left), only matching objects individually gives incorrect results (center). The geometry of both objects needs to be considered (right).*

SIGGRAPH Asia 2016, December 5-8, 2016, Macao, China
ISBN: 978-1-4503-4514-9/16/12
DOI: http://dx.doi.org/10.1145/2980179.2982410

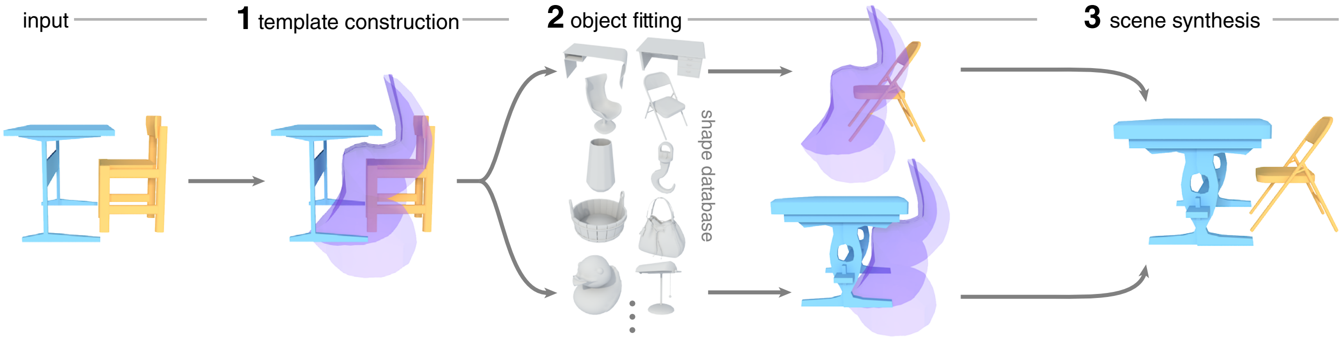

Figure 3: *Overview of our pairwise synthesis. Given two example objects, we compute a* relationship template *that describes the geometric relationship between the two objects, shown in transparent blue. We can then fit objects from a database independently to each side of the template, guaranteeing that any fitted object pair will be in a relationship similar to the exemplar.*

Instead of matching object pairs to determine if their relations are similar to the example, we can match objects to the relationship template independently, reducing the complexity from quadratic to linear. We base the template on IBS, augmented by a feature that we call space coverage feature (SCF), which compactly encodes the rich geometry of surrounding objects in the frequency domain. This setup allows us to efficiently position and orient objects to match a given template using a multi-step optimization procedure.

Our approach allows existing scene synthesis approaches to handle complex relationships in their framework, which increases the range of scenes they can synthesize. The relationship template can also handle complex relations of more than two objects by introducing a hierarchical structure. We show that our scheme can produce scenes with complex relations such as flowers inside a vase, hats hooked on racks, a person sitting on a chair in front of a desk, and a cup with a toothbrush on a bathroom basin. These scenes are produced by adding our method either to our own minimal scene synthesis pipeline or to a more sophisticated existing synthesis pipeline [Fisher et al. 2012]. We evaluate the proposed method via both qualitative and quantitative evaluations, and compare against state-of-the-art scene synthesis algorithms.

2 Related Work

Learning object structures. Several co-analysis methods have been developed to discover the structure of man-made objects (see survey [Mitra et al. 2013]). For example, Wang et al. [2011] compute hierarchical pyramids of single objects based on the symmetry and contact information; Kalogerakis et al. [2012] produce a probabilistic model of the shape structure from examples; van Kaick et al. [2013a] use a co-hierarchical analysis to learn models; Several approaches [Zheng et al. 2013a; Su et al. 2016; Huang et al. 2016] build a graph structure from an object based on the spatial relations of the components. Researchers have also proposed new features based on pairwise points to encode the spatial context of shapes [van Kaick et al. 2013b; Zheng et al. 2013b]. The probabilistic models learned by such methods can be applied for synthesizing novel objects [Kalogerakis et al. 2012; Zheng et al. 2013a]. We next describe the above techniques that have been extended for scenes composed of multiple objects.

Learning scene layouts. Research to learn the contextual and spatial structures of living spaces is currently advancing due to the large amount of room data available in the world, for example, in the Trimble 3D Warehouse. Methods have been proposed to either synthesize variations from such example scenes, or create new scenes based on rules provided by an artist. Shao et al. [2012] match surface data obtained from RGBD cameras to individual example objects using a random forest classifier trained on features such as spin images and geometry moments. Nan et al. [2012] also match data captured by RGBD cameras with objects in the database, and produce discrete versions of the scene. Matching individual objects works well if they are clearly separated, but matching objects in close proximity or objects that have complex spatial interactions may result in collisions and mismatches. In these cases it is necessary to take into account the spatial relations between objects. Early work [Coyne and Sproat 2001; Coyne et al. 2010] manually annotated areas of interest in the open space around an object to handle spatial relationships. More recently, Yu et al. [2011] evaluate spatial relations of furniture based on ergonomic factors, while Fisher et al. model spatial relations in a scene using density estimation over relative distances [Fisher and Hanrahan 2010] or a graph structure based on contact information [Fisher et al. 2011]. These two approaches are further extended in [Fisher et al. 2012] to handle more general relationship based on relative centroid positions and orientations. Xu et al. [2014] propose to classify scenes based on different focal points, where a context-specific graph structure is produced based on the spatial relations of a sub-set of the objects. Majerowicz et al. [2014] learn the style of object alignment and symmetry from a single image of a shelf and apply it to filling in new shelves with varied object arrangements. Chen et al. [2014] also use contacts and relative centroid locations as a spatial relationship representation and produce scenes from RGBD camera data regularized by an example scene database. Liu et al. [2014] learn the contextual grammar of scenes formalized as scene graphs. Yeh et al. [2012] sample new scenes using MCMC sampling in the null space of the constraints described by collisions and spatial relations. However, these methods cannot handle complex spatial relations due to their simple centroid-based relationship representation. Additionally, several methods require an extensive scene database as input, while we only work with a single example scene. In Section 7, we show that our method could be used to augment some of these methods.

Relevant to our work, Zhao et al. [2014] compute the Interaction Bisector Surface (IBS) that describes the open space between objects to represent complex relations such as enclosure and hooking. Hu et al. [2015] extend the idea with a feature called *interaction context* to estimate the functionality of a 3D object in a given scene and further develop this idea to co-analyze the functionality of object categories and localize surface areas where interactions take place [Hu et al. 2016]. In this work, we go beyond analysis to directly generate scenes with complex spatial relations, taking a step towards applying these methods in example-based scene synthesis.

Human-Object relations for scene analysis and synthesis. Researchers have also augmented scene descriptions with human-object relations as indoor scenes are typically designed with human activities in mind. Jiang et al. [2013] learn human object relationships from the data and apply them to label novel objects in point clouds; while, Savva et al. [2014] decompose objects into primitive structures such as cuboids and make use of human activity data to predict the likelihood of activities being performed at scene locations. More recently, affordance of objects has been utilized to analyze and populate 3D scenes. Kim et al. [2014] consider the local geometry of the contact area with the human body, the object symmetries, the human body articulation and postures in the example data as additional factors to evaluate whether a human can perform certain actions with an object. Fisher et al. [2015] and Savva et al. [2016] synthesize novel scenes based on the interactions of a human proxy with the environment. These approaches focus on relationships of objects with human actors. In this work, we support object-object relationships, which require a different approach than human-centric relationships.

3 Overview

Given an example scene with complex object interactions and a 3D object database, our goal is to synthesize new scenes by replacing objects in the example scene with database objects that preserve the nature of the original spatial relations. The three main stages of the algorithm are described next. Figure 3 provides an overview.

Template construction. We first build a *relationship template* for the input exemplar. The template consists of a set of cells and cell features. Each cell corresponds to one object in the scene, and describes the open space between the object and neighboring scene objects. We augment the cell by a set of features, including a novel feature called Space Coverage Feature (SCF), that describes the spatial relations between individual points in space and surrounding objects (see Section 4.4).

Object fitting. In order to synthesize new scenes, we match novel database objects to each template cell. To reduce the search space for a matching pose, we start with a heuristic search for volumes where the SCF features around the novel object statistically match the template cell, filtering out large volumes that are unlikely to contain good matches (see Section 5.3). This approach is especially useful for objects with many potential matches. The novel object is then fit to the template cell by matching the open space around it to the features stored in the template. This initial match is refined by optimizing translation and orientation of the novel object with respect to the cell boundary (see Section 5.5).

Scene synthesis. After matching novel objects to each cell of the template, we combine them to build the final scene (see Section 6), restoring detected contact relations. We also present a hierarchical approach for larger-scale scene synthesis, such as room layouts.

4 Relationship Template

In order to synthesize novel scene variations from an example scene, we need to extract a descriptor of the example scene and use it as a guidance for synthesis. To this end, we introduce a concise *relationship template*, which encodes the rich spatial relationships between objects and can be applied for scene synthesis, as well as quantitatively rating the similarity of relationships in synthesized scenes.

4.1 Definitions

We start by introducing definitions and notations related to the relationship template and give an overview of its structure.

- *Scene*: A scene is an entity that is composed of N 3D objects located in the Cartesian space. An isolated object can be considered as a special scene.

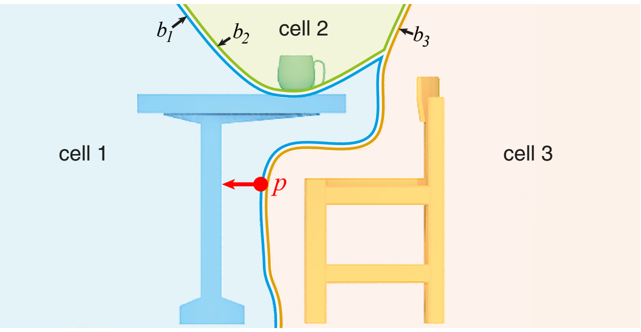

Figure 4: *The IBS divides the scene space into one cell per scene object. We compute features on the boundaries of these cells to describe object relationships. Based on these features, new objects can be fit into each cell to synthesize new scene variations.*

- *Interaction Bisector Surface (IBS)*: The IBS of a scene is a surface consisting of the set of points equidistant from neighboring objects in a scene (see Section 4.2).
- *Open Space*: The open space of a scene is defined as the free space outside the objects. We crop the infinite open space using a box surrounding the scene as in [Zhao et al. 2014].
- *Cell*: The IBS divides the open space of the scene into N subspaces, which we refer to as cells. Each cell corresponds to one object in the scene, and its boundary is a subset of the IBS (see Section 4.2).
- *Template*: The relationship template consists of a group of cells and their features. The computation process of the cell features is described in Section 4.3.

Figure 4 shows an example of a template in a scene that contains three objects: a table, a chair and a cup. Cells of the template are shown in blue, green and red background colors. The lines between the cells illustrate the IBS of the scene. The relationship template can be considered as a 'connector' or an 'adaptor' of objects in a scene. Each cell is one 'joint' of the connector/adaptor and defines how an object connects to the template.

4.2 IBS and Cell Construction

Here we briefly review how we compute the IBS and construct all cells in the scene. The IBS describes interactions between objects and was used for scene classification and relation-based 3D object retrieval in Zhao et al. [2014]. The IBS is the set of points that are equidistant from the two neighboring objects, i.e., it is a subset of the external medial axis, or the Voronoi diagram. We compute IBS via the quickhull algorithm [Barber et al. 1996]. Points are sampled on the surface of scene objects and a set of polygons named *ridges* that are equidistant from the sample points are extracted and merged to approximate the medial axis. Among these ridges, those produced by points from two different objects are used to form the IBS. Thus the IBS describes the open space between interacting objects. Recall that each cell c_i corresponds to an object. The boundary b_i of the cell is the IBS subset produced by this object, shown in Figure 4 in the same color as the cell. We store only the boundaries b_i ($i \in [1, N]$) of the cells.

4.3 Cell Features

In order to provide a descriptive representation of each cell, we first sample a set of points on the cell boundary and then compute several point-wise features for each sample point.

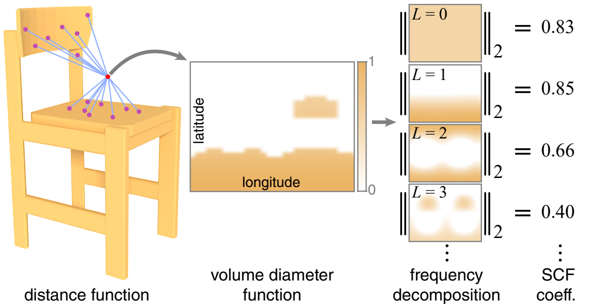

Figure 5: *SCF computation steps. To compute the SCF at the red point (left), we sample the boundary distance around the point and normalize it to get the* volume diameter function*. Our SCF coefficients are the power of this function in each frequency band.*

Sampling on boundary. We first sample the entire IBS of the scene with $200 \times (N-1)$ points, following the approach in [Zhao et al. 2014]. Intuitively, sampling density is higher where the distance between the IBS and the object is shorter and the angle between the IBS normal and the closest point on the objects is smaller. We call these sample points template points. Note that each template point is used for the two cells on both side of the boundary.

Feature computation. Next, we compute a set of features for each template point on boundary b_i that describe the spatial relation to the object contained in the corresponding cell. For example, in Figure 4, we compute features at template point p that describe its relation to the table. Three types of features are computed for each template point p with respect to the corresponding object:

the shortest distance between p and the object (denoted here by f_{dis}), the direction from p to the closest point on the object (denoted here by f_{dir}), and the SCF feature at point p with respect to the object (denoted here by f_{scf}), which we will describe next.

4.4 Space Coverage Feature (SCF)

The space coverage feature (SCF), effectively quantifies the relationship between a point in open space and a nearby object. Figure 5 provides an overview of the computation steps. The key of the SCF feature is to encode the geometry of the open space around objects in the frequency domain using spherical harmonics; a similar feature has been applied for describing object geometry [Saupe and Vranic 2001; Kazhdan et al. 2003; Kazhdan et al. 2009], but not for relationships.

Given a point P in open space, a spherical depth map is computed at P. We define a unit sphere centered at P and sample the distance d from P to all surrounding surfaces. Rays are cast from P in a set of directions obtained by uniformly sampling n points along the latitude and longitude of the sphere, for a total of $n \times n$ rays. The north pole points towards the object's closest point and we set n to 30 in our experiments. If a ray does not hit an object, d is set to infinity. This approach is inspired by Shapira et al. [2008] where the rays are cast within the object to produce a feature of the object shape; here we cast from outside to produce a feature of the open space between objects.

Next, we define a discrete spherical function called the *volume diameter function* for P as

$$F_{vdf}(i,j) := \left\{ \frac{d^{min}+e}{d(i,j)+e} \;\middle|\; 0 \le i,j \le n-1 \right\}, \qquad (1)$$

where i, j are the ray indices along the longitude and latitude directions, respectively, d^{min} is the minimum distance among all rays and used for normalization, and e is an offset whose value is set to the mean of all non-infinite distances. We use e to keep F_{vdf} descriptive even when P is sampled at the surface of the object. Note that the value range of F_{vdf} is $[0,1]$ due to the normalization, which makes the feature scale invariant. Infinite ray results in a value of 0, while the ray with minimum distance gets a value close to 1.

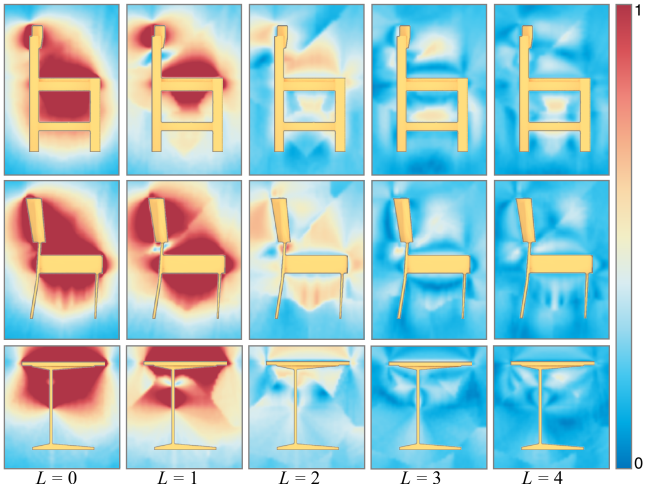

Figure 6: *SCF coefficients in the open space around different objects are shown as color maps. Values are clamped to 1 to preserve contrast for smaller coefficients. In each row, we show coefficient values of bands 0 to 4 for the object shown in the foreground.*

Our SCF feature vector is the rotation-invariant power spectrum of the volume diameter function. Similar to [Kazhdan et al. 2003], we first compute the spherical harmonics expansion $F_{vdf} = \sum_{l=0}^{\infty} \sum_{|m| \le l} a_{l,m} Y_l^m$, where $a_{l,m}$ are the spherical harmonics coefficients of F_{vdf} and Y_l^m are the spherical harmonics at frequency l. We then define the SCF features as:

$$SCF(F_{vdf}) = \{a_0, a_1, \ldots, a_n\}, \qquad (2)$$

where a_l is the power of the function in frequency band l:

$$a_l = \| \sum_{|m| \le l} a_{l,m} Y_l^m \|_2 = \sqrt{\sum_{|m| \le l} (a_{l,m})^2}. \qquad (3)$$

Note that the last equality holds due to the orthonormality of spherical harmonics. In practice, we use SpharmonicKit [Kostelec 2008] for computing the spherical harmonics coefficients.

Describing the open space in the frequency domain makes our descriptor more robust to small variations in the geometry and topology of the open space. In Figure 6, we visualize the first five frequency bands for three different objects. Note that the coefficients are similar for objects of similar type (first two rows), even though their geometry and topology is slightly different; while objects of different type (last row), have a larger difference in their coefficients.

5 Fitting a Novel Object to a Cell

Scenes with complex relations are synthesized by fitting novel objects into template cells. A template cell of the input scene is used as a 'connector' that only accepts objects with suitable geometry. Fitting a novel object to a cell is done by searching for a rigid transformation of the object that maximizes the similarity between the cell features computed for the transformed object and the cell features stored from the example scene. Note that this is equivalent to rigidly transforming the template to match the novel object. In this section, we will use these concepts interchangeably, as suitable. We first define a similarity function that measures the fitting quality (see

Section 5.1), and then find an object pose that maximizes this similarity in a three-step optimization procedure. As a pre-computation step, we compute features in a uniform grid around the novel object (see Section 5.2). This allows reducing our search space to so-called *regions of interest* (ROIs - see Section 5.3), and to efficiently get a coarse initial match to the template (Section 5.4), before optimizing for feature similarity (see Section 5.5).

5.1 Similarity Function

We define a similarity function that evaluates how well a novel object with a given rigid transformation fits the template. This similarity is based on comparing the cell features for the novel object to the cell features for the example scene:

$$S_{final} := (1 - d_{dis})(1 - d_{dir})(1 - d_{scf}), \text{ where} \tag{4}$$

$$\begin{aligned} d_{dis} &= \frac{1}{N_i}\sum_{j=1}^{N_i}\frac{|f^j_{dis} - {f'}^j_{dis}|}{\alpha_{dis}}, \\ d_{dir} &= \frac{1}{N_i}\sum_{j=1}^{N_i}\mathbf{1}\left(\angle(f^j_{dir}, {f'}^j_{dir}) < \alpha_{dir}\right), \\ d_{scf} &= \frac{1}{N_i}\sum_{j=1}^{N_i}\frac{\|f^j_{scf} - {f'}^j_{scf}\|_2}{\alpha_{scf}}, \text{ with} \end{aligned} \tag{5}$$

N_i being the number of template points on the boundary of cell c_i, $(f^j_{dis}, f^j_{dir}, f^j_{scf})$ are the distance, direction, and SCF features at the template point p_j in the original scene, and $({f'}^j_{dis}, {f'}^j_{dir}, {f'}^j_{scf})$ are the features computed for the transformed novel object. $\mathbf{1}(\chi)$ is an indicator function that returns 1 if χ is true and 0 otherwise, and α_{dir} is an angular threshold within which the orientation is considered similar, which is set to $\pi/2$. We normalize d_{dis} and d_{scf} with the upper bounds α_{dis} and α_{scf} to keep them in $[0, 1]$; see Appendix B for details of these bounds.

5.2 Precomputing Features

We precompute two quantities to improve the efficiency of our search for good relative transformations between template and novel object: statistics over the template features to focus our search on regions that are likely to contain good candidates, and features values at a uniform grid of points in the open space around the novel object, referred to as *open space points*.

Representative template points. In order to reduce the cost of matching the template points with the open space points, we cluster the template points based on their SCF feature using k-means clustering, and compute the average SCF feature and distance feature for each cluster. The cluster number M is set to 5 in all our experiments and represent them as,

$$\begin{aligned} \mathbf{c}^{ibs}_{scf} &= \{c^1_{scf}, c^2_{scf}, ..., c^M_{scf}\} \\ \mathbf{c}^{ibs}_{dis} &= \{c^1_{dis}, c^2_{dis}, ..., c^M_{dis}\}. \end{aligned} \tag{6}$$

Open space points. We first compute a tight axis-aligned bounding box of the novel object and then add a margin of $2\max(\mathbf{c}^{ibs}_{dis})$. Open space points are uniformly sampled within this volume and their SCF, distance, and direction features are computed. Note that the resolution of sampling the open space points affects the performance: a sparse sampling may lead to bad matching results while a dense sampling leads to heavy computation cost. We determine the sampling interval dx based on the distribution of the distance feature on the template. If there are many points where the distance is small, we need to sample more densely to precisely fit the novel object, where a coarse sampling is fine when the distance is always large. In practice, we set the interval of uniform sampling as half the average distance feature of the template.

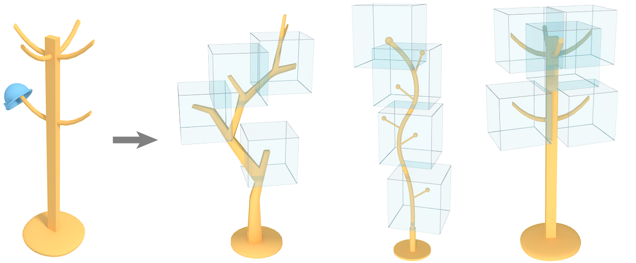

Figure 7: *Regions of interest. To reduce the search space for poses with relationships similar to the exemplar (left), we heuristically choose regions of interest that potentially contain good candidate poses (right) and discard all poses outside these regions.*

5.3 Region of Interest

We reduce our search space to regions that are likely to contain good candidates. This is similar to the bag of visual words concept [Fei-Fei and Perona 2005], where the goal is to search for a volume that contains point sets with high matching score. We use a 'sliding window' to test all different sub-regions of the open space. Specifically, we use sphere shaped windows with radius r set to half the the longest edge of the cell's axis-aligned bounding box (see Figure 7 - for clearer illustration we show cubes instead of spheres).

Window matching score. In each window, H candidate open space points are selected for computing the similarity to the cluster averages c^m_{dis} and c^m_{scf}. A high similarity score means the point is more likely to fit the template around the cluster. So we first compute the matching score between an open space point and a cluster average as:

$$s^m_i := \begin{cases} 0 & |f^i_{dis} - c^m_{dis}| > \theta_{dis} \\ 1 - \frac{\|f^i_{scf} - c^m_{scf}\|}{\alpha_{scf}} & \text{otherwise,} \end{cases} \tag{7}$$

where α_{scf} normalizes the matching score, similar to Eq. 5. Now, assume the number of template points is N_{total}, and the number of template points in each cluster is $N_m(m = 1, ...M)$. In the window, for the m-th template point cluster, we select h_m candidate points with the best matching score s^m_i, where $h_m = H \times \frac{N_m}{N_{total}}$. The window matching score is then given as:

$$S := \sum_{m\in[1,M]}\sum_{i\in w} s^m_i / h_m, \tag{8}$$

where w is the number of candidates picked for a cluster. Higher values for S increase the chance of finding a good fit in the window.

Sliding window. The window slides across the open space along the x, y, and z axes and the interval is set to be $r/5$. We compute the similarity score for all the windows, keep the best ROI and restrict our search to this region. This process can be repeated to pick a different ROI when the matching fails or there are multiple possible matching locations in the novel object. Each time we exclude all windows that have already been used before and recompute the score for all remaining windows.

5.4 Initial Matching

We now describe how to establish an initial match between the template cell and the extracted ROI in the open space around the novel objects. A good match of the open space can be obtained even when the fine geometry of the object is different from that of the original scene, thanks to the abstraction provided by our frequency domain representation. We adopt geometric hashing [Wolfson and Rigoutsos 1997], which is a generalization of the RANSAC algorithm and is robust against outliers. (Note that alternate matching schemes can be applied in this step.)

Selecting the candidate points. After extracting the ROI, the H candidate points used for computing Eq. 8 are used for fitting the novel object to the cell. We set $H = 90$ in all our experiments. The H points are labeled with the *id* of the corresponding cluster and the label is used for voting in geometric hashing.

Matching by geometric hashing. Individual steps for geometric hashing are similar to Gal et al. [2006], where geometric hashing is applied to partial matching of 3D surface data:

(i) All the template points are used to build the geometric hashing table. More specifically, all the triplets of template points are used as 'bases' and other points are labelled with the base id and inserted into the hash table.

(ii) Each triplet in the candidate points then votes in the hash table. A triplet of the template points and a triplet of the candidate points are selected, and a transformation from the object to the template cell is computed. Using this transformation, all the rest of the candidate points are transformed to the coordinates of the template triplet. For each candidate point, if both its location and cluster id match with a point in the hash table, the candidate point is considered as an inlier, and the vote for the triplet is increased by 1.

(iii) All triplets of the candidate points are tested, and the triplet with the largest vote (number of inliers) is selected as the best fit.

This fitting process only produces an approximate match due to the finite sampling density of open space points. Next, we describe how to further refine the relative transformation between relationship template and novel object.

5.5 Refinement

We iteratively refine the coarse matching results in Section 5.4 by maximizing the similarity function in Eq. 4. Similar to Zheng et al. [2009], at every iteration, we find an *ideal* location for each template point using the following method. We treat the distance feature of the open space points as a distance field and decide to move the template either in the positive or negative gradient direction by comparing f_{dis} and $f_{dis}^{'}$: if $f_{dis} > f_{dis}^{'}$, we move towards the object, otherwise we move away from it. This decision is made individually for each template point. We fit a rigid transformation of the template to the updated template point locations and use this transformation as input for the next iteration. We iterate until S_{final} becomes stable using an updating scheme described in Appendix A.

Note that this refinement process is local, and during the refinement, the SCF feature of each template points does not change much. This simplifies the optimization. Specifically, we consider the SCF feature to be fixed, only update the state based on the distance feature and move the template points along the gradient of the distance field. We use an average step size of approximately 0.1% of the scene's longest side.

We show an example of applying the above method in Figure 8. In this example, the table is the novel object. The chair is from the original scene and keeps its relative location with respect to the cell boundary. The location of the cell boundary is updated iteratively with the chair, as it moves towards a location that produces the same spatial relationship as in the original scene. Although the initial matching puts the chair only roughly in front of the table, its location is improved after a few iterations.

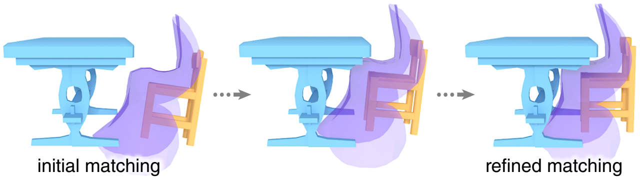

Figure 8: *Snapshots of the refinement process. Given an initial matching, we iteratively refine the template pose to optimize its fit to the table.*

6 Synthesizing Scenes

We now describe how to use our approach to synthesize variations of a scene that includes complex relations. For this purpose, we build a simple scene synthesis pipeline around our method. Alternatively, we can include our method in existing synthesis pipelines, as we will show in Section 7. Given an example scene, we replace the individual scene objects by database objects to generate new scenes with similar inter-object relationships. If there are only two objects in the scene, we simply apply the fitting approach described in Section 5 to both cells. Relationships in larger-scale scenes are often structured hierarchically such as a set of objects put into a basket that is hanging off a handle; or a box with a stack of towel is on top of a shelf (see Section 6.1). We describe extensions of our method to capture semantic constraints of scenes (see Section 6.2), and spatially repeated relationships, such as many hats/caps hung on a cap holder (Section 6.3).

6.1 Hierarchies for Larger-scale Scenes

For scenes composed of more than two objects, directly placing objects into the template cells can be overly restrictive. Take for example a pushcart carrying a pot and a basket hooked on the handle (Figure 12 (d)); we cannot directly exchange the cart with a wider or shorter handle cart, because it would not fit the cell delimited by both pot and basket. In the real world however, the pot and basket positions can easily be changed to accommodate the new cart.

To allow for more flexibility in larger-scale scene synthesis, we build a hierarchy on our cells based on the distance and area of the cell boundary, following the approach in Zhao et al. [2014]. In this hierarchy, siblings are usually smaller objects placed near a larger parent object. To decompose scene synthesis into a sequence of pairwise synthesis problems, we establish an ordering among siblings and consider one pairwise relationship at a time, starting at the leaves and working our way up breadth-first towards the root. In the pushcart scene, we first synthesize the sub-scene consisting of cart and pot, then we synthesize the basket containing food, and finally hook the basket with food onto the cart with pot. In each step, we only use the template part corresponding to the current pair's relationship, avoiding the problem of overly constrained placements. Note that the synthesized scene might have a different relationship template than the exemplar, but with similar pairwise relationships.

The scenes in our experiments are usually not sensitive to the sibling order. For example, the order of placing the lamp and book in the desk scene (Figure 12 (e)) is not relevant, so siblings can often be ordered arbitrarily. Some cases are possible where the ordering among siblings influences the final result. In this case the order can be defined manually. We consider the the problem of determining the order automatically a separate research problem that is out of scope of this paper.

6.2 Capturing Scene Semantics

Realistic scenes usually contain constraints that are not captured purely by the geometry of relationships. Since the focus of our method lies on geometry, we either model these semantic constraints explicitly or let the user input additional information to get semantically plausible scenes.

Resolving placement ambiguity. For a template such as a book placed on a table, there are many candidate locations over the new table that can produce a similar relationship. In this case, the choice of location depends on the semantics of the object being placed. To capture these semantics, the user can specify the relative vector between the object center and the IBS center as an additional attribute. Such a relative vector is encoded into the scene hierarchy and used as an additional criterion for selecting the ROI in Section 5.3: The matching score of each candidate window is multiplied by a factor inversely proportional to the distance between the preferred relative vector and the vector from novel object center to the window center. This option is effective for synthesizing examples such as putting a book on the right-hand side over the table.

Contact constraints. Since novel objects are usually not a perfect match for the distance values stored in the template, contacts between objects on both sides of the template can not always be reproduced accurately. To ensure preservation of desired contacts, the user can indicate if a contact between two objects should be preserved. For these object pairs, we find and store a *contact point* on the relationship template as the closest point to both of the objects (note that closest point to the first object is also the closest point to the second object). As a post-processing step after the refinement stage in Section 5.5, we project the contact point back to the synthesized objects along the template normal and register these projected points with the contact point, thus restoring (near-)contact between the synthesized object pair, if allowed by their geometry.

Upright constraint. We assume that objects come with predefined upright directions, and always impose this constraint unless explicitly released by the user. We examine how much the object deviates from being upright after geometric hashing. If the rotation angle is larger than a threshold (set to $\pi/6$ in our tests), we discard the variation, otherwise we rotate the object back to the upright direction.

Floor constraint. The floor constraint enforces the lowest point of the object to be on to the floor and is applied to objects such as furniture. Whether the floor constraint should be imposed can be deduced from the input scene: objects with the lowest point within a threshold of the floor surface at height 0 are constrained. We set the threshold to 1 (the height of a normal desk is about 21) in our experiments. After the refinement, we check if the lowest points of constrained objects are on the ground level. Results with objects that exceed the threshold are discarded. Otherwise, object positions are slightly nudged to make contact with the floor. Optionally, floor constraints can be turned off for objects that are not supported by the floor but are placed close to the floor level.

Collision constraint. This is imposed to ensure that objects do not inter-penetrate. Although the novel object fits well to a cell boundary at one side, it may collide with other objects on the other side. Again, the results are discarded if such collisions persist after fitting objects to both sides of the template.

6.3 Spatially Repeated Relationships

For synthesizing scenes where the same spatial relations can be found at various locations, such as hats hung at multiple hat-holders, and books kept in a bookshelf, we find multiple candidate locations around a novel object. To this end, we scan the scene after each object placement to find a new ROI that does not intersect a volume that is already occupied. Specifically, the open space points within the occupied ROIs are excluded from computing the matching score (Eq. 8) in subsequent rounds.

7 Experimental Results and Evaluation

In this section, we provide qualitative and quantitative evaluation of our synthesis results. We show several examples of pairwise synthesis results, compare them with a baseline method and demonstrate synthesis with spatially repeated relationships. Complex scene synthesis is evaluated in a user study that compares the plausibility of our results with a state-of-the-art method. Finally, we evaluate the effect of SCF parameter settings and the impact of semantic constraints on our results, compare our method to a baseline brute-force approach, and analyze the computational complexity of our method.

Pairwise synthesis. Pairwise scenes are produced by computing a relationship template from a given example object pair and fitting novel objects to both sides of the template. We synthesize results from four different example pairs, shown in Figure 9 (leftmost pairs of each type). Novel objects are retrieved from a database of unlabeled objects based on their fitting score to the exemplar's template (see Equation 4). We synthesize all possible combinations of retrieved object pairs and rank them based on their combined fitting score. We established a ground truth by manually labeling synthesized pairs as correct or incorrect. Since there is usually very little ambiguity in the correct/incorrect decision and there was practically no disagreement between our choices, we opted not to conduct a Mechanical Turk user study. Figure 9 shows our labels for the 10 best results of each query and labels for the full dataset are available in the supplementary material; highlighted objects are incorrect. The total number of correct pairs in the database is set to the number of pairs having the same type as the input exemplar.

Our method successfully retrieves and synthesizes similar pairs from the database, based only on their fit to the relationship template. Results are shown in Figure 9 to the right of each example scene. Note that similar relationships can be produced even though objects exhibit large variations in their geometry and topology. The relatively few false positives (e.g. the toothbrush in the vase) mainly occur due to object pairs of different types being separated by a similar open space as in the input exemplar. In this experiment, we always place objects at the optimal poses our method can find, therefore the objects in the shelf scene are placed at similar locations in results with similar shelves. Note that semantic constraints are imposed on the results, as discussed in Section 6.2.

Table 1 shows which constraints are applied to at least one of the objects in each of the scenes. Most constraints are applied automatically. An exception is the contact constraint, which is enabled manually, since it depends on the semantics of the contact. In the vase-flower scene, the collision constraint is disabled manually, since

scenes	contact	upright	floor	collision
hook-bag		✓		✓
desk-chair		✓	✓	✓
vase-flower		✓		
shelf-object	✓	✓	✓	✓
bench	✓	✓	✓	✓
cart	✓	✓	✓	✓
study	✓	✓	✓	✓
kitchen	✓	✓	✓	✓
bathroom	✓	✓	✓	✓

Table 1: *Semantic constraints that were applied to at least one of the objects in a scene are indicated with a checkmark.*

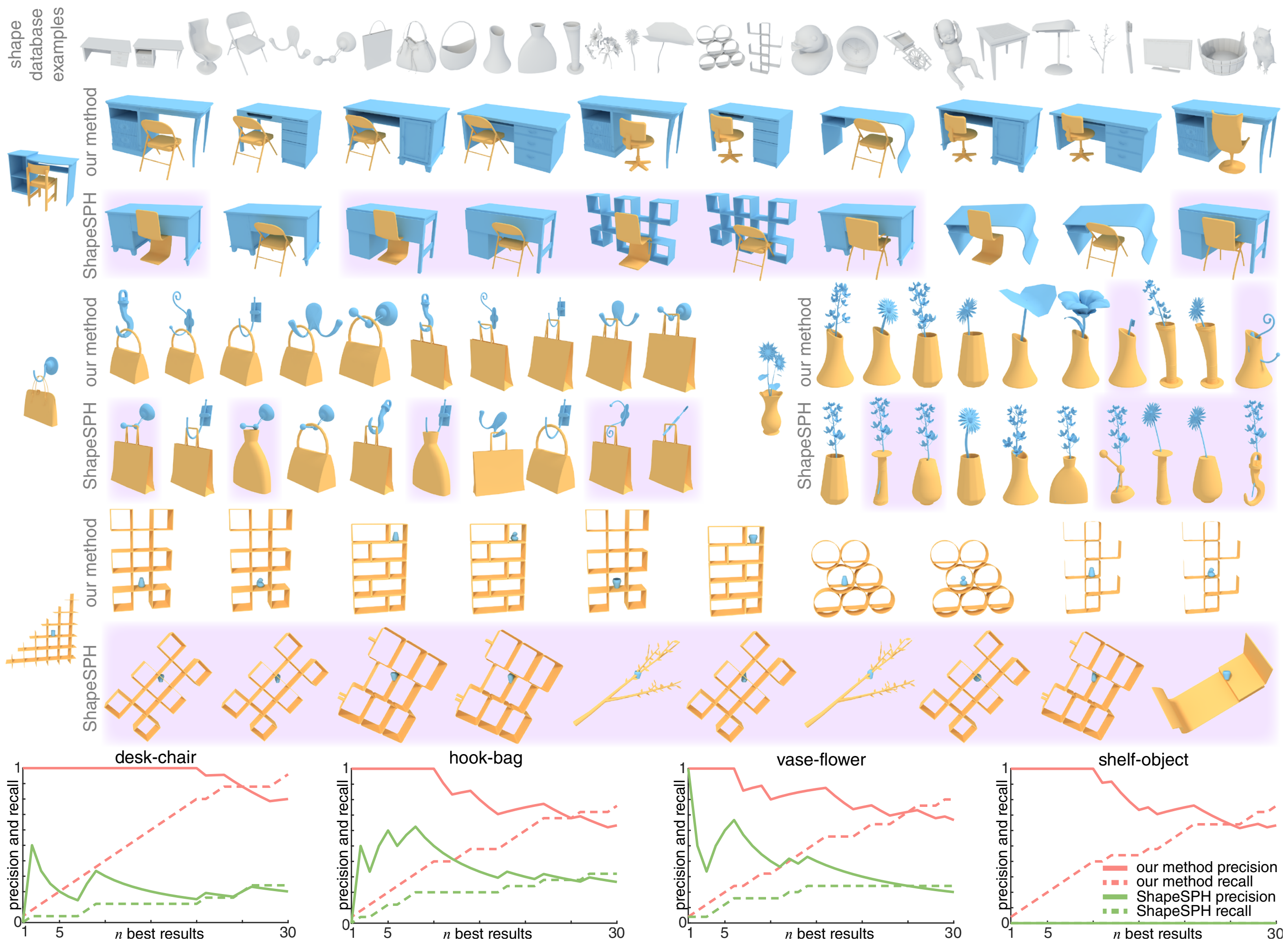

Figure 9: *Pairwise object synthesis results in four different types of complex relationships. We compare to traditional shape matching using ShapeSPH [Funkhouser et al. 2004] as a baseline. Models are taken from a mixed shape database, a few examples are shown in the top row. Below, we show the four relationship exemplars on the left, followed on the right by the 10 best results of both our method and the baseline. Incorrect results have a purple background. In the bottom row, we provide precision and recall for up to the 30 best results of both methods.*

the input contains near-collisions and results contain some borderline collisions as well, due to the limited open space sampling density.

We compare our results to a baseline shape matching method that does not take relationship between example objects into account. Objects in the example pair are replaced separately with database objects using global shape alignment with Wigner-D functions [Funkhouser et al. 2004]. We rank the resulting scenes by the minimum correlation score between fitted objects and exemplar. Results are shown in Figure 9 under those of our method. As a result of taking into account the geometry of *individual* example objects only, this method produces a large number of false positives.

We provide quantitative comparisons in Figure 9, bottom. Taking the example relationship into account allows our method to achieve significantly higher performance than our baseline that fits individual objects only. The reason is easy to see in the desk-chair example: aligning two different desks with the baseline method may cause the space for the chair to be at a place different from the exemplar, resulting in badly mismatched desk-chair pairs. A quantitative evaluation of pose synthesis without object retrieval, is shown in Figure 10, where we only retrieve objects from the subset of the database with the correct object type. Our method finds a correct pose in almost all of the results, while the baseline performs significantly worse due to not accounting for spatial relations.

Spatially repeated pairwise relations. Next we show examples where a given relationship is synthesized at multiple locations, such as hats and caps hung on hat holders and objects held in shelves. As discussed in Section 6.3, we search for multiple ROIs, giving us multiple candidates for object poses that produce similar relationships. Results of two example scenes are shown in Figure 11. Objects are correctly placed in shelves, and caps and hats are successfully hung at multiple locations on a stand despite the wide variation in object geometry.

Larger-scale scene synthesis. Scenes with more than two objects are synthesized by replacing individual objects in a given input scene using our scene hierarchy (see Section 6.1). Results of scenes synthesized from 5 different input scenes are shown in Figure 12 (c-g). Inputs range from relatively simple scenes with three objects, like the ‘garden’ scene to larger scenes like the ‘kitchen’ or ‘bathroom’ scenes. In Figure 12 (a) and (b), we also include two pairwise synthesis results to get a wider range of object counts and to examine how complex scene synthesis quality compares to pairwise synthesis quality. For each scene type, we synthesize 10 scenes, a subset of which are shown in Figure 12. Please refer to the supplementary material for the complete set of synthesized scenes.

We compare our results against results obtained from Fisher et al. [2012] using the same input scene. In the interest of a fair

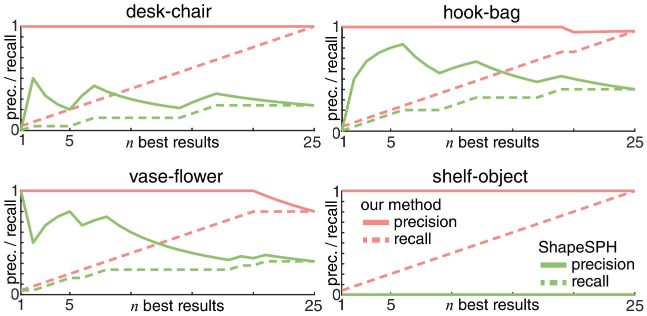

Figure 10: *Precision and recall of pairwise object synthesis using only objects of correct type. Our method (red) finds significantly better poses than the baseline method (green).*

comparison, we only use parts of Fisher's method concerned with object placement: the arrangement and layout models (described in Sections 7 and 8 of their paper). The list of objects to be used in each synthesized scene is selected by our method and used in Fisher's method as well. Note that this gives a slight advantage to Fisher's method, as only objects that are guaranteed to geometrically fit the scene are used. The arrangement model, capturing the distribution of object poses, is trained on the input scene using data augmentation, as described in Fisher et al. [2012]. Our objects are not always supported by a planar surface, therefore we extend the arrangement model to include full 3D orientations for objects without planar supporting surface. Results are displayed in the second row of each scene type in Figure 12. Objects on planar contact surfaces are placed correctly, but more complex contact surfaces like the woven baskets in Figure 12 (g) or the stack of plates in Figure 12 (f) cannot be handled by Fisher's method and result in several floating and misplaced objects. Since pairs are placed as close to their original relative pose as possible, relationships that strongly depend on the geometry of object pairs, like the chair-desk relationship in Figure 12 (e), result in sub-optimal placements.

For the bathroom and kitchen scenes (f and g), we combine our approach with Fisher's method to show that our method can be used as a component of existing scene synthesis pipelines. Fisher's method is used as described above, but objects in complex relationships, such as the stack of plates or the baby in the bathtub, are synthesized by our method, then merged and placed as single objects into the scene using Fisher's method. The input scene with similarly merged objects is used to train Fisher's method.

We rated plausibility of each synthesized scene via a user study. Users are shown images of two randomly picked scenes and asked to judge which object arrangement is more realistic. Using the Bradley-Terry Model [Hunter 2004], we obtained a 'realism' or 'plausibility' score for each image from the results of these pairwise comparisons, as well as probabilities for each image to be judged more realistic than any other image in the dataset. The study was performed in Mechanical Turk. For each scene type (e.g., 'bathroom,' 'kitchen,' etc.), we exhaustively compared all scene pairs taken from a pool of both our and Fisher's method, with 5 different users per pair, for a total of 950 comparisons per scene type. Results are shown in Figure 12, bottom row. The histograms show that our results are likely to be judged more plausible (probability > 0.5) in a significantly larger set of pairwise comparisons than Fisher's method. Since Fisher's method does not take object geometry into account, complex relationships different from support surface relationships, such as persons sitting at a desk or stacks of plates in a kitchen can not be handled.

Evaluation of SCF features. The SCF features we introduce in this work provide additional information that is not captured by IBS features proposed in [Zhao et al. 2014] alone. Each SCF captures geometric information from a large part of the model, not only from the closest point, making the SCF features more discriminative. This allows us to use a fast feature pooling approach to efficiently identify regions of interest. We demonstrate this advantage in Figure 13, where we compare synthesis results of the vase-flower scene with and without using SCF features. Regions of interest found without SCF features are incorrect, due to the limited discriminative power of IBS features. Distance features alone do not carry enough information to distinguish between the stem and other parts of the flower in our feature pooling approach.

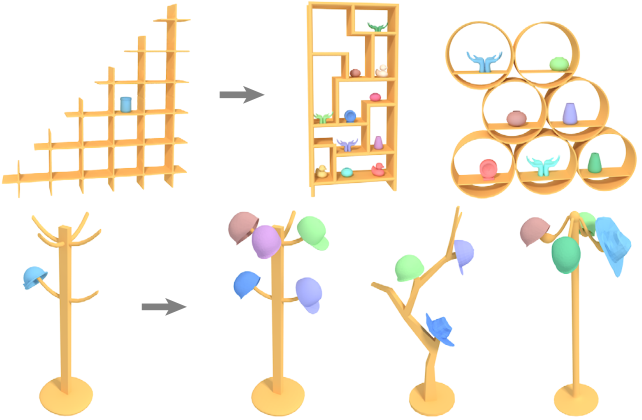

Figure 11: *Instead of synthesizing a single object for each object in the exemplar, we can synthesize multiple objects that have similar relations to the scene as the exemplar. This allows us to quickly populate scenes containing objects like shelves or stands with correctly placed objects.*

There are two parameters that can be tuned when computing SCF features: the sample density n used to sample the surrounding geometry and compute the spherical harmonics (SH) coefficients, and the number of SH bands L, corresponding to the frequency bandwidth of the resulting SCF features. We performed experiments on a small dataset consisting of 30 models typical for our scenes to determine the best bandwidth setting. Results are shown in Figure 14. Bandwidths that are too low miss important geometric detail, while too high bandwidths are not robust to small geometric variations. Our experiments suggest that $L = 10$ performs best and we use this value in our experiments. The sample density n needs to be high enough to capture all frequencies representable with the given number of SH bands. Since the maximum frequency representable with L bands is L, we could set $n = 2L$ (the Nyquist rate), but we prefer $n = 3L$ to increase robustness to sampling noise.

Impact of semantic constraints. To evaluate the impact of semantic constraints on our results, we provide a qualitative evaluation of a study area scene (Figure 12 (e)) synthesized with and without using semantic constraints in Figure 15. Without contact constraints, objects like the lamp and person (encircled in red) may float slightly above contact surfaces, due to our limited open space sampling resolution; without upright constraint, all objects may be slightly tilted, since the difference in geometry between example scene and novel object may cause a slight tilt to result in a better fit; and for the same reason, objects like the chair and the desk may be floating slightly above or below the floor when not using floor constraints. These changes are geometrically minor, but do improve the realism of the synthesized scenes.

Comparison to brute-force fitting. We compare our fitting approach to a brute-force approach that evaluates a full regular grid of poses in the open space around an object. We use a regular grid with $n_x \times n_y \times n_z \times n_\theta$ equally spaced samples. The number of directions n_θ is fixed to 100. We increase the spatial grid resolution,

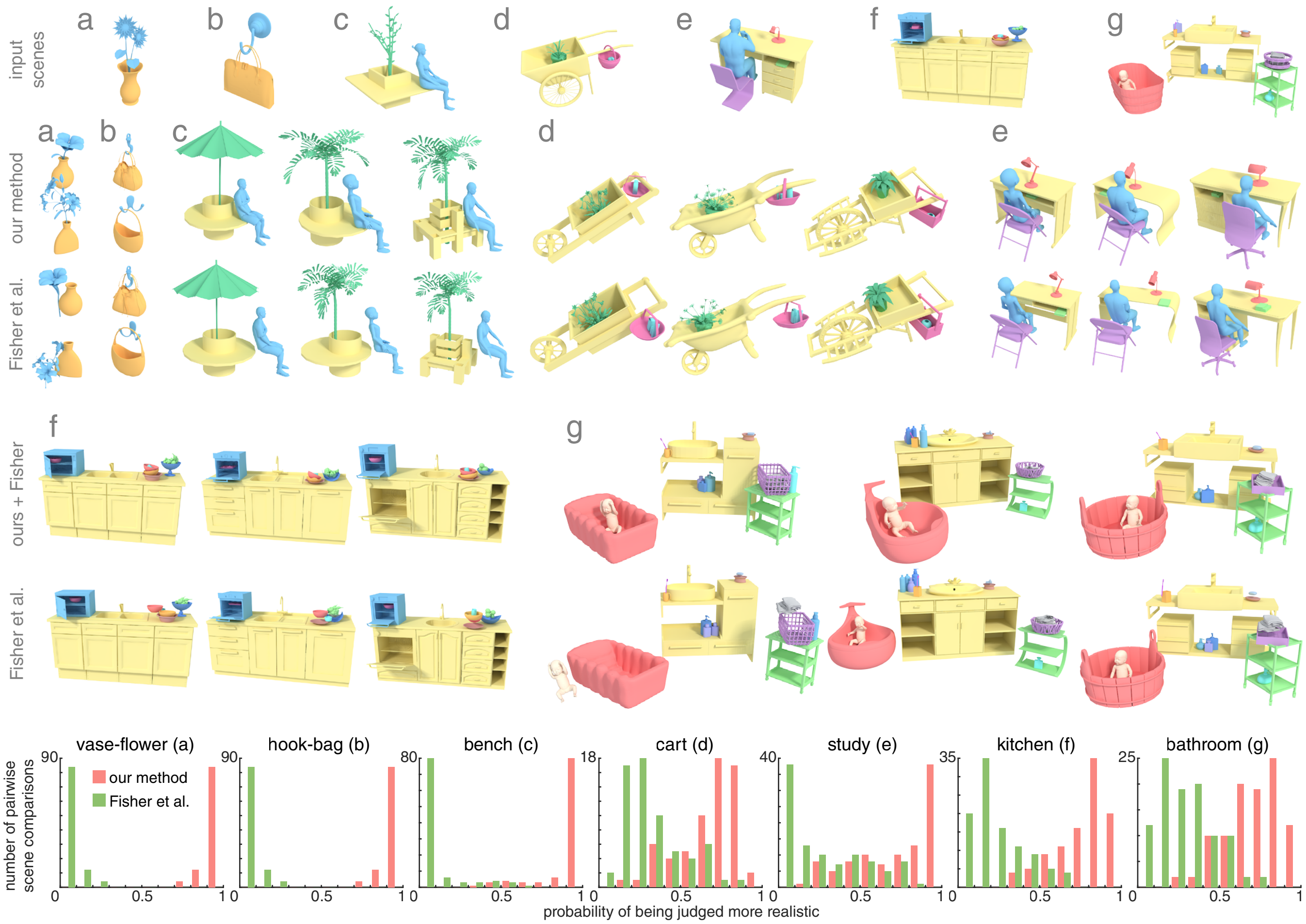

Figure 12: *Larger-scale scene synthesis results. We synthesize scenes from 5 different input exemplars (c-g) and compare to a state-of-the-art scene synthesis method [Fisher et al. 2012]. For completeness, we also include two pairwise synthesis results (a and b). In the top row, we show the input exemplars, followed below by several results from both our and Fisher's method. Note the geometrically complex relationships, like sitting persons and hanging baskets, that are difficult to handle with previous methods. Results of a user study comparing the plausibility of synthesized scenes is shown in the bottom row. We compute the probability that synthesis results from either our method (red) or Fisher's method (green) will be judged more realistic by study participants and construct histograms over all pairs of synthesis results in one scene type.*

which improves the score and increases the fitting time, and compare to our method with increasing open space point density. In Figure 16, we show the objective score vs. time required for both approaches. The brute-force approach quickly becomes unfeasible at grid densities that are necessary for acceptable scores. Our approach allows finding high-scoring poses much more efficiently.

Timings. The elementary operation of our synthesis system is fitting an object to a cell. We can divide this process into two main parts: (1) *precomputation* of open space features, and (2) *matching* which includes ROI search, initial matching and refinement. Around 60% of fitting times goes to the precomputation part, 32.1 seconds on average in our experiments. The time required for the second part takes around 13 seconds on average. For example, in Figure 9, the table-chair scene takes 9.58 seconds for matching on average while the vase-flower scene takes 15.06 seconds on a computer with i7-2600 CPU, 3.4Hz and 16G RAM.

As we replace each object in the input scene to produce new scene variations, the complexity of synthesizing a new scene is approximately linear in the number of objects of the input scene. More specifically, for an input scene with n objects and a hierarchy which defines $n-1$ pairwise combinations of these objects, the runtime is proportional to $2(n-1)$; that is a complexity of $O(n)$ for both the precomputation and the matching steps. Synthesis of spatially repeated relationships usually takes less time. When building a new scene with n objects, we have 1 base object that interacts with all other objects (e.g. the shelf), and the remaining $n-1$ objects (e.g. the objects on the shelf) only interact with one object. The runtime of the precomputation step is proportional to only n and the matching step to $2(n-1)$.

8 Discussion

As shown in the experimental results, our method is not a scheme to replace existing scene synthesis approaches, but is rather complementary to them. For example, when used as a component of an existing synthesis pipeline, our method enables scenes with more complex relations. We also do not aim to learn scene semantics from large databases in a data-driven manner, instead our goal is to describe, search for, and synthesize complex geometrical relations from a single exemplar.

A possible alternative to our approach is to apply 3D shape matching to the individual objects and swap them if they match. In our experiments, however, we have shown that simple global shape matching does not work well when objects are in close proximity, since all other objects in the scene are ignored. In contrast, our method uses the shape of the open space between objects as the representation of

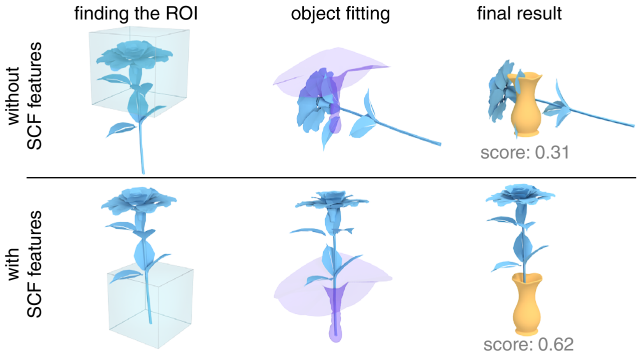

Figure 13: *Evaluating the impact of SCF features. In the top row we show the results of the ROI search step and the object fitting step, as well as the final result for our method without using SCF features (i.e. only using IBS features). IBS features alone are not discriminative enough for feature pooling in our ROI search step, resulting in a poor choice of ROI. SCF features (bottom row) provide the information necessary to find a good ROI.*

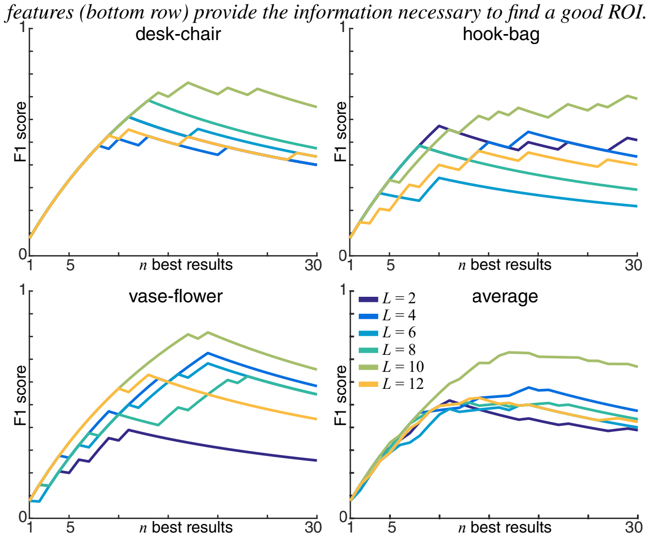

Figure 14: *Evaluation of the bandwidth parameter L of the SCF features. We show the F1 score up to the 30 best results of three input exemplars under different bandwidth settings. The average over all three input exemplars is shown on the bottom right. In our experiments we use a value of $L = 10$.*

their interactions. This is along the line of techniques such as spin images [Johnson and Hebert 1999], where a depth image produced at each vertex of the object are used as a feature. The main difference of our method is that the feature is computed from outside the object, at a location where the context is occurring. When looking at similar interactions, the shape of this open space is often more consistent than the object geometry, due to functional or ergonomic requirements of the objects' interacting geometry. We also convert it into a multi-scale representation such that the influence of high frequencies can be eliminated, especially when the objects are farther away.

The pose of objects in real scenes might be caused by various physical and semantic (i.e. human) factors. For example, in a hat-stand scene similar to Figure 11, the pose of a hat might be caused by the position where the hat was 'let go' and subsequent forces acting on it to find a physically stable resting pose on the hook. Our example-based approach tries to make the synthesis problem tractable by circumventing the extremely difficult problem of simulating all these semantic and physical factors and focusing instead on finding a geometrically similar end result.

Limitations. Currently, we only consider rigid IBS when constructing the relationship template. Therefore, it is difficult to fit objects that would produce contextually similar relationships but have shapes which are significantly different. An example is given in Figure 17, where a person is sitting on a chair. It is difficult to replace this object pair with a person lying in a reclining seat. Adding some constrained form of flexibility to the template, allowing adaption of the shape to the geometry of new objects, may enable these edits and could be an interesting extension to our work.

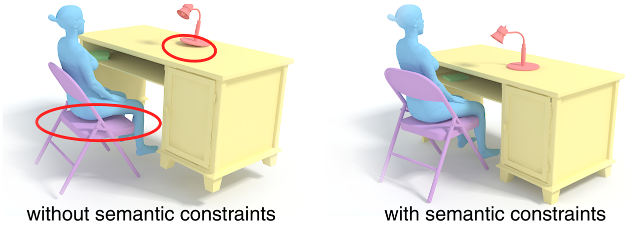

Figure 15: *Semantic constraints introduce small geometric changes to our synthesis results that make scenes semantically more realistic. They ground floating objects (encircled in red), make sure objects on the floor are placed at the correct height and keep objects like the chair or table upright.*

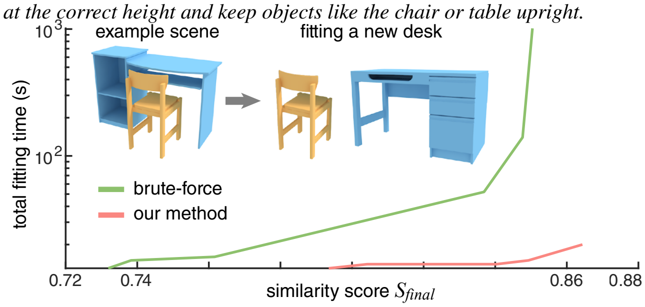

Figure 16: *Comparison to a baseline method. Brute-force object fitting using a regular grid of sample poses exhibits an exponential increase in fitting time (note the log-scale) to achieve scores comparable to our method, due to the required grid density. Our method performs significantly better.*

Another problematic scenario for our method is the restoration of contacts for objects with multiple separate contact surfaces, like handrails, or very small contact surfaces, like in the hook-bag example. In the first case, it may be hard to decide which contact to restore, in the second case it is often hard to find any contact at all, due to our limited sample density.

9 Conclusion and Future Work

We propose a method for synthesis of scenes with complex relationships, i.e. scenes where relationships are tightly coupled to the geometry. We have shown that our method based on a robust description of the open space between objects is useful for describing, searching for, and synthesizing scenes with complex relationships. Our method can be used to augment existing methods that focus on other areas of scene synthesis and performs significantly better than the state-of-the-art for scenes with complex relationships.

Avenues for future work include adding flexibility to the relationship templates to allow handling a wider range of contextually similar relationships, or to learn a parametric model of the relationship template from multiple example scenes.

Acknowledgements

We thank the anonymous reviewers for their comments and constructive suggestions, and the anonymous Mechanical Turk users. This work was supported in part by the China Postdoctoral Science Foundation (2015M582664), the National Science Foundation for Young Scholars of China (61602366),

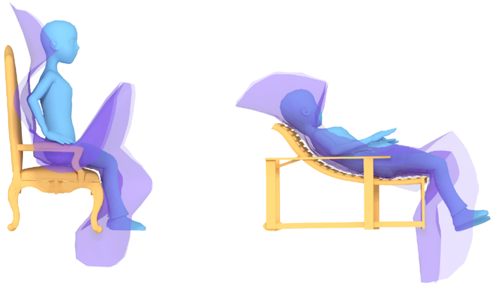

Figure 17: *A limitation of our method is the restriction to rigid IBS. The person in the chair on the right can not be matched to the exemplar on the left, since that would require deforming the IBS.*

NSFC(61232011, 61602311), Guangdong Science and Technology Program (2015A030312015, 2016A050503036), Shenzhen Innovation Program (JCYJ20151015151249564), the ERC Starting Grant SmartGeometry (StG-2013-335373), Marie Curie CIG 303541, the Open3D Project (EPSRC Grant EP/M013685/1), the Topology-based Motion Synthesis Project (EPSRC Grant EP/H012338/1) and the FP7 TOMSY.

References

BARBER, C. B., DOBKIN, D. P., AND HUHDANPAA, H. 1996. The quickhull algorithm for convex hulls. *ACM Trans. Math. Soft. 22*, 4, 469483.

CHEN, K., LAI, Y.-K., WU, Y.-X., MARTIN, R., AND HU, S.-M. 2014. Automatic semantic modeling of indoor scenes from low-quality rgb-d data using contextual information. *ACM TOG 33*, 6.

COYNE, B., AND SPROAT, R. 2001. Wordseye: An automatic text-to-scene conversion system. In *ACM SIGGRAPH*, ACM, New York, NY, USA, 487–496.

COYNE, B., SPROAT, R., AND HIRSCHBERG, J. 2010. Spatial relations in text-to-scene conversion. In *Computational Models of Spatial Language Interpretation, at Spatial Cognition 2010*.

FEI-FEI, L., AND PERONA, P. 2005. A bayesian hierarchical model for learning natural scene categories. In *IEEE CVPR*, vol. 2, IEEE, 524–531.

FISHER, M., AND HANRAHAN, P. 2010. Context-based search for 3d models. In *ACM SIGGRAPH Asia*, ACM, New York, NY, USA, 182:1–182:10.

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

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

FISHER, M., SAVVA, M., LI, Y., HANRAHAN, P., AND NIE\S SNER, M. 2015. Activity-centric scene synthesis for functional 3d scene modeling. *ACM TOG 34*, 6, 179.

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

GAL, R., AND COHEN-OR, D. 2006. Salient geometric features for partial shape matching and similarity. *ACM TOG 25*, 1, 130–150.

HU, R., ZHU, C., VAN KAICK, O., LIU, L., SHAMIR, A., AND ZHANG, H. 2015. Interaction context (icon): Towards a geometric functionality descriptor. *ACM TOG 34*, 4, 83:1–83:12.

HU, R., VAN KAICK, O., WU, B., HUANG, H., SHAMIR, A., AND ZHANG, H. 2016. Learning how objects function via co-analysis of interactions. *ACM SIGGRAPH 35*, 4 (July), 47:1–47:13.

HUANG, S. S., FU, H., WEI, L. Y., AND HU, S. M. 2016. Support substructures: Support-induced part-level structural representation. *IEEE TVCG 22*, 8 (Aug), 2024–2036.

HUNTER, D. R. 2004. Mm algorithms for generalized bradley-terry models. *Ann. Statist. 32*, 1 (02), 384–406.

JIANG, Y., KOPPULA, H. S., AND SAXENA, A. 2013. Hallucinated humans as the hidden context for labeling 3d scenes. In *IEEE CVPR*.

JOHNSON, A. E., AND HEBERT, M. 1999. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE PAMI 21*, 5, 433–449.

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

KAZHDAN, M., FUNKHOUSER, T., AND RUSINKIEWICZ, S. 2003. Rotation invariant spherical harmonic representation of 3 d shape descriptors. In *SGP*.

KAZHDAN, M., SIMARI, P., MCNUTT, T., WU, B., JACQUES, R., CHUANG, M., AND TAYLOR, R. 2009. A shape relationship descriptor for radiation therapy planning. In *MICCAI 2009*. Springer, 100–108.

KIM, V. G., CHAUDHURI, S., GUIBAS, L., AND FUNKHOUSER, T. 2014. Shape2Pose: Human-centric shape analysis. *ACM TOG 33*, 4.

KOSTELEC, P. J., 2008. SpharmonicKit.

LIU, T., CHAUDHURI, S., KIM, V. G., HUANG, Q.-X., MITRA, N. J., AND FUNKHOUSER, T. 2014. Creating consistent scene graphs using a probabilistic grammar. *ACM TOG 33*, 6.

MAJEROWICZ, L., SHAMIR, A., SHEFFER, A., AND HOOS, H. H. 2014. Filling your shelves: Synthesizing diverse style-preserving artifact arrangements. *IEEE TVCG 20*, 11, 1507–1518.

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

NAN, L., XIE, K., AND SHARF, A. 2012. A search-classify approach for cluttered indoor scene understanding. *ACM TOG 31*, 6, 137.

SAUPE, D., AND VRANIC, D. V. 2001. 3d model retrieval with spherical harmonics and moments. In *23rd DAGM-Symposium on Pattern Recognition*, Springer-Verlag, 392–397.

SAVVA, M., CHANG, A. X., HANRAHAN, P., FISHER, M., AND NIESSNER, M. 2014. Scenegrok: Inferring action maps in 3d environments. *ACM TOG 33*, 6.

SAVVA, M., CHANG, A. X., HANRAHAN, P., FISHER, M., AND NIESSNER, M. 2016. Pigraphs: Learning interaction snapshots from observations. *ACM SIGGRAPH 35*, 4 (July), 139:1–139:12.

SHAO, T., XU, W., ZHOU, K., WANG, J., LI, D., AND GUO, B. 2012. An interactive approach to semantic modeling of indoor scenes with an rgbd camera. *ACM TOG 31*, 6, 136:1–136:11.

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

SU, X., CHEN, X., FU, Q., AND FU, H. 2016. Cross-class 3d object synthesis guided by reference examples. *Computers & Graphics 54*, 145 – 153.

VAN KAICK, O., XU, K., ZHANG, H., WANG, Y., SUN, S., SHAMIR, A., AND COHEN-OR, D. 2013. Co-hierarchical analysis of shape structures. *ACM TOG 32*, 4, 69:1–69:10.

VAN KAICK, O., ZHANG, H., AND HAMARNEH, G. 2013. Bilateral maps for partial matching. In *CGF*, Wiley Online Library.

WANG, Y., XU, K., LI, J., ZHANG, H., SHAMIR, A., LIU, L., CHENG, Z.-Q., AND XIONG, Y. 2011. Symmetry hierarchy of man-made objects. *CGF 30*, 2, 287–296.

WOLFSON, H. J., AND RIGOUTSOS, I. 1997. Geometric hashing: An overview. *Computational Science & Engineering, IEEE 4*, 4, 10–21.

XU, K., MA, R., ZHANG, H., ZHU, C., SHAMIR, A., COHEN-OR, D., AND HUANG, H. 2014. Organizing heterogeneous scene collection through contextual focal points. *ACM TOG 33*, 4.

YEH, Y.-T., YANG, L., WATSON, M., GOODMAN, N. D., AND HANRAHAN, P. 2012. Synthesizing open worlds with constraints using locally annealed reversible jump mcmc. *ACM TOG 31*, 4, 56.

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 TOG 30*, 4, 86.

ZHAO, X., WANG, H., AND KOMURA, T. 2014. Indexing 3d scenes using the interaction bisector surface. *ACM TOG 33*, 5.

ZHENG, B., ISHIKAWA, R., OISHI, T., TAKAMATSU, J., AND IKEUCHI, K. 2009. A fast registration method using IP and its application to ultrasound image registration. *IPSJ Trans. Comp. Vis. and App. 1*, 209–219.

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

ZHENG, Y., TAI, C.-L., ZHANG, E., AND XU, P. 2013. Pairwise harmonics for shape analysis. *IEEE TVCG 19*, 7, 1172–1184.

A Configuration Update

Assuming there are N points sampled on the template in the k-th step, the next state of the template points can be computed as:

$$\mathbf{X}^{k+1} = \mathbf{X}^k + \mathbf{M}, \tag{9}$$

where $\mathbf{X}^k$ is a $3 \times N$ matrix encoding the location of all the template points $\mathbf{x}$ in the k-th step, $\mathbf{X}^{k+1}$ a matrix representing the goal positions in the next iteration, and $\mathbf{M}$ a $3 \times N$ matrix encoding the direction vectors at each point. The columns of $\mathbf{M}$ are computed by $\mathbf{m}_i = a_i\, \mathbf{g}(\mathbf{x}_i^k)$, where $\mathbf{g}(\mathbf{x})$ is the gradient vector of the distance field corresponding to the open space distance feature at the point $\mathbf{x}$, and signed a_i is called a move parameter, whose sign indicates whether the move is in the gradient direction or in the negative gradient direction.

The optimal rigid transformation to register $\mathbf{X}^k$ to $\mathbf{X}^{k+1}$ is computed by first computing a cross-covariance matrix $\mathbf{A}$:

$$\mathbf{A} = (\mathbf{X}^k - \bar{\mathbf{X}}^k)(\mathbf{X}^{k+1} - \bar{\mathbf{X}}^{k+1})^T, \tag{10}$$

where $\bar{\mathbf{X}}$ is a matrix whose columns are $\bar{\mathbf{x}}$. $\bar{\mathbf{x}}$ is the mean location of all points in $\mathbf{X}$. We apply a singular value decomposition to $\mathbf{A}$ as $\mathbf{A} = \mathbf{USV}^T$. The rotation matrix of the rigid transformation is obtained by $\mathbf{R} = \mathbf{UV}^T$, and the translation vector can be computed by: $\bar{\mathbf{x}}^k - \bar{\mathbf{x}}^{k+1}$.

The rigid transformation is applied to $\mathbf{X}^k$ to get the updated location of the template points, which are fed into $\mathbf{X}^k$ in Equation 9 for the next iteration. After every iteration, we compute D and check if it is small enough. We stop the iteration when either D or the update in D are smaller than thresholds.

B Normalizing Cell Feature Distances

To normalize the absolute difference of distance features, we use the upper bound $\alpha_{dis} = maxf_{dis}^{SDF} - minf_{dis}^{CELL}$ where $maxf_{dis}^{SDF}$ are the maximum values of the novel object's open space point distance features and $minf_{dis}^{CELL}$ is the minimum distance feature of cell c_i.

The L_2 distance between two SCF features can be bounded by $\alpha_{scf} = 2\sqrt{\pi}$, to see this, let $F_{vdf}(i,j)$ and $G_{vdf}(i,j)$ be two spherical functions. The corresponding scf features of them can be denoted as:

$$f_{scf} = SCF(F_{vdf}) = \{\|f_0(\theta,\phi)\|, \ldots, \|f_n(\theta,\phi)\|\}$$

$$g_{scf} = SCF(G_{vdf}) = \{\|g_0(\theta,\phi)\|, \ldots, \|g_n(\theta,\phi)\|\}$$

with

$$f_l(\theta,\phi) = \sum_{|m|\leq l} a_{l,m} Y_l^m(\theta,\phi). \tag{11}$$

We can then bound their squared L_2 distance as:

$$\begin{aligned}
\|f_{scf} - g_{scf}\|^2 &= \sum_{l=0}^{n} (\|f_l(\theta,\phi)\| - \|g_l(\theta,\phi)\|)^2 \\
&\leqslant \sum_{l=0}^{\infty} (\|f_l(\theta,\phi)\| - \|g_l(\theta,\phi)\|)^2 \\
&\leqslant \sum_{l=0}^{\infty} (\|f_l(\theta,\phi) - g_l(\theta,\phi)\|)^2 \\
&= \|F_{vdf} - G_{vdf}\|^2 \\
&= \int_\theta \int_\phi |F_{vdf} - G_{vdf}|^2 \, d\theta d\phi \\
&\leqslant \int_\theta \int_\phi 1 \, d\theta d\phi \\
&= 4\pi
\end{aligned} \tag{12}$$