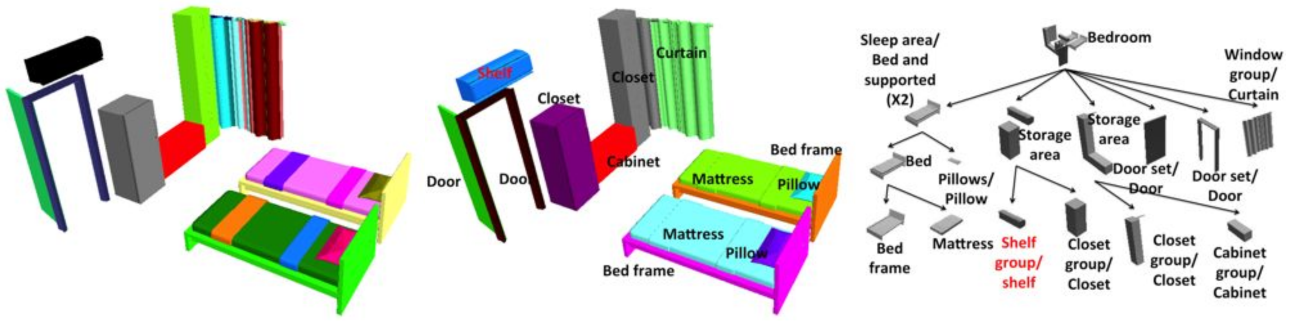
Creating Consistent Scene Graphs Using a Probabilistic Grammar

Tianqiang Liu[1] Siddhartha Chaudhuri[1,2] Vladimir G. Kim[3] Qixing Huang[3,4] Niloy J. Mitra[5] Thomas Funkhouser[1]

[1]Princeton University [2]Cornell University [3]Stanford University [4]Toyota Technological Institute at Chicago [5]University College London



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

Figure 1: *Our algorithm processes raw scene graphs with possible over-segmentation (a), obtained from repositories such as the Trimble 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.*

Abstract

Growing numbers of 3D scenes in online repositories provide new opportunities for data-driven scene understanding, editing, and synthesis. Despite the plethora of data now available online, most of it cannot be effectively used for data-driven applications because it lacks consistent segmentations, category labels, and/or functional groupings required for co-analysis. In this paper, we develop algorithms that infer such information via parsing with a probabilistic grammar learned from examples. First, given a collection of scene graphs with consistent hierarchies and labels, we train a probabilistic hierarchical grammar to represent the distributions 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 these algorithms, we find that: they work effectively for scene graphs for indoor scenes commonly found online (bedrooms, classrooms, and libraries); they outperform alternative approaches that consider only shape similarities and/or spatial relationships without hierarchy; they require relatively small sets of training data; they are robust to moderate over-segmentation in the inputs; and, they can robustly transfer labels from one data set to another. As a result, the proposed algorithms can be used to provide consistent hierarchies for large collections of scenes within the same semantic class.

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

Keywords: scene understanding, scene collections

Links: DL PDF WEB DATA CODE

1 Introduction

The abundance of 3D scenes in online repositories offers valuable input for a multitude of data-driven interfaces for exploring and synthesizing novel scenes. Previous work offers tools for sketch-based scene modeling [Xu et al. 2013], context-based object retrieval [Fisher and Hanrahan 2010], scene retrieval [Fisher et al. 2011], scene organization [Xu et al. 2014], and automatic scene synthesis [Fisher et al. 2012]. Unfortunately, these interfaces 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 1a that shows a scene downloaded from the Trimble 3D Warehouse [Trimble 2012]. 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 meaningful 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).

Our goal is to develop algorithms that build a consistent representation for the hierarchical decomposition of a scene into semantic components. 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 1a, we produce the scene graph shown in Figure 1b,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 [Kalogerakis et al. 2010; Kim et al. 2013], correspondences [Huang et al. 2011], and hierarchies [van Kaick et al. 2013] 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 [Zhao and Zhu 2013] and/or work only on special types of layouts, such as building facades [Zhang et al. 2013].

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 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).

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 [Socher et al. 2011] or two-dimensional facades [Martinović and Van Gool 2013], 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 [Bishop 2006].

We use our method to semantically label several datasets drawn from various publicly available scene repositories, including the Trimble (Google) 3D Warehouse and the Sketch2Scene collection [Xu et al. 2013]. 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.

2 Related Work

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 [Golovinskiy and Funkhouser 2009; Huang et al. 2011; Sidi et al. 2011; Hu et al. 2012; Zheng et al. 2014] and joint shape matching [Nguyen et al. 2011; Kim et al. 2012; Huang et al. 2012; Kim et al. 2013; Huang and Guibas 2013]. However, these methods are designed for collections of individual objects (e.g., chairs) and assume relatively small numbers of subparts 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.

Hierarchical shape analysis. Several previous techniques demonstrate the advantages of a hierarchical representation. Wang et al. [2011] 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. [2013] present a method that infers consistent part hierarchies for a collection of shapes. The method takes as input a set of shapes, each presegmented 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.

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 [Teboul et al. 2013], buildings [Mathias et al. 2011; Boulch et al. 2013], and facades [Martinović and Van Gool 2013; Zhang et al. 2013; Wu et al. 2014], which exhibit 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 this paper.

In the computer vision community, several algorithms have been proposed to parse images of indoor environments using annotated 3D geometry [Choi et al. 2013; Zhao and Zhu 2013]. 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.

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, Št'ava et al. [2010] derived L-systems from plants; Bokeloh et al. [2010] discovered repeating units and connections to form a procedural model for shapes; while, Talton et al. [2012] 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.

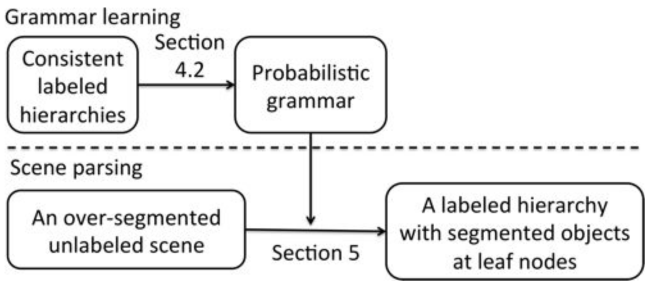

Grammar learning
Consistent labeled hierarchies
Section 4.2
Probabilistic grammar
Scene parsing
An over-segmented unlabeled scene
Section 5
A labeled hierarchy with segmented objects at leaf nodes

Figure 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.*

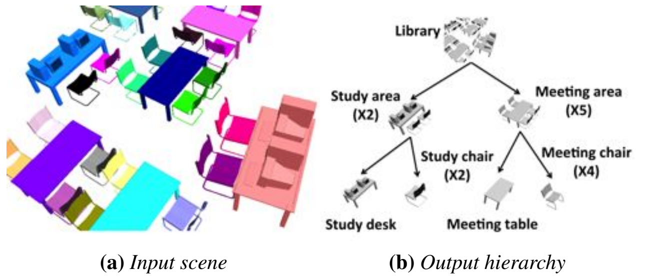


Figure 3: *An example library scene. By grouping objects, we are not only able to detect interesting intermediate-level structures, e.g. study area and meeting area, but also distinguish objects based on their functionalities, e.g. study chair and meeting chair.*

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. [2011] and Kalogerakis et al. [2012] train generative models of component-based shape structure from compatibly segmented and labeled models for shape synthesis, while Fisher et al. [2012] and Yeh et al. [2012] 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 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 1a, 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. It contains several meeting and study areas, where each area provides 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 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 5).

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.

4.1 Grammar specification

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

$$\mathcal{G} =< \mathbf{L}, \mathbf{R}, \mathbf{P} > \tag{1}$$

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 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 λ is the set of right-hand-side labels. For example, a production rule could be:

$$bed \ \to \ bed\text{-}frame \ \ mattress.$$

Since our grammar is non-recursive, λ 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 $l \rightarrow \lambda$ 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_{card} assigns cardinality probabilities independently to each child. For example, the observed productions:

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

are combined into a single production:

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

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.children, matches rule $l \rightarrow \lambda$ as:

$$\begin{aligned} P_{\text{prod}}(x) = & \; P_{\text{nt}}(x.\text{label}) \\ & \times \prod_{r \in \lambda} P_{\text{card}}[x.\text{label}, r] \left(\sum_{y \in x.\text{children}} \mathbf{1}_{y.\text{label}=r} \right) \end{aligned} \quad (2)$$

where x is a node in the parse tree labeled as x.label with a set of children x.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, 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. The descriptor is described in detail in Appendix A. We assume that the individual features in the descriptor are independent, and model the distribution of the i^{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 $A_g[l]$:

$$P_g(l, x) = \prod_{i=1\ldots21} \frac{1}{\sqrt{2\pi}\sigma_{l,i}} \exp\left(-\frac{(D_{g,i}(x) - \mu_{l,i})^2}{2\sigma_{l,i}^2} \right) \quad (3)$$

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^{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. We define a 7-dimensional descriptor $D_s(x, y)$ that describes the pairwise relationship of two nodes x and y. This descriptor is also described in detail in Appendix B. Intuitively, the descriptor captures support and vertical relationships, horizontal separation, and overlap 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 [Parzen 1962], 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^{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:

$$P_s(l_p, l_x, l_y, x, y) = \prod_{i=1\ldots7} \sum_{w \in W[l_p, l_x, l_y]} K_{w,i}\left(D_s(x, y)\right) \quad (4)$$

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

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 λ concatenates all labels that act as children of l across all scenes, generalizing from the individual observed productions. The derivation probability P_{nt} and cardinality probability P_{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.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.

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_{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_{leaf} as leaf nodes, assigning them special "object-subpart" labels from the grammar in the case of oversegmentation.

In the rest of this section, we formulate the objective function that is equivalent to maximizing $P(H|S, \mathcal{G})$ (Section 5.1), and propose an efficient dynamic programming algorithm to find the optimal hierarchy (Section 5.2).

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:

$$P(H|S, \mathcal{G}) \propto P(H|\mathcal{G}) \cdot P(S|H, \mathcal{G}). \tag{5}$$

$P(H|\mathcal{G})$ is the product of production probabilities of rules P_{prod} (Equation 2) in H:

$$P(H|\mathcal{G}) = \prod_{x \in H} P_{\text{prod}}(x)^{T(x)} \tag{6}$$

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:

$$P(S|H, \mathcal{G}) = \prod_{x \in H} P_g(x)^{T(x)} P_s^*(x)^{T(x)} \tag{7}$$

where the geometry term P_g is defined in Equation 3 and the full per-node spatial probability $P_s^*(x)$ is derived from the pairwise terms P_s (Equation 4):

$$\log P_s^*(x) = \frac{\sum_{p,q \in x.\text{children}} \log P_s(x.\text{label}, p.\text{label}, q.\text{label}, p, q)}{|x.\text{children}| \times (|x.\text{children}| - 1)} \tag{8}$$

Our final objective function is the negative logarithm of Equation 5:

$$\mathbf{E}(H) = \sum_{x \in H} E(x) \tag{9}$$

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

5.2 Algorithm

The main challenge in optimizing Equation 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 [2013] 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 [Zhao and Zhu 2013]) 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 this paper).

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

$$\begin{aligned} \mathbf{E}(H) &= \bar{E}(\mathbf{Root}(H)) \\ \bar{E}(x) &= E(x) + \sum_{y \in x.\text{children}} \bar{E}(y) \end{aligned} \tag{10}$$

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 [Earley 1970], 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 (Section 5.2.1). Next, it binarizes the grammar (Section 5.2.2). 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 (Section 5.2.3).

5.2.1 Proposing candidate groups

Given a scene S with a set of leaves S_{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_{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.

5.2.2 Grammar binarization

The goal of this step is to produce a grammar that is similar to the input grammar, but has a branching factor ≤ 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 $(l \to \lambda) \in \mathbf{R}$ into a set of rules with at most two right-hand labels:

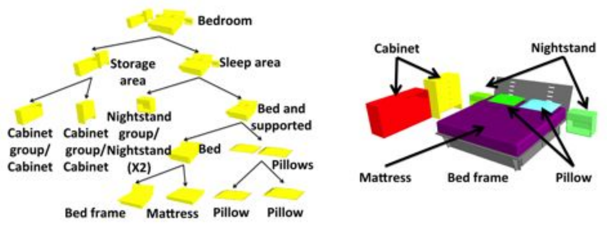
$$\begin{array}{lcll} l & \to & l'k & \text{for each } k \in \lambda \\ l & \to & jk & \text{for each } j,k \in \lambda \\ l & \to & k & \text{for each } k \in \lambda \\ l & \to & l'l' & \\ l' & \to & l'k & \text{for each } k \in \lambda \\ l' & \to & jk & \text{for each } j,k \in \lambda \\ l' & \to & l'l'. & \end{array} \tag{11}$$

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 in Section 5.2.3.

5.2.3 Dynamic programming

Now we describe an efficient dynamic programming algorithm for minimizing the Equation 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



(a) *Ground-truth hierarchy* **(b)** *Ground-truth leaf nodes*

Figure 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.*

equations as follows,

$$\begin{aligned} Q(x,l) &= \min\{Q_u(x,l), Q_b(x,l)\} \\ Q_u(x,l) &= \min_{l' \in \mathbf{RHS}(l)} E_2(x, l, \mathbf{K}(x, l')) + S(x, l') \\ Q_b(x,l) &= \min_{\substack{y,z \in \mathbf{Part}(x) \\ l_y, l_z \in \mathbf{RHS}(l)}} E_2(x, l, \mathbf{K}(y, l_y) \cup \mathbf{K}(z, l_z)) \\ &\quad + S(y, l_y) + S(z, l_z) \\ S(x,l) &= \sum_{[y, l_y] \in \mathbf{K}(x,l)} Q(y, l_y) \end{aligned} \tag{12}$$

where Q_u is the optimal energy of applying grammar rules with a single right-hand child ($l \to k$ in Equation 11), and Q_b is the optimal energy of applying grammar rules with two right-hand children (all other rules in Equation 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 not 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 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 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 $Q(y, l_y)$ 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.

6 Results

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 4a) and manually assigned a ground-truth parse for each input scene graph (Figure 4b). 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.

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 6 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 10 (a complete set of results can be found in supplemental materials). These examples show that our algorithm is able to create functionally 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 three bedroom scenes in the top row; and, it differentiates teacher areas from student desk areas in the classroom shown in the fourth 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.

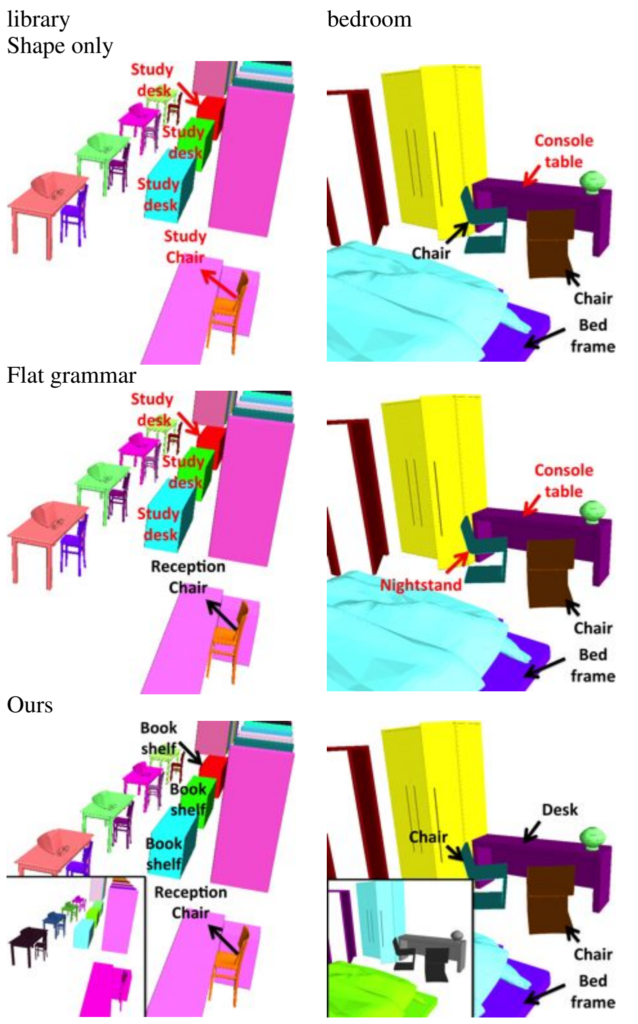


Figure 5: *Comparison to alternative methods. Classifying objects only by their geometry (first row) cannot differentiate between objects of similar shape in different categories, e.g. short bookshelf and study desk, or console table and study desk. Even if contextual information is leveraged, relations among objects can be wrongly interpreted (e.g. short book shelf and study chair (second row left), chair and bed (second row right)) 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 row show the object groups predicted by our method. Black labels are correct, and red labels are incorrect.*

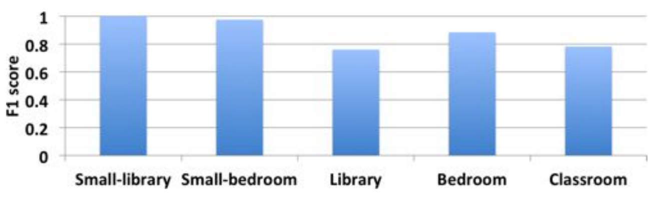


Figure 6: *Performance of object grouping. Our method achieves almost 100% on illustrative datasets, and ∼80% on Trimble 3D Warehouse scenes.*

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 representative of previous methods that leverage spatial context, but not hierarchy, for object classification [Fisher and Hanrahan 2010; Fisher et al. 2011; Fisher et al. 2012; Xu et al. 2013].

Results are shown in Figure 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 5 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).

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."

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.

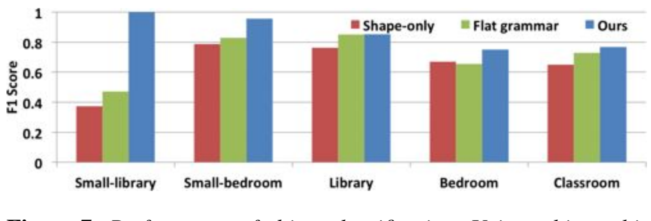


Figure 7: *Performance of object classification. Using a hierarchical grammar clearly outperforms alternatives.*

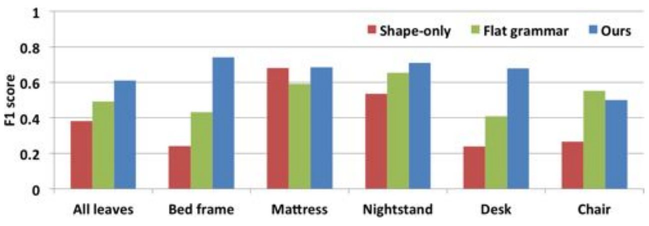


Figure 8: *Performance on over-segmented bedroom scenes. Our method significantly outperforms shape-only classification in most object categories except mattresses, which are rarely over-segmented, and can be distinguished from other classes based on their distinctive geometry. Our method outperforms the "flat" grammar, with spatial relations but no hierarchy, in all object categories except for chairs.*

Results of this experiment are shown in Figure 8, 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 [Xu et al. 2013] 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 (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 9. 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 (far right) 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 same study area). By leveraging hierarchy we can learn relations between objects that belong to the same group, and thus learn stronger layout priors.

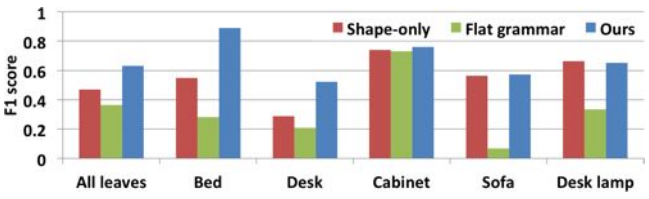


Figure 9: *Parsing scenes in the Sketch2Scene dataset [Xu et al. 2010]. We reuse the grammar learned in Section 6.2 to parse scenes in Sketch2Scene, and compare the performance to those of alternative methods. Using a flattened grammar is not effective because spatial relations are not discriminatory 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).*

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

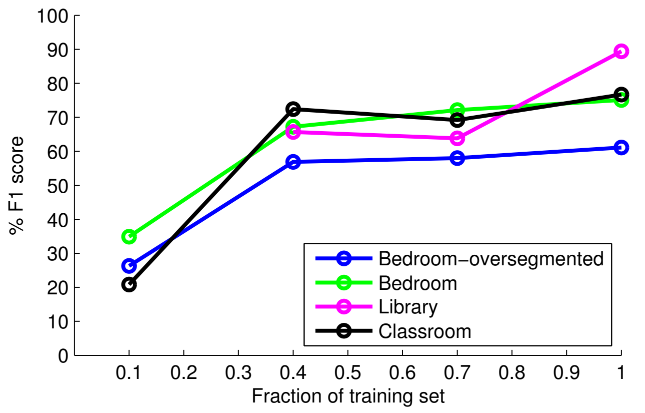


Figure 11: *Impact of size of training set. Labeling accuracy increases on all datasets with more training examples.*

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. The experiment result in Figure 13 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 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 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

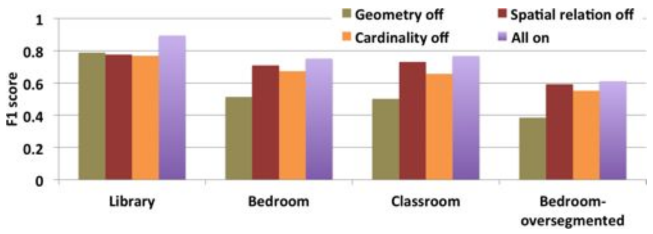


Figure 12: *Impact of individual energy terms on object classification. Each energy term contributes to the overall performance in each dataset.*

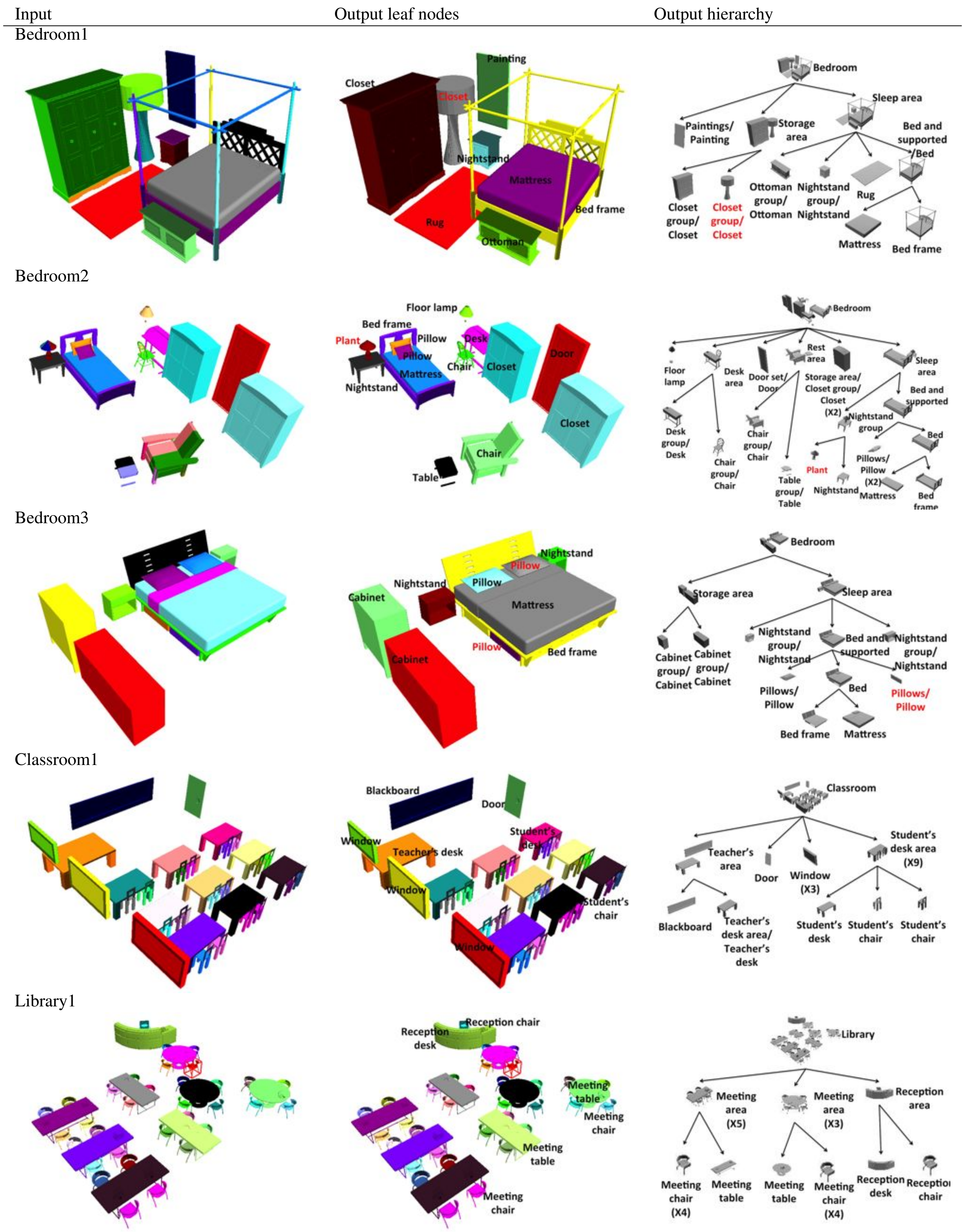


Figure 10: *Examples of parsing results. We show the leaf nodes of the input scene graph (column 1), and the leaf nodes (column 2) and hierarchy (column 3) output by our algorithm. Red labels indicate either wrong labels or incorrect segmentation. In column 3, to save space, we merge a child with its 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.*

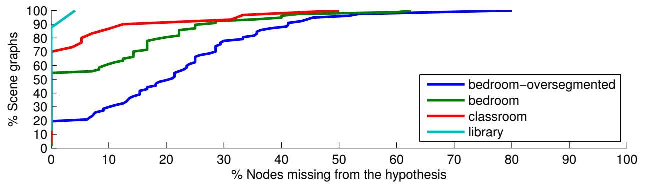


Figure 13: *Fraction of ground truth internal nodes missing from the predicted hierarchies. The Y-value of each point on a curve denotes the fraction of scenes in which the number of missing ground truth internal nodes is at most the X-value. For libraries, our algorithm successfully proposes all ground-truth nodes, except one, in the entire dataset. Oversegmented input scene graphs are in general more challenging for our method.*

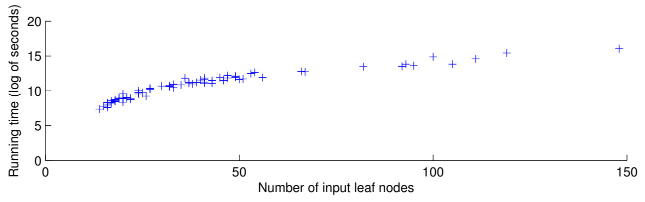


Figure 14: *Relationship between number of input leaf nodes and running time on oversegmented bedroom scene graphs. Our method scales reasonably well for complex scenes.*

(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 14 shows the result relating the number 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.

7 Conclusion

This paper 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 hierarchial 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 the paper 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, the paper focuses 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.

Acknowledgments

We acknowledge Kun Xu for distributing Sketch2Scene dataset. We thank Christiane Fellbaum, Stephen DiVerdi, and the anonymous reviewers for their comments and suggestions. The project was partially supported by the NSF (IIS-1251217, CCF-0937137), an ERC Starting Grant (StG-2013-335373), Intel (ISTC-VC), Google, and Adobe.

A Shape Descriptors for Geometry Attributes

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 $z_{\text{min}}, z_{\text{max}}, l_z = z_{\text{max}} - z_{\text{min}}, l_1 = \max(x_{\text{max}} - x_{\text{min}}, y_{\text{max}} - y_{\text{min}}), l_2 = \min(x_{\text{max}} - x_{\text{min}}, y_{\text{max}} - y_{\text{min}}), l_2/l_1, l_z/l_1, l_z/l_2$
- 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_{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.

B Layout Descriptors for Spatial Attributes

The relative layout of two nodes x and y is described with a 7-dimensional descriptor $D_s(x, y)$:

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

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.

References

BISHOP, C. M. 2006. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc.

BOKELOH, M., WAND, M., AND SEIDEL, H.-P. 2010. A connection between partial symmetry and inverse procedural modeling. *ACM Trans. Graph. 29*, 4, 104.

BOULCH, A., HOULLIER, S., MARLET, R., AND TOURNAIRE, O. 2013. Semantizing complex 3D scenes using constrained attribute grammars. In *Computer Graphics Forum*, vol. 32, Wiley Online Library, 33–42.

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

CHOI, W., CHAO, Y. W., PANTOFARU, C., AND SAVARESE, S. 2013. Understanding indoor scenes using 3D geometric phrases. In *CVPR*.

EARLEY, J. 1970. An efficient context-free parsing algorithm. *Communications of the ACM 13*, 2, 94–102.

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

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

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

GOLOVINSKIY, A., AND FUNKHOUSER, T. 2009. Consistent segmentation of 3D models. *Computers & Graphics 33*, 3, 262–269.

HU, R., FAN, L., AND LIU, L. 2012. Co-segmentation of 3D shapes via subspace clustering. In *Computer Graphics Forum*, vol. 31, Wiley Online Library, 1703–1713.

HUANG, Q.-X., AND GUIBAS, L. 2013. Consistent shape maps via semidefinite programming. In *Computer Graphics Forum*, vol. 32, Wiley Online Library, 177–186.

HUANG, Q., KOLTUN, V., AND GUIBAS, L. 2011. Joint shape segmentation with linear programming. In *ACM Trans. Graph.*, vol. 30, ACM, 125.

HUANG, Q.-X., ZHANG, G.-X., GAO, L., HU, S.-M., BUTSCHER, A., AND GUIBAS, L. 2012. An optimization approach for extracting and encoding consistent maps in a shape collection. *ACM Trans. Graph. 31*, 6, 167.

KALOGERAKIS, E., HERTZMANN, A., AND SINGH, K. 2010. Learning 3D mesh segmentation and labeling. In *SIGGRAPH*.

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

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

KIM, V. G., LI, W., MITRA, N. J., CHAUDHURI, S., DIVERDI, S., AND FUNKHOUSER, T. 2013. Learning part-based templates from large collections of 3D shapes. *ACM Trans. Graph.*.

MARTINOVIĆ, A., AND VAN GOOL, L. 2013. Bayesian grammar learning for inverse procedural modeling. In *CVPR*.

MATHIAS, M., MARTINOVIC, A., WEISSENBERG, J., AND VAN GOOL, L. 2011. Procedural 3D building reconstruction using shape grammars and detectors. In *3DIMPVT*.

NGUYEN, A., BEN-CHEN, M., WELNICKA, K., YE, Y., AND GUIBAS, L. 2011. An optimization approach to improving collections of shape maps. In *CGF*, vol. 30, 1481–1491.

PARZEN, E. 1962. On estimation of a probability density function and mode. *Ann. Math. Stat. 33*, 3, 1065–1076.

SIDI, O., VAN KAICK, O., KLEIMAN, Y., ZHANG, H., AND COHEN-OR, D. 2011. Unsupervised co-segmentation of a set of shapes via descriptor-space spectral clustering. In *ACM Trans. Graph.*, vol. 30, ACM, 126.

SOCHER, R., LIN, C. C., NG, A., AND MANNING, C. 2011. Parsing natural scenes and natural language with recursive neural networks. In *ICML*, 129–136.

ŠT'AVA, O., BENEŠ, B., MĚCH, R., ALIAGA, D. G., AND KRIŠTOF, P. 2010. Inverse procedural modeling by automatic generation of L-systems. In *Computer Graphics Forum*, vol. 29, Wiley Online Library, 665–674.

TALTON, J., YANG, L., KUMAR, R., LIM, M., GOODMAN, N., AND MĚCH, R. 2012. Learning design patterns with bayesian grammar induction. In *UIST*, ACM, 63–74.

TEBOUL, O., KOKKINOS, I., SIMON, L., KOUTSOURAKIS, P., AND PARAGIOS, N. 2013. Parsing facades with shape grammars and reinforcement learning. *Trans. PAMI 35*, 7, 1744–1756.

TRIMBLE, 2012. Trimble 3D warehouse, http://sketchup.google.com/3Dwarehouse/.

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

WANG, Y., XU, K., LI, J., ZHANG, H., SHAMIR, A., LIU, L., CHENG, Z., AND XIONG, Y. 2011. Symmetry hierarchy of man-made objects. In *Computer Graphics Forum*, vol. 30, Wiley Online Library, 287–296.

WU, F., YAN, D.-M., DONG, W., ZHANG, X., AND WONKA, P. 2014. Inverse procedural modeling of facade layouts. *ACM Trans. Graph. 33*, 4.

XU, K., CHEN, K., FU, H., SUN, W.-L., AND HU, S.-M. 2013. Sketch2Scene: sketch-based co-retrieval and co-placement of 3D models. *ACM Trans. Graph. 32*, 4, 123:1–123:12.

XU, K., MA, R., ZHANG, H., ZHU, C., SHAMIR, A., COHEN-OR, D., AND HUANG, H. 2014. Organizing heterogeneous scene collection through contextual focal points. *ACM Transactions on Graphics, (Proc. of SIGGRAPH 2014) 33*, 4, to appear.

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

ZHANG, H., XU, K., JIANG, W., LIN, J., COHEN-OR, D., AND CHEN, B. 2013. Layered analysis of irregular facades via symmetry maximization. *ACM Trans. Graph. 32*, 4, 121.

ZHAO, Y., AND ZHU, S.-C. 2013. Scene parsing by integrating function, geometry and appearance models. CVPR.

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