# Analyzing, Optimizing and Synthesizing Scenes by Reasoning About Relationships Between Objects

Tianqiang Liu

A Dissertation
Presented to the Faculty
of Princeton University
in Candidacy for the Degree
of Doctor of Philosophy

Recommended for Acceptance
by the Department of
Computer Science
Adviser: Thomas A. Funkhouser

September 2015

# Abstract

3D scene modeling has many applications, including virtual social worlds, massively multiplayer online games, and production of catalog images. However, scene modeling is extremely tedious and challenging, due to the requirement of realism and the involvement of large numbers of objects. Therefore, automated methods for 3D scene modeling are needed. A promising approach is to first analyze the existing 3D scenes, such as the ones in online repositories (e.g. Trimble 3D Warehouse), and then use the knowledge obtained from the analysis step to create new scenes.

Although a significant amount of research has focused on developing algorithms for scene analysis and scene modeling, these processes still remain challenging for two reasons. First, the large variability of 3D scenes makes it hard to capture the commonality among scenes for scene analysis. Second, the highly constrained space of realistic 3D scenes makes it challenging to automatically create satisfactory scenes.

This dissertation pushes the limits of existing efforts on analyzing, synthesizing, and optimizing 3D scenes by reasoning about relationships between objects. First, it describes an algorithm that segments and annotates 3D scenes by considering relationships between objects in a hierarchical representation. Second, it describes a tool that optimizes a 3D scene to produce compositions by considering relationships between objects in the image space and the scene space. Finally, it focuses on style compatibility between objects, which is a relationship that has never been considered in previous scene modeling tools, and it presents a method for learning to predict the stylistic compatibility between 3D furniture models from different object classes.

In this dissertation, we find that relationships between objects are comparatively stable across scenes, and that they can serve as a strong cue for inferring annotation and segmentation of scenes. Furthermore, we also find that modeling relationships between objects helps ensure the realism of synthesized scenes. Therefore, reasoning about relationships between objects greatly facilitates scene analysis and synthesis.

# Acknowledgements

Foremost, I would like to thank my adviser, Thomas Funkhouser, for his continuous support and guidance throughout my PhD studies. In these five years, he has taught me everything from how to think and write logically, to how to approach a challenging problem and do good research. More importantly, he has set a lifetime role model for me demonstrating what it means to be a meticulous and responsible person.

I was fortunate to collaborate with Wilmot Li, Aaron Hertzmann, Jim McCann, Niloy J. Mitra, Vladimir G. Kim, Siddhartha Chaudhuri, and Qi-Xing Huang. This work would not have been possible without their insightful suggestions and invaluable advice. I am especially grateful to my internship mentors, Wilmot Li, Aaron Hertzmann, and Jim McCann for sharing knowledge, insights, and expertise with me during and after my internships at Adobe Research. I would also like to thank Adam Finkelstein, Szymon Rusinkiewicz, and Jianxiong Xiao for their help and suggestions in almost every project that I have been working on. I am also thankful to everyone in the Princeton Graphics Group. My PhD life would not have been so amazing without them.

I thank many other people who helped me in my research. I acknowledge Kun Xu for distributing Sketch2Scene data set, Evangelos Kalogerakis for distributing code. I thank Martin Enthed and Helen Crowther for giving interviews, and hundreds of anonymous workers on Amazon Mechanical Turk for participating in user studies.

Finally, I am grateful to my parents and my wife for their persistent support and encouragement.

My graduate studies and this work were supported by Princeton Fellowship, NSF (IIS-1251217, CCF-0937137), Intel (ISTC-VC), Google, and Adobe.

To the memory of my grandfather, Quanren.

To my parents.

To Xinke.

To Lucas.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

3D virtual scenes have been widely used for building virtual social worlds and massively multiplayer online games, such as Second Life [8]. Recently, furniture companies such as IKEA have begun to produce images for their catalogs by creating and rendering 3D virtual scenes [27]. Compared to photographing real scenes, rendering 3D virtual scenes provides a more efficient solution with larger flexibility for customization.

Despite a broad range of applications of 3D scenes, it is tedious and challenging to create 3D virtual scenes manually. All these applications require the resulting virtual scenes to be realistic in order to provide users with an immersive experience in games or ensure that catalog images are indistinguishable from real photos. Consequently, human modelers have to spend days or even weeks going through huge databases of 3D shapes, selecting hundreds of objects for the scene, choosing materials for each object, and carefully placing them in the scene. Therefore, (semi-) automated methods for scene modeling are needed.

Researchers have proposed several methods for scene modeling [1, 104, 71, 103], and a promising research direction is to leverage existing 3D virtual scenes [29, 102] to model new ones. The idea is to first learn what makes a scene realistic from existing

3D virtual scenes, and then use the knowledge to create new scenes. In order to obtain precise knowledge in the first step, the algorithms usually require the training examples to have perfect segmentation and annotation. Although online repositories (e.g. Trimble 3D Warehouse [92]) offer a great number of 3D scenes, the scenes are usually not semantically segmented and annotated, and the algorithms require a lot of manual work to perform segmentation and annotation in advance. Therefore, automated methods for segmenting and annotating 3D virtual scenes are needed.

Although a significant amount of research has focused on developing algorithms for analyzing, synthesizing, and optimizing 3D scenes, these problems still remain challenging. On the one hand, 3D scenes exhibit large variability in terms of object classes, object positions and geometries, which makes it hard to capture the commonality between scenes for scene analysis. On the other hand, the space of 3D scenes is highly constrained: the arrangement of objects in a scene has to be plausible (e.g. a dining table and a dining chair should be close and facing each other); the combination of objects has to be stylistically compatible (e.g. a casual, contemporary coffee table should not appear in front of a formal, antique sofa). Moreover, if the goal is to create a 2D composition from the 3D scene, the scene has to satisfy additional constraints in order to ensure that the output composition is aesthetically compelling and/or certain objects are highlighted. All these constraints make it hard to automatically create satisfactory 3D scenes.

In this dissertation, we find that reasoning about relationships between objects significantly helps address these challenges. First, although there is large variability among 3D scenes, spatial relationships between objects, particularly spatial relationships between objects within a semantic subgroup, are comparatively stable across scenes, which provide a strong cue for scene analysis. Second, reasoning about relationships between objects can greatly help ensure the plausibility and aesthetics of the output scenes in scene synthesis. For example, constraining the spatial relationships

between a chair and a table helps ensure the plausibility of a dining area; modeling style compatibility helps avoid a combination of different styles in the same room. In this dissertation, we push the limits of existing efforts on analyzing, synthesizing and optimizing 3D scenes by reasoning about relationships between objects.

First, we focus on the problem of scene analysis, and we describe an algorithm that infers the segmentation and annotation of 3D scenes by considering priors of geometries and relationships between objects in a hierarchical representation of scenes [61]. Given a collection of scene graphs with consistent hierarchies and labels, we train a hierarchical probabilistic grammar to represent the distribution of shapes, cardinalities, and spatial relationships of semantic objects within the collection. Then, we use the learned grammar to parse new scenes to assign them segmentations, labels, and hierarchies consistent with the collection. During experiments with the algorithm, we find that: it works effectively for indoor scenes commonly found online (bedrooms, classrooms, and libraries); it outperforms alternative approaches that consider only shape similarities and/or spatial relationships without hierarchy; it robustly handles moderate over-segmentation in the inputs.

Second, we focus on the problem of scene optimization, and we describe a tool that optimizes a 3D virtual scene to produce good compositions by considering relationships between objects in the image space and the scene space [63]. We define an energy function that models a variety of design constraints and image composition rules that are important for producing effective product images. Given an initial scene description and a set of high-level constraints provided by a stylist, the tool automatically generates an optimized scene by locally adjusting the 3D object transformations, surface materials, and camera parameters. The value of the tool is demonstrated in a variety of applications motivated by product catalogs. Results of a perceptual study indicate that our system produces images preferable for product advertisement compared to a more traditional camera-only optimization.

Finally, we focus on a relationship between objects that has never been considered in existing scene modeling tools: style compatibility [62]. We present a method for learning to predict the stylistic compatibility between 3D furniture models from different object classes. 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. During experiments with these methods, we find that they are effective at predicting style compatibility agreed upon by people. We also find in user studies that the learned compatibility metric is useful for a variety of novel interactive tools.

This dissertation makes the following contributions:

- an idea of modeling hierarchical structure for analyzing 3D scenes.
- a probabilistic grammar that characterizes geometric properties and spatial relationship in a hierarchical manner.
- a novel scene parsing algorithm based on dynamic programming that can efficiently update labels and hierarchies of scene graphs based on our probabilistic grammar.
- an optimization approach that takes into account a large variety of design constraints and image composition rules, and simultaneously manipulates object transformations, surface materials, and the camera view.
- a first method for learning style compatibility between 3D models from different object classes.
- a part-aware geometric feature vector that encodes style-dependent information.
- a new asymmetric embedding distance that is appropriate for estimating compatibility between objects of different classes.

# Chapter 2

# Related Work

## 2.1 Analyzing 3D virtual scenes

Although a lot of research has focused on analyzing individual CAD objects [34, 43, 83, 43, 52, 94], not much work has focused on analyzing 3D virtual scenes. Previous tools for scene retrieval [30] and scene organization [101] require as input consistently and semantically segmented and annotated virtual scenes. Therefore, they are not able to work directly on the scenes downloaded from online repositories.

Researchers have also developed methods on inferring the segmentation and annotation for point clouds or RGBD scans of scenes based on shape [35, 58] and contextual information [20, 59]. There has also been a body of work that infers such information in the image domain [79, 80, 107]. Although these methods can be applied to analyzing 3D virtual scenes, they are unable to recover the missing hierarchy of functional groups in scenes, which is critical for recognition of scene semantics at multiple scales and context-based disambiguation of object roles (a coffee table is used differently from a bedside table, even if the two are geometrically similar).

Several algorithms have been proposed to parse buildings [70, 15] and facades [91, 69, 106, 99] using a grammar. However, these methods rely on a high degree of geo-

metric and spatial regularity, and their grammar definitions and parsing algorithms are typically specific to those application domains and do not apply to the general 3D virtual scenes. In the computer vision communities, several algorithms have been proposed to parse images of indoor environments using annotated 3D geometry [21, 108]. While our goal is conceptually similar to these works, our problem setting has two main differences. First, the number of labels we consider is significantly larger than that in previous approaches. Second, parsing 3D layouts creates both opportunities and challenges for modeling geometric and spatial variations. These differences necessitate novel methods for learning spatial relationships, computing geometric similarities, and pruning the parsing search space.

## 2.2 Synthesizing and optimizing 3D virtual scenes

Several systems have been developed to assist people in creating new shapes or scenes by combining 3D parts or objects from online repositories. They suggest new parts or objects based on spatial context [102], probabilistic models [19, 48, 105], physical simulations [93], and interior design guidelines [71]. Other systems aim to create scenes completely automatically by learning object compatibilities based on substructure symmetries [110], spatial contexts [29], and object contacts [1]. These methods focus on scene plausibility only in 3D, without considering the goal of generating 2D compositions from scenes. In addition, no prior system has explicitly considered stylistic compatibility when selecting objects or parts to combine in a virtual world.

Our work is also related to scene synthesis based on aesthetic relations [67]. The method learns the target style of artifact arrangements from a single input exemplar, and then generates diverse arrangements according to the target style for different layout dimensions. Rather than styles of arrangements, we focus on learning style compatibility between objects, and optimizing the 2D composition produced by a 3D

scene layout. Several other methods have incorporated principles for image aesthetics and compositions in optimization algorithms for camera control in 3D rendering systems [76, 36, 22, 6, 5].

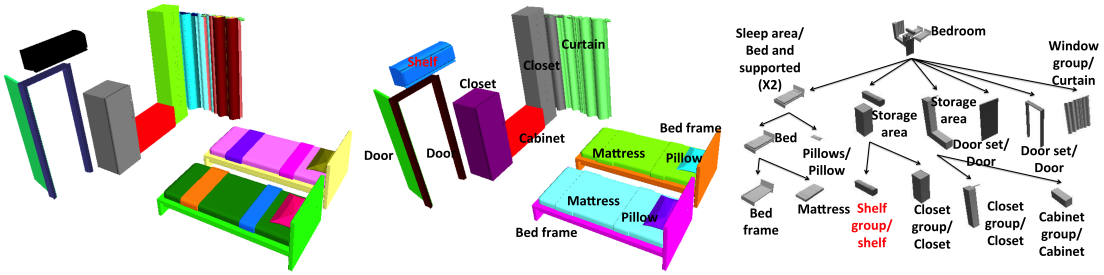
Scene synthesis is also related to 3D shape retrieval. Researchers previously have developed search algorithms for 3D models based on shape similarities [31], symmetries [51], part structures [82], and other geometric cues [89]. These methods are generally aimed to retrieve similar shapes from the same object class. Fisher et al. [28, 30] developed algorithms to perform context-based shape retrieval. Instead of specifying a shape, the user only needs to specify a region in the scene they are modeling, and the system returns a list of related objects based on the spatial relationship between the query and other objects in the scene. However, none of the previous methods have modeled style compatibility between objects or have supported style-aware scene synthesis or shape retrieval.

# Chapter 3

# Analyzing 3D Scenes Using a Probabilistic Grammar

## 3.1 Introduction

Given that manually creating 3D virtual scenes is tedious and challenging, researchers have proposed data-driven methods for synthesizing novel scenes [28, 30, 29, 102, 101], of which the common idea is to use the knowledge extracted from existing 3D virtual scenes to create new ones. Although the abundance of 3D virtual scenes in online repositories offers valuable input for the data-driven methods, these methods require consistently and semantically segmented and annotated input, and thus cannot directly leverage the typical scenes available in existing online repositories. For example, consider Figure 3.1(a) that shows a scene downloaded from the Trimble 3D Warehouse [92]. While this scene has polygons grouped into connected components and a sparse grouping of connected components into a scene graph hierarchy, many objects in the scene are not explicitly represented in the scene graph (e.g., curtain, mattress), few of the scene graph nodes are explicitly annotated with a semantic label (e.g, "table", "chair", etc.), and the scene graph hierarchy is void of any meaning-




(a) Input (b) Output leaf nodes (c) Output hierarchy

Figure 3.1: Our algorithm processes raw scene graphs with possible over-segmentation (a), obtained from repositories such as the Trimble 3D Warehouse, into consistent hierarchies capturing semantic and functional groups (b,c). The hierarchies are inferred by parsing the scene geometry with a probabilistic grammar learned from a set of annotated examples. Apart from generating meaningful groupings at multiple scales, our algorithm also produces object labels with higher accuracy compared to alternative approaches.

ful functional groups (e.g., sleeping area, storage area). This (missing) hierarchy of functional groups is critical for recognition of scene semantics at multiple scales and context-based disambiguation of object roles (a coffee table is used differently from a bedside table, even if the two are geometrically similar).

In this chapter, we focus on the problem of inferring semantic segmentation and annotation for 3D scenes, and we develop an algorithm that infers such information by building a consistent representation for the hierarchical decomposition of a scene into semantic components, which encodes priors of geometries and relationships between objects in a hierarchical representation of scenes. We achieve it in two stages. First, given a collection of consistently-annotated scene graphs representing a category of scenes (e.g., bedroom, library, classroom, etc.), we learn a probabilistic hierarchical grammar that captures the scene structure. Second, we use the learned grammar to hierarchically segment and label newly downloaded scenes. For example, for the scene depicted in Figure 3.1(a), we produce the scene graph shown in Figure 3.1(b),(c), where every functional object has been separated into a leaf node, annotated with a semantic label, and clustered hierarchically into labeled semantic groups represented

by interior nodes of the scene graph. Such a representation is useful for applications that require not only segmenting scenes into objects and clustering similar objects into semantic classes (e.g., chairs, beds, lamps), but also establishing functional roles and relationships of objects (e.g., *dining* table, *bedside* lamp, table-*and*-chairs), which are critical components of scene understanding.

Achieving such a level of scene understanding is extremely challenging. Previous methods for predicting part segmentations [49, 52], correspondences [43], and hierarchies [94] are mainly designed for single objects (e.g., chairs), which exhibit significantly less variability in the types, numbers, shapes, and arrangements of objects in comparison to scenes (e.g., bedrooms). Previous methods designed for scenes usually focus on parsing images [108] and/or work only on special types of layouts, such as building facades [106].

In our setting, the grammar specification includes hierarchical generation rules, rule probabilities, distributions of object descriptors, and spatial relationships between sibling nodes. These parameters are learned from a set of manually and consistently annotated example scene graphs, where consistency means that: i) all functionally equivalent objects and groups are assigned the same label, and ii) all hierarchical parent-child relations are the same across all scene graphs.

The learned grammar is then used to parse new scenes so that labeled object hierarchies are consistent with the training data.

In comparison to previous work on probabilistic modeling of scenes in computer graphics, a key aspect of our approach is that we explicitly learn and leverage the hierarchical structure of scenes. Prominent semantic and functional relationships exist at multiple scales in most scenes. For example, in Figure 3.1, the *bedroom* decomposes functionally into *sleeping area* and *storage area*, and each area decomposes further into objects, such as pillow, bed, cabinet and so on. Since the types of objects, numbers of objects, and spatial relationships amongst the objects are unique for each

type of area, representing the scene with a hierarchical representation (probabilistic grammar) provides great advantages for scene understanding (see also Figure 3.3).

However, using a probabilistic grammar to represent hierarchical relationships within scenes poses several novel technical challenges. In particular, parsing arbitrary arrangements of three-dimensional objects with a probabilistic grammar is a distinctly different challenge from parsing one-dimensional text [84] or two-dimensional facades [69], which allow exploitation of sequential and grid structures. The space of all possible groupings is exponentially large and intractable to explore exhaustively. Unfortunately, methods derived for lower-dimensional patterns do not directly carry over. We develop a new approach for 3D scene parsing, based on dynamic programming for belief propagation in a pruned search space. Our method binarizes the grammar, proposes a large set of candidate recursive groupings based on spatial proximity, and efficiently minimizes an energy function to find the optimal parse tree. The procedure effectively performs approximate MAP estimation of the most probable output of the hierarchical model [12].

We use our method to semantically label several datasets drawn from various publicly available scene repositories, including the Trimble 3D Warehouse [92] and the Sketch2Scene collection [102]. Our experiments demonstrate that hierarchical analysis infers more accurate object labels than (i) a descriptor-based shape classifier that does not incorporate contextual information, and (ii) an approach that uses both a shape classifier and knowledge of spatial relationships, but no hierarchical structure. Of particular note is the fact that we are able to better disambiguate similar objects used in different functional roles, e.g., "study table" vs "meeting table", which is difficult to achieve without a rich context model. Our results can be directly applied for a range of applications including scene retrieval, organization, and synthesis.

## 3.2 Related Work

### 3.2.1 Joint shape analysis

Recently, there has been a growing interest in data-driven shape analysis, which aims to aggregate information from a collection of related shapes to improve the analysis of individual shapes. Significant progress has been made in the areas of joint shape segmentation [34, 43, 83, 41, 109] and joint shape matching [74, 53, 45, 52, 42]. However, these methods are designed for collections of individual objects (e.g., chairs) and assume relatively small numbers of sub-parts and largely consistent overall layouts. This assumption does not hold for 3D scenes, which exhibit significantly greater variability in type, number, and arrangements of sub-objects.

### 3.2.2 Hierarchical shape analysis

Several previous techniques demonstrate the advantages of a hierarchical representation. Wang et al. [95] propose hierarchical decompositions of man-made objects into symmetric subgroups. However, their method does not apply to general indoor environments where semantic object groups are not necessarily symmetric. Van Kaick et al. [94] present a method that infers consistent part hierarchies for a collection of shapes. The method takes as input a set of shapes, each pre-segmented into primitive parts. Candidate part hierarchies are built up by recursive grouping, and the set of hierarchies clustered by similarity. Within each cluster, a representative hierarchy is used as a template to re-parse the shapes. The method assumes that the collection can be split into discrete clusters, where each cluster contains shapes with essentially identical part hierarchies. This assumption is often violated in 3D scenes, where each scene layout is in general only partially similar to others, with corresponding sub-layouts but no overall commonality. We use a probabilistic grammar to model

the different regions, at different scales, of different scenes with different rules of a common generative process.

### 3.2.3 Layout parsing

In the computer graphics community, grammar-based scene parsing has been an active research area. However, most existing methods in this area have focused on parsing cities [91], buildings [70, 15], and facades [69, 106, 99], which exhibit a high degree of geometric and spatial regularity. The grammar definitions and parsing algorithms being developed are typically specific to those application domains and do not apply to the scenes considered in our work.

### 3.2.4 Inverse procedural modeling

Several researchers have also studied the problem of inverse procedural modeling: recovering a generative grammar from a set of shapes assumed to have self-repeating hierarchical structures. For example, Šťava et al. [86] derived L-systems from plants; Bokeloh et al. [14] discovered repeating units and connections to form a procedural model for shapes; while, Talton et al. [88] applied Bayesian Model Merging to induce a compact grammar for a collection of shapes. These methods are complementary to ours: they focus on learning a grammar from existing training data for the purpose of shape synthesis, instead of trying to derive structure for novel data.

### 3.2.5 Synthesis with probabilistic models

Our work is also related to works on generating new shapes and scenes with data-driven probabilistic modeling. Chaudhuri et al. [19] and Kalogerakis et al. [48] train generative models of component-based shape structure from compatibly segmented and labeled models for shape synthesis, while Fisher et al. [29] and Yeh et al. [103]

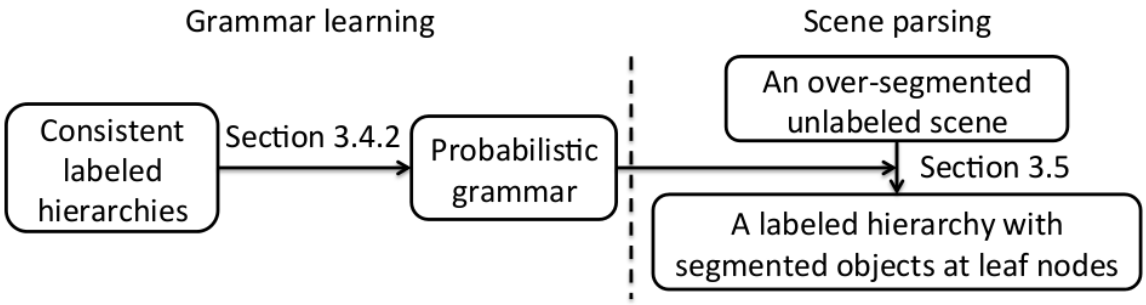


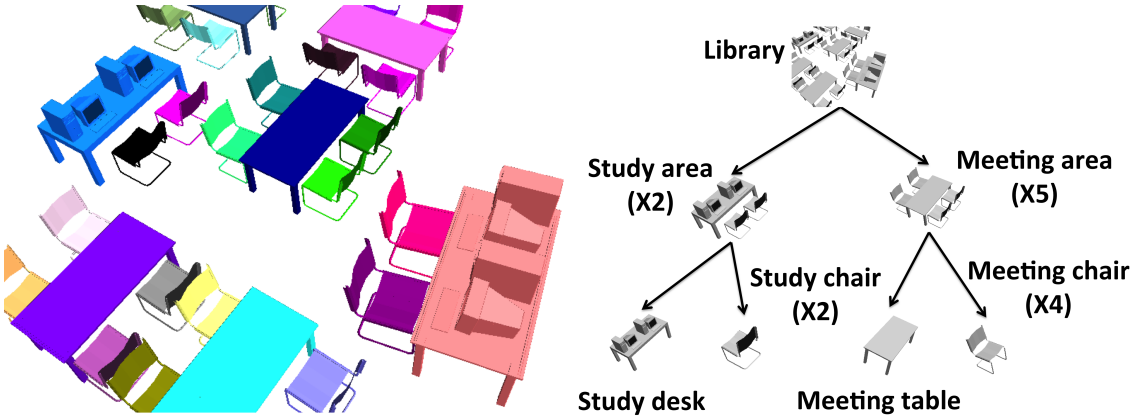

Figure 3.2: Flow chart of our approach. We learn a probabilistic grammar from consistently annotated training hierarchies. We then leverage this grammar to parse new scenes (which might include over-segmented objects). The output is a labeled hierarchy consistent with the grammar and assigned a high probability by it.

characterize spatial relationships among objects in 3D scenes for scene synthesis. Although these models are very effective for synthesis, they are not applicable to segmentation and labeling of novel scenes, and do not have a rich representation of hierarchical context. As we show in our evaluations, hierarchical contexts can greatly aid recognition tasks and improve accuracy.

## 3.3 Overview

The main objective of this work is to automatically create consistent annotated scene graphs for a collection of related scenes. To achieve this goal, our system starts by learning a probabilistic grammar from a training set of annotated 3D scenes with consistent hierarchical structure. Then, given a new input scene described by unlabeled non-semantic scene graph, such as the one presented in Figure 3.1(a), we use the learned grammar to produce a semantic hierarchical labeling of a scene with objects at the leaves.

Our hierarchical representation and analysis tools are motivated by the observation that semantic and functional relationships are often more prominent within some subregions or subgroups of objects. For example, consider the library scene in Figure 3.3. It contains several meeting and study areas, where each area provides




(a) Input scene (b) Output hierarchy

Figure 3.3: An example library scene. By grouping objects, we are not only able to detect interesting intermediate-level structures, e.g. study area and meeting area, but also distinguish objects based on their functionalities, e.g. study chair and meeting chair.

a strong prior on spatial relationships between the objects, and types and numbers of the objects. In particular, meeting area is likely to have chairs arranged so that people could face one another, while study area is likely to provide more personal space on a desk (and thus, would have fewer chairs). In addition, hierarchy provides the necessary context to distinguish functional categories of shapes that otherwise have very similar geometry such as meeting and study chairs.

Our approach is defined by two stages. In the first stage, we learn a probabilistic grammar from a set of example scenes. In particular, given a set of consistently annotated hierarchical scene graphs as the training data, we produce hierarchical production rules, production probabilities, distributions of object descriptors and spatial relationships between sibling nodes, which define our grammar (see Section 3.4). Then, in the second stage of our pipeline, we use the learned grammar to compute consistent scene graphs for novel 3D scenes. We assume that the new scenes come from an online repository, and thus unlikely to have semantic annotations or consistent scene graphs. A typical scene from a Trimble 3D Warehouse is missing some important hierarchical nodes, has nodes that corresponds to meaningless groups, does

not have objects as leaf nodes, since it further subdivides them into meaningless geometric parts (which we refer to as an over-segmentation problem). We solve the challenging problem of matching these geometry soups to meaningful objects, and then organizing the objects into consistent hierarchies by using an efficient dynamic programming algorithm (see Section 3.5).

## 3.4 Probabilistic Grammar

In this section, we first define the probabilistic grammar, and then describe how we learn the grammar parameters from annotated training data.

### 3.4.1 Grammar specification

We define an attributed, non-recursive, probabilistic grammar $\mathcal{G}$ represented by a tuple:

$$\mathcal{G} =< \mathbf{L}, \mathbf{R}, \mathbf{P} > \tag{3.1}$$

where $\mathbf{L}, \mathbf{R}$ define the topology of the grammar, and $\mathbf{P}$ are its probabilistic parameters. We model $\mathcal{G}$ to be non-recursive as object groups in indoor scenes are not expected to be functionally equivalent to any of the group's components.

**Labels.** The *label set* $\mathbf{L}$ is a list containing a label for each object category (e.g., *bed*, *chair*) and object group (e.g., *sleeping-area*, *table-and-chairs*). We include a special label $w$ that denotes the axiom of $\mathcal{G}$. We also include a special label for each object category that denotes a non-semantic subpart of the complete object, such as the curtain pieces in Figure 3.1. Introducing these labels helps us parse oversegmented scenes where the leaf levels of input scene graphs are below the object level.

**Rules.** The *rule set* $\mathbf{R}$ comprises production rules of the grammar. Each production rule $r \in \mathbf{R}$ is in the form of $l \to \lambda$, where $l \in \mathbf{L}$ is the left-hand-side label, and $\lambda$ is the set of right-hand-side labels. For example, a production rule could be:

$$\textit{bed} \;\to\; \textit{bed-frame} \;\; \textit{mattress}.$$

Since our grammar is non-recursive, $\lambda$ should not include $l$ or any label that has $l$ in its expansion. In other words, the labels $\mathbf{L}$ can always be topologically sorted.

**Probabilities.** The parameters $\mathbf{P}$ include production probabilities and attributes. The probability of a production rule $l \to \lambda$ is the product of two terms. The *derivation term* $P_{\text{nt}}(l)$ denotes the probability that a non-terminal with label $l$ is composed of sub-objects according to the rule, given its parents. The *cardinality term* $P_{\text{card}}[l, r](i)$ denotes the probability that a node with label $l$, expanded according to the rule, has exactly $i$ children labeled $r$. We represent the distribution $P_{\text{card}}[l, r](i)$ by recording probabilities for four possible cardinalities: $i = 1, 2, 3, 4+$, where $4+$ denotes cardinalities of 4 or greater. The purpose of the cardinality term is to avoid introducing a new production rule for each combination of child labels and cardinalities. Instead, $\lambda = \text{RHS}(l)$ exhaustively lists all possible children of $l$, and $P_{\text{card}}$ assigns cardinality probabilities independently to each child. For example, the observed productions:

$$\begin{aligned} \textit{storage-area} \;&\to\; \textit{cabinet} \;\; \textit{trunk} \\ \textit{storage-area} \;&\to\; \textit{closet} \;\; \textit{trunk} \end{aligned}$$

are combined into a single production:

$$\textit{storage-area} \;\to\; \textit{cabinet} \;\; \textit{closet} \;\; \textit{trunk}.$$

Thus, our learned grammar has exactly one rule for each left-hand label, with independent cardinality distributions for each right-hand label. The purpose of this relaxation is to generalize in a reasonable manner from a small number of training examples. For instance, in the above example, we generalize to include storage areas with both cabinets and closets, which is not an uncommon scenario. While this relaxation can theoretically miss some co-occurrence constraints, we found it gave good results in practice.

With this setup, we define the probability that a proposed node $x$ in a parse tree, with children $x.\text{children}$, matches rule $l \to \lambda$ as:

$$\begin{aligned} P_{\text{prod}}(x) &= P_{\text{nt}}(x.\text{label}) \\ &\quad \times \prod_{r \in \lambda} P_{\text{card}}[x.\text{label}, r] \left( \sum_{y \in x.\text{children}} \mathbf{1}_{y.\text{label}=r} \right) \end{aligned} \tag{3.2}$$

where $x$ is a node in the parse tree labeled as $x.\text{label}$ with a set of children $x.\text{children}$, and $\mathbf{1}$ is the indicator function.

**Attributes.** We identify two types of attributes that are important for scene understanding: *geometry attributes* $A_g$ which describe the shape of objects, and *spatial attributes* $A_s$ which describe the relative layout of objects in a group. For example, in a library scene such as the one in Figure 3.3, $A_g$ would help in distinguishing tables and chairs since they have distinctive geometry, and $A_s$ would capture the distinctive spatial arrangement of chairs in a meeting area in contrast to a study area.

A geometry attribute $A_g$ is associated with each label $l \in \mathbf{L}$ and represented as a multivariate normal distribution over 21-dimensional shape descriptors $D_g$. To build the shape descriptors, we uniformly sample 1024 points on each shape, and then compute the following values:

- Dimensions of the axis-aligned bounding box of a shape (8 dimensions). We assume that $z$ is pointing up, and we compute $z_{\min}, z_{\max}, l_z = z_{\max} - z_{\min}, l_1 = \max(x_{\max} - x_{\min}, y_{\max} - y_{\min}), l_2 = \min(x_{\max} - x_{\min}, y_{\max} - y_{\min}), l_2/l_1, l_z/l_1, l_z/l_2$

- Descriptors from PCA analysis (7 dimensions). We perform PCA analysis for all points on the ground plane and on the upward, $z$-axis, separately. We denote the mean of $z$ values by $z_{\text{mean}}$, variance on $z$ axis by $V_z$, and variances on the ground plane by $V_1, V_2 (V_1 \geq V_2)$, and we include $z_{\text{mean}}, V_1, V_2, V_z, V_2/V_1, V_z/V_1, V_z/V_2$.

- Descriptors of "uprightness" (2 dimensions). We compute the fraction of points that have "up" as the principle direction of their normal. We denote the fraction by $r$ and include $r$ and $1 - r$ as features.

- Point distribution along the upward direction (4 dimensions). We compute a 4-bin histogram of points according to their $z$ coordinates.

We assume that the individual features in the descriptor are independent, and model the distribution of the $i^{\text{th}}$ feature with a Gaussian $G_{l,i}$. Given a new node $x$, we estimate the probability of it being labeled $l$ via the geometry attribute $A_g[l]$:

$$P_g(l, x) = \prod_{i=1\ldots21} \frac{1}{\sqrt{2\pi}\sigma_{l,i}} \exp\left(-\frac{(D_{g,i}(x) - \mu_{l,i})^2}{2\sigma_{l,i}^2}\right) \tag{3.3}$$

where $\sigma_{l,i}$ and $\mu_{l,i}$ are respectively the mean and the variance of $G_{l,i}$, and $D_{g,i}(x)$ is the $i^{\text{th}}$ component of $D_g(x)$.

A spatial attribute $A_s$ describes a probability distribution over object layouts. We assume that objects that appear in the same group in the hierarchy have a stronger prior on their relative spatial relations. Thus, we only capture $A_s$ for label pairs that are siblings on the RHS of a rule: the attribute is conditional on the LHS label. To generalize from sparse training data, we factor the joint distribution of the layout of a group of objects into a product of pairwise layouts. The relative layout of two nodes

$x$ and $y$ is described with a 7-dimensional descriptor $D_s(x, y)$:

$$
\begin{aligned}
D_s(x,y) = [&x.z_{\min} - y.z_{\min}, \\
&x.z_{\min} - y.z_{\max}, \\
&x.z_{\max} - y.z_{\min}, \\
&\textbf{Dist}(x.\text{box.center}, y.\text{box.center}), \\
&\textbf{Dist}(x.\text{box}, y.\text{box}), \\
&Area(x.\text{box} \cap y.\text{box})/Area(x.\text{box}), \\
&Area(x.\text{box} \cap y.\text{box})/Area(y.\text{box})]
\end{aligned}
$$

where $x$.box is the bounding box of the object $x$ on the ground plane, **Dist** is the distance between two points or two bounding boxes. Intuitively, 1-3 represents support and vertical relationships, 4-5 represents horizontal separations, and 6-7 represents overlaps between objects.

Note that these pairwise relations are typically not distributed around a single mean value. For example, the spacing between all pairs of chairs arranged evenly around a table jumps discretely as the table grows larger and accommodates more chairs. Thus, we use kernel density estimation [78], a non-parametric technique, to represent the probability distribution. For each triplet of labels $l_p, l_x, l_y$, where $l_p$ is the parent of $l_x$ and $l_y$ according to a grammar rule, we find matching parent-child triplets $p, x, y$ in training scenes, and store the pairwise descriptor of each such pair $x, y$ in the set $W[l_p, l_x, l_y]$. As for the geometry attribute, we assume the individual features vary independently. The $i^{\text{th}}$ dimension of each exemplar descriptor $w$ in the set is associated with a local Gaussian kernel $K_{w,i}$ centered at $w$ that describes the distribution in its proximity. The overall probability at any point in the descriptor space, for any pair of sibling objects $x, y$, is the product of the sums of these 1D

kernels:

$$P_s(l_p, l_x, l_y, x, y) = \prod_{i=1...7} \sum_{w \in W[l_p, l_x, l_y]} K_{w,i}\left(D_s(x, y)\right) \tag{3.4}$$

By taking the product over sums, instead of the sum over products, we again encourage generalization from a few examples.

### 3.4.2 Learning the grammar from consistent labeled hierarchies

**Scene labeling.** Given a collection of 3D scenes from a public repository with their default (possibly non-semantic) scene graphs, an annotator builds consistent hierarchies for the scenes. We instructed the annotator to follow the following four steps:

1. Identify leaf-level objects in each scene either by selecting a node in the existing scene graph or by grouping multiple non-semantic nodes to form an object.

2. Provide a label for each object in a scene.

3. Group objects that belong to the same semantic group and provide a group label that is consistent across all scenes. This step is performed recursively until only one group (the axiom) is left.

4. Summarize the resulting annotations in a form of a grammar. The annotator is presented with all production rules and is asked to remove redundancies in the grammar and potentially relabel the scenes that include these redundancies. This step is introduced to favor consistent annotations after all scenes are labeled.

In our experiments, the annotation took about 15 minutes per scene. The first step was only required for over-segmented scenes and could take up to 30 minutes for a scene with 300 segments.

**Grammar generation.** The set of all unique labels in the training scenes defines $\mathbf{L}$. For each non-terminal label $l \in \mathbf{L}$, we create a rule $(l \rightarrow \lambda) \in \mathbf{R}$, where $\lambda$ concatenates all labels that act as children of $l$ across all scenes, generalizing from the individual observed productions. The derivation probability $P_{\text{nt}}$ and cardinality probability $P_{\text{card}}$ of each rule are directly learned from occurrence statistics in training data.

We then proceed to compute the geometric and spatial attributes. The means and variances of geometry attribute Gaussians are estimated from the set of descriptors of observed instances of each label. The kernel bandwidths (variances) of spatial attributes, for each pair of observed siblings $x, y$ are chosen differently for each dimension, based on the type of relation that we expect to capture. In particular, for dimensions describing vertical separations, we predefine a small bandwidth of 7.5cm since we expect support and co-planarity relations to hold almost exactly up to minor misalignments introduced by a modeler. For spatial relations on the ground plane, we estimate the bandwidth as $0.2 \times \min\{x.\text{box.diagonal}, y.\text{box.diagonal}\}$, where $a.\text{box.diagonal}$ is the bounding-box diagonal of $a$, since we expect the variance in these to be proportional to the object size. For overlap-related dimensions, we predefine a tiny bandwidth of 0.05cm, since we generally do not expect objects to intersect.

## 3.5 Scene Parsing

Given a learned grammar $\mathcal{G}$ and an input scene graph $S$, our goal is to produce an annotated hierarchy $H$ on scene geometry that is a valid parse of $S$ according to $\mathcal{G}$. We first extract the set of leaf nodes $S_{\text{leaf}}$ from $S$, which forms a partition of the scene. These leaves do not necessarily correspond to semantic objects or groups. We assume that $H$ has $S_{\text{leaf}}$ as leaf nodes, assigning them special "object-subpart" labels from the grammar in the case of over-segmentation.

In the rest of this section, we formulate the objective function that is equivalent to maximizing $P(H|S,\mathcal{G})$ (Section 3.5.1), and propose an efficient dynamic programming algorithm to find the optimal hierarchy (Section 3.5.2).

### 3.5.1 Objective function

Given a grammar $\mathcal{G}$ and an input scene $S$, our goal is to produce an annotated hierarchy $H^* = \arg\max_H P(H|S,\mathcal{G})$. We rewrite $P(H|S,\mathcal{G})$ using Bayes' rule, dropping the $P(S)$ in the denominator because it does not affect the optimal solution:

$$P(H|S,\mathcal{G}) \propto P(H|\mathcal{G}) \cdot P(S|H,\mathcal{G}). \tag{3.5}$$

$P(H|\mathcal{G})$ is the product of production probabilities of rules $P_{\text{prod}}$ (Equation 3.2) in $H$:

$$P(H|\mathcal{G}) = \prod_{x \in H} P_{\text{prod}}(x)^{T(x)} \tag{3.6}$$

where $T(x)$ is a weight that is used to compensate for decreasing probability values as $H$ has more internal nodes. We define $T(x) = 1$ for leaves and internal nodes that have a single child, and $T(x) = |x.\text{children}| - 1$ for all others.

$P(S|H,\mathcal{G})$ is the *data likelihood*, which is the probability of $S$ being a realization of the underlying parse $H$. We define the data likelihood of scene as a product of per-node likelihoods:

$$P(S|H,\mathcal{G}) = \prod_{x \in H} P_g(x)^{T(x)} P_s^*(x)^{T(x)} \tag{3.7}$$

where the geometry term $P_g$ is defined in Equation 3.3 and the full per-node spatial probability $P_s^*(x)$ is derived from the pairwise terms $P_s$ (Equation 3.4):

$$\log P_s^*(x) = \frac{\sum_{p,q \in x.\text{children}} \log P_s(x.\text{label}, p.\text{label}, q.\text{label}, p, q)}{|x.\text{children}| \times (|x.\text{children}| - 1)} \tag{3.8}$$

Our final objective function is the negative logarithm of Equation 3.5:

$$\mathbf{E}(H) = \sum_{x \in H} E(x) \tag{3.9}$$

where $E(x) = -T(x) \log \left( P_{\text{prod}}(x) P_g(x) P_s^*(x) \right)$.

### 3.5.2 Algorithm

The main challenge in optimizing Equation 3.9 is the size of the solution space. For example, if there are $n$ nodes at the leaf level, even a single group can be formed in $2^n - 1$ different ways. Previous approaches such as Zhao and Zhu [108] use simulated annealing, which requires a good initial guess and typically takes a long time to converge. While this approach is feasible for a small number of labels (e.g., 11 labels are used in the bedroom grammar of [108]) we had to develop an alternative technique to handle the typical scenes available in online repositories (e.g., there are 132 semantic labels in our grammar for the bedroom scenes presented in our work).

Our idea is to conduct a dynamic programming optimization. We start by rewriting the objective function recursively, as

$$\begin{aligned} \mathbf{E}(H) &= \bar{E}(\mathbf{Root}(H)) \\ \bar{E}(x) &= E(x) + \sum_{y \in x.\text{children}} \bar{E}(y) \end{aligned} \tag{3.10}$$

where $\bar{E}(x)$ represents the total energy of the subtree rooted at node $x$, and $\mathbf{Root}(H)$ is the root node of $H$. This recursive formulation naturally leads to a dynamic programming optimization where we choose the optimal tree structure and labels in a bottom-up manner. We define a *state* in our dynamic programming for a node $x$ and a label $l$, and we store a variable $Q(x, l)$ for the state that represents the

optimal energy of the subtree rooted at node $x$ and label $l$. Given this definition, $\mathbf{E}(H) = Q(\mathbf{Root}(H), w)$.

Since it is impractical to conduct dynamic programming algorithm directly due to the large search space, we propose two relaxations that lead to an approximated but efficient solution. First, we precompute a set of good candidate groups and assume that the hierarchy only includes nodes from these groups. Although this reduces the search space significantly, the number of ways to map a collection of nodes to the right-hand-side of a grammar production is still exponential if the branching factor of the grammar is not limited. Thus, inspired by grammar binarization techniques in natural language processing [26], we convert each rule with more than two right-hand labels into a set of rules with only one or two children. This reduces the number of states in a dynamic programming solution from exponential to polynomial in $n$. After we get a valid parse with the binarized grammar, we transform it to a valid parse with the original grammar. Although there are no guarantees that this procedure produces the optimal parse with respect to the original grammar, our experiments demonstrate that it produces semantic hierarchies with high accuracy.

In summary, our scene parsing works in three steps. First, it creates candidate nodes based on spatial proximity. Next, it binarizes the grammar. Finally, our method finds the optimal binary hierarchy with an efficient dynamic programming algorithm and then converts it to a valid hierarchy of the original grammar.

### Proposing candidate groups

Given a scene $S$ with a set of leaves $S_{\text{leaf}}$, our algorithm narrows down the search space by proposing a set of candidate groups $\mathbf{C}$ from which we build the hierarchy $H$. Each group $X \in \mathbf{C}$ is a subset of leaves in $S_{\text{leaf}}$, and our grouping heuristic for constructing $X$ stems from the assumption that only shapes that are close to one another produce semantically meaningful groups.

We iteratively build the set of subsets $\mathbf{C}$, increasing the cardinality of subsets with each iteration. In the first iteration, we set $\mathbf{C}_1 = S_{\text{leaf}}$, i.e., all subsets of cardinality 1. In iteration $k$, we enumerate all subsets of cardinality $k$ that can be created by merging pairs of subsets in $\mathbf{C}_{k-1}$. Each subset is scored using a *compactness* metric $\mathbf{M}$ that favors tight arrangements of nearby objects. Specifically, for a given subset $X$, we build a graph $A$ on $X$ where the weight of an edge is the distance between bounding boxes of its endpoints. $\mathbf{M}(X)$ is the cost of the minimum spanning tree of $A$. We add the $c$ most compact new subsets to $\mathbf{C}_k$. The iterations terminate when $k = |S_{\text{leaf}}|$. Note that this procedure guarantees that the maximal size of $\mathbf{C}$ is $O(c|S_{\text{leaf}}|^2)$. We set $c = 5$ in our experiments.

**Grammar binarization**

The goal of this step is to produce a grammar that is similar to the input grammar, but has a branching factor $\leq 2$, i.e., each rule has one or two right-hand labels. We derive the binarized grammar $\mathcal{G}_2 =< \mathbf{L}', \mathbf{R}', \mathbf{P}' >$ from the original grammar $\mathcal{G} =< \mathbf{L}, \mathbf{R}, \mathbf{P} >$ by splitting each rule into multiple equivalent rules. First, for each label $l \in \mathbf{L}$ we add two labels to $L'$: $l$ itself, which we call a *full label*, and $l'$, which we call a *partial label* of $l$. Then, we decompose each production rule $(l \rightarrow \lambda) \in \mathbf{R}$ into a set of rules with at most two right-hand labels:

$$
\begin{array}{lcll}
l & \rightarrow & l'k & \text{for each } k \in \lambda \\
l & \rightarrow & jk & \text{for each } j, k \in \lambda \\
l & \rightarrow & k & \text{for each } k \in \lambda \\
l & \rightarrow & l'l' & \\
l' & \rightarrow & l'k & \text{for each } k \in \lambda \\
l' & \rightarrow & jk & \text{for each } j, k \in \lambda \\
l' & \rightarrow & l'l'. &
\end{array}
\tag{3.11}
$$

Since the binarized grammar lacks the cardinality term, we introduce recursion to represent multiple instances of the same object. There are many possible binarization expansions that would lead to a language that is equivalent to the original grammar, each with a different number of rules and a different number of states to be searched when parsing. We did not aim to minimize the number of rules, since more rules lead to more states in the dynamic programming algorithm, thus the algorithm is more likely to find a lower energy solution. We will discuss more details next.

### Dynamic programming

Now we describe an efficient dynamic programming algorithm for minimizing the Equation 3.9. Note that given the two relaxations described above, the solution of our algorithm can only approximate the optimal solution.

In order to define the state transfer equations, we introduce an auxiliary variable for each state $[x, l]$, $\mathbf{K}(x, l)$, which represents the annotated partition of $x$ into nodes with full labels that produces $Q(x, l)$. $\mathbf{K}(x, l)$ is an array of pairs, and each pair consists of a descendant of $x$ and its label. Now we can define the state transfer equations as follows,

$$\begin{aligned} Q(x,l) &= \min\{Q_u(x,l), Q_b(x,l)\} \\ Q_u(x,l) &= \min_{l' \in \mathbf{RHS}(l)} E_2(x, l, \mathbf{K}(x, l')) + S(x, l') \\ Q_b(x,l) &= \min_{\substack{y,z \in \mathbf{Part}(x) \\ l_y, l_z \in \mathbf{RHS}(l)}} E_2(x, l, \mathbf{K}(y, l_y) \cup \mathbf{K}(z, l_z)) \\ &\quad + S(y, l_y) + S(z, l_z) \\ S(x,l) &= \sum_{[y,l_y] \in \mathbf{K}(x,l)} Q(y, l_y) \end{aligned} \tag{3.12}$$

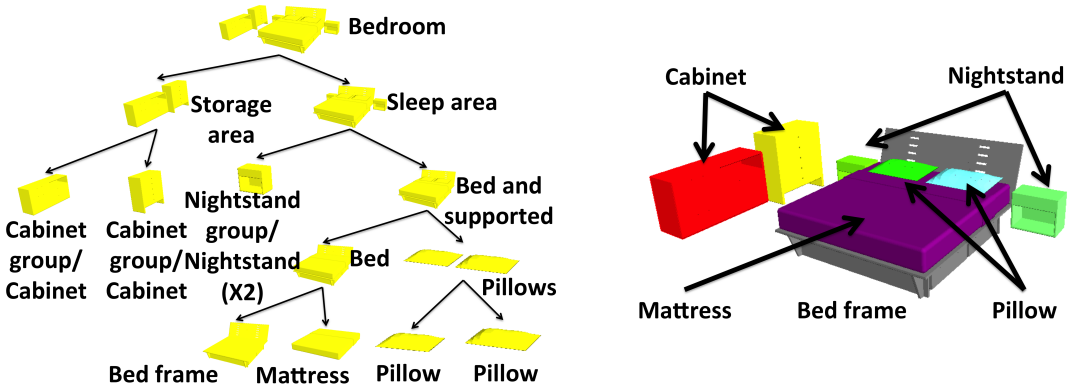
where $Q_u$ is the optimal energy of applying grammar rules with a single right-hand child ($l \to k$ in Equation 3.11), and $Q_b$ is the optimal energy of applying grammar

rules with two right-hand children (all other rules in Equation 3.11). $\mathbf{Part}(x)$ is the set of partitions of $X$ into two subsets from $\mathbf{C}$. $E_2$ is similar to $E$, but nodes and labels are specified in the argument list. $\mathbf{RHS}(l)$ is the set of right-hand-side labels derivable from $l$ in $\mathcal{G}_2$. $S(x,l)$ is the total energy of partition $\mathbf{K}(x,l)$. $\mathbf{K}(x,l)$ can be updated accordingly given the optimal $l', y, z, l_y, l_z$ for computing $Q(x,l)$.

Note that there are no guarantees that $Q(x,l)$ is the optimal energy of the subtree rooted at node $x$ and label $l$. If $\bar{\mathbf{K}}(\cdot,\cdot)$ represents the optimal $\mathbf{K}(\cdot,\cdot)$, and $\{[y_i, l_{y_i}], [z_i, l_{z_i}]\}$ represents all binary partitions of $\bar{\mathbf{K}}(x,l)$, $Q(x,l)$ is suboptimal when none of $y_i, z_i$ is in $\mathbf{Part}(x)$, or none of $\bar{\mathbf{K}}(y_i, l_{y_i}) \cup \bar{\mathbf{K}}(z_i, l_{z_i})$ constructs $\bar{\mathbf{K}}(x,l)$. Redundancy in grammar binarization (Equation 3.11) leads to a larger set of $\{[y_i, l_{y_i}], [z_i, l_{z_i}]\}$, which is likely to enable our algorithm to find a lower energy solution. As we will show in Section 3.6, we can always find solutions with reasonably low energies (i.e. equal to or lower than the ground-truth hierarchy) in our experiments.

Given the state transfer equations, the only remaining problem is to compute $Q(x,l)$ in the correct order. To ensure that the values on the right-hand-side of the binary term $Q_b(x,l)$ are available, we compute $Q(x,l)$ in the order of increasing cardinality of $X$. This ensures that $Q(y,l_y)$ and $Q(z,l_z)$ are computed before $Q_b(x,l)$. Among the states $(x,l)$ with the same $x$, we compute $Q(x,l)$ based on the topological order of label $l$ in $\mathcal{G}_2$, which ensures $Q(x,l')$ is available when computing $Q_u(x,l)$ if $l'$ is derivable from $l$.

Finally, we transform the optimal binary hierarchy $\arg\min \mathbf{E}(H)$ to a hierarchy in the original grammar $\mathcal{G}$ by removing all nodes with partial labels and attaching their children to their parents.




(a) Ground-truth hierarchy

(b) Ground-truth leaf nodes

Figure 3.4: Each test data set includes (a) a manually-created hierarchical grammar and (b) a set of scene graphs with manually-labeled nodes representing a "ground truth" parse of the scene.

## 3.6 Results

### 3.6.1 Datasets and evaluation methods

**Datasets:** We tested our algorithms on scene graphs representing three types of scenes downloaded from the Trimble 3D Warehouse: Bedroom (77 scenes), Classroom (30 scenes), and Library (8 scenes).

For two types of these scenes, we additionally created small datasets with simple scene graphs representing 17 bedrooms and 8 libraries, respectively. These scenes have only the basic objects commonly found in such scenes and thus serve as a "clean" dataset for testing the core elements of our algorithms independent of the noise found in real-world data sets.

For each scene graph in all five of these data sets, we enforced a canonical scaling (one unit equals one inch), removed polygons representing walls and floors, and removed scene graph nodes representing small parts of objects. While these steps could be performed automatically, we performed them manually for this experiment to avoid confounding our main results with errors due to preprocessing heuristics.

**Evaluation methods:** To evaluate the results of our scene parsing algorithm, we manually specified a hierarchical grammar for each type of scene (Figure 3.4(a)) and manually assigned a ground-truth parse for each input scene graph (Figure 3.4(b)). Then, we tested our parsing algorithms in a series of leave-one-out experiments. Specifically, for each scene, we trained a grammar on the other scenes of the same type, used that grammar to parse the leaf nodes of the left-out scene, and then measured how accurately the topology and labels of the predicted scene graph match those of the ground truth parse. Note that since the resulting scene graphs are all valid parses of the probabilistic grammar they have consistent hierarchical parent-child relations.

To measure the label consistency of a predicted parse with the ground truth, we used precision, recall, and $F_1$ score (F-measure) statistics. Since the interior nodes of the predicted scene graph can be different than those of the ground truth for the same scene, calculation of the standard form of those metrics is not possible. Instead, we computed measures that account for the fractions of surface area labeled correctly. For example, to compute precision for a particular label $l$, we computed the fraction of all surfaces in the subtrees rooted at nodes predicted to have label $l$ that appear in a subtree rooted at a node labeled $l$ in the ground truth. Our final results are averages weighted by surface area over all label types and all scenes.

### 3.6.2 Benefit of hierarchy

**Hierarchical parsing results.** In our first experiment, we evaluate how well the scene graphs predicted by our hierarchical parsing algorithm match the ground-truth data. Figure 3.5 shows the results: the height of each bar indicates the average $F_1$ score for a different dataset, where 1.0 is perfect and higher bars are better. On average, our method achieves almost 100% accuracy on small datasets, and 80% on the Trimble 3D Warehouse datasets. Example parsing results can be seen in Figure 3.6. These examples show that our algorithm is able to create functionally

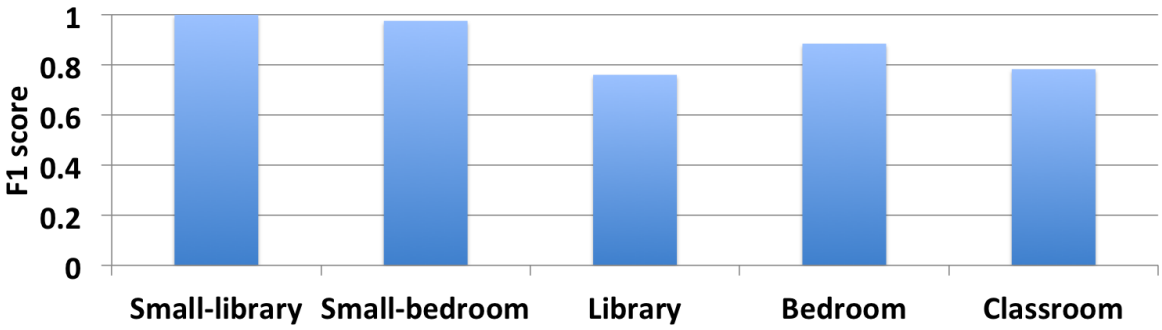



Figure 3.5: Performance of object grouping. Our method achieves almost 100% on illustrative datasets, and ∼80% on Trimble 3D Warehouse scenes.

relevant hierarchies for many different types of scenes, even though the input scene graphs have very little hierarchy, if any at all. For example, it correctly parses the sleep areas (bed, nightstand, etc.) and storage areas (closets, cabinets, etc.) in the bedroom scenes in the top two rows; and, it differentiates teacher areas from student desk areas in the classroom shown in the third row, even though the shapes of the individual objects (desk and chairs) are geometrically very similar. Incorrectly labeled nodes are highlighted in red – errors usually occur due to limited amounts of training data.

**Comparison to alternative methods.** In a second experiment, we test whether parsing scenes with our hierarchical grammar provides more accurate object labels than simpler alternatives. To test this hypothesis, we compare our results with the following two alternative methods:

- **Shape only.** This method selects the label that maximizes the geometry term $E_g$ for each input node. It is representative of previous methods that perform object classification based solely on similarities of shape descriptors.

- **Flat grammar.** This method executes our algorithm using a flattened grammar that has only one production rule that connects all terminals directly to the axiom. The geometric and spatial attributes of the flattened grammar are learned from ground-truth flattened graphs. Thus, this method is representa-

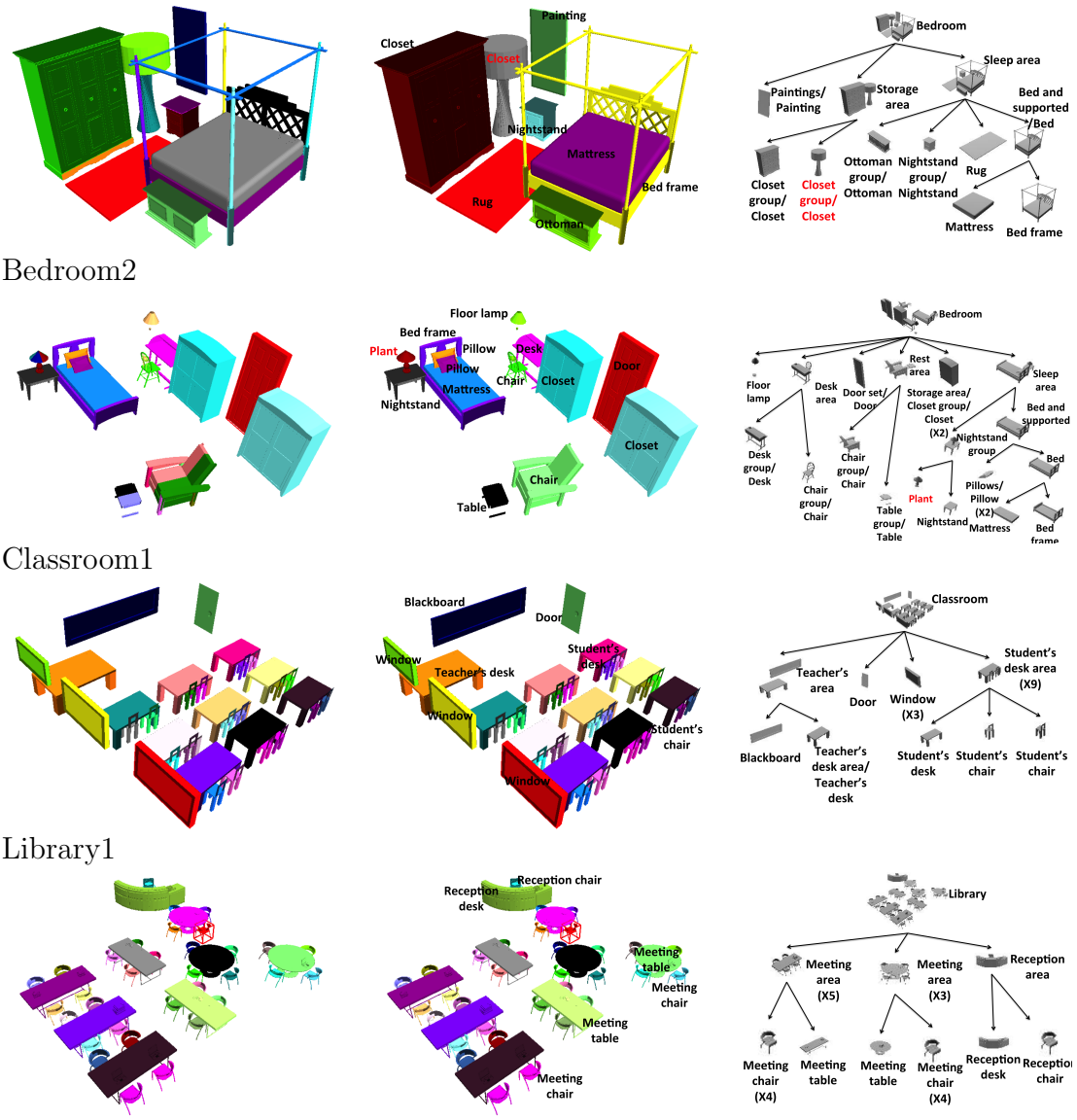



Figure 3.6: Examples of parsing results. We show the leaf nodes of the input scene graph (column 1), and the leaf nodes (column 2) and hierarchy (column 3) output by our algorithm. Red labels indicate either wrong labels or incorrect segmentation. In column 3, to save space, we merge a child with its parent if it is the only child, and use '/' to separate the labels of the child node and the parent node. Also to save space, we use 'X' to represent multiple occurrences of the same geometry in the parse tree (note that we do not detect identical geometries in our algorithm; this is only for visualization purposes). The input scenes of the top three examples are oversegmented.

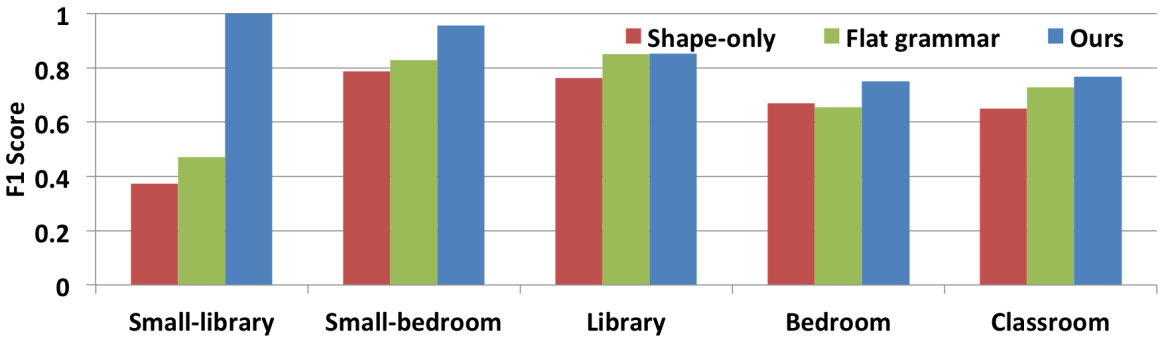



Figure 3.7: Performance of object classification. Using a hierarchical grammar clearly outperforms alternatives.

tive of previous methods that leverage spatial context, but not hierarchy, for object classification [28, 30, 29, 102].

Results are shown in Figure 3.7: each set of bars shows a comparison of our method (blue bar on right) with the two alternatives running on a given test dataset. Since the alternative methods predict labels only for objects (i.e., do not produce hierarchy), we compare their results only for labels predicted at leaf nodes by our algorithm.

From the results we see that methods based on parsing with our probabilistic grammar (green and blue) outperform a method based purely on matching shape descriptors. Moreover, we find that parsing with a hierarchical grammar (blue) is better than with a flat grammar (green). Figure 3.8 shows representative failures of alternative methods (highlighted with red labels). The method based only on matching shape descriptors fails when geometries of different object classes are similar (e.g., short book shelf vs. study desk in the library example; console table vs. study desk in the bedroom example). The method based on flattened grammars fails when spatial relationships between objects are context dependent (e.g., the relation between the study chair and the short book shelf is wrongly interpreted in the library example).

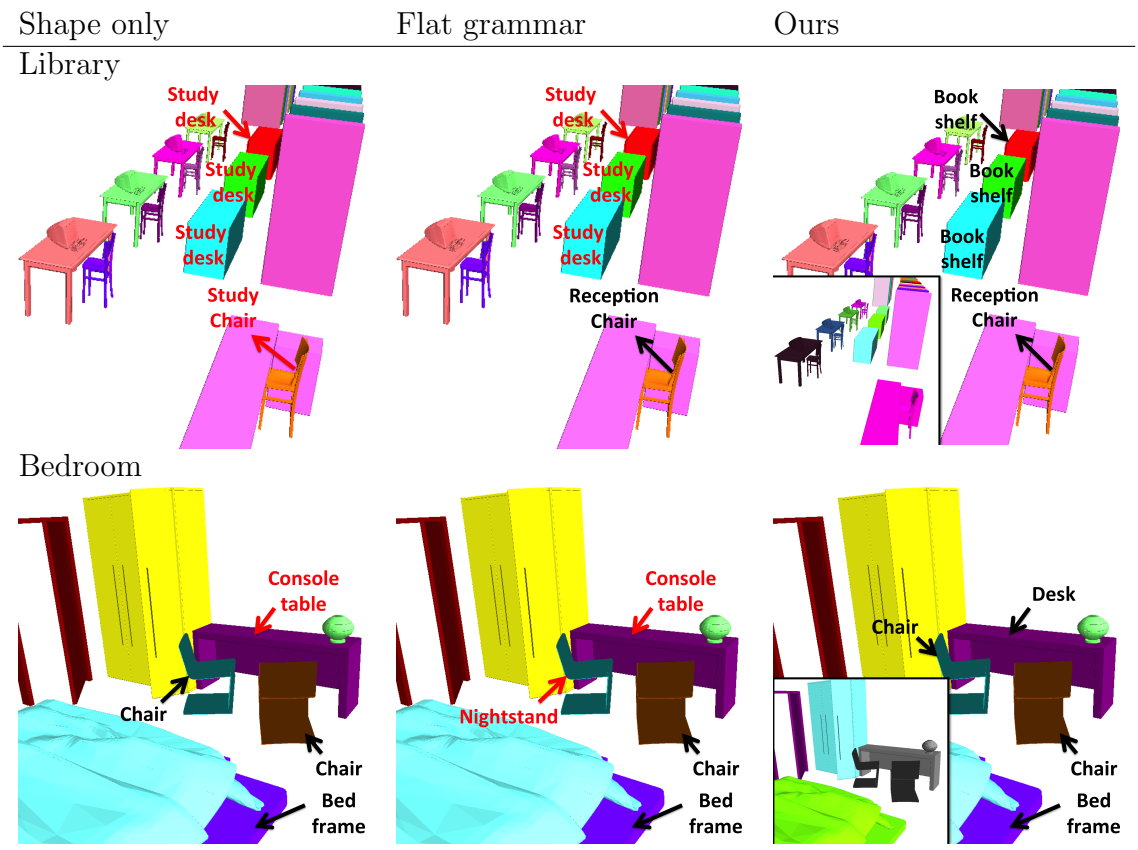



Figure 3.8: Comparison to alternative methods. Classifying objects only by their geometry (first column) cannot differentiate between objects of similar shape in different categories, e.g. short bookshelf and study desk, or console table and study desk. Even if contextual information is leveraged, relations among objects can be wrongly interpreted (e.g. short book shelf and study chair (second column top), chair and bed (second column bottom)) in the absence of a hierarchy of semantic contexts at various scales. Our method exploits such a hierarchy to yield more accurate object recognition. The inset images of the third column show the object groups predicted by our method. Black labels are correct, and red labels are incorrect.

### 3.6.3 Generalization of our approach

**Handling over-segmentation.** In a third experiment, we test whether our method is able to parse Bedroom scene graphs with moderate levels of over-segmentation. In this test, the leaves of the input scene graphs are not necessarily representative of basic category objects, but instead can represent parts of objects as determined by the leaf nodes of the scene graphs originally downloaded from the Trimble 3D Warehouse. We call this new data set "Bedroom-oversegmented."

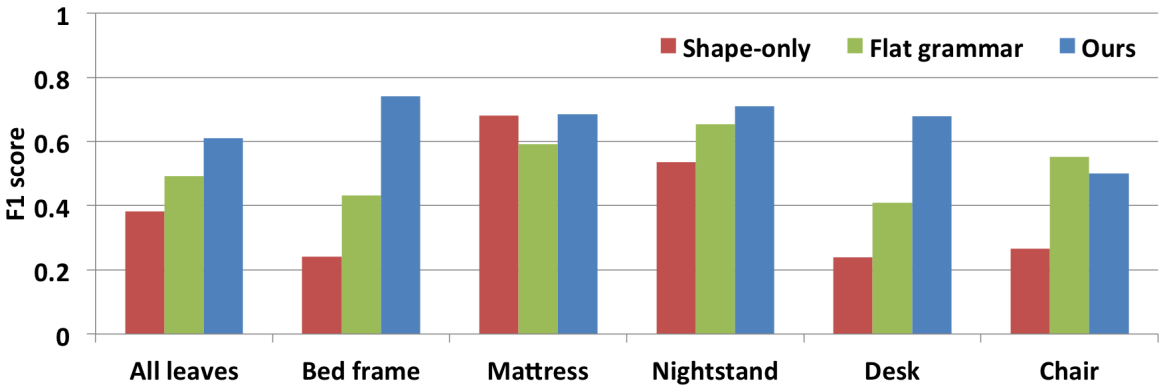



Figure 3.9: Performance on over-segmented bedroom scenes. Our method significantly outperforms shape-only classification in most object categories except mattresses, which are rarely over-segmented, and can be distinguished from other classes based on their distinctive geometry. Our method outperforms the "flat" grammar, with spatial relations but no hierarchy, in all object categories except for chairs.

This test is much more difficult than the previous one, because it requires the parsing algorithm to determine what level of each input scene graph represents the basic category objects in addition to assigning labels and creating a meaningful hierarchy.

We compare the methods described above with a few changes. In our method and the flat grammar method, the grammar is augmented with an extra layer of labels at the bottom of the hierarchy representing object parts (e.g., "part of a chair"). These new types of labels are necessary to allow the parser to find an appropriate "segmentation" of the scene by assigning them to over-segmented nodes of the input scene graphs while grouping them into new interior nodes with basic object category labels.

Results of this experiment are shown in Figure 3.9, with the overall results shown in the left set of three bars and results for individual object labels shown to the right.

Not surprisingly, the shape-only method (red bars) performs the worst. Since it does not parse the scene and therefore cannot create new nodes representing groups of leaf nodes, it is unable to correctly label any objects not represented explicitly by a node in the input scene graph. Also since it does not leverage spatial relationships when assigning labels, it is difficult for it to distinguish some object classes from

others with similar shapes. Our parsing method using a hierarchical grammar has better overall performance than using a flattened grammar. This is because it better captures the spatial and cardinality distributions specific to semantic groups of objects represented by interior nodes of the grammar. For example, without those cues, *bed frame* can be easily confused with *bed*, as they share similar geometries and spatial relationships.

**Parsing other datasets.** In a fourth experiment, we test whether our algorithm can learn a hiearchical grammar on one data set and then use it to parse a different data set. For this test, we downloaded the Sketch2Scene Bedroom dataset [102] and then parsed each of the Bedroom scene graphs using the grammar learned our Bedroom dataset. Since the Sketch2Scene dataset was constructed by retrieval using keywords, it includes scenes that are obviously not bedrooms, which were excluded from our experiments. Additionally, we excluded Sketch2Scene scenes that were very similar (or duplicate) with any in our dataset. In the end, we were left with 90 scenes for testing.

We ran our parsing algorithm (and the two alternative methods) trained on our Bedroom set to predict a scene graph hierarchy for each of the 90 scenes in the Sketch2Scene bedroom dataset *without any change to the algorithm or parameters* – i.e., the algorithm was frozen and parameters learned before even looking at the Sketch2Scene data for the first time.

To evaluate the results, we use the manually-specified ground truth labels for all basic object category objects provided with the Sketch2Scene dataset. Since the Sketch2Scene data has no hierarchy, we evaluate our results only for leaf nodes. Since the Sketch2Scene ground-truth label set is different from ours, we created a mapping from our label set to theirs so that labels predicted by our parser could be compared to their ground truth. Unfortunately, the Sketch2Scene label set is coarser-grained than ours, often not separating functionally different objects with similar shapes

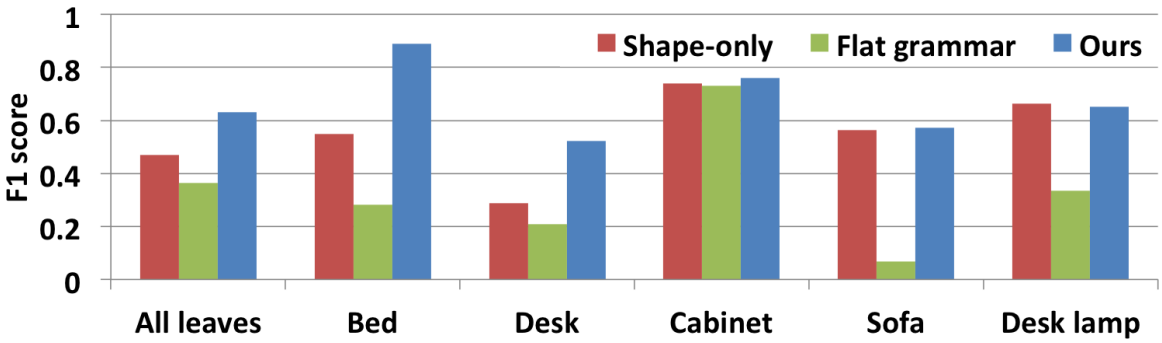



Figure 3.10: Parsing scenes in the Sketch2Scene dataset [Xu et al. 2010]. We reuse the grammar learned in Section 3.6.2 to parse scenes in Sketch2Scene, and compare the performance to those of alternative methods. Using a flattened grammar is not effective because spatial relations are not discriminatory enough without meaningful object groups. Shape-only classification performs comparably to our method in object categories where geometry is distinctive, but is surpassed by our method when contextual information is important for disambiguation (e.g. desk and bed).

(e.g., *nightstand*, *cabinet*, and *closet*) are all mapped to one label called *cabinet* in the Sketch2Scene. This reduction of ground-truth labeling granularity and the lack of hierarchy in the ground truth hides key differences in the evaluation of our results, but we use it none-the-less since it provides an objective evaluation of our method with respect to a third-party data set.

As in the previous experiments, we compare the performance of our hierarchical parsing algorithm to the shape-only and flat-grammar methods. Results are shown in Figure 3.10. Note how the results for this new data set are similar to the ones previously reported for the leave-one-out experiment. Hierarchical parsing provides the best average results overall ("All leaves") and significant improvements for most object labels (e.g., desk). This result verifies the robustness of the algorithm to handle different input scene graphs.

Interestingly, the flat grammar method performs worse than shape-only for several object categories. This is because spatial attributes learned for pairs of objects within a scene are mixed in the flat grammar (e.g., the spacing between desks and chairs is learned from all pairs across the entire room rather than just the pairs within the

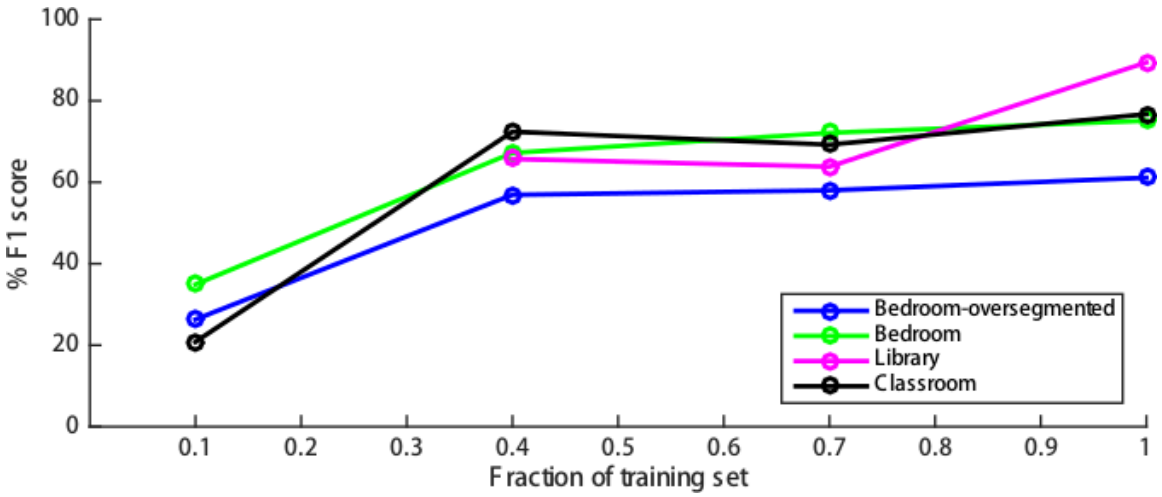



Figure 3.11: Impact of size of training set. Labeling accuracy increases on all datasets with more training examples.

same study area). By leveraging hierarchy we can learn relations between objects that belong to the same group, and thus learn stronger layout priors.

### 3.6.4 Sensitivity analysis

**Impact of training set size.** We tested how the performance of our algorithm is affected by the size of the training set. For each scene graph, we trained a grammar on $X\%$ of the other scenes selected randomly (for $X = 10\%, 40\%, 70\%$, and $100\%$), used that grammar to parse the scene, and then evaluated the results. Figure 3.11 shows that the results. From this test, it seems that training on approximately 40% of the scenes provides results approximately as good as training on 100% in all datasets except for Library, which has only 8 scenes in total.

**Impact of individual energy terms.** We ran experiments to show the impact of each energy term on the final results by disabling each one and re-running the first experiments. The results of this experiment (Figure 3.12) suggest that the performance becomes worse if we disable any of the energy terms. Interestingly, terms have different impact on different datasets. For instance, the geometry term is more

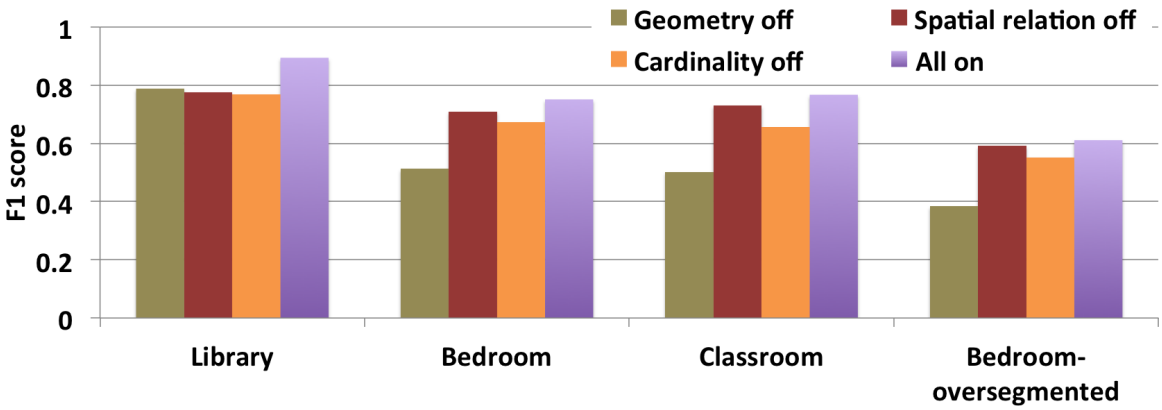



Figure 3.12: Impact of individual energy terms on object classification. Each energy term contributes to the overall performance in each dataset.

important in bedrooms, while the spatial and cardinality terms are more important in libraries, probably because hierarchical structure is more prominent there.

**Impact of optimization approximations.** We next ran experiments to evaluate the impact of approximations made by our parsing algorithm to narrow the search space, i.e., proposing candidate groupings based on spatial proximity and binarizing the grammar.

To evaluate the approximations, we compare the output of our algorithm to the output of exhaustive search. Because the computation complexity of exhaustive search is exponential in the size of input, we do this comparison only for the small dataset of bedrooms, where each scene contains no more than 10 nodes. Our experiment result shows that our approximations are able to get the globally optimal solutions in 16 out of the 17 cases. In the only failure case, the candidate node selection algorithm misses one internal node in the ground truth. On average, exhaustive search takes 35 minutes for the scene with 10 leaf nodes, while our method takes only 3 seconds.

We also evaluate the impact of our approximations on parsing the Trimble 3D Warehouse scenes. Since it is impractical to get the globally optimal solution for

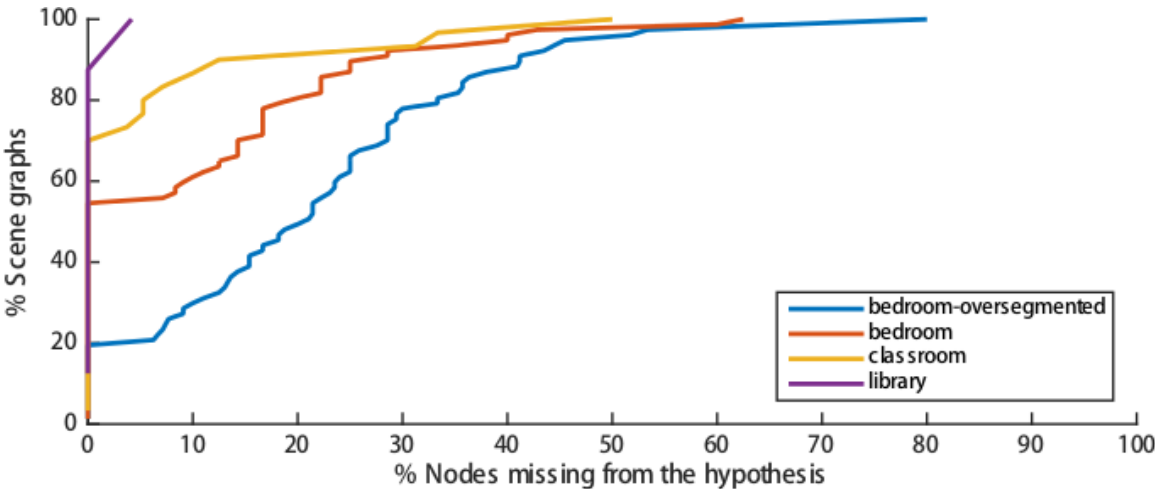



Figure 3.13: Fraction of ground truth internal nodes missing from the predicted hierarchies. The Y-value of each point on a curve denotes the fraction of scenes in which the number of missing ground truth internal nodes is at most the X-value. For libraries, our algorithm successfully proposes all ground-truth nodes, except one, in the entire dataset. Oversegmented input scene graphs are in general more challenging for our method.

these scenes, we study the impact of our approximations only with statistics gathered during the search.

First, to evaluate the impact of selecting candidate nodes based on spatial proximity, we measure the fractions of internal ground truth nodes that are not considered as candidate nodes by our algorithm (Figure 3.13). The results show that the approximation misses very few nodes for cases where the input scene graph is well-segmented at the leaf nodes, but provides mixed results when the input is over-segmented.

Second, to evaluate the impact of grammar binarization, we investigate how often our algorithm outputs a hierarchy with higher energy than the ground-truth hierarchy. If we consider only the examples where the ground-truth solution is included in our search space (85% of scenes), then there is only one case where our method produces a solution with higher energy than the ground-truth, which indicates that grammar binarization is not significantly affecting the accuracy of our final results.

**Timing results.** We measured the computational complexity of our parsing algorithm on the Bedroom data set. Figure 3.14 shows the result relating the number

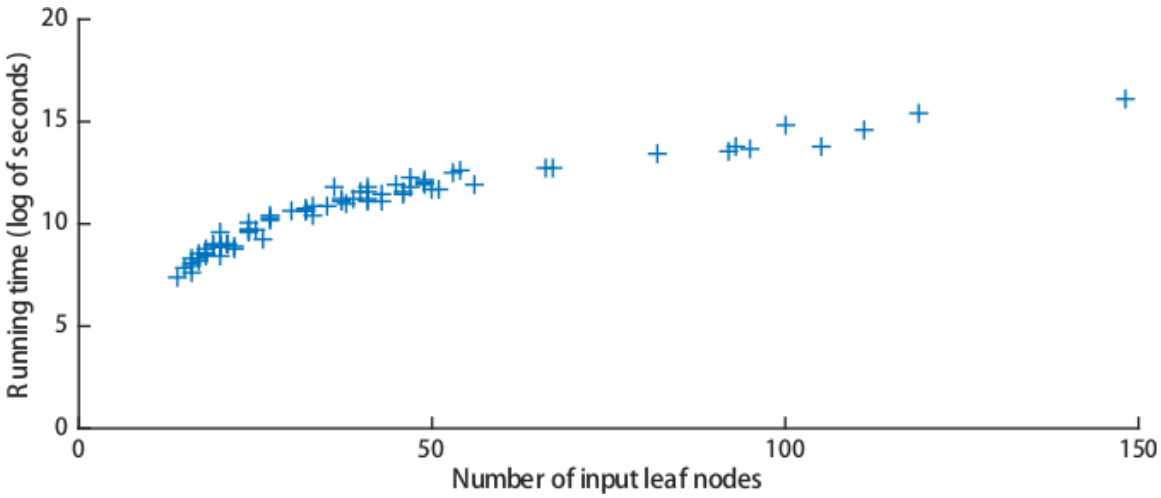



Figure 3.14: Relationship between number of input leaf nodes and running time on oversegmented bedroom scene graphs. Our method scales reasonably well for complex scenes.

of input leaf nodes and the running time. Our algorithm is far from real-time, but scales well for scenes with large numbers of input leaf nodes.

## 3.7 Discussion

This chapter presents a method for parsing scene graphs using a hierarchical probabilistic grammar. Besides this main idea, we offer two technical contributions. First, we formulate a probabilistic grammar that characterizes geometric properties and spatial relationship in a hierarchical manner. Second, we propose a novel scene parsing algorithm based on dynamic programming that can efficiently update labels and hierarchies of scene graphs based on our probabilistic grammar. Experimental results show that: i) the hierarchy encoded in the grammar is useful for parsing scenes representing rooms of a house; ii) our algorithms can be used to simultaneously segment and label over-segmented scenes; and iii) the grammar learned from one data set can be used to parse scene graphs from a different data set (e.g., Sketch2Scene). To the best of our knowledge, this is the first time that a hierarchical grammar has been used to parse scene graphs containing 3D polygonal models of interior scenes. So,

the highest-level contribution of this chapter is demonstrating that this approach is feasible.

Our method is an early investigation and thus has several limitations that suggest topics for future work. First, our current grammar does not capture the correlations between co-occurrences of sibling labels. For instance, couch and chair are interchangeable in a rest area, so the occurrences of them are highly related. It would be interesting to augment the grammar with higher-order relationships, which might be leveraged to improve prediction accuracy. Second, our algorithm learns the probabilistic grammar from labeled examples, which may not always be available. It would be nice to develop methods to detect repeated shapes and patterns in scenes and use them to derive grammars automatically, although it would be hard to guarantee the semantic relevance of such grammars. Finally, we focus mainly on methods for representing and parsing scenes with a grammar. Although there are several obvious applications for these methods in computer graphics, including scene database exploration, scene synthesis, semantic labeling of virtual worlds, etc., it would be nice to explore applications in computer vision, robotics, and other fields.

# Chapter 4

# Composition-Aware Scene Optimization for Product Images

## 4.1 Introduction

Many applications of 3D scene modeling have emerged in the last decade. One major category of the applications is to provide users with an immersive experience in the synthesized 3D scenes, for example, modeling 3D virtual scenes for building virtual social worlds and massively multiplayer online games [8]. Rather than taking the synthesized 3D scenes as the output of the scene modeling process, another growth application of scene modeling is creation of images for product advertisements and catalogs [27, 85, 87]. As photorealistic rendering algorithms have improved, it has become practical to synthesize images that are indistinguishable from photographs for many types of scenes commonly found in product advertisements (e.g., kitchens, bathrooms, living rooms, etc.). As a result, several furniture and home goods companies are beginning to create product images for their catalogs by synthesizing and rendering 3D models rather than photographing physical objects [85]. For example,

IKEA has reported that 75% of scenes shown in its most recent catalog were rendered from 3D models [87].

There are many advantages to creating catalog images from 3D models [27]. Rendering virtual scenes is much less expensive than photographing real scenes because it does not require building physical sets in large photo studios, storing physical objects in large warehouses, and scheduling actors, stylists, and photographers to meet for photo shoots. Moreover, digital assets make it easier to customize images in a variety of ways, such as producing multiple images of the same scene with different objects of interest, adapting scene composition to the resolution and aspect ratio of the display device, adapting the size and placement of text labels for different languages.

Despite these advantages, producing good product images from 3D models is still difficult. Based on interviews with professionals who work on product catalogs, we learned that the typical workflow for a *stylist* (person who designs scenes for product images) is to first create an approximate scene layout with the appropriate objects (furniture, accessories, etc.) in roughly the desired configuration (e.g., a sofa with an end table on the left, a lounge chair on the right, and a coffee table with some plants and coasters sitting on top). Based on this rough layout, the stylist will then carefully refine the positions, orientations and materials of objects as well as the camera viewpoint to compose several different images of the scene that highlight different objects of interest, fit different display devices (e.g., iPhone, desktop, print, etc.), and in some cases, target different cultures (e.g., IKEA makes 62 variants of its catalog for 43 countries). This refinement step is challenging because it requires taking into account the image-space position, size, visibility, and color contrast of objects of interest, as well as the overall composition of each image. As a result, stylists often spend multiple days refining the initial rough scene layouts to produce all the necessary product images.

In this chapter, we focus on the problem of scene modeling for creation of product images. We present a tool that facilitates the creation of product images by automatically refining rough scene layouts to produce good compositions. Our tool starts with an approximate scene description provided by a stylist that includes which objects should appear in the scene, which objects rest upon which other objects, and which materials can be used for which objects, plus an initial configuration for object positions/orientations, surface materials, lighting parameters, and (optionally) camera views. Our tool then optimizes the camera view, object transformations and surface materials to meet user-specified design goals (e.g., highlight these objects of interest, fit the image within a specific form factor, leave space for a text box) while maintaining the compositional quality of the image.

The main contribution of our work is an optimization approach that takes into account a large variety of design constraints and image composition rules that are important for producing effective product images. Since the plausibility and the quality of image compositions is determined by scene layouts in both the 2D image space and the 3D scene space, we define a (highly non-convex) energy function that takes into account object positions and relationships between objects in both spaces. We then introduce an iterative optimization procedure that minimizes the energy. One key feature of our method is that we simultaneously manipulate object transformations, surface materials, and the camera view, all of which have a significant impact on the quality of the resulting image. We show that our approach produces better results than a more traditional camera-only optimization, and we also demonstrate how our tool can help create product images for a variety of practical applications.

## 4.2 Related Work

Our work draws upon previous work in image composition and aesthetics, image analysis and optimization, virtual camera control, and automatic scene synthesis.

### 4.2.1 Image composition and aesthetics

Our work is inspired by composition "rules" that have been established to guide photographers and graphics designers towards better scene compositions and aesthetics [2, 10, 23, 38, 55, 68, 90]. Well-known examples include the "rule of thirds," visual balance, diagonal dominance, and color contrast. Although previous work has considered subsets of these rules for automatic image composition, one of our key contributions is in determining a set of rules suitable for product images and applying these rules to a challenging 3D scene optimization problem. Compared to existing methods, our approach considers a larger set of 3D scene parameters — object transformations, surface materials, and the camera view — that are important for generating effective product images.

### 4.2.2 Image analysis and optimization

Several papers have used these rules to quantify [24, 25] and enhance the compositional and aesthetic quality of images. However, all previous methods operate only on edits to 2D images: for example, rotating and cropping images [50, 60, 47], or adjusting locations of foreground regions [11, 64] – they do not optimize 3D scene parameters, such as cameras, materials, and/or object transformations, as our system does. Our work is also related to the technique that allows the user to alter the composition after 3D rendering [39], but our system is able to automatically optimize the composition by refining 3D scene layouts and materials.

### 4.2.3 Camera optimization

Several methods have incorporated principles of image aesthetics and composition in optimization algorithms for camera control in 3D rendering systems [76, 36, 22, 6, 5]. While these papers provide motivation for our work, they consider only camera control – we additionally optimize object transformations and surface materials, which can significantly improve image compositions, but require solving a more difficult optimization problem.

### 4.2.4 Scene optimization

Several recent papers have proposed methods for automatically placing objects in scenes to produce plausible furniture layouts based on examples and design guidelines [104, 29, 71]. These methods focus on scene plausibility without concern for any particular camera viewpoint and/or image composition principles. As a result, they produce scenes that may not support generation of aesthetic images from any camera viewpoint. Our work is also related to arrangement synthesis based on aesthetic relations [67]. Instead of optimizing relations among objects, our goal is to optimize the 2D composition produced by a 3D scene layout.

We believe that ours is the first system to optimize aesthetics and composition of rendered images with simultaneous control over camera parameters, object transformations, and surface materials. We investigate this optimization problem for the novel application of image synthesis for product catalogs.

## 4.3 Overview

The core of our work is a method for optimizing 2D compositions of rendered 3D scenes by adjusting camera parameters, object transformations, and surface materials.

Our system works with a database of 3D object models, where each model is annotated with an object class (e.g. chair, table) and a list of possible materials. Each object class is associated with a set of 1-4 canonical views, which can be manually specified or predicted automatically [13, 36, 81]. Some object classes are also associated with a set of semantic spatial constraints. For examples, picture frames on walls should not be rotated, and the distance between a dining chair and a dining table should be maintained.

Given the model database, a stylist begins the process of creating a set of product images by specifying a *scene configuration*, which consists of a set of object models, light sources, and a rough location/orientation for every object. With the rough scene in place, the stylist can instruct our system to create a specific image by specifying an *image configuration*, which consists of desired *aspect ratio* and *focus objects* $\mathbf{O_F}$ to highlight. The stylist may optionlly specify initial camera parameters and *context objects* $\mathbf{O_C}$ that should remain visible for context.

From this input, our system optimizes the scene description to generate a set of rendered images. In particular, our method optimizes the following scene parameters (plus other application-specific variables described in Section 4.6), with the degrees of freedom listed in parentheses:

- ***Camera (6):*** position (3), direction (2), and field of view (1) (camera roll is constrained to be zero).
- ***Object transformations (3 per object):*** position of the object centroid on its support surface (2) and rotation of the object around the normal of its support surface (1).
- ***Materials (1 per object):*** choice of a material/texture definition amongst a list of possible candidates.

Our system optimizes these parameters according to an energy function that accounts for image composition, aesthetic principles, and object focus, while maintaining 3D spatial constraints and 2D image space layout constraints.

The following sections describe our energy function and optimization procedure in detail.

## 4.4 Energy Function

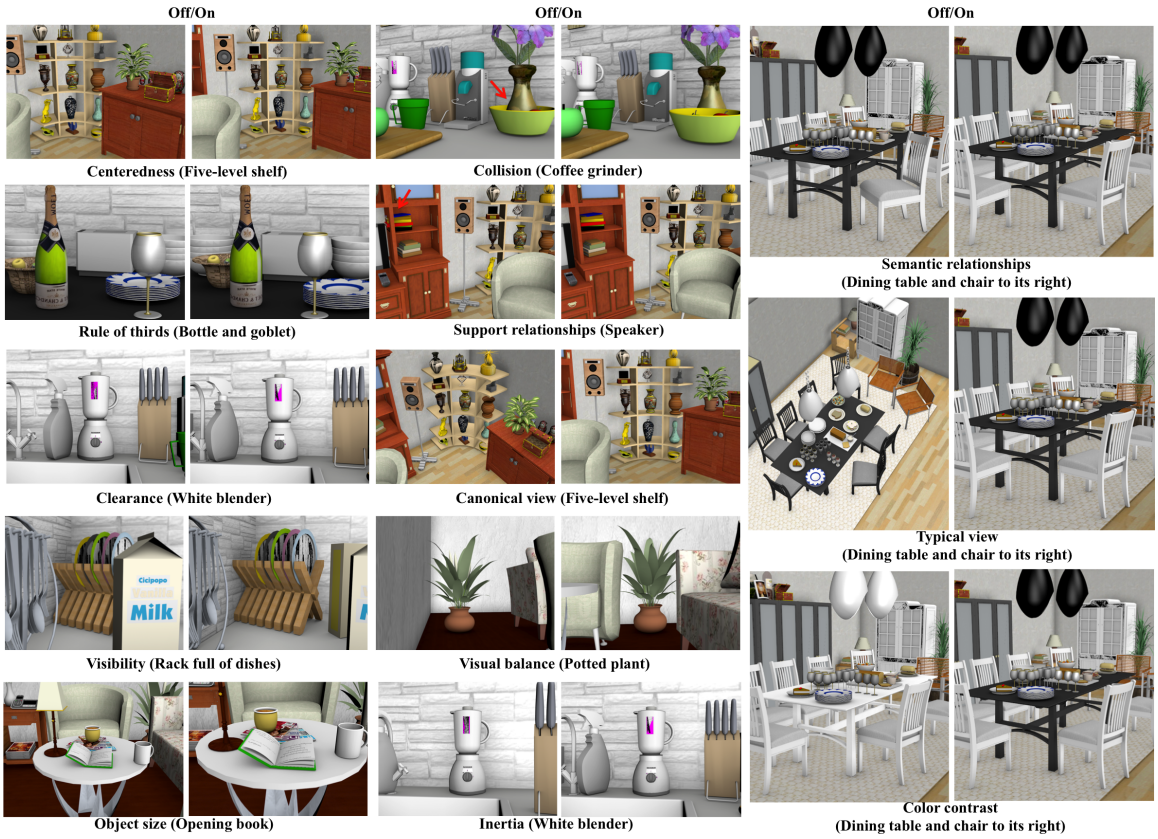



Figure 4.1: Effects of disabling energy function terms. For each energy term, we compare the result with the term disabled (left) to our result (right). The focus object(s) is specified in the parentheses.

Our energy function estimates how effectively an image advertises the product(s) it depicts.

| | |
|---|---|
| $F$ | The 2D image frame (viewport) |
| $\mathbf{O}$ | The set of all objects |
| $\mathbf{O_F}/\mathbf{O_C}$ | Focus/context objects |
| $O_i$ | An object in set $\mathbf{O}$ |
| $P_i$ | $O_i$'s projection into image space |
| $V_i$ | the part of $P_i$ visible to the camera |
| $\mathcal{V}(\cdot)$ | Volume in scene space |
| $\mathcal{A}(\cdot)$ | Area in image space |
| $\mathcal{B}(\cdot)$ | Boundary contour in image space |
| $\mathcal{F}(\cdot)$ | Projection onto XY plane in scene space |
| $\mathcal{R}(\cdot)$ | Diagonal radius |
| $\mathcal{C}_2(\cdot)/\mathcal{C}_3(\cdot)$ | Centroid in screen/scene space |
| $d_2(\cdot,\cdot)/d_3(\cdot,\cdot)$ | Euclidean distance in screen/scene space |
| $c(\cdot,\cdot)$ | Color difference in L*ab space |

Table 4.1: Symbols used in the energy function definition.

To design the energy function, we interviewed professionals responsible for creating scenes for popular product catalogs, worked with them to identify a set of principles important for product image composition, and encoded how well those principles are satisfied into a mathematical error function. We interviewed two professionals: 1) a professional stylist who lays out scenes for Pottery Barn catalogs, and 2) the lead of image synthesis for IKEA catalogs. They identified twelve different factors as critical for composition of product images. The following lists these factors, grouped roughly into six categories, along with an explanation of why each is important (Figure 4.1).

Our energy function is the summation of the value of each energy term.

$$\begin{aligned} E = {} & E_{rt} + E_{ce} + E_{cl} + E_{sa} + \\ & E_{sr} + E_{co} + E_{su} + E_{cv} + \\ & E_{tv} + E_{vb} + E_{cc} + E_{ir} \end{aligned} \tag{4.1}$$

Next, we will provide the definition of each energy term. Symbols used in the definition can be found in Table 4.1. In the rest of this section, we use $w$ to represent the weight of the term with the same subscripts.

- **Object placement within the 2D frame.** For product advertisements, some of the most significant factors affecting image quality are the positions of focus objects, which we encode with the following terms.

  - **Rule of thirds.** In general, focus objects should align with vertical or horizontal lines that divide the viewport into thirds and/or be centered at the intersections formed by them [4, 5, 17, 24, 36, 60, 96].

$$E_{rt} = \frac{w_{rt}}{\mathcal{R}(F)^2} \sum_{O_i \in \mathbf{O_F}} \left( \frac{w^2}{h^2} d_h(\mathcal{C}_2(P_i))^2 + \frac{h^2}{w^2} d_v(\mathcal{C}_2(P_i)) \right)^2$$

    where $w$ and $h$ are the width and height of the bounding box of $P_i$, respectively, and $d_h$ and $d_v$ stand for minimum distance to the closest horizontal and vertical third lines respectively.

  - **Centeredness.** For some product images (e.g., zoomed views of single objects), focus objects should appear in the center of the image [2].

$$E_{ce} = \frac{w_{ce}}{\mathcal{R}(F)^2} \sum_{O_i \in \mathbf{O_F}} d_2\left(\mathcal{C}_2(F), \mathcal{C}_2(P_i)\right)^2$$

  - **Clearance.** Since other objects should not compete with the focus objects in the composition, we introduce a term to penalize objects that are close to focus objects.

    For an object $O_i$, we consider its signed distance to the bounding circle of a focus object in image space (a negative distance means $O_i$ is inside the bounding circle), and normalize the distance by the radius of the bounding circle in order to avoid favoring small objects. We use $r(O_i)$ to denote the minimum of all the distances,

$$r(O_i) = \min_{O_j \in \mathbf{O_F}} \frac{d_2(\mathcal{C}_2(P_i), \mathcal{C}_2(P_j)) - \mathcal{R}(P_j)}{\mathcal{R}(P_j)}$$

The clearance term is then defined as,

$$E_{cl} = \frac{w_{cl}}{N} \sum_{O_i \in \mathbf{O_F}} \left( e^{-\max\{0, r(O_i)\}^2} + \max\{0, -r(O_i)\}^2 \right)$$

where $N = |\mathbf{O} \setminus \mathbf{O_F}|$. The $\max\{0, -r(O_i)\}^2$ term handles the case where $O_i$ is inside a focus object's bounding circle.

- **Object saliency within the 2D frame.** The visibility and image-space size of objects contribute to their perceived importance [6, 5, 7, 76].

  - **Visibility.** An object is perceived as less important if it is partially occluded by another object or partially clipped by the frame. Thus, focus objects should be more visible than context objects. To quantify the effect of visibility on saliency, we use the following term:

$$\mathscr{V}_r(O_i) = \max\left\{0, r - \frac{\mathcal{A}(V_i)}{\mathcal{A}(P_i)}\right\}^2 + \mathscr{D}(O_i)$$

  where $A(P_i)$ is the total area of the object projection $P_i$ assuming no occlusions by other objects or clipping by the image frame. $A(V_i)$ and $A(P_i)$ can be computed efficiently using hardware occlusion queries [37]. We introduce the term $\mathscr{D}(O_i)$ to encourage objects outside the frame to move towards the center of the frame:

$$\mathscr{D}(O_i) = \begin{cases} d_2(\mathcal{C}_2(F), \mathcal{C}_2(P_i))^2, & \text{if } \mathcal{A}(V_i) = 0 \\ 0, & \text{else} \end{cases}$$

  - **Object size.** Similarly, focus objects should take up more image space than context objects, since larger objects are perceived as more important. To quantify the effect of object size on saliency, we introduce the following

term:

$$\mathscr{S}_r(O_i) = \max\left\{0, r - \frac{\mathcal{A}(V_i)}{\mathcal{A}(F)}\right\}^2$$

where $A(V_i)$ is the area of the visible part $V_i$ of $O_i$, $r$ is the minimum required size of $O_i$. We describe how we choose $r$ later.

Visibility and object size are both essential for focus objects. However, for context objects, we observe that there are two scenarios. If the context object is small compared to the focus object, visibility is important while its absolute size in the viewport is not (e.g. items on the dining table in Figure 4.3 left). On the other hand, if the context object is largely occluded, it must maintain some minimum size in the composition. To handle these cases, we compute both energies $\mathscr{V}_r$ and $\mathscr{S}_r$ for each context object and select the minimum. Thus, the complete form of our object saliency energy term is

$$E_{sa} = w_{sf} \sum_{O_i \in \mathbf{O_F}} \left(\mathscr{V}_{v_f}(O_i) + \mathscr{S}_{s_f}(O_i)\right) + w_{sc} \sum_{O_i \in \mathbf{O_C}} \min\left\{\mathscr{V}_{v_c}(O_i), \mathscr{S}_{s_c}(O_i)\right\}$$

There are four parameters that reflect parameter $r$ in the visibility term $\mathscr{V}_r$. In all experiments, we set $v_f = 100\%, s_f = 10\%, v_c = 80\%, s_c = 5\%$. which means a focus object is required to be fully visible *and* cover 10% of the viewport; a context object is required to be at least 80% visible *or* cover at least 5% of the viewport.

- **Object constraints within the 3D scene.** Scene plausibility and physical laws both impose constraints on object positions. We therefore introduce several terms to enforce these constraints.

– **Semantic constraints.** Semantic spatial constraints that are specified in the model database should be satisified during the optimization.

We consider two types of constraints. The first type is defined for a single object class, which constrains the object to only transform in a subspace in 3D, e.g. a picture frame cannot rotate. We implemented it as hard constraints – i.e. the degrees of freedom are reduced in the optimization. The second type is defined for a pair of object classes, which constrains the change of the pairwise distance between two objects, e.g. the distance between a dining table and a dining chair cannot change drastically in the optimization. We implemented it as soft constraints using the method similar to [16, 71]:

$$E_{sr} = w_{sr} \sum_{\{O_i, O_j\} \in \mathbf{C}} \sigma_{i,j} d_3(\mathcal{C}_3(O_i), T_i^{-1}(\mathcal{C}_3(O_j)))^2$$

where $\mathbf{C}$ is a set of constrained object pairs, and $T_i^{-1}$ is the initial transformation from the scene space into the local coordinate frame of object $O_i$, and $\sigma_{i,j}$ controls how much the spatial relationship can change. $\sigma_{i,j}$ can be set by the stylist empirically in practice, and we set $\sigma_{i,j} = 1$ by default in all of the results shown in our work.

– **Collision relationships.** Object inter-penetrations should be avoided to improve the physical plausibility of the scene.

$$E_{co} = w_{co} \sum_{\substack{O_i \in \mathbf{O} \\ \mathcal{A}(V_i) > 0}} \sum_{\substack{O_j \in \mathbf{O} \\ \mathcal{A}(V_j) > 0}} \frac{\mathcal{V}(O_i \cap O_j)}{\mathcal{V}(O_i)}$$

– **Support relationships.** The support relationships that exist in the input scene layout must be maintained during the optimization. We infer the support relationships in the input scene layout by using the method similar

to [28]. Then we penalize placement of an object off its support object by measuring the fraction of its projected area outside its support surface [29]:

$$E_{su} = w_{su} \sum_{O_i \in \mathbf{O}} \left(1 - \frac{\mathcal{A}(\mathcal{F}(O_i) \cap \mathcal{F}(S_i))}{\mathcal{A}(\mathcal{F}(O_i))}\right)^2$$

where $S_i$ is the object supporting object $O_i$.

- **Camera placement.** Product images generally depict scenes from viewpoints that are "natural" for people. We introduce two terms that capture the notion of natural viewpoints.

  - **Canonical views.** In most product images with one focus object, stylists favor canonical views of the object class [13, 36, 81]. We manually defined a set of 1-4 canonical view directions for each object *class* [13, 36], and then deviations from them are measured as:

$$E_{cv} = w_{cv} \min_i ||(\theta, \phi) - (\hat{\theta}_i, \hat{\phi}_i)||$$

  where $(\theta, \phi)$ is the view direction of the camera, and $\{(\hat{\theta}_i, \hat{\phi}_i)\}$ are canonical view directions for the object class.

  - **Typical views.** Product images that depict large scenes with multiple focus objects often use camera viewpoints that match how a human would typically see the scene.

$$E_{tv} = w_{ch}(h - h_0)^2 + w_{ca}\phi^2$$

  where $h$ is the height of camera off the floor, $h_0 = 5ft$ is the typical height of a human eye, and $\phi$ is the pitch of the camera (where 0 is horizontal). This term penalizes viewpoints that deviate from a typical human eye height and tilt the camera upwards/downwards.

- **Image composition.** Several well-established composition guidelines are used by stylists to create aesthetically pleasing images. We have included several in our system.

  - **Visual balance.** Images whose "center of mass" is close to the center of the image frame generally have better aesthetics [2, 60, 64].

$$E_{vb} = \frac{w_{vb}}{\mathcal{R}(F)^2} d_2 \left( \mathcal{C}_2(F), \frac{\sum \mathcal{C}_2(P_i)\mathcal{A}(P_i)}{\sum \mathcal{A}(P_i)} \right)^2$$

    We penalize by the distance between the frame centroid and the center of mass of all objects in the frame.

  - **Color contrast.** Greater color contrast at object contours can help a viewer understand boundaries between shapes in a scene [54, 98].

$$E_{cc} = \frac{w_{cc}}{\mathcal{R}(F)^2} \sum_{O_i \in \mathbf{O_F}} \sum_{p \in \mathcal{B}(V_i)} \frac{1}{\left( \text{avg}_{q \in N(p) \setminus V_i} c(p,q) \right)^2 + \epsilon}$$

  where $N(p) \setminus V_i$ denotes the neighborhood of pixel $p$, excluding the visible pixels in $V_i$.

- **Regularization.** Finally, we add a regularization term that encourages small changes to the scene with respect to the initial configuration provided by the stylist.

$$E_{ir} = w_{ir} \sum_{O_i \in \mathbf{O}} \left( \frac{x_i^2}{\sigma_t^2} + \frac{y_i^2}{\sigma_t^2} + \frac{\theta_i^2}{\sigma_r^2} \right) + \sum_{i=0}^{5} \frac{c_i^2}{\sigma_c[i]^2}$$

  where $(x_i, y_i, \theta_i)$ describe the translation and rotation of object $O_i$, respectively, and $c_i$ describe the change to camera parameters, with $\sigma_t = 0.5, \sigma_r = 0.5, \sigma_c = [0.17, 0.17, 20, 20, 20, 0.17]$ controlling the flexibility of object movement and camera manipulation.

These energy terms are weighted by coefficients that adjust for scale differences and control their effects on the final results. By default, the weights are set to $w_{rt}$ = 10000, $w_{ce}$ = 10000, $w_{cl}$ = 500, $w_{sf}$ = 10000, $w_{sc}$ = 500, $w_{sr}$ = 100, $w_{co}$ = 10000, $w_{su}$ = 10000, $w_{cv}$ = 10000, $w_{ch}$ = 10000, $w_{ca}$ = 10000, $w_{vb}$ = 20000, $w_{cc}$ = 1.0, and $w_{ir}$ = 1.0. These weightings were determined empirically and are kept the same for all examples in our work, except that $w_{ce} = w_{cv} = 0$ for overview images of scenes (e.g., session A in Figure 4.2, Figure 4.5, 4.7 and 4.8), and $w_{rt} = w_{ch} = w_{ca} = 0$ for zoomed-in images of specific objects (e.g., session B in Figure 4.2, Figure 4.4 and 4.6). *It is not expected that a user has to tweak these weights to get good results for specific scenes.*

The large number of terms in this energy function reflects the inherent complexity of composing a good product image. The necessity of each and every term has been confirmed by our experts and validated experimentally. As an example, Figure 4.1 shows the effects of removing some of the energy terms: each omission yields a product image of inferior quality. Overall, our conclusion is that composition of product images is very difficult as many competing considerations must be balanced.

## 4.5 Optimization

Our optimization procedure searches for camera parameters, object placements, and surface materials that minimize the energy function.

This is a difficult optimization problem for several reasons: 1) there are many free variables (six for the camera, three for each object transformation, one for each surface with multiple candidate materials); 2) some of the variables are continuous (camera and object transformations) while others are discrete (surface materials); and 3) the energy function is highly non-convex, with strong dependencies between

multiple variables (e.g., camera and object movements). As a result, we can only hope to find a good local minimum.

Our approach is to decompose the problem into two simpler, more tractable optimizations that we interleave in an iterative algorithm. Within each iteration, the first step is to optimize the discrete choices of materials with camera parameters and object transformations fixed. Then, we optimize the continuous camera parameters and object transformations with the materials fixed. The iterations terminate when neither step changes the scene significantly in the same iteration.

### 4.5.1 Discrete optimization

We use a discrete steepest-descent algorithm to optimize materials. The input to the algorithm is a scene and a list of candidate definitions for each surface material, and the output is a selection of one candidate definition for each material that minimizes the energy function. Note that the color contrast term $E_{cc}$ is the only one affected by material switches. Higher color contrast for focus objects corresponds to a lower overall energy.

The algorithm first builds a list of visible objects with multiple candidate materials. Then, it iteratively optimizes the materials for each such object in order from the one with the lowest to the highest color contrast. For each object, the algorithm selects the material switch that produces the highest color constrast. The algorithm stops iterating over the objects when no material switches are possible to improve the color contrast further, which usually occurs within 2-3 iterations through all objects.

### 4.5.2 Continuous optimization

We use a continuous steepest-descent algorithm to optimize camera parameters and object transformations. The following paragraph describes how the direction and magnitude of each step is computed.

Since the energy function contains terms whose partial derivatives are difficult to compute analytically (e.g., visibility), we compute the derivative of the energy with respect to each free variable via finite differences. Of course, a brute force implementation of centered differences for each variable would be extremely slow: a typical scene has approximately 150 free variables (3 for each of ∼50 object transformations plus 6 camera parameters), and thus the energy would have to be computed 300 times for each steepest descent step. Instead, we use a technique similar to stochastic gradient descent. Specifically, we keep estimates for all partial derivatives and re-estimate only a subset after most steps. Specifically, every $k$ steps, we estimate partial derivatives for all variables, except ones for transformations of objects outside the view frustum, and make a move along the direction of steepest descent determined by all partial derivatives. We also build a list of objects $T$ that have non-zero partial derivatives. Then, during the intervening steps, we re-estimate partial derivatives only for the camera parameters and $k$ randomly selected objects from $T$ and make a steepest descent move based on these derivatives. We choose $k = \sqrt{|T|}$, which provides a nice trade-off between efficiency and accuracy, leveraging the fact that fewer objects have significant effect on the energy as the optimization converges. To compute the magnitude of each steepest descent step, we conduct a line search along the direction of the estimated derivative.

### 4.5.3 Timing

The full optimization procedure takes approximately 20 minutes for the most complex examples in our work. The discrete optimization step is usually very fast (<10 seconds), since there are relatively few (∼10) candidate materials in most scenes. The continuous optimizations are slower, since there are many possible object transformations in most scenes (∼60 objects per scene in our examples) and computing partial derivatives for each transformation variable requires rendering the scene mul-

tiple times. In our experience, computing partial derivatives takes ∼90 seconds for all variables (every $k$ steps), but only ∼10 seconds for our randomly chosen subsets (intervening steps), at no observed accuracy difference. All times are reported for a 2660 MHz Intel Core i7 processor with 8 GB of memory.

## 4.6 Applications

In this section, we describe several applications of our scene optimization framework. These applications were chosen based on the suggestions of the same two professionals we interviewed to determine the relevant rules for creating effective product images.

### 4.6.1 Refining rough compositions

The primary application of our system is to facilitate the refinement stage of digital catalog image creation. Given a set of focus objects and a rough scene configuration as initialization, we can apply the optimization procedure described in the previous section to automatically adjust the camera, object positions, and materials.

To evaluate whether our system can assist this application, we ran an informal experiment in which we asked a user with formal training in image composition to go through the full process of creating and refining a scene for a product image using an interactive modeling tool, and then investigated how our tool could have helped during the modeling process. We instructed the user to create a 3D scene motivated by an image highlighting a dining room table and chair in the IKEA catalog (Figure 4.3 left), which she could refer to as she modeled. During the session, she started with a set of objects, candidate materials, and a random camera viewpoint (Figure 4.2, A0), and then edited the scene interactively to achieve the final result shown in Figure 4.3 left. We then asked her to further refine the composition to recreate the image in Figure 4.3 right, which highlights the three goblets on the dining table. The

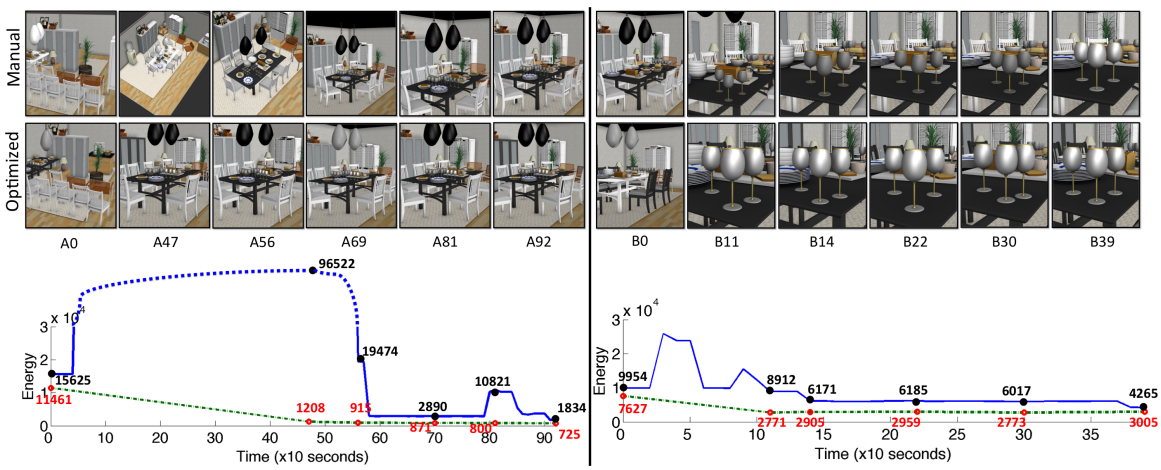



Figure 4.2: Snapshots of an interactive session (top row) and the results of refining them by our optimization tool (second row). In the first session (left), the user's goal is to achieve the composition in Figure 4.3 left, while in the second session (right), her goal is to achieve Figure 4.3 right. The plots on the bottom show the evaluation of these scenes using our energy function, with the blue representing the energy of interactive snapshots and the red points representing our optimized results. Note that the dotted section of the lefthand blue curve has been compressed to save space.

experiment was performed exactly once with no feedback from the system regarding composition quality.

During these interactive sessions, we logged a "snapshot" scene file every 10 seconds representing the user's progress (several examples are shown in the top row of Figure 4.2). After the session was finished, we used the snapshot scenes to: 1) analyze whether our energy function explains changes made interactively by the user, and 2) to study at what point in the modeling process our optimization procedure could have been used to assist the user by refining the scene automatically.

The blue curve in the plot at the bottom of Figure 4.2 shows the value of our energy function for each snapshot of the user's interactive session. Note that for each session, the curve reveals two phases: a period of "large-scale layout" when the scene energy goes up and down ($A0 \rightarrow A47$ and $B0 \rightarrow B11$), followed by a period of "fine-scale refinements" where the energy decreases almost steadily ($A47 \rightarrow A92$

and $B11 \rightarrow B39$). This behavior suggests that the energy function correctly captures image quality differences of improvements made by the user.

The second row of images in Figure 4.2 shows the results of running our optimization procedure on each of the snapshot scenes shown in the top row, and the red dots in the plot below show the energy function of the optimized results (connected by a green curve). Note that the optimized results of the snapshots (bottom row) captured in the latter half of each user session (A47-A92 and B11-B39) are qualitatively similar to the final scene created by the user (top-right image), and their corresponding energy function values are comparable, or even less. These results suggest that more than half of the time the user spent on scene refinement could have been off-loaded to the computer.

### 4.6.2 Generating detail images from an overview

In many cases, catalogs provide an overview image that shows how various objects can fit together in a room, and then one or more detail images that focus on individual products of interest. Detail images are almost never simply cropped and zoomed-in versions of the overview image. Stylists typically choose different viewpoints and move objects slightly in order to highlight the shape and relevant features of the focus object (e.g. Figure 4.3).

To reduce this effort, stylists can use our system to automatically create detail images. For each detail object, our optimization framework initializes all object positions to the arrangement in the overview image and generates a set of candidate detail images using each canonical view of the detail object as a different starting point for the camera. By default, we choose the candidate image with the lowest energy as the result.

As a test of our method, we generated detail images for three scenes: a kitchen, study, and living room. For each scene, we generated detail images for 20 random


Figure 4.3: Overview and detail images in IKEA catalog. In addition to the overview image on the left, IKEA provides a detail image that advertises the glasses on the table. Note how the viewpoint and object positions are adjusted from the overview image (reprinted with permission from the 2013 IKEA catalog).

objects. In many cases, our optimization was able to put the camera close to canonical views only by moving objects that would otherwise occlude the detail object. For example, the gray chair is moved to different positions in Figure 4.4(b) and Figure 4.4(c) in order to reduce occlusions for different focus objects. Overall, most of our detail images produce reasonable compositions, and our perceptual study (Section 4.7) indicates that our full optimization produces better results than camera-only optimizations in a large majority of cases.

### 4.6.3 3D views for room planner

Home furnishing companies have recently started to provide online tools that let users create arrangements of furniture customized for their own rooms, e.g. IKEA's Home Planner. After designing a room in this manner, users often want images of the room to share with others and to help them evaluate the design. Thus, another application

(a) Overview (b) Speaker (c) Shelf

Figure 4.4: Detail images generated from overview. From an overview image of a living room (a), we automatically generate detail images that highlight the speaker (b) and shelf (c). Notice how the chair moves to the right in (b) and to the left in (c) to provide an unobstructed view of the focus object (results without moving objects can be found in Figure 4.9).

of our system is to provide an automated solution for generating well composed images of user-designed rooms.

After generating the 3D arrangement of objects, the user selects a set of focus objects (likely the objects he is considering for purchase) and then asks our system to generate a composition. Unlike the previous two applications, we do not expect the user to provide an initial viewpoint for the scene. As a result, we modify our optimization to first search globally for the best camera parameters, which we then use as an initialization to our full optimization.

For our global camera search, we first generate a set of "plausible" initial camera parameters. We sample camera positions within the walls of the room at roughly 2 ft intervals and restrict the height to be at human eye level; we take 100 samples for the camera direction with a polar angle between $\pi/2$ (looking horizontally) and $\pi/2+0.28$ (looking slightly down); and we consider 4 uniformly spaced field-of-view values from 0.3 to 0.6. We then prune very poor samples based on the fractional image-space area of every focus object bounding box within the viewport (threshold = 50%). Next, we do $k$-means clustering (with $k = 4$) of the camera parameters. Within each cluster, we pick the camera with the lowest energy as the initial viewpoint and run

(a) Initializations (b) Our method (c) Camera-only (optimal) (d) Camera-only (initial as (b))

Figure 4.5: Optimizing without a starting camera for a room planner application. Our system starts by sampling plausible cameras to generate a set of initial views (a), which are then optimized (b). Running camera-only optimization over all initial views yields (c), while starting from the same initialization as (b) yields (d).

our optimization. Finally, we pick the optimized composition with the lowest energy as our result. Without parallelizing the optimization for different initial viewpoints, the entire process took 40 to 60 minutes in our experiments.

We used this optimization procedure to generate the results in Figure 4.5. Here, we chose the couch, coffee table and ottoman as the focus objects. In the final composition (Figure 4.5(b)), all of the focus objects are visible and the image provides a good overview of the scene from a plausible camera angle. For comparison, we show the result of a camera-only optimization in Figure 4.5(c), which uses a different initial viewpoint than Figure 4.5(b). In Figure 4.5(d), we show the camera-only result generated from the same initial viewpoint as image (b). With the capability of moving objects, our full optimization is able to achieve a better balance between multiple factors, and achieve a better overall composition.

### 4.6.4 Object replacement

Multinational furniture companies like IKEA usually customize their catalog images for different countries to match cultural preferences. This customization often involves choosing different materials or replacing objects within a scene. In many cases, the size and shape of new objects can be significantly different from the original, which

(a) Original (b) Objects replaced (c) Optimized

Figure 4.6: Object replacement. From the original composition (a) the chair, side table and coffee table are replaced (b). Our optimization eliminates collisions and produces a better composition for these objects (c).

means that a stylist will have to spend a significant amount of additional effort adjusting the camera parameters and object positions to achieve a good composition for each customization. As with the previous applications, our optimization framework can automatically make these adjustments to reduce the amount of human effort required to perform these cultural customizations.

Figure 4.6 shows an example where we replace the grey seat, side table and coffee table. When we swap in the new objects, there is a collision between the chair and plant, and in general, the composition feels cramped. When we optimize the composition, the collision is resolved and the camera pulls back to keep all the relevant objects in the frame.

### 4.6.5 Text-incorporated composition

Most catalog images have text overlays that describe the depicted scene. Such text is typically positioned over regions with nearly constant color so that it is easy to read and often appears in roughly the same location on every page (e.g., corners) so that the viewer knows where to look to find textual information. Our optimization framework can automatically position text based on all of these criteria.

In addition to a set of focus objects and an initial composition, the stylist also specifies a set of rectangles $\mathbf{R}$ where she would like overlay text to appear in the frame. Our system treats each rectangle as just another object in the scene, but one that only has a 2D position and can only move within the viewport.

We apply the visibility and inertia terms to text rectangles as well. Specifically, the overlapping region of a focus object or a context object with a text rectangle is treated as occlusion.

We introduce two extra terms to make the text with constant color stand out. First, we use a contrast term $E_{tc}$ to keep the background light or dark, so the text can be made a contrasting color.

$$E_{tc} = w_{tv} \sum_{R_i \in \mathbf{R}} \min_{wb=w,b} \frac{1}{\mathcal{A}(R_i)} \sum_{p \in R} \frac{1}{d_l(p, wb)^2 + \epsilon}$$

where $p$ is a pixel in the rectangle $R_i$, $d_l(\cdot, \cdot)$ is the difference between the luminance of two pixels, $w$ is white and $b$ is black, $w_{tc} = 2.0$.

Second, we use a variance term $E_{tv}$ to keep a low variance in luminance within each rectangle to reduce clutter behind overlaid text.

$$\bar{L_{R_i}} = \frac{1}{\mathcal{A}(R_i)} \sum_{p \in R_i} L(p)$$
$$E_{tv} = w_{tv} \sum_{R_i \in \mathbf{R}} \frac{1}{\mathcal{A}(R_i)} \sum_{p \in R_i} \left(L(p) - \bar{L_{R_i}}\right)^2$$

where $w_{tv} = 10000$.

Figure 4.7 presents a composition optimized with two different initial positions for the text. Notice how the objects in the scene are moved to create low contrast, low variance regions of the image where the text can be overlaid.


(a) Input (b) Ours (c) Camera-only

Figure 4.7: Retargeting for different text layouts. The artist provides a rough position for the text box and specifies the champaign bottle and the goblet as focus objects. Our optimization adjusts object positions, viewpoint and text positions to increase contrast, reduce clutter and remove occlusion of the focus objects. By contrast, only optimizing the camera produces an inferior result. Note how the readability of the text is reduced due to the fruit bowl.

(a) Input (b) Ours (c) Camera-only

Figure 4.8: Retargeting for different aspect ratios (focus objects: champagne bottle and goblet). We start with the initial aspect ratio 1:2 (left), and retarget it to a different aspect ratio 4:3 (middle). We compare our result to the one where only the camera is optimized (right).

### 4.6.6 Retargeting for different aspect ratios

Our system can also automatically retarget catalog images to aspect ratios that are appropriate for different display formats. Simple cropping is usually not sufficient to create a good retargeted composition because the relative arrangement of objects in image space remains fixed. In contrast, our optimization framework has the ability

to adjust camera parameters and object positions to produce good compositions for different aspect ratios. For this application, we use the viewpoint and object positions from the input composition as initialization and solve for a new image with the specified dimensions.

In Figure 4.8, we start with the optimal composition in one aspect ratio and then retarget it to another. For comparison, we generate images using our optimization method but without adjusting object positions. Notably, the greater flexibility creates better retargeted images.

## 4.7 Perceptual Study

A natural question to ask when considering our system is whether the additional freedom afforded by moving objects makes a positive impact on the results, or if – to the contrary – similarly good results could be obtained by performing a camera-only optimization. We investigated this question by asking people to compare 36 pairs of compositions created using our optimization procedure with object movement enabled (our method) and disabled (camera-only).

**Study design:** Our selection of scene compositions to compare includes all the examples shown in Section 4.6, plus thirty detail images generated for different objects in three scenes: living room, study, and kitchen (Figure 4.9). For each of the three scenes, we selected – from among all the large furniture and a random subset of the smaller objects – the ten objects where the detail image generated with camera-only optimization differed most from the full optimization.

We showed these pairs of scene compositions to study participants in randomized order, with images in each pair flipped left-right randomly. For each pair, the user selected a radio button to indicate that one composition was better at showcasing the


Figure 4.9: A subset of the image pairs compared in our perceptual study. Our system is able to satisfy multiple composition constraints simultanuously, which cannot be achieved by changing viewpoint only.

specified focus objects (listed in the title), or that the two compositions are of the same quality.

We administered the study to two groups: *experts* who work professionally on scene layout for catalog images, and *non-experts*.

The first group was recruited through personal contacts and completed a single-page web-based survey without compensation. Given the small number of experts in this field, we were only able to administer the survey to four participants.

The second group was recruited through Amazon Mechanical Turk, and each participant was compensated 10 cents. To exclude "lazy" participants from our results, we tested the consistency of each participant's responses. Specifically, each participant completed a multiple-page (one comparison per page) survey, where each comparsion was asked twice, with compositions swapped left-right. We excluded any input from participants for each question where their two answers for the same pair were inconsistent, and we excluded all input from any participant whose answers were inconsistent for more than 25% of the questions. After running the study for 200 participants, these consistency checks yielded 49 to 75 answers per comparison.

| ID | Ours | Camera only | No preference |
|---|---|---|---|
| Expert 1 | 22 | 12 | 2 |
| Expert 2 | 17 | 14 | 3 |
| Expert 3 | 22 | 11 | 3 |
| Expert 4 | 21 | 12 | 3 |

Table 4.2: Expert study. Our method is preferred in general.

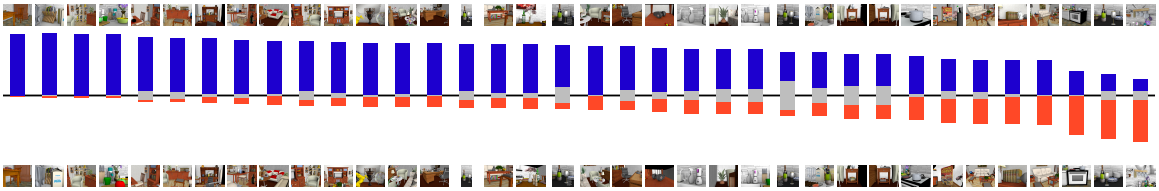


Figure 4.10: Amazon Mechanical Turk Study results. We asked participants to compare 36 pair of images generated with full optimization (top) and camera-only optimization (bottom). Bars represent the proportion of participants who favored each image (dark blue: full, red: camera-only, light grey: no preference).

**Study results:** Results of the expert study and the Amazon Mechanical Turk study are summarized in Table 4.2 and Figure 4.10 respectively. Generally, we find that full scene optimization is preferred to camera-only optimization. If the "no preference" answer is treated as half a vote for our optimization result and half a vote for camera-only, our optimization results received $\geq$ 75% of the votes in 20 cases, 50%-75% of the votes in 13 cases, and $<$ 50% of the votes in 3 cases in the Mechanical Turk study. With a significance level of 0.05, our method is significantly preferred in 26 cases, while the camera-only optimization is significantly preferred in 2 cases. In the remaining 8 cases, the null hypothesis (i.e. no preference) cannot be rejected. Note that the Holm-Bonferroni method was used in the analysis to control the familywise error rate in our experiments. According to comments provided by participants, the main benefits of our full optimization are that it moves objects to avoid unwanted occlusions, generates better contrast, and avoids awkward camera views in the final composition.

## 4.8 Discussion

In this chapter, we have introduced a technique for optimizing 2D compositions of 3D scenes. Our key contribution is an optimization approach that simultaneously adjusts camera parameters, object transformations, and surface materials. Our results and user evaluation show the benefits of optimizing over all of these scene parameters simultaneously. In particular, the comparisons between images generated by only adjusting the camera and those generated by our full optimization clearly indicate that moving objects and altering materials significantly improves the quality of compositions in many cases. We have demonstrated how our optimization framework benefits a variety of applications related to the creation of digital catalog images.

Our system has several limitations. First, the current implementation of our optimization procedure does not run at interactive rates. This is largely because speed has been sacrificed for flexibility in our prototype; we believe a production-oriented implementation could likely run orders of magnitude faster. In particular, the computation of partial derivatives, which is the bottleneck of the speed of our optimization, is highly parallelizable, because partial derivatives with respect to different free variables could be computed independently. Second, we use OpenGL rendering during our optimization, which does not account for global illumination effects that can impact the composition of the final image. Third, there may be additional composition rules that could improve the quality of the results. Early on in the project, we implemented energy terms for diagonal dominance, symmetry, and focusing with vanishing points, but found them less useful in our target applications. Using our system to systematically investigate which energy terms are most effective for which applications would be an interesting topic of further study.

Given that companies are increasingly relying on computer-generated imagery for catalogs and other product advertisements, there are many opportunities for future work related to the automated generation of such images. For example, we imagine

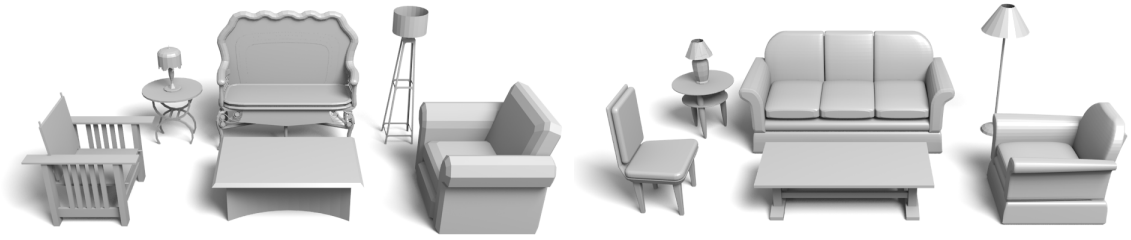
new advertising applications that choose furniture arrangements based on how a room will look from key viewpoints (e.g., the front door). Film, game, and real-estate companies could automatically optimize scenes for sequences of camera viewpoints (e.g., for movie shots or virtual tours). On-line advertisers could adapt product images to wide varieties (millions) of user preferences with automatically optimized aesthetics. We believe composition-aware scene modeling is a useful approach for all of these applications and as such represents a promising research direction for the computer graphics community.

# Chapter 5

# Style Compatibility for 3D Furniture Models

## 5.1 Introduction

In scene modeling, ensuring the plausibility and realism of synthesized 3D scenes is one of the biggest challenges because scene plausibility and realism are determined by many factors. For example, since a scene usually has well-defined functionalities (e.g. a living room should provide seats and make it easy for people to have a conversation), objects need to be selected and arranged so that (virtual) humans are able to interact with the scene in a reasonable way. While many existing tools help users select the appropriate categories and placements of objects when modeling a 3D scene [71, 104], they generally ignore style compatibility — the degree to which objects "exist together in harmony" [72]. For example, while the scene shown in Figure 5.1(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.



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

Figure 5.1: This chapter 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 arrangment but with furniture pieces chosen to optimize stylistic compatibility according to our metric.

In this chapter, we 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 [73] 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 comparisons 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 5.3). 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 5.4). We then learn a compatibility metric using the part-aware geometric features and crowdsourced preferences (Section 5.5); quantitative evaluation shows that this method gives more accurate predictions than existing methods (Section 5.6). The learned metric can then be used to reason about style compatibility in retrieval and interactive modeling applications (Section 5.7).

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.

## 5.2 Related Work

### 5.2.1 Shape styles

Researchers have developed methods to model styles of 2D and 3D shapes [100, 57, 66]. For example, Xu et al. [100] investigate style variations that are caused by anisotropic part scaling. Li et al. [57] 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. [44] propose a fine-grained classification approach by learning a distance metric from a 3D model collection with partial and noisy labels. Kalogerakis et al. [48] 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.

### 5.2.2 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 [75] and illustration styles [32]. Relative attributes have been learned from visual features for image analysis [77] and from shapes for 3D model creation [18]. 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. [65] 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.

### 5.2.3 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 [31], symmetries [51], part structures [82], and other geometric cues [89]. 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.

### 5.2.4 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 [28], probabilistic models [19, 48], physical simulations [93], and interior design guidelines [71]. Other systems have aimed to create scenes completely automatically, for example learning object compatibilities based on substructure symmetries [110], spatial contexts [29, 30], and object contacts [1]. However, no prior system has explicitly considered stylistic compatibility when selecting objects to combine in a virtual world.

## 5.3 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

Figure 5.2: 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.

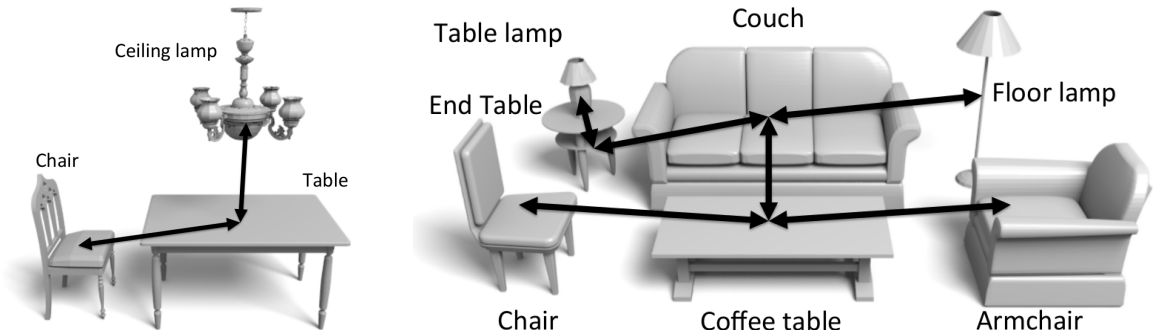
with it. $B$ and $C$ are always from the same object class, and $A$ is always from a different class.

To gather triplets efficiently, we use the grid technique proposed by Wilber et al. [97]. 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 5.2). 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, and the reference object. As demonstrated by Wilber et al. [97], 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 5.3 using Amazon Mechanical Turk. 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 Amazon Mechanical Turk 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.




(a) Dining room (b) Living room

Figure 5.3: Pairs of object classes for which style compatibility preferences are collected in our study. We chose pairs with close proximity and functional interactions.

## 5.4 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 [73]. 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. [52]. 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. [48].

## 5.5 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 [56, 75, 32] has employed distance functions of the form

$$d_{\text{symm}}(\mathbf{x}_i, \mathbf{x}_j) = ||\mathbf{W}(\mathbf{x}_i - \mathbf{x}_j)||_2 \tag{5.1}$$

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 [56]. 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:

$$d_{\text{asymm}}(\mathbf{x}_i, \mathbf{x}_j) = ||\mathbf{W}_{c(i)}\mathbf{x}_i - \mathbf{W}_{c(j)}\mathbf{x}_j||_2 \tag{5.2}$$

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 5.4). We refer to this as the *asymmetric embedding distance*. This model is related to Canonical Correlation Analysis [40] and Neighborhood Components Analysis [33], 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 [75, 32]. The probability that a rater evaluates object $A$ as more compatible to $B$ than to $C$ is modeled as a logistic function:

$$P^A_{B,C} = \frac{1}{1 + \exp(d(\mathbf{x}_A, \mathbf{x}_B) - d(\mathbf{x}_A, \mathbf{x}_C))} \tag{5.3}$$

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

$$E(\mathbf{W}_{1:M}) = -\frac{1}{|\mathcal{D}|} \sum_{(A,B,C)\in\mathcal{D}} \log P^A_{B,C} + \frac{\lambda}{M} \sum_{1\le c\le M} R(\mathbf{W}_c) \tag{5.4}$$

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

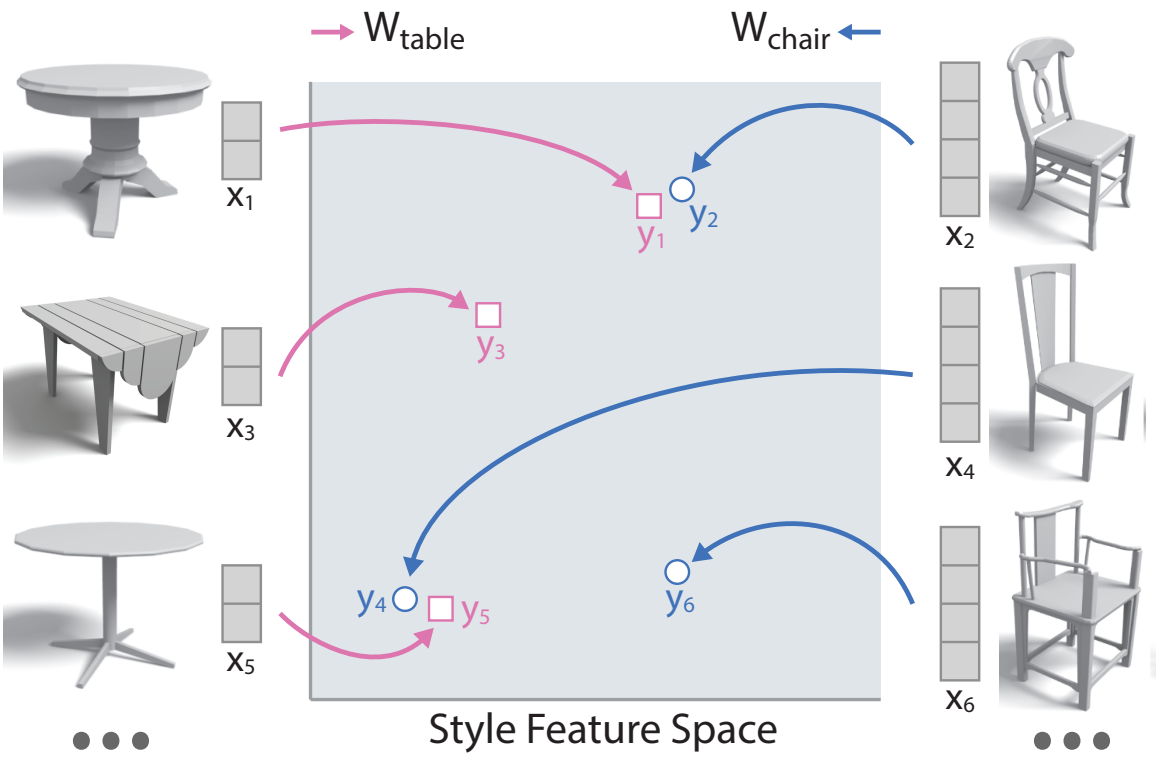



Figure 5.4: Mapping into shared feature space. We learn separate embedding matrices for each object class that map shapes into a shared feature space where objects that are stylistically compatible are close to each other. Here, old-fashioned chairs and tables are clustered at the top, while modern objects are clustered in bottom left. Feature vectors for different classes may have different dimensionality based on the number of parts.

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 [111], 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 [32]. 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:

$$R(\mathbf{W}_c) = \frac{1}{KD} \sum_{1 \leq \ell \leq D} ||\mathbf{w}_\ell||_2 \tag{5.5}$$

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. [3], 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 $\mathbf{W}_{1:M}$.

## 5.6 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.

As discussed previously, many of the triplets in the crowdsourcing study described in Section 5.3 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 $\sim 250$ 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.

Table 5.1 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

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

Table 5.1: 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.

| 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 5.2: 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.

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 5.2. 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 5.2. 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 5.3), 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.

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

Table 5.3: Impact of shared models. Accuracy of style compatibility rankings for our algorithm using different training sets.

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 5.3. 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.

## 5.7 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.

### 5.7.1 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,

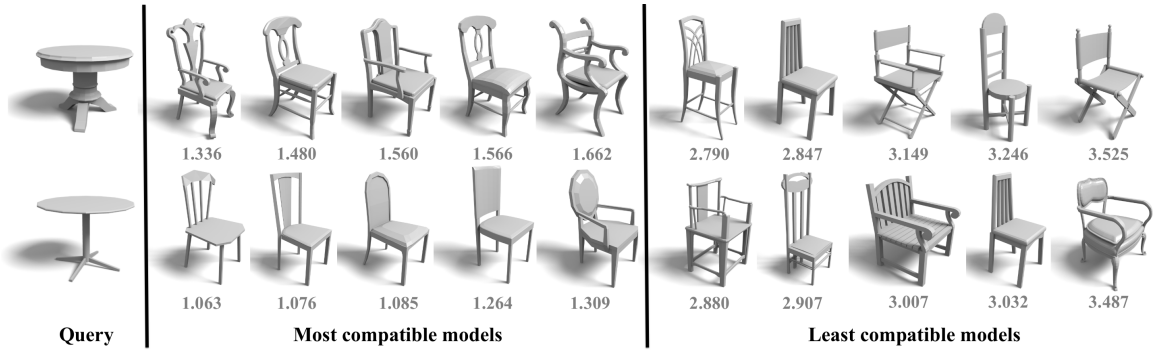



Figure 5.5: Style-aware shape retrieval. Given a query (dining table), our system returns 5 dining chairs that are most stylistically compatible to the query as predicted by our learned metric. We also list the 5 most incompatible models for comparison. The numbers are the compatibility distance $d$ between the chairs and the query, with lower values being more compatible.

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 5.5 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.

### 5.7.2 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 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.

Figure 5.6: 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.

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:

$$F(\{x_i\}) = \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{P}} d(\mathbf{x}_i, \mathbf{x}_j) \tag{5.6}$$

where $\{\mathbf{x}_i\}$ are the models in the scene, $\mathcal{P}$ is the set of linked model pairs shown in Figure 5.3, and $d(\mathbf{x}_i, \mathbf{x}_j)$ is the compatibility distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ (Equation 5.2). 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 (two examples are shown in Figure 5.6) 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 Amazon Mechanical Turk. 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

| | 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 5.4: 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 Amazon Mechanical Turk workers to the one suggested by B.

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 Amazon Mechanical Turk 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 5.4 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*

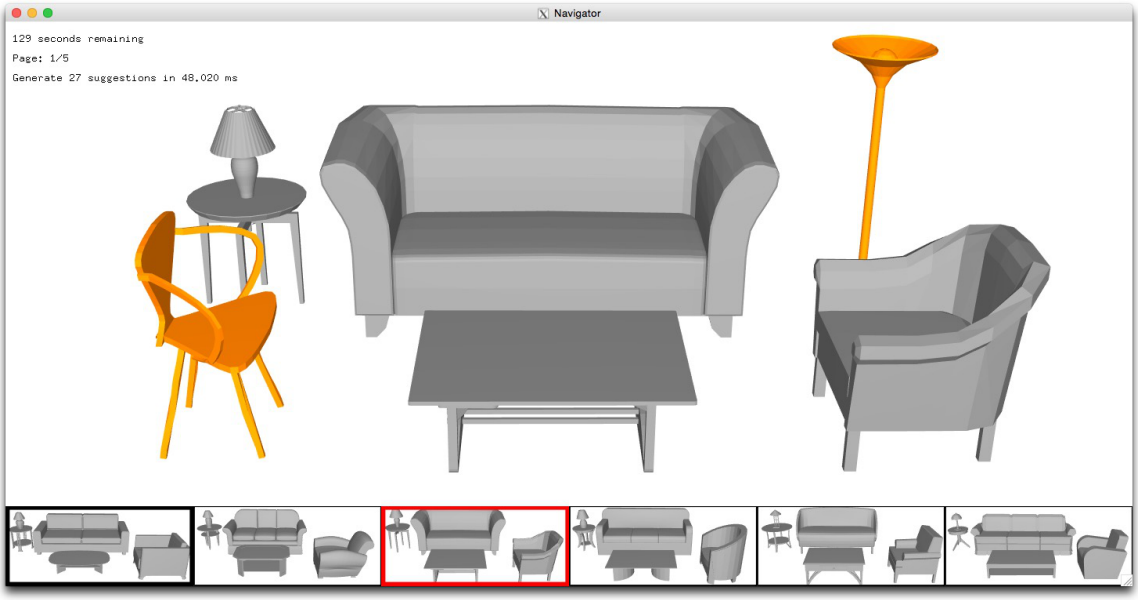



Figure 5.7: Interface of style-aware scene builder. The user is allowed to free (in gray) or fix (in orange) any number objects in the scene, and our system suggests combinations of free objects in order of style compatibility (at the bottom of the window). When the user selects any suggestion from the list (red box), the relevant models are updated in place within the scene.

$<$ *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.

### 5.7.3 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 5.7). 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 5.6). 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 5.7.2), 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 5.3) 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.

**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

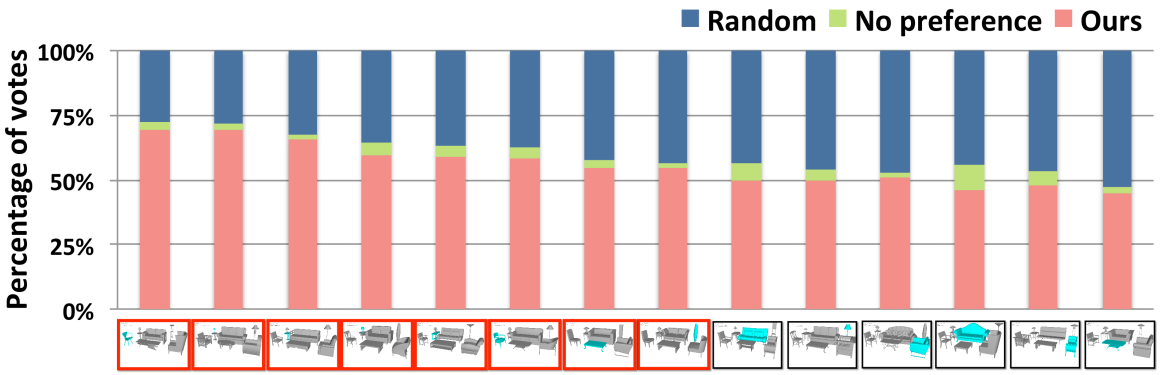



Figure 5.8: Style-aware scene building. Preferences from Amazon Mechanical Turk for the style compatibility of scenes created with our system's suggestions versus the alternative. The tasks are ranked by the descending order of the percentages of votes that favor the results creating by using our system (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).

participants, with the two warm-up tasks first. Participants were given 3 minutes for each task.

To evaluate the two conditions, we asked Amazon Mechanical Turk 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 num-

bers 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 [75, 32].

Moreover, if we look at the data for individual tasks (Figure 5.8), 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.

In an effort to understand the utility of our suggestions in more detail, we analyzed the ranks of the suggestions selected by participants using the two different interfaces. In Figure 5.9, we compare the means and standard errors of the ranks of scenes selected by users in the list of suggestions, with *Ours* in red and *Random* in blue. We find that people selected suggestions with better rankings in our system on average (in 11 of 14 tasks). If two distributions are considered to be significantly different when the effect size is larger than 0.8, then the participants selected suggestions with

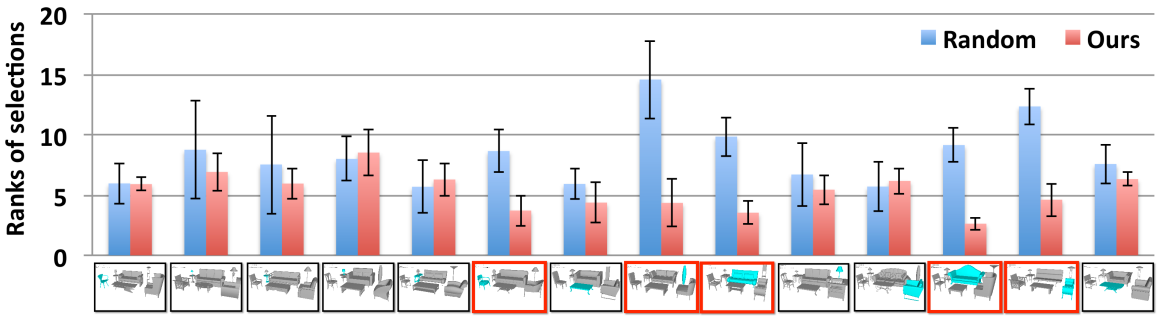



Figure 5.9: Style-aware scene builder analysis. Comparison of the ranks of suggestions selected by users of the scene builder with our algorithm (red bars) versus the alternative (blue bars). Black lines represent standard errors. The participants selected suggestions with smaller ranks using our system than the alternative in 11 out of 14 tasks, with statistical significance in 5 of them (in red boxes).

better ranks using our system in 5 tasks (in red boxes), and there is no significant difference in the remaining 9 tasks.

## 5.8 Discussion

This chapter presents a method for computing style compatibility between 3D furniture models using crowdsourced data and distance learning technique. Besides proposing the first method for learning style compatibility between 3D models from different object classes, we offer two technical contributions. First, we introduce a part-aware geometric feature vector that is better suited for characterizing style-dependent information. Second, we present a new asymmetric embedding distance that is appropriate for estimating compatibility between objects of different classes. 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 [46], 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.

# Chapter 6

# Conclusion and Future Work

## 6.1 Summary

This dissertation describes several algorithms on analyzing, synthesizing, and optimizing 3D scenes by reasoning about relationships between objects. First, we present a method for parsing 3D virtual scenes using a hierarchical probabilistic grammar, and experimental results show that modeling hierarchy significantly improves inference of segmentation and annotation of 3D virtual scenes. Second, we introduce a technique for 2D compositions of 3D scenes that adjusts camera parameters, object transformations, and surface materials. We take into account design principles and user's constraints by modeling object positions and relationships between objects in both the image space and the scene space. Third, we present a method for computing style compatibility between 3D furniture models using crowdsourced data and distance learning technique, and experimental results show that the learned metric can be used effectively to create scenes with higher style compatibility in applications ranging from shape retrieval to interactive scene building.

In this dissertation, we find that reasoning about relationships between objects greatly facilitates scene analysis, scene optimization, and scene synthesis. First, re-

lationships between objects are strongly related to the plausibility and aesthetics of scenes. For example, we have shown that style compatibility between furniture models is important for scene plausibility, and color contrast between adjacent objects in the image space is critical for the quality of compositions created from 3D scenes. Therefore, it is effective to impose constraints in scene synthesis and optimization by modeling relationships between objects. Second, relationships between objects, particularly the ones between objects within a semantic subgroup, are comparatively stable across scenes. Because relationships between objects are related to the plausibility and semantics of 3D scenes, scenes usually share similar relationships, even if they are very different in other aspects, such as geometry and object positions. For example, the spatial relationships between beds and nightstands are nearly invariant, although the sizes and geometries of nightstands can be significantly different in different bedrooms. This is because a nightstand needs to be reachable from a bed beside it. Consequently, relationships between objects provide a strong cue for scene understanding.

## 6.2 Future Work

We identify several directions for future work on analyzing, synthesizing, and optimizing 3D virtual scenes.

First, it would be interesting to explore other sources of scene examples that the knowledge of scene realism can be learned from. Although virtual scenes provide opportunities for data-driven scene modeling, the number of 3D virtual scenes in online repositories is still not comparable to the numbers of many other sources of scenes. For example, there are millions of high-quality product images of indoor scenes on the Internet, which are great examples for learning design constraints and image composition rules. Furthermore, as RGB-D cameras become more popular and

cheaper, it will become easier to acquire a large number of scans of real scenes, which would provide even more opportunities for data-driven scene modeling. However, similar to the scenario of virtual scenes, scene analysis tools would probably require good segmentation and annotation on the exemplar images or RGB-D scans in order to maximize the usefulness of the data, which would require better representations of scene grammars and more sophisticated analysis tools.

Second, it would also be interesting to investigate other factors that are related to scene plausibility and scene semantics. This dissertation majorly focuses on spatial relationships between objects in the scene space and the image space, and style compatibility between objects that is solely computed from geometric properties. One factor that has not been fully investigated is materials. In this dissertation, we consider materials for maximizing color contrast when generating compositions of virtual scenes, but we believe materials may play a larger role in ensuring the plausibility of a scene in scene synthesis. We leave a thorough investigation for the role of materials in scene synthesis for future work. Furthermore, we also imagine materials can provide a strong cue for scene understanding. In particular, materials are strongly related to object categories, as pointed out by previous work [46, 9]. Therefore, it would be fruitful to consider materials when inferring segmentation and annotation on images or RGB-D scans. Besides materials, many other properties of 3D models, such as construction methods and affordances, could also be strongly related to scene plausibility, and we leave them for future work.

# Bibliography


[1] Yoshiaki Akazawa, Yoshihiro Okada, and Koichi Niijima. Automatic 3d scene generation based on contact constraints. In *Proc. Conf. on Computer Graphics and Artificial Intelligence*, pages 593–598, 2005.

[2] Rudolf Arnheim. *The Power of the Center*. University of California Press, 1988.

[3] Francis Bach, Rodolphe Jenatton, Julien Mairal, and Guillaume Obozinski. Optimization with sparsity-inducing penalties. *Foundations and Trends in Machine Learning*, 4(1):1–106, 2012.

[4] Serene Banerjee and Brian L Evans. Unsupervised automation of photographic composition rules in digital still cameras. In *Electronic Imaging 2004*, pages 364–373. International Society for Optics and Photonics, 2004.

[5] William Bares. A photographic composition assistant for intelligent virtual 3d camera systems. In *Smart Graphics*, pages 172–183. Springer, 2006.

[6] William Bares, Scott McDermott, Christina Boudreaux, and Somying Thainimit. Virtual 3d camera composition from frame constraints. In *Proceedings of the eighth ACM international conference on Multimedia*, pages 177–186. ACM, 2000.

[7] Blaine Bell, Steven Feiner, and Tobias Höllerer. View management for virtual and augmented reality. In *Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 101–110. ACM, 2001.

[8] Lori Bell, Tom Peters, and Kitty Pope. Get a (second) life!: Prospecting for gold in a 3-d world. *Computers in Libraries*, 27(1):10–15, 2007.

[9] Sean Bell, Paul Upchurch, Noah Snavely, and Kavita Bala. Opensurfaces: A richly annotated catalog of surface appearance. *ACM Transactions on Graphics (TOG)*, 32(4):111, 2013.

[10] Ray Bethers. *Composition in Pictures*. Pitman, 1956.

[11] Subhabrata Bhattacharya, Rahul Sukthankar, and Mubarak Shah. A framework for photo-quality assessment and enhancement based on visual aesthetics. In *Proceedings of the international conference on Multimedia*, pages 271–280. ACM, 2010.

[12] Christopher M Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc., 2006.

[13] Volker Blanz, Michael J Tarr, Heinrich H Bülthoff, and Thomas Vetter. What object attributes determine canonical views? *Perception-London*, 28(5):575–600, 1999.

[14] Martin Bokeloh, Michael Wand, and Hans-Peter Seidel. A connection between partial symmetry and inverse procedural modeling. *ACM Trans. Graph.*, 29(4):104, 2010.

[15] Alexandre Boulch, Simon Houllier, Renaud Marlet, and Olivier Tournaire. Semantizing complex 3D scenes using constrained attribute grammars. In *Computer Graphics Forum*, volume 32, pages 33–42. Wiley Online Library, 2013.

[16] Richard W Bukowski and Carlo H Séquin. Object associations: a simple and practical approach to virtual 3d manipulation. In *Proceedings of the 1995 symposium on Interactive 3D graphics*, pages 131–ff. ACM, 1995.

[17] Zachary Byers, Michael Dixon, William D Smart, and Cindy M Grimm. Say cheese! experiences with a robot photographer. *AI magazine*, 25(3):37, 2004.

[18] Siddhartha Chaudhuri, Evangelos Kalogerakis, Stephen Giguere, and Thomas Funkhouser. Attribit: content creation with semantic attributes. In *Proc. UIST*, pages 193–202. ACM, 2013.

[19] Siddhartha Chaudhuri, Evangelos Kalogerakis, Leonidas Guibas, and Vladlen Koltun. Probabilistic reasoning for assembly-based 3d modeling. *ACM Trans. Graph.*, 30(4), 2011.

[20] Kang Chen, Yu-Kun Lai, Yu-Xin Wu, Ralph Martin, and Shi-Min Hu. Automatic semantic modeling of indoor scenes from low-quality rgb-d data using contextual information. *ACM Transactions on Graphics*, 33(6), 2014.

[21] Wongun Choi, Yu-Wei Chao, Caroline Pantofaru, and Silvio Savarese. Understanding indoor scenes using 3d geometric phrases. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 33–40. IEEE, 2013.

[22] Marc Christie, Patrick Olivier, and Jean-Marie Normand. Camera control in computer graphics. In *Computer Graphics Forum*, volume 27, pages 2197–2218. Wiley Online Library, 2008.

[23] Jack Clifton. *The Eye of the Artist*. North Light Publishers, 1973.

[24] Ritendra Datta, Dhiraj Joshi, Jia Li, and James Z Wang. Studying aesthetics in photographic images using a computational approach. In *Computer Vision–ECCV 2006*, pages 288–301. Springer, 2006.

[25] Ritendra Datta and James Z Wang. Acquine: aesthetic quality inference engine-real-time automatic rating of photo aesthetics. In *Proceedings of the international conference on Multimedia information retrieval*, pages 421–424. ACM, 2010.

[26] Jay Earley. An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2):94–102, 1970.

[27] Martin Enthed. 3D at IKEA. In *3D Modeling Standards*. SIGGRAPH Birds of a Feather, 2012.

[28] Matthew Fisher and Pat Hanrahan. Context-based search for 3d models. *ACM Trans. Graph.*, 29(6):182, 2010.

[29] Matthew Fisher, Daniel Ritchie, Manolis Savva, Thomas Funkhouser, and Pat Hanrahan. Example-based synthesis of 3d object arrangements. *ACM Trans. Graph.*, 31(6):135, 2012.

[30] Matthew Fisher, Manolis Savva, and Pat Hanrahan. Characterizing structural relationships in scenes using graph kernels. *ACM Trans. Graph.*, 30(4), 2011.

[31] Thomas Funkhouser, Patrick Min, Michael Kazhdan, Joyce Chen, Alex Halderman, David Dobkin, and David Jacobs. A search engine for 3d models. *ACM Trans. Graph.*, 22(1):83–105, 2003.

[32] Elena Garces, Aseem Agarwala, Diego Gutierrez, and Aaron Hertzmann. A similarity measure for illustration style. *ACM Trans. Graph.*, 33(4), 2014.

[33] Jacob Goldberger, Sam Roweis, Geoff Hinton, and Ruslan Salakhutdinov. Neighbourhood components analysis. *Advances in Neural Information Processing Systems*, 2004.

[34] Aleksey Golovinskiy and Thomas Funkhouser. Consistent segmentation of 3D models. *Computers & Graphics*, 33(3):262–269, 2009.

[35] Aleksey Golovinskiy, Vladimir G. Kim, and Thomas Funkhouser. Shape-based Recognition of 3D Point Clouds in Urban Environments. *ICCV*, 2009.

[36] Bruce Gooch, Erik Reinhard, Chris Moulding, and Peter Shirley. Artistic composition for image creation. In *Rendering Techniques 2001: Proceedings of the Eurographics Workshop in London, United Kingdom, June 25-27, 2001*, page 83. Springer Verlag Wien, 2001.

[37] Naga K Govindaraju, Stephane Redon, Ming C Lin, and Dinesh Manocha. Cullide: Interactive collision detection between complex models in large environments using graphics hardware. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, pages 25–32. Eurographics Association, 2003.

[38] Tom Grill and Mark Scanlon. *Photographic composition*. Amphoto Books, 1990.

[39] Cindy Grimm. Post-rendering composition for 3d scenes. *Eurographics short papers*, 20(3), 2001.

[40] Harold Hotelling. Relations between two sets of variates. *Biometrika*, 28(3-4):321–377, 1936.

[41] Ruizhen Hu, Lubin Fan, and Ligang Liu. Co-segmentation of 3D shapes via subspace clustering. In *Computer Graphics Forum*, volume 31, pages 1703–1713. Wiley Online Library, 2012.

[42] Qi-Xing Huang and Leonidas Guibas. Consistent shape maps via semidefinite programming. In *Computer Graphics Forum*, volume 32, pages 177–186. Wiley Online Library, 2013.

[43] Qi-Xing Huang, Vladlen Koltun, and Leonidas Guibas. Joint shape segmentation with linear programming. In *ACM Trans. Graph.*, volume 30, page 125. ACM, 2011.

[44] Qi-Xing Huang, Hao Su, and Leonidas Guibas. Fine-grained semi-supervised labeling of large shape collections. *ACM Trans. Graph.*, 32(6):190, 2013.

[45] Qi-Xing Huang, Guo-Xin Zhang, Lin Gao, Shi-Min Hu, Adrian Butscher, and Leonidas Guibas. An optimization approach for extracting and encoding consistent maps in a shape collection. *ACM Trans. Graph.*, 31(6):167, 2012.

[46] Arjun Jain, Thorsten Thormählen, Tobias Ritschel, and Hans-Peter Seidel. Material memex: Automatic material suggestions for 3d objects. *ACM Trans. Graph.*, 31(5):143, 2012.

[47] Yong Jin, Qingbiao Wu, and Ligang Liu. Aesthetic photo composition by optimal crop-and-warp. *Computers & Graphics*, 36(8):955–965, 2012.

[48] Evangelos Kalogerakis, Siddhartha Chaudhuri, Daphne Koller, and Vladlen Koltun. A probabilistic model for component-based shape synthesis. *ACM Trans. Graph.*, 31(4):55, 2012.

[49] Evangelos Kalogerakis, Aaron Hertzmann, and Karan Singh. Learning 3d mesh segmentation and labeling. In *ACM Trans. Graph.*, volume 29, page 102. ACM, 2010.

[50] Hong-Cheng Kao, Wan-Chun Ma, and Ming Ouhyoung. Esthetics-based quantitative analysis of photo composition. In *Pacific Graphics*, 2008.

[51] Michael Kazhdan, Bernard Chazelle, David Dobkin, Thomas Funkhouser, and Szymon Rusinkiewicz. A reflective symmetry descriptor for 3d models. *Algorithmica*, 38(1):201–225, 2004.

[52] Vladimir G. Kim, Wilmot Li, Niloy J Mitra, Siddhartha Chaudhuri, Stephen DiVerdi, and Thomas Funkhouser. Learning part-based templates from large collections of 3d shapes. *ACM Trans. Graph.*, 32(4), 2013.

[53] Vladimir G. Kim, Wilmot Li, Niloy J Mitra, Stephen DiVerdi, and Thomas A Funkhouser. Exploring collections of 3d models using fuzzy correspondences. *ACM Trans. Graph.*, 31(4):54, 2012.

[54] Michael A Kowalski, John F Hughes, Cynthia Beth Rubin, and Jun Ohya. User-guided composition effects for art-based rendering. In *Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 99–102. ACM, 2001.

[55] Bert Krages. *Photography: the art of composition*. Skyhorse Publishing, Inc., 2012.

[56] Brian Kulis. Metric learning: A survey. *Foundations and Trends in Machine Learning*, 5(4):287–364, 2012.

[57] Honghua Li, Hao Zhang, Yanzhen Wang, Junjie Cao, Ariel Shamir, and Daniel Cohen-Or. Curve style analysis in a set of shapes. In *Computer Graphics Forum*, volume 32, pages 77–88. Wiley Online Library, 2013.

[58] Yangyan Li, Angela Dai, Leonidas Guibas, and Matthias Nießner. Database-assisted object retrieval for real-time 3d reconstruction. In *Computer Graphics Forum*, volume 34. Wiley Online Library, 2015.

[59] Dahua Lin, Sanja Fidler, and Raquel Urtasun. Holistic scene understanding for 3d object detection with rgbd cameras. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 1417–1424. IEEE, 2013.

[60] Ligang Liu, Renjie Chen, Lior Wolf, and Daniel Cohen-Or. Optimizing photo composition. In *Computer Graphics Forum*, volume 29, pages 469–478. Wiley Online Library, 2010.

[61] Tianqiang Liu, Siddhartha Chaudhuri, Vladimir G. Kim, Qi-Xing Huang, Niloy J. Mitra, and Thomas Funkhouser. Creating consistent scene graphs using a probabilistic grammar. *ACM Trans. Graph.*, 33(6), December 2014.

[62] Tianqiang Liu, Aaron Hertzmann, Wilmot Li, and Thomas Funkhouser. Style compatibility for 3D furniture models. *ACM Trans. Graph.*, 34(4), August 2015.

[63] Tianqiang Liu, Jim McCann, Wilmot Li, and Thomas Funkhouser. Composition-aware scene optimization for product images. *Computer Graphics Forum (Proc. Eurographics)*, 34(2), May 2015.

[64] Simon Lok, Steven Feiner, and Gary Ngai. Evaluation of visual balance for automated layout. In *Proceedings of the 9th international conference on Intelligent user interfaces*, pages 101–108. ACM, 2004.

[65] Zhaoliang Lun, Evangelos Kalogerakis, and Alla Sheffer. Elements of style: Learning structure-transcending perceptual shape style similarity. *ACM Trans. Graph.*, 34(4), 2015.

[66] Chongyang Ma, Haibin Huang, Alla Sheffer, Evangelos Kalogerakis, and Rui Wang. Analogy-driven 3d style transfer. In *Computer Graphics Forum*, volume 33, pages 175–184. Wiley Online Library, 2014.

[67] Lucas Majerowicz, Ariel Shamir, Alla Sheffer, and H Hoos. Filling your shelves: Synthesizing diverse style-preserving artifact arrangements. 2013.

[68] Benjamin Martinez. *Visual forces: an introduction to design*. Pearson College Division, 1995.

[69] Andelo Martinović and Luc Van Gool. Bayesian grammar learning for inverse procedural modeling. In *CVPR*, 2013.

[70] Markus Mathias, Andelo Martinović, Julien Weissenberg, and Luc Van Gool. Procedural 3d building reconstruction using shape grammars and detectors. In *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2011 International Conference on*, pages 304–311. IEEE, 2011.

[71] Paul Merrell, Eric Schkufza, Zeyang Li, Maneesh Agrawala, and Vladlen Koltun. Interactive furniture layout using interior design guidelines. *ACM Trans. Graph.*, 30(4):87, 2011.

[72] Merriam-Webster. *Merriam-Webster Dictionary*. Merriam-Webster Mass Market, July 2004.

[73] Judith Miller. *Furniture*. Penguin, 2005.

[74] Andy Nguyen, Mirela Ben-Chen, Katarzyna Welnicka, Yinyu Ye, and Leonidas Guibas. An optimization approach to improving collections of shape maps. In *CGF*, volume 30, pages 1481–1491, 2011.

[75] Peter O'Donovan, Jānis Lībeks, Aseem Agarwala, and Aaron Hertzmann. Exploratory font selection using crowdsourced attributes. *ACM Trans. Graph.*, 33(4):92, 2014.

[76] Patrick Olivier, Nicolas Halper, Jon Pickering, and Pamela Luna. Visual composition as optimisation. In *AISB Symposium on AI and Creativity in Entertainment and Visual Art*, pages 22–30, 1999.

[77] Devi Parikh and Kristen Grauman. Relative attributes. In *Proc. ICCV*, pages 503–510, 2011.

[78] Emanuel Parzen. On estimation of a probability density function and mode. *Ann. Math. Stat.*, 33(3):1065–1076, 1962.

[79] Scott Satkin, Jason Lin, and Martial Hebert. Data-driven scene understanding from 3D models. In *Proceedings of the 23rd British Machine Vision Conference*, 2012.

[80] Alexander G Schwing, Sanja Fidler, Marc Pollefeys, and Raquel Urtasun. Box in the box: Joint 3d layout and object reasoning from single images. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 353–360. IEEE, 2013.

[81] Adrian Secord, Jingwan Lu, Adam Finkelstein, Manish Singh, and Andrew Nealen. Perceptual models of viewpoint preference. *ACM Transactions on Graphics (TOG)*, 30(5):109, 2011.

[82] Lior Shapira, Shy Shalom, Ariel Shamir, Daniel Cohen-Or, and Hao Zhang. Contextual part analogies in 3d objects. *International Journal of Computer Vision*, 89(2-3):309–326, 2010.

[83] Oana Sidi, Oliver van Kaick, Yanir Kleiman, Hao Zhang, and Daniel Cohen-Or. Unsupervised co-segmentation of a set of shapes via descriptor-space spectral clustering. In *ACM Trans. Graph.*, volume 30, page 126. ACM, 2011.

[84] Richard Socher, Cliff C Lin, Andrew Ng, and Chris Manning. Parsing natural scenes and natural language with recursive neural networks. In *ICML*, pages 129–136, 2011.

[85] Alex Southern. Real or rendered? how 3d imagery is changing the way you shop. *Technomy*, October 2012.

[86] Ondrej Šťava, Bedrich Beneš, R Měch, Daniel G Aliaga, and Peter Krištof. Inverse procedural modeling by automatic generation of L-systems. In *Computer Graphics Forum*, volume 29, pages 665–674. Wiley Online Library, 2010.

[87] Scott Stump. Say it ain't so: Ikea reveals 75 *Today Home*, 2014. http://www.today.com/home/say-it-aint-so-ikea-reveals-75-catalog-images-are-1D80127051.

[88] Jerry Talton, Lingfeng Yang, Ranjitha Kumar, Maxine Lim, Noah Goodman, and Radomír Měch. Learning design patterns with bayesian grammar induction. In *UIST*, pages 63–74. ACM, 2012.

[89] Johan WH Tangelder and Remco C Veltkamp. A survey of content based 3d shape retrieval methods. *Multimedia tools and applications*, 39(3):441–471, 2008.

[90] Edward Canby Taylor. *The How and Why of Photographic Composition*. The Galleon Publishers, 1938.

[91] Olivier Teboul, Iasonas Kokkinos, Loic Simon, Panagiotis Koutsourakis, and Nikos Paragios. Parsing facades with shape grammars and reinforcement learning. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(7):1744–1756, 2013.

[92] Trimble. Trimble 3D warehouse, http://sketchup.google.com/3Dwarehouse/ 2012.

[93] Nobuyuki Umetani, Takeo Igarashi, and Niloy J Mitra. Guided exploration of physically valid shapes for furniture design. *ACM Trans. Graph.*, 31(4):86, 2012.

[94] Oliver van Kaick, Kai Xu, Hao Zhang, Yanzhen Wang, Shuyang Sun, Ariel Shamir, and Daniel Cohen-Or. Co-hierarchical analysis of shape structures. *ACM Trans. Graph.*, 32(4):69, 2013.

[95] Yanzhen Wang, Kai Xu, Jun Li, Hao Zhang, Ariel Shamir, Ligang Liu, Zhiquan Cheng, and Yueshan Xiong. Symmetry hierarchy of man-made objects. In *Computer Graphics Forum*, volume 30, pages 287–296. Wiley Online Library, 2011.

[96] Peter Ward. *Picture composition for film and television.* Taylor & Francis, 2003.

[97] Michael J Wilber, Iljung S Kwak, and Serge J Belongie. Cost-effective hits for relative similarity comparisons. In *Proc. HCOMP*, 2014.

[98] Lai-Kuan Wong and Kok-Lim Low. Saliency retargeting: An approach to enhance image aesthetics. In *Applications of Computer Vision (WACV), 2011 IEEE Workshop on*, pages 73–80. IEEE, 2011.

[99] Fuzhang Wu, Dong-Ming Yan, Weiming Dong, Xiaopeng Zhang, and Peter Wonka. Inverse procedural modeling of facade layouts. *ACM Trans. Graph.*, 33(4), 2014.

[100] Kai Xu, Honghua Li, Hao Zhang, Daniel Cohen-Or, Yueshan Xiong, and Zhi-Quan Cheng. Style-content separation by anisotropic part scales. *ACM Trans. Graph.*, 29(6):184, 2010.

[101] Kai Xu, Rui Ma, Hao Zhang, Chenyang Zhu, Ariel Shamir, Daniel Cohen-Or, and Hui Huang. Organizing heterogeneous scene collection through contextual focal points. *ACM Transactions on Graphics, (Proc. of SIGGRAPH 2014)*, 33(4):to appear, 2014.

[102] Kun Xu, Kang Chen, Hongbo Fu, Wei-Lun Sun, and Shi-Min Hu. Sketch2scene: Sketch-based co-retrieval and co-placement of 3d models. *ACM Trans. Graph.*, 32(4):123, 2013.

[103] Yi-Ting Yeh, Lingfeng Yang, Matthew Watson, Noah D Goodman, and Pat Hanrahan. Synthesizing open worlds with constraints using locally annealed reversible jump mcmc. *ACM Transactions on Graphics (TOG)*, 31(4):56, 2012.

[104] Lap-Fai Yu, Sai Kit Yeung, Chi-Keung Tang, Demetri Terzopoulos, Tony F Chan, and Stanley Osher. Make it home: automatic optimization of furniture arrangement. *ACM Trans. Graph.*, 30(4):86, 2011.

[105] Lap-Fai Yu, Sai-Kit Yeung, and Demetri Terzopoulos. The clutterpalette: An interactive tool for detailing indoor scenes. *IEEE Transactions on Visualization and Computer Graphics*, 21, 2015.

[106] Hao Zhang, Kai Xu, Wei Jiang, Jinjie Lin, Daniel Cohen-Or, and Baoquan Chen. Layered analysis of irregular facades via symmetry maximization. *ACM Trans. Graph.*, 32(4):121, 2013.

[107] Yinda Zhang, Shuran Song, Ping Tan, and Jianxiong Xiao. Panocontext: A whole-room 3d context model for panoramic scene understanding. In *Computer Vision–ECCV 2014*, pages 668–686. Springer, 2014.

[108] Yibiao Zhao and Song-Chun Zhu. Scene parsing by integrating function, geometry and appearance models. CVPR, 2013.

[109] Youyi Zheng, Daniel Cohen-Or, Melinos Averkiou, and Niloy J. Mitra. Recurring part arrangements in shape collections. *Computer Graphics Forum (Special issue of Eurographics 2014)*, 2014.

[110] Youyi Zheng, Daniel Cohen-Or, and Niloy J Mitra. Smart variations: Functional substructures for part compatibility. *Computer Graphics Forum*, 32(2pt2):195–204, 2013.

[111] Ciyou Zhu, Richard H Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM TOMS*, 23(4):550–560, 1997.