Sub-Scene Level Analysis and Synthesis of 3D Indoor Scenes

by

Rui Ma

M.Sc., Jilin University, 2012
B.Sc., Jilin University, 2010

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Doctor of Philosophy

in the
School of Computing Science
Faculty of Applied Science

SIMON FRASER UNIVERSITY
Fall 2017

Approval

Name:	**Rui Ma**
Degree:	**Doctor of Philosophy (Computing Science)**
Title:	***Sub-Scene Level Analysis and Synthesis of 3D Indoor Scenes***
Examining Committee:	**Chair:** Dr. KangKang Yin Associate Professor

Dr. Hao (Richard) Zhang
Senior Supervisor
Professor _____

Dr. Ze-Nian Li
Supervisor
Professor _____

Dr. Yasutaka Furukawa
Internal Examiner
Assistant Professor
School of Computing Science _____

Dr. Oliver Deussen
External Examiner
Professor
Department of Computer and
Information Science
University of Konstanz _____

Date Defended: 13 September 2017

Abstract

3D indoor scenes are ubiquitous in computer graphics applications such as 3D games and interior design. With the emerging applications in VR/AR, there is an increasing demand of realistic 3D scene data. However, designing 3D indoor scenes requires proficient 3D modeling skills and is often time-consuming. A promising solution to the content-creation bottleneck of scenes is to utilize the existing scene data for data-driven 3D scene generation. Recent research about data-driven indoor scene processing in computer graphics usually takes a holistic view and operates at the object-level for scene analysis and synthesis. The main limitation of existing methods is their applicability to characterizing and modeling complex scenes. In this thesis, we address the problems of data-driven 3D indoor scene analysis and synthesis via sub-scene level processing. Our goal is to improve the understanding of scene structures through sub-scene level analysis and develop efficient systems to create complex scenes by manipulating sub-scenes instead of individual objects.

To this end, we first introduce *focal points*, the representative sub-scenes, for characterizing, comparing, and organizing collections of complex and heterogeneous data, and apply the developed concepts and algorithms to collections of 3D indoor scenes. Then, we propose a framework for action-driven evolution of 3D indoor scenes. Human actions learned from annotated photographs are applied to trigger appropriate object placements at a sub-scene level, inducing a more compact way of scene generation. Finally, we present a novel framework that uses natural language to generate 3D indoor scenes. We demonstrate advantages of focal-centric scene comparison and organization over existing approaches. We show results of our action-driven and language-driven scene synthesis that lead to realistic, messy and complex 3D scenes, and evaluate the plausibility and naturalness of the scenes by user studies.

Keywords: 3D indoor scenes; sub-scene processing; scene analysis; scene synthesis; data-driven; action model; natural language processing

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

3D indoor scenes composed of precise 3D CAD models are ubiquitous in computer graphics. In 3D games, the virtual characters perform actions to explore and interact with the 3D indoor environment. In the field of interior design, artists create 3D indoor scenes to show their design ideas to customers. With the recent advance of the VR/AR technologies and their emerging applications, there is an increasing demand of realistic 3D scene data. For example, in AR, a user can view a 3D scene rendered on the online capture of the real world environment to help with the interior design and furniture layout.

Currently, most existing 3D indoor scenes in online repositories are manually created using general 3D modeling softwares such as 3DS Max and SketchUp, or professional indoor scene modeling tools, e.g., Autodesk Homestyler [5], Sweet Home 3D [3], or Planner 5D [2]. A traditional modeling process includes a user iteratively inserts individual objects into the 3D scene and adjusts their locations by mouse and keyboard operations. Creating a scene by these 3D modeling softwares often requires proficient 3D designing and modeling skills, while designing detailed and complex 3D indoor scenes requires even more expertise and is very time-consuming.

Data-driven indoor scene processing Comparing to manually designing the 3D indoor scenes, a promising solution to address the content-creation bottleneck of indoor scenes is to use the existing scene data for data-driven scene generation. Current online scene repositories usually contain large-scale and heterogeneous scene collections, such as 3D Warehouse [1]. To utilize scenes in these databases, one problem is to efficiently organize and explore a collection of scenes so that we could have a better understanding of the structure of the scene collection and quickly find the relevant scenes for data-driven scene creation. The other problem is to develop efficient algorithms that learn and apply knowledge from scene databases to generate realistic and complex scenes for graphics and VR/AR applications.

1.1 Sub-scene level indoor scene processing

This thesis address the above two problems of data-driven indoor scene processing by performing *sub-scene level* analysis and synthesis of 3D indoor scenes. Our goal is to improve the understanding of scene structures through sub-scene level analysis and develop efficient systems to create complex scenes by manipulating sub-scenes instead of individual objects.

Why using sub-scenes? There is substantial amount of recent work in graphics on geometry and structure analysis of 3D indoor scenes and synthesis of plausible and novel scenes using the data-driven approach. However, existing methods usually take a *holistic* view and operate at the *object-level* for scene analysis [23] and synthesis [22, 12]. As a result, they cannot achieve satisfied performance on characterizing and modeling complex scenes. In contrast, our scene processing at the sub-scene level is more flexible and could provide more comprehensive scene characterization and produce more complex scene synthesis results. Moreover, the sub-scenes we work with are related to the functional substructures of scenes. For example, our scene synthesis discovers the functional substructure like a bed with two nightstands as a representative sub-scene for the bedrooms; our synthesis utilizes sub-scenes involved with human actions to generate functionally plausible results. Scene analysis and synthesis using such sub-scenes allow us to perform function-aware scene processing.

Scene analysis. The goal of scene analysis is to process the existing 3D scenes in the databases and learn the knowledge and constraints that could be used for arranging objects in new scenes, i.e., scene synthesis. To perform scene analysis, a common first step is to convert the scene into a graph representation which captures the relationships between the objects in the scene. Fisher et al. [23] propose to represent a scene as a relationship graph and define a graph kernel to compare 3D scenes. By considering the geometry features and semantic labels of objects encoded in the nodes and structural relationships encoded in the edges, the graph kernel works well for characterizing 3D scenes and largely improves the performance of scene retrieval.

However, as the graph kernel accumulates the similarities between every object pair in two scenes, it takes a holistic view for scene comparison and the performance will be reduced if the scenes are in large scale and complex, e.g., a scene with several functional regions or belongs to multiple semantic categories. Analyzing complex and heterogeneous data is difficult without references to certain points of attention or focus, i.e., the focal points. For example, comparing New York City to Paris as a whole will unlikely yield a useful answer. The comparison is a lot more meaningful if it is focused on particular aspects of the cities, e.g., architectural style or fashion trends. For complex indoor scenes, a scene characterization measure which focuses more on the interested regions in a scene and operates on the sub-scene level will produce better analysis results (Figure 1.1).

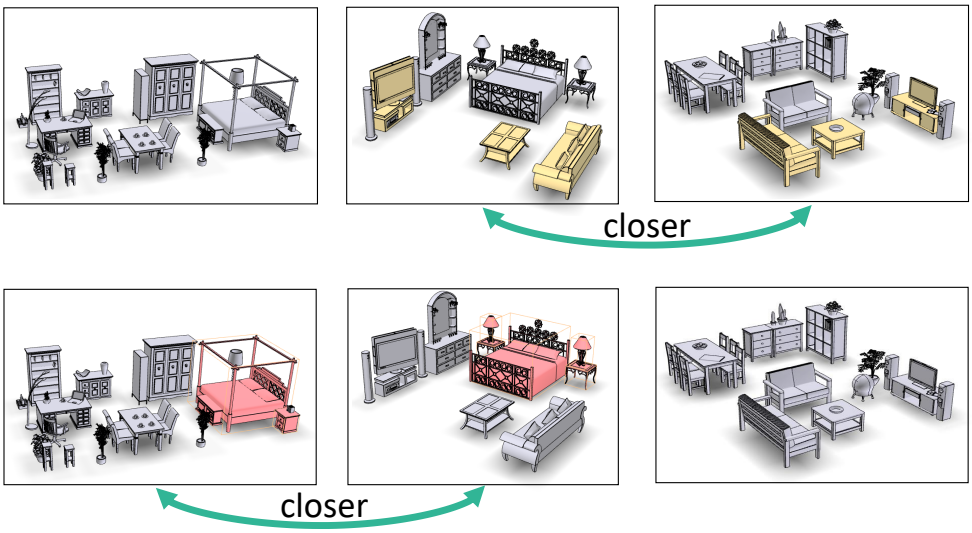


Figure 1.1: Comparing complex scenes using focal points, the representative sub-scenes of a scene collection. Given same set of scenes in the first and second row, different scene similarities are yielded by choosing different focal points (highlighted sub-scenes).

Therefore, we propose to use *focal points*, the frequent sub-scenes in a scene collection to characterize and compare complex scenes, and organize the scene collection. We develop a co-analysis algorithm to extract the contextually representative focal points by interleaving with frequent pattern mining and scene clustering. By applying the extracted focal points for scene analysis, we are able to efficiently organize and explore a large scene collection with heterogeneous scenes.

Scene synthesis. Many efforts have been made to develop algorithms for automatic scene generation and design intelligent tools to assist interactive scene modeling. Among various scene synthesis works, the example-based method from Fisher et al. [22] produces the most intriguing results by applying probabilistic models for object arrangements learned from both the example and database 3D scenes. As the object distributions are trained for the whole scene, the example-based synthesis is only evaluated with small-scale scenes, such as arrangements on or around an office desk. For large-scale scenes, learning the distributions of all the objects will be intractable. Therefore, it is better to learn object distributions for selected objects in a sub-scene, e.g., objects related to some human action or activity.

The recent work activity-centric scene synthesis [21] learns activity models which encode the distributions of objects involved in specific activities and performs functional scene modeling from indoor scans based on the predicted activities. The activity models offer a more compact view of object arrangements and result in semantic reconstruction of large-scale scenes. Instead of targeting at reconstruction or modeling from indoor scans, we propose to use human actions to evolve a given scene. This idea is motivated by how real life scenes are generated. When we look at the real world indoor scenes around us, they

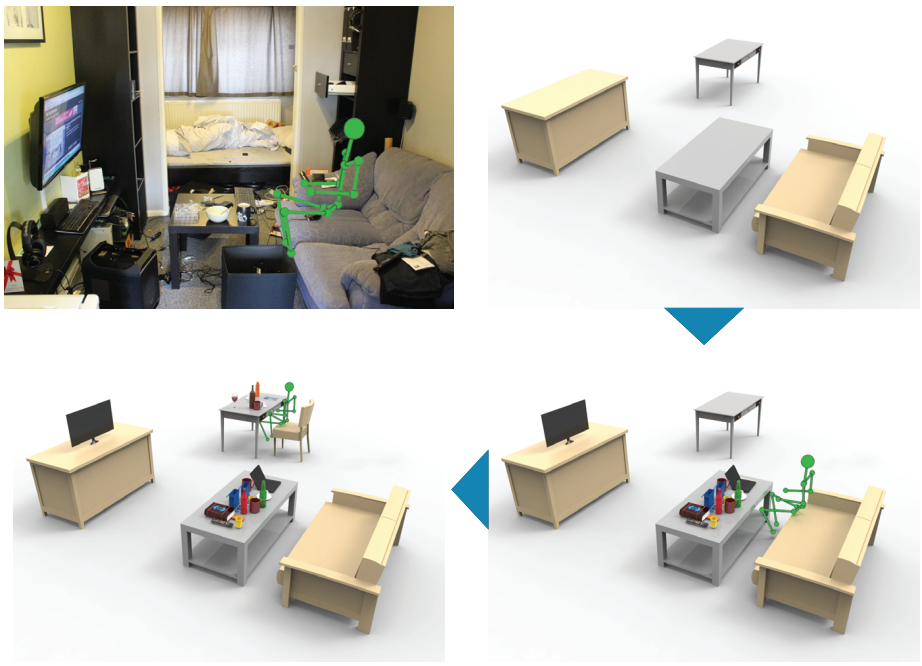


Figure 1.2: Real life scenes (top-left) are generated by human actions. We take this idea and use actions to evolve an initial 3D scene (top-right) by progressively adding objects of sub-scenes. A continuous scene evolution sequence with realistic and messy 3D scenes are generated (second row, from right to left).

are not static environments. The scenes evolve over time and can reach a high complexity and messiness, driven by object movements at the sub-scene level resulting from human actions. When generating 3D scenes, it is also natural to use an action-driven manner to progressively create and alter arrangements of objects in sub-scenes (Figure 1.2).

We define an action model as the combination of information about human poses, object categories and their spatial configurations which summarize the object-object and object-human relations for the action. Applying actions to an initial scene will trigger appropriate object placements including relocation of existing objects or insertion of new objects into the scene, which will produce a continuous scene evolution with realistic and messy scenes at the end.

The other notable line of work for scene synthesis is the text-to-scene generation. Generating 3D scenes from text has been a long and on-going pursuit since the pioneering work WordsEye [13], which maps explicit scene arrangement languages to object placements in a scene. The recent work of Chang et al. [12] improves the text-to-scene generation by utilizing the spatial and the common sense knowledge learned from 3D scene database to infer the unstated facts from the input text. As these systems work on arranging individual objects based on the text, synthesizing complex scenes is tedious and difficult due to the number of objects and the inherent ambiguity of language.

Figure 1.3: Using natural languages, we manipulate sub-scenes to efficiently generate and edit 3D indoor scenes in an interactive manner.

Thus, we propose to develop a text-to-scene framework that aims to create complex 3D scenes using compact and high-level natural language commands (Figure 1.3). We focus on language-driven scene synthesis and editing at the sub-scene level rather than the object-level. We learn how to manipulate the sub-scenes by analyzing available 3D scenes from the database and perform the sub-scene level modeling to improve the efficiency of text-driven synthesis for generating complex results.

1.2 Challenges

There are several challenges of executing the idea of sub-scene level processing for both scene analysis and synthesis. The first challenge is to extract the *representative* sub-scenes as focal points from a heterogeneous scene collection and apply them for scene comparison and organization. The second challenge is to learn the action models from appropriate and sufficient data for the proposed action-driven scene evolution. The last challenge is to build a mapping between the natural scene modeling language and arrangements of complex scenes or sub-scenes for language-driven scene synthesis.

Extracting contextual focal points from heterogeneous scenes. For focal-centric scene analysis, by representing an indoor scene by a graph of its constituent objects, we define a focal point as a sub-scene or a substructure in a scene which corresponds to a subgraph. However, we are not interested in all sub-scenes. To analyze and organize a heterogeneous scene collection which contains scenes belonging to multiple categories, we are only interested in the representative focal points in the given scene collection. For a focal to be representative, it must occur frequently. However, extracting focal points by frequency analysis alone is not enough. For example, chairs are likely to be found in almost all scenes, but they can hardly be regarded as representative of any meaningful scene group, e.g., bedrooms or living rooms.

Based on the above observation, a representative focal point must also be discriminative so that it could be used for characterizing the nature of scenes, especially for the scenes with complex structures. Moreover, the representativity is related to a notion of coherence or compactness of the group of scenes the focal point is to represent or characterize. This means focal points should be extracted from a group of similar scenes, i.e., a scene clustering result. Given a scene collection, focal extraction is coupled with scene clustering, each of which is unknown and difficult to solve without information from the other problem.

Action learning from appropriate training data. A key question facing any data-driven approach is the choice of the data. To learn the action models for action-driven scene generation, 3D data of human actions and human-object interactions are most directly applicable. However, acquiring such data in large volume is costly with challenges from reconstruction, tracking, and annotation. Annotating existing 3D scenes as in [21] is an option, but such scenes are limited in number and variety, and they were mostly designed without human presence or intimate connections to human actions.

One solution to this data limitation is to use the vast source of *photographs* of indoor scenes with daily human activities. The Microsoft COCO (Common Objects in Context) database [53] offers a solid baseline for our data requirement: a large number of photos with object segmentations, labels, and text captions describing the contents, including human actions, in each photo. Yet, to learn our action model, much information about human poses and inter-object relations is still missing. Recovering necessary 3D action data to drive 3D scene synthesis is a challenging problem in general.

Mapping natural language to complex scenes. Using natural language to generate complex scenes is challenging because it is tedious and redundant to describe all the details of the expected scenes. Instead of asking the user to provide explicit commands to affect every single object, incorporating the implicit or common knowledge from the text input for object arrangements will reduce the language redundancy and improve the efficiency of scene modeling. 3D scene database provides a rich knowledge source for learning the object relationships and the text-to-scene mapping. However, there lacks an efficient model which summarizes the arrangement of object groups or sub-scenes w.r.t scene descriptions in natural language. We define and build the relational models from annotated 3D scene databases to characterize semantic relationships among objects. The main challenge is how to represent and extract the semantic relationships from 3D scene data.

Applying the relational models is also difficult due to the ambiguity of natural language and the missing of a canonical scene representation which connects the 3D scenes and the scene descriptions in natural language. To enable the mapping between natural language to 3D scenes, we propose the Semantic Scene Graph (SSG) as a uniform scene representation. The problems then become how to convert the natural language and 3D scenes into

consistent SSGs and how to utilize the SSGs and the learned relational models to create complex 3D scenes.

1.3 Contributions

In this thesis, we address the challenges for analysis and synthesis of 3D indoor scenes using the sub-scene level processing. By working at the sub-scene level, we improve the performance of scene retrieval and enable new applications for scene analysis, such as scene organization and exploration. We also develop efficient frameworks that utilize probabilistic models of sub-scenes for scene synthesis. Our main contributions are as follows:

- We introduce the focal points to characterize and organize complex 3D indoor scenes. The focal-centric similarity provides new perspective for comparing complex scenes or data in other forms.
- We present a co-analysis algorithm to extract the contextual and representative focal points from heterogeneous scene collections.
- We propose a novel framework for action-driven scene evolution. Action models of human-object relations are learned from annotated photographs and applied to generate continuous sequences of realistic 3D scenes.
- We develop a novel and efficient interactive system for language-driven scene synthesis. A language-driven "retrieve-and-accommodate" scheme is adopted to manipulate sub-scenes for progressive generation of complex scenes.
- We propose the Semantic Scene Graph as a uniform scene representation for text-to-scene mapping and learn relational models from 3D scene databases to encode semantic relationships of objects.

1.4 Thesis organization

This thesis is organized in the following way: in Chapter 2, we first review the background of indoor scene analysis and synthesis, then investigate the substructure level processing of 3D objects and the related scene processing work that involves sub-scenes. In Chapter 3, we develop a co-analysis algorithm to extract focal points from scene collections and show results of focal-based scene organization, retrieval and exploration. In Chapter 4, we introduce a novel framework of action-driven action evolution, describing how to learn action models from photographs and how to apply the learned actions to generate continuous series of realistic 3D scenes. In Chapter 5, we propose an interactive system for using natural language to generate and edit 3D indoor scenes, and show results of complex scenes generated

with compact natural language sentences. Finally, in Chapter 6, we summarize this thesis with contributions and present several directions for future work.

Related publications. This thesis includes previously published material and a finished work which is currently in submission. The following is a list of the papers and their corresponding chapters:

- The survey of related work presented in Chapter 2 is based on the technical report: Rui Ma. Analysis and Modeling of 3D Indoor Scenes. *AxXiv e-prints (SFU-CMPT TR 2017-55-3)*, 2017. [57]

- The focal-based scene co-analysis approach in Chapter 3 appeared in the paper: Kai Xu, Rui Ma, Hao Zhang, Chenyang Zhu, Ariel Shamir, Daniel Cohen-Or, Hui Huang. Organizing Heterogeneous Scene Collections through Contextual Focal Points. *ACM Transactions on Graphics*, 2014. [103]

- The action-driven scene evolution framework presented in Chapter 4 appeared in the paper: Rui Ma, Honghua Li, Changqing Zou, Zicheng Liao, Xin Tong, Hao Zhang. Action-Driven 3D Indoor Scene Evolution. *ACM Transactions on Graphics*, 2016. [58]

- The language-driven scene synthesis system described in Chapter 5 appeared in the paper: Rui Ma, Matthew Fisher, Soeren Pirk, Binh-Son Hua, Sai-Kit Yeung, Xin Tong, Leonidas Guibas, Hao Zhang. Language-Driven Synthesis of 3D Scenes from Scene Databases. *ACM Transactions on Graphics*, 2017 (in submission).

Chapter 2

Background

In this chapter, we first summarize recent works on analysis and synthesis of 3D indoor scenes. Then, we review methods that exploit substructures for shape and scene processing. Specifically, we start by introducing the graph-based scene representation and then explore the problems and applications for scene analysis. Next, we survey various techniques for 3D indoor scene modeling and synthesis, and study in detail the human-centric scene modeling and text-based scene generation. Lastly, we investigate the contextual-based shape analysis and substructure-based shape modeling as well as the related scene processing work that involves sub-scenes.

2.1 3D scene analysis

In the following we first introduce a representative graph-based scene representation which encodes structural relationships between objects [23]. Then, we study scene representations that could encode complex object relationships and their applications in 3D scene analysis [115, 84]. In the end, we survey techniques for co-analysis and processing of a collection of scenes [103, 55].

Structural scene representation. The work of Fisher et al. [23] takes a 3D scene and transforms it into a *relationship graph*, a representation which encodes structural relationships between objects, such as support, contact or enclosure (Figure 2.1 left). The nodes of a relationship graph correspond to meaningful objects in the scene and the edges represent the relationship between these objects. Representing scenes as relationship graphs greatly facilitates comparing scenes and their structures (Figure 2.1 right). A graph kernel is defined for comparison of two relationship graphs: similarities between the graph nodes and edges are computed and accumulated to produce an overall similarity of two graphs. The limitation of graph kernel is that it compares two scenes globally. Thus, it is difficult to disambiguate two complex scenes using the graph kernel.

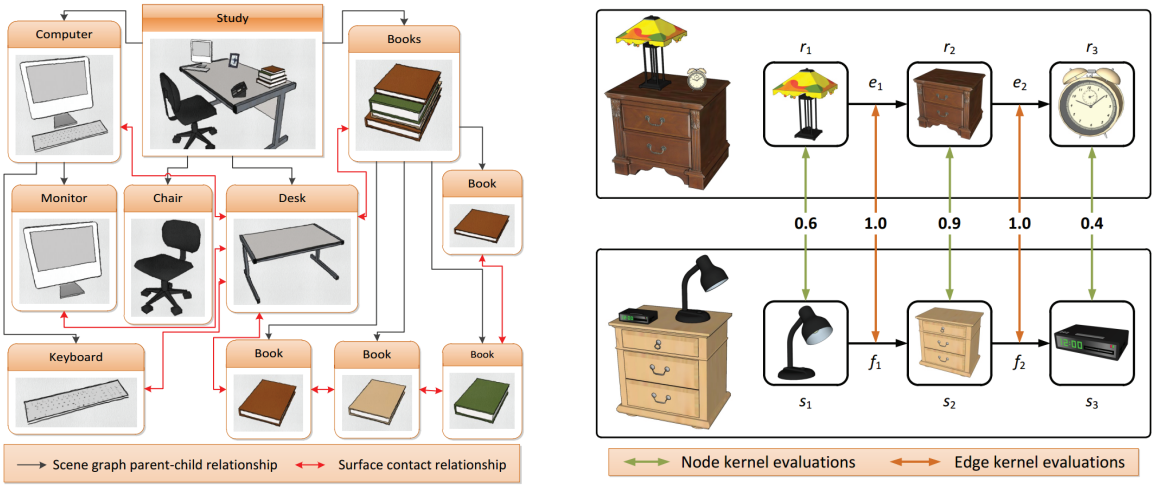


Figure 2.1: Graph-based scene representation and comparison in Fisher et al. [23]: a scene and its representation as a relationship graph (left); comparisons of two graph walks (right).

Characterization of complex relationships. To further represent more complex relationships between objects, Zhao et al. [115] propose a new relationship descriptor, Interaction Bisector Surface (IBS), to describe both topological relationships such as whether an object is "wrapped in", "linked to" or "tangled with" others, and geometric relationships such as the distance between objects. Automatic construction of scene hierarchies and content-based relationship retrieval are studied using the IBS-based measures. Sharf et al. [84] relate object functionalities to a set of predominant motions that can be performed by the object or its subparts. These motions are denoted as *mobility*, which defines the specific degrees of freedom, types, axes and limits of motions. A set of sophisticated controllers which allow semantical editing operations are defined based on the detected mobilities and used for high-level scene manipulation. The complex relationships captured by IBS and the mobility are useful to characterize and analyze scenes with complex geometry and structures. It remains a problem of how to generate complex scenes based on such relationships.

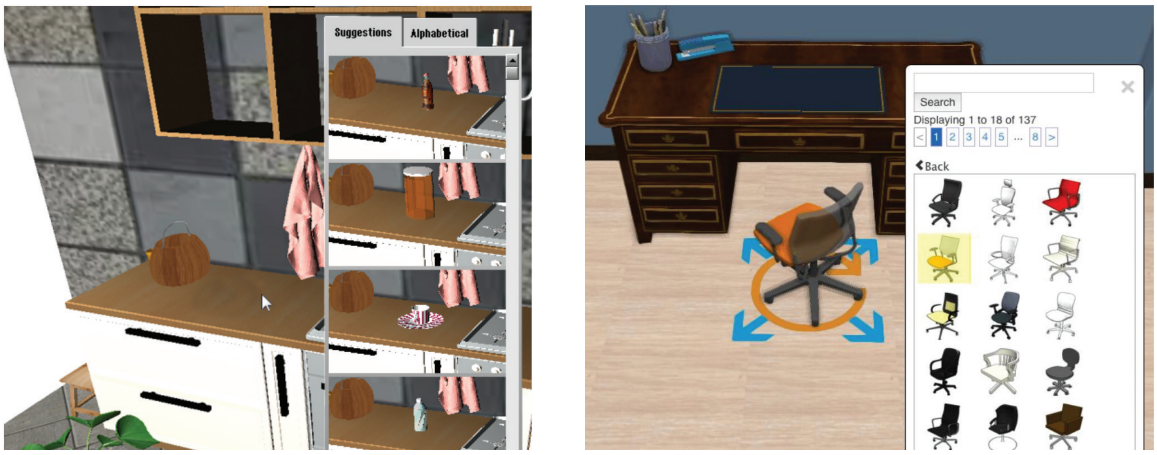
Consistent scene graphs. Although most of 3D scenes downloaded from online scene databases, e.g., 3D Warehouse, are accompanied with scene graphs which are generated during the scene design process, the graphs usually lack consistent and semantic object segmentations and category labels. Pre-processing to consolidate the initial scene graphs are performed in [23] so that the graph nodes represent meaningful objects, but consistent scene hierarchies (e.g., functional groups) are not guaranteed since such information must be inferred from multiple scenes. Liu et al. [55] develop algorithms that build a consistent representation for the hierarchical decomposition of a scene into semantic components. Given a collection of consistently-annotated scene graphs representing a category of scenes (e.g., bedroom, library, classroom, etc.) as the training set, they learn a probabilistic

hierarchical grammar that captures the scene structure and apply the learned grammar to hierarchically segment and label novel scenes and create consistent scene graphs. As the grammar needs to be learned from scenes of the same categories and with similar structures, it is hard to extract the grammar from the existing online scene collections which contain heterogeneous scenes.

2.2 3D scene modeling and synthesis

To resolve the bottleneck of content-creation for indoor scenes, many efforts have been made in the graphics and vision communities to develop intelligent tools and methods for 3D scene modeling and synthesis [57]. In this section, we first review suggestive tools developed for interactive scene modeling [20, 111, 75]. Next, we study different modeling techniques for creating 3D scenes [107, 22] and furniture layouts [110, 63]. Then, we introduce human-centric 3D scene modeling which exploits the human-object context for scene analysis [77] and synthesis [21, 58]. Finally, we review existing works for text-to-scene generation [17, 12, 9].

Suggestive scene modeling. Early scene modeling work by Xu et al. [106] used a set of intuitive placement constraints such as non-interpenetration of objects and object stability to allow the simultaneous manipulation of a large number of objects. Recent user-centric systems have explored the interactive context-based model suggestion [20, 111, 75], by which scene modeling could be simplified as a set of point-and-click operations in the suggestive modeling interfaces (Figure 2.2).



ClutterPalette SceneSuggest

Figure 2.2: Suggestive scene modeling interfaces of ClutterPalette [111] and SceneSuggest [75].

To make appropriate model suggestions, different constraints are extracted from existing database. Fisher and Hanrahan [20] learn object co-occurrences and their spatial relationships from existing 3D scenes and rank each object in the database according to how well it fits to a query box given by the user as well as its relationships to the context objects. The ClutterPalette tool from Yu et al. [111] uses the co-occurrences of objects extracted from the NYU Depth Dataset [88] to provide suggestions based on objects that have already been added into the scene. The recent system SceneSuggest presented by Savva et al. [75] learns continuous distributions of object position and orientation from existing 3D scenes [22] and combines the priors with the context of a user specified region to suggest a list of relevant 3D models. In these works, only individual objects are suggested for scene modeling and creating a complex scene with tens to hundreds of objects will need quite a lot of operations.

Scene modeling from X. Data-driven scene modeling or reconstruction from X has also gained much interest lately where X could be a sketch [107], a photograph [56, 39], or indoor scans [66, 47, 82, 14], just to name a few. In these cases, X provides inspirations and a target for scene modeling (Figure 2.3 first row). The objects and their arrangements are inferred from X to guide the retrieval and placement of suitable 3D objects from a model repository (Figure 2.3 second row).

Sketch-based user interface is commonly adopted for intuitive content creation. For sketch-based scene modeling, early works [87, 51] typically repeat the following process for individual objects one by one: first input a 2D sketch of an object, then retrieve and place a 3D model that best matches the input sketch. Focusing on joint processing of a set of

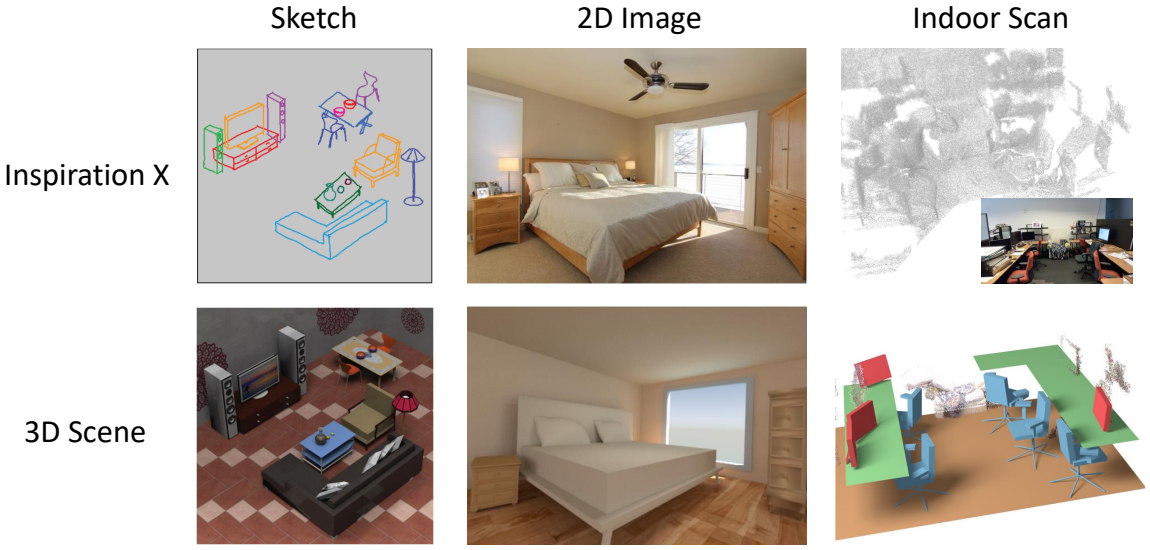


Figure 2.3: Different inspirations X (first row) are used to create 3D scenes (second row). Scene modeling results are produced by Xu et al. [107] (left), Izadinia et al. [39] (middle) and Kim et al. [47] (right).

sketched objects, Xu et al. [107] present Sketch2Scene, a framework that automatically turns a sketch inferring multiple scene objects to semantically valid, well arranged scenes of 3D models by performing *co-retrieval* and *co-placement* of a set of relevant 3D models.

Comparing to sketches, 2D images and photographs of indoor scenes are more accessible and contain more rich object compositions which could inspire the 3D scene modeling or reconstruction. 2D image to scene generation is recently tackled in [56, 39]. Given a single 2D image depicting an indoor scene as the input, Liu et al. [56] reconstruct a 3D scene in two stages: image analysis which obtains object segmentations and their 3D location using [50]; 3D model retrieval based on matching the line drawings extracted from 2D objects and 3D models. Using the state-of-the-art object recognition and scene understanding algorithms, Izadinia et al. [39] present IM2CAD which automatically produces high-quality scene modeling results on challenging photos from interior home design websites. Instead of using handcrafted features as in [56], the IM2CAD leverages deep features trained by convolutional neural nets (CNNs) [48] to reliably match between photographs and CAD renderings extracted from ShapeNet [10].

Creating 3D indoor scenes as the digital replica of our living environments is a common interest of computer vision and graphics. There has been a great deal of work on 3D indoor scene reconstruction and modeling which takes as input one or more images or depth scans and aims to reconstruct the captured scene geometry or semantics. To address the problem of missing data and scanning noise from cluttered indoor scans, data-driven and model-based approaches are proposed to learn the prior knowledge of objects from the scene or 3D model databases for semantic scene understanding and modeling [66, 47, 82, 14]. Unlike the model-based methods which require a training phase, unsupervised methods are also studied to use the intrinsic geometry properties [46] , the repetition of indoor objects [62], or the physical stability of object arrangements [81], to identify and reconstruct the objects from the indoor scans.

Recently, online scene understanding and modeling during the process of real-time scene scanning is becoming more popular. Object retrieval from shape databases [52] and structure analysis [113] are implemented for real-time scene modeling. As detailed scene scanning by human is laborious, especially for large indoor scenes containing numerous objects, Xu et al. [102] propose a framework for robot-operated autonomous scene scanning or autoscanning. The basic system setup is a mobile robot holding a depth camera performing real-time scene reconstruction [67]. The autoscanning and reconstruction is object-aware, guided by object-centric analysis which incorporates online learning to the task of object segmentation. In the follow-up work of [104], the problem of autonomously exploring unknown objects in a scene by robot-operated consecutive depth acquisitions is addressed. They propose a 3D Attention Model to select the next-best-views (NBVs) for depth acquisition around an object of interest and conduct part-based recognition to tolerate occlusion.

As these works on scene modeling from X focus on creating 3D scenes with the geometry and semantics constrained by the inspiration X, they are not targeted at generating novel scene arrangements. Also, due to various limitations, e.g., the complexity of the input sketches, the intrinsic problem of 3D geometry estimation from 2D images and the quality of the indoor scans, it is challenging to reconstruct very complex and detailed 3D scenes from X.

Furniture-layout optimization. 3D scenes can also be generated by furniture layout optimization [27, 63, 110] for a given room with a given set of furniture or objects. The common problem is to automatically generate object arrangements that satisfy a set of indoor scene layout constraints or rules. Germer and Schwarz [27] arrange a room by procedurally placing objects constrained by manually specified room semantics and furniture layout rules. Merrell et al. [63] present an interactive furniture layout system that assists users by suggesting furniture arrangements that are based on interior design guidelines. The guidelines are encoded as terms in a density function and layout suggestions are generated by sampling this function while respecting user's constraints. Similar to [63], the *Make it Home* system of Yu et al. [110] encodes spatial relationships for furniture objects into a cost function and automatically synthesizes furniture layouts by optimizing the function. The difference is the spatial relationships of objects, such as relative distance and orientation as well as support relations are learned from 3D scene exemplars instead of manual specification in [63].

Comparing to generating layouts with fixed objects instances, the automatic *open world* layout synthesis is more appealing as the goal is to generate diverse layouts with unspecified number of objects. Yeh et al. [109] use factor graphs, a type of graphical model, to encode complex object relationships as constraints, and propose a variant of Markove Chain Monte Carlo (MCMC) method to sample the possible layouts. The algorithm succeeds at synthesizing interior environments, such as coffee shops, with varying shapes and scales. However, the resultant scenes are restricted with patterns that are only handcrafted in the code.

Example-based scene synthesis. In contrast to 3D scene modeling from observations, 3D scene synthesis aims to generate *novel* scenes with plausible object compositions and arrangements. Given only one or a few exemplars which could be 3D scenes or images, example-based approaches [22, 59, 114] are able to synthesize scenes that are similar to the input, but with certain diversity (Figure 2.4).

The best known method for synthesizing realistic 3D scenes is the example-based scene synthesis proposed by Fisher et al. [22]. Given a few user-provided examples, the system can synthesize a diverse set of plausible new scenes by learning from a 3D scene database. The main challenge for example-based synthesis is generating a *variety* of results while retaining plausibility and similarity to the examples. Two ways are exploited in [22] to

Figure 2.4: Different example-based scene synthesis approaches. Scenes with similar object occurrences and relationships (left), arrangement style (middle), and complex relationships (right) are synthesized based on the corresponding input exemplars.

improve the diversity of synthesized scenes: firstly, extract *contextual categories* from the scene database by using object neighborhoods in scenes to group together objects that are likely to be considered interchangeable; secondly, treat the scene database as a prior over scenes and train a probabilistic model on both the examples and relevant scenes retrieved from the database using [23]. Using the contextual categories and the *mixed* probabilistic model, the algorithm successfully synthesizes scenes with a large variety of both objects and arrangements.

Focusing on enriching scenes with more details, i.e., small objects on the shelves, Majerowicz et al. [59] present an example-based method which automatically populates empty shelf-like surfaces with diverse arrangements of artifacts in a given style. In [59], the style of an arrangement is defined as a combination of object-level and global measures. The object-level measures capture local arrangement properties, such as the percentage of instances of a particular object and the relative location of objects, while the global measures compare high-level characteristics of the two arrangements, such as density and symmetry. To generate scenes with complex relations such as one object is "hooked on", "surrounded by" or "tucked into" another object, Zhao et al. [114] take an example-based approach to synthesize new scenes by replacing objects in the example scene with database objects while preserving the original complex spatial relations in the example.

Progressive scene synthesis. Unlike the example-based methods which take a holistic view of scene generation by targeting overall similarity between the generated scene and the exemplars, Sadeghipour et al. [74] propose the progressive scene synthesis which inserts objects progressively into an initial scene based on probabilistic models learned from a large-scale annotated RGB-D dataset, in particular, the SUN RGB-D databset [91]. By progressively selecting and placing objects, the scene synthesis process attains more local controllability and ensures global coherence and holistic plausibility of the synthesized scenes. However, it is intractable to learn probabilistic models to encode the distributions of all objects and capture all details in large-scale and complex scenes.

Human-centric scene modeling. We live in a 3D world, performing activities and interacting with objects in the indoor environments everyday. It is easy for humans to understand the surrounding scenes or arrange objects based on their functionality. However, it is challenging for an agent, e.g., a robot, to automatically generate behaviors and interact with the 3D environments since the robot lacks knowledge of the objects as well as their functionalities. There has been a great deal of work in robotics and computer vision on utilizing human-centric approaches in the scene analysis and modeling tasks, e.g., scene geometry estimation [25], object labeling [42] and robot placement [43], just to name a few.

Recently, human-object contexts are exploited in computer graphics for functional scene understanding [77]. By observing humans performing different actions, e.g., "use a desktop PC", correlations between human body poses and the surrounding scene geometry are extracted in [77] to train action classifiers which can transfer interaction knowledge to unobserved scenes. Given a new, unobserved scene, action maps which encode the probability of specific actions taking place in the scene are predicted based on the classifiers and the scene geometry.

Instead of learning the supervised action classifiers, recent works build probabilistic models for human-object interaction, i.e., actions or activities, and learn the parameters from multitude data sources, such as depth scans and 3D scenes. Targeting at the generation of realistic 3D scenes, Fisher et al. [21] present a novel method to synthesize scenes that allow the same activities as real environments captured by noisy and incomplete 3D scans (Figure 2.5 left). In order to synthesize object arrangements based on human activities, they train activity models that encode distributions of objects involved in specific activities from annotated 3D scene and model databases in a pre-processing step. Given a 3D scan of an environment as input, semantic scene reconstructions are synthesized by placing objects iteratively in the scenes based on the the geometric and activity properties of the scan.

Following their work for action-based scene understanding, Savva et al. [78] learn a probabilistic model connecting human poses and arrangements of object from real-world observations captured using the same framework as in [77]. The learned probability distributions over pose and object geometry are encoded in the *PiGraphs* and then used to

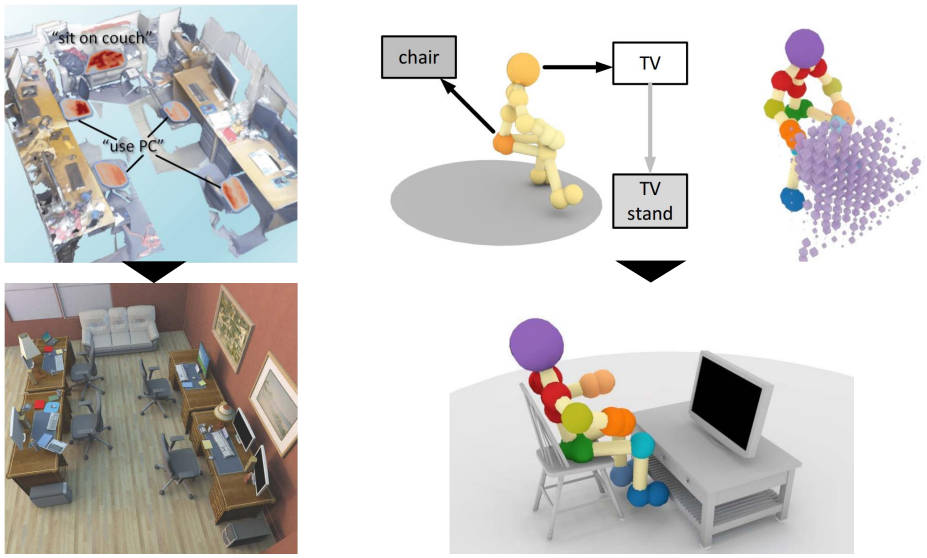


Figure 2.5: Human-centric scene modeling methods that synthesize static 3D scenes based on the activities predicted from the indoor scan (top-left) or learned probabilistic models of human poses and object arrangements (top-right).

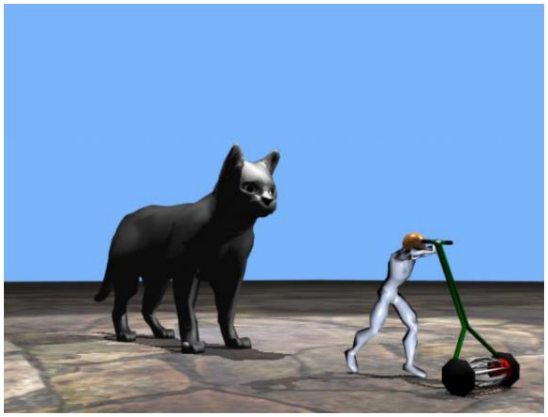
generate *interaction snapshots*, which are static depictions of human poses and relevant objects during human-object interactions (Figure 2.5 right).

The human-centric indoor scene modeling focuses more on the functionality and plausibility of the results, making one step closer to the generation of realistic scenes. However, with the similar limitation of the scene modeling from X works, activity-centric scene modeling only focuses on synthesizing static scenes based on the input scan [21] or the semantic instruction [78]. As real-world scenes are not static environments, instead they evolve over time, driven by object movements resulting from human, it is natural to think about using human actions to evolve a 3D scene in a progressive manner and generate novel and open-ended results.

Text-to-scene generation. Similar to sketches, text or natural language is another form for intuitive content generation. Text-to-scene generation has been studied since the pioneering work WordsEye [17]. The early works focus on directly mapping explicit scene arrangement languages [17, 80, 16] to object placements in 3D scenes (Figure 2.6 left). Improvements over the early text-to-scene systems are made by Chang et al. [11, 12, 9], in which spatial knowledge is learned from the 3D scene database, and utilized to provide unstated facts or common sense knowledge for scene generation (Figure 2.6 right).

In the representative work of Chang et al. [12], the input text is first parsed into a *scene template*, a graph representation that captures semantics and spatial relations of a scene. The scene template is then grounded into a geometric 3D scene by querying a 3D model

"The lawn mower is 5 feet tall. John pushes the lawn mower. The cat is 5 feet behind John. The cat is 10 feet tall"



Coyne et al. [17]

"There is a room with a table and a cake. There is a red chair to the right of the table."

Chang et al. [12]

Figure 2.6: The WordsEye system [17] (left) maps explicit scene arrangement languages to 3D scenes. Chang et al. [12] (right) use spatial knowledge learned from the 3D scene database to improve the text-to-scene generation, e.g., introducing a plate to support the cake although the plate is not mentioned in the input sentences.

database and arranging the objects based on the constraints encoded in the template and the spatial knowledge priors learned from the 3D scene database [22]. Their latest SceneSeer system [9] further extends the pineline in [12] with interactive text-based scene editing operations, e.g., adding, removing, replacing objects, and mouse-based scene manipulation to refine a generated scene. Since current text-to-scene systems work on an object-level for scene generation, it is difficult to use them to create complex scenes.

2.3 Substructure and sub-scene level processing

On the conceptual level, our sub-scene level scene processing is related to the substructure level shape processing. Structure-aware shape processing has drawn a lot of research interests in recent years [65]. Beyond local and low-level geometry processing, structure-aware shape analysis and synthesis work on a higher level to understand the global inter and intra semantic relations among the parts of the shapes and use such structural knowledge for creation of novel shapes. When regarding a 3D scene as a whole shape and the objects in the scene as the parts, it is possible to apply the ideas and algorithms developed for structure-aware shape processing for scene analysis and scene synthesis. In this section, we investigate the shape processing works that utilize substructures or multiple parts for analysis and synthesis of 3D objects. In addition, we review the works that consider sub-scenes in processing of 3D scenes.

Contextual-based analysis of shapes and scenes. Rather than only considering the local geometry features for shape matching and understanding, contextual-based shape analysis combines neighborhood information of nearby geometry or substructures to characterize shape features at a higher level. In the early work of [6], Belongie et al. introduced *shape context*, a contour-based shape descriptor which captures the distribution of nearby contour points relative to the reference position, for robust digit recognition and silhouette-based shape matching.

With the increasing heterogeneity in 3D model databases, shapes serving the same function may have a large difference in geometry. Context information of shape *structures*, which characterizes the relationships between the central object part and others, becomes more reliable when finding similar semantic parts [83, 49]. Shapira et al. [83] define a similarity measure between two 3D object parts based not only on their local signatures and geometry, but also on the context extracted from their shape partitioning hierarchy. Similar to the context-aware similarity measures that build on the graph kernels for image classification [30] and scene comparison [23], Laga et al. [49] model the context of a shape part as walks in the graph encoding shape parts and their structural relationships, and apply the context-aware part similarity to find part-wise semantic correspondences between 3D shapes. Context-based shape descriptors achieve better performance in part-in-whole shape matching comparing to only using local features. However, it is unknown which shape parts or substructures are more discriminative to characterize the nature of the corresponding shapes.

Similar to the part-in-whole shape matching, object-in-scene type of 3D model retrieval that uses context information learned from existing 3D scenes is introduced in Fisher and Hanrahan [20]. Also, contextual information is utilized in 3D scene understanding by training classifiers that combine contextual relationships between objects for object recognition [14, 85]. Contextual relationships such as object co-occurrence and spatial arrangements are exploited in Chen et al. [14] to assist object segmentation and classification for automatic semantic scene modeling from indoor scans. Shi et al. [85] propose a unified framework that detects both individual objects and object groups by performing contextual analysis using classifiers learned from 3D model and scene databases. In these works, the context encoded in the substructures or sub-scenes is only regarded as a feature of the central objects instead of a scene feature.

Our sub-scene level scene analysis takes the benefit of contextual modeling for a deeper understanding of scene structures. Moreover, our contextual modeling works at two levels: at a lower level, we treat the representative focal points as a new context-based scene feature for characterizing complex scenes; at a higher level, we extract focal points by co-analysis of a scene collection, exploring the broader context of scene clusters to determine the discriminative focal points.

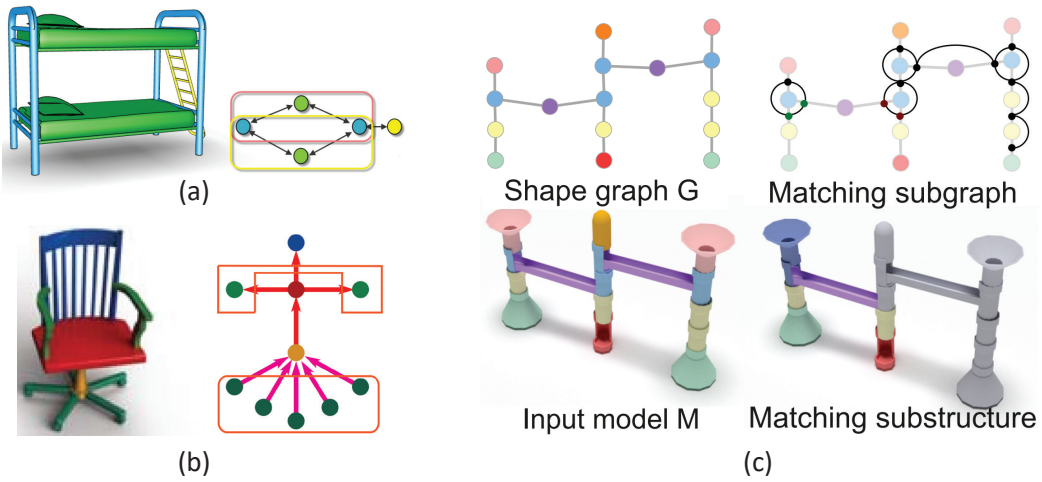


Figure 2.7: Substructures for part-based shape modeling. (a) A 3D object and its symmetry-induced substructures in Zheng et al. [116]; (b) a 3D object and its support-induced substructures in Huang [37]; (c) Matching replaceable substructures in Liu [54].

Substructure-based shape modeling. Content creation by combining parts from existing models has been explored for part-based shape modeling in [26, 105, 40]. There are also data-driven methods which learn probabilistic models for part compositions of shapes in certain categories [13, 45]. However, these methods can only produce shape variations in the same category of objects. Recombining the substructures of the shapes instead of the individual parts can handle shapes with large geometry and structure difference, and produce interesting, plausible and functionally valid shape variations [116, 37, 54].

Certain substructures are commonly shared by objects from different categories and they are related to the functionality of the objects. Based on this observation, Zheng et al. [116] identify the symmetric function arrangements, which are special arrangements among symmetrically related substructures (Figure 2.7(a)), and propose a purely geometric approach based on such substructures to match, replace, and position triplets of parts to create non-trivial, yet functionally plausible, shape variations without the need of extensive training data.

Besides the symmetry-induced substructures, support-induced substructures, which are a set of stable and self-supporting arrangement of object parts, are introduced in [37] (Figure 2.7(b)). A bottom-up approach is proposed to identify such substructures in a support relation graph. The derived support substructures are applied to structure reshuffling, rearrangement and synthesis. As the support substructures are not based on the symmetric arrangements, shape modeling using support substructures can produce richer results which are difficult to achieve with symmetry-induced substructures [116]. In contrast, the functionality of the results may be reduced since support relationships themselves might mot be sufficient to capture relationships between parts.

Also aiming to use substructures for part-based shape modeling, Liu et al. [54] develop an algorithm that efficiently enumerates all replaceable substructures that lead to shape modification operations. The replaceable substructures are defined as arrangements of parts that can be interchanged during part-based modeling (Figure 2.7(c)). By converting a shape into a shape graph in which the nodes represent the shape parts and the edges encode the connection between parts, the replaceable substructures or subgraphs are characterized as the graph cuts that will respect all of the neighbor connections when one subgraph is exchanged with another subgraph. Such substructures are discovered in polynomial time by performing efficient subgraph matching on the input shape graphs. Plausible and complex shape variations which contains tens of part nodes are generated efficiently by swapping the matched replaceable substructures for both in-model and across-model synthesis.

Scene synthesis by reshuffling substructures. Inspired by part-based shape modeling [105, 40, 116, 35, 54], reshuffle-based methods are proposed for scene synthesis. Xie et al. [101] develop a system to reshuffle the furnitures between different input scenes. Their synthesis results are limited as only simple scene structures and relations are studied. Recently, Huang et al. [36] propose to reshuffle the scene structures to create scene variations. They establish a graph matching between structure graphs representing two scenes and construct an *Augmented Graph* (AG) to encode the scene structures of all example scenes. By reshuffling objects corresponding to the subgraphs based on the AG, scenes with plausible structure variations are synthesized efficiently.

Comparing to using substructures for shape synthesis, scene synthesis by matching and reshuffling substructures is difficult since many indoor scenes contain rich and loose object relationships which will lead to complex structure graphs with edges encoding multiple relationships between objects. One the other hand, substructure or sub-scene level scene synthesis is intriguing since it shares the similar advantage of substructure-based shape modeling, i.e., instead of synthesizing from scratch, the object arrangements and scene semantics encoded in the sub-scenes could be directly used to generate new scenes.

Our scene synthesis frameworks take the advantage of sub-scene level object arrangements and utilize knowledge extracted from existing scene databases for scene generation. Our action model of human-object relations and relational model which encodes the semantic object relationships for text-to-scene mapping are indeed the probabilistic models of object arrangements corresponding to sub-scenes. We build the models through learning stages and apply these compact models of sub-scenes for interactive and progressive scene synthesis.

Chapter 3

Focal-Centric Scene Analysis

In this chapter, we introduce *focal points* for 3D indoor scene analysis. We represent each scene by a graph of its constituent objects and define focal points as *representative* sub-structures or sub-scenes in a scene collection. We propose a co-analysis algorithm to extract focal points from a heterogeneous scene collection and cluster the scenes based on the focal points. We demonstrate advantages of *focal-centric* scene comparison and organization over existing approaches, particularly in dealing with *hybrid* scenes, scenes consisting of elements which suggest membership in different semantic categories.

3.1 Introduction

Recent works on organizing and exploring 3D visual data have mostly been devoted to object collections [68, 40, 47, 95, 38]. In contrast, we are interested in analyzing and organizing visual data at a larger scope, namely, 3D indoor scenes. Even a moderately complex indoor scene would contain tens to hundreds of objects. Compared to the individual objects therein, a scene is more complex with looser structural and spatial relations among its components and a more diverse mixture of functional substructures. The latter point is attested by *hybrid* scenes which contain elements reminiscent of different semantic categories. For example, the middle scene in Figure 3.1 is partly a bedroom and partly a living room. The greater intra-class variabilities and richer characteristics in scene data motivate our work to go beyond providing only a holistic and singular view of a scene or a scene collection.

We introduce the use of *focal points* for characterizing, comparing, and organizing collections of complex data and apply the concepts and algorithms developed to 3D indoor scenes. In particular, we are interested in organizing scenes in a *heterogeneous* collection, i.e., scenes belonging to multiple semantic categories. Analyzing complex and heterogeneous data is difficult without references to certain points of attention or focus, i.e., the focal points. For example, comparing New York City to Paris as a whole will unlikely yield a useful answer. The comparison is a lot more meaningful if it is focused on particular aspects of the cities,

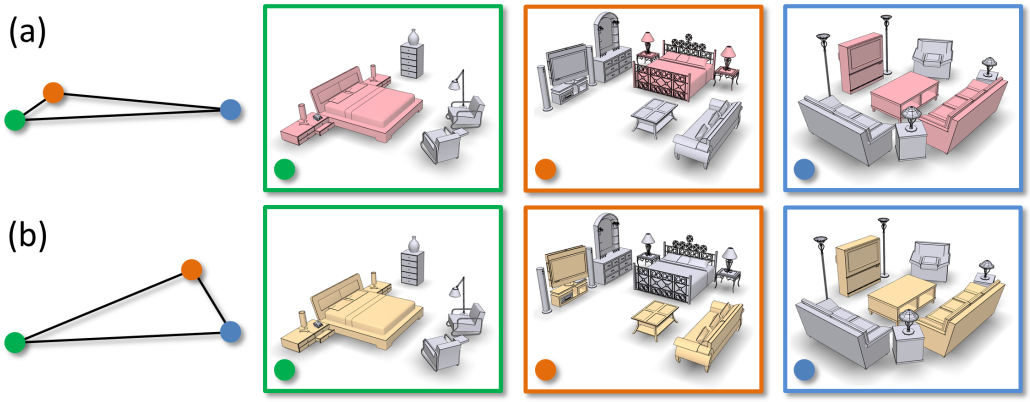


Figure 3.1: We analyze and organize 3D indoor scenes in a heterogeneous collection from the perspective of *focal points* (sub-scenes in color). Scene comparisons may yield different similarity distances (left) depending on the focal points.

e.g., architectural style or fashion trends. One of the natural consequences of the focal point driven data view is that scene comparison may yield different similarity distances depending on the focal points; see Figure 3.1 for an illustration.

We represent an indoor scene by a graph of its constituent objects. A focal point, or focal, for short, is a *substructure* in a scene and corresponds to a subgraph or a sub-scene. However, we are not interested in all sub-scenes. A key premise of our work is that meaningful focals should be determined *contextually*, in a set (Figure 3.2), and through a *co-analysis*. To illustrate, there are probably too many notable aspects about Paris. When putting London and Paris together, one's focuses narrow down to, e.g., European capitals.

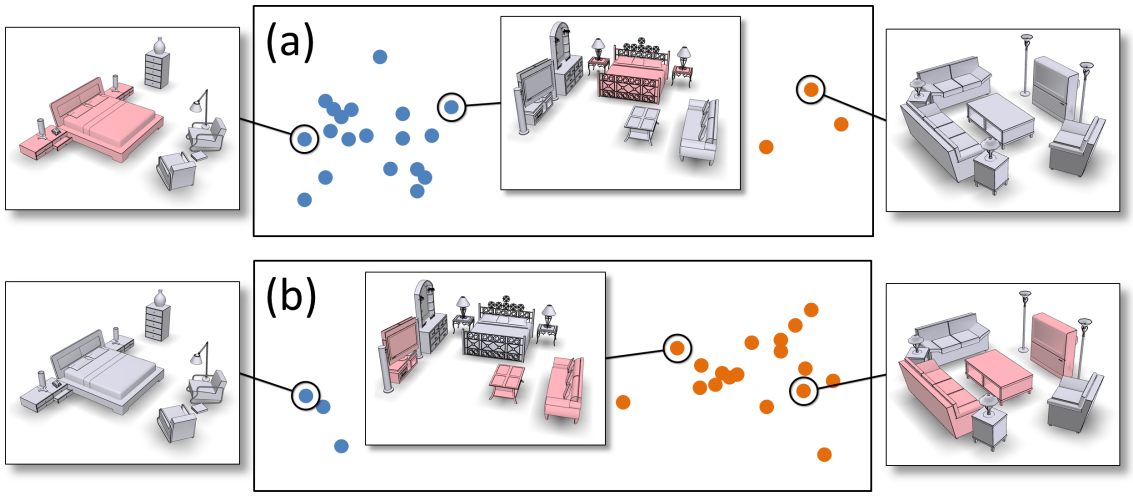


Figure 3.2: Focal points (marked red in the scenes) are *contextual* and depend on scene composition in a collection. With more bedrooms (a) or more living rooms (b), different focals were extracted and hybrid scenes are pulled towards one of the clusters.

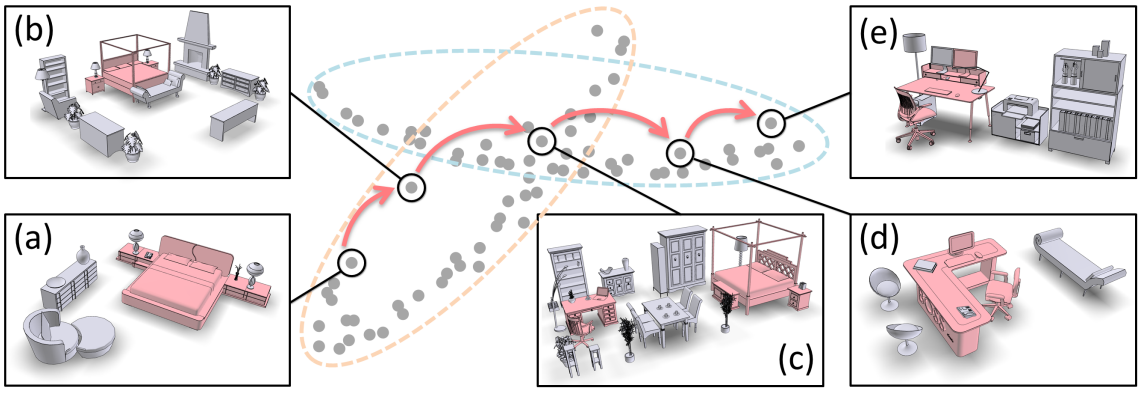


Figure 3.3: Focal-driven scene clustering produces overlapping clusters. An exploratory path, (a) to (e), through an overlap, which often contains hybrid scenes (c) possessing multiple focals, can smoothly transition between the scene clusters. These scene clusters often characterize meaningful scene categories. In this example, the transition is from bedroom scenes to offices.

If we throw New York and Milan into the mix, then most people are first reminded that the four cities are the fashion capitals of the world.

In this work, we are interested in extracting contextual focal points that are *representative* in a given scene collection. For a focal to be representative, it must occur sufficiently *frequently*. However, frequency analysis alone is insufficient. We stipulate that representativity is also tied to a notion of coherence or *compactness* of the group of scenes the focal point is to represent or characterize. Therefore, frequency analysis for focal extraction is intermixed with clustering, which computes compact groups of scenes, where the scenes in each cluster are closely connected when viewed from the perspective of the representative focals of the cluster. Once again, the representative focals occur frequently in the cluster and they must also induce a compact cluster. To solve the two coupled problems simultaneously, we develop a co-analysis algorithm which interleaves *frequent pattern mining* [29] and *subspace clustering* [96].

Focal points play a key role in our organization of a heterogeneous scene collection. First, we define compactness of a cluster based on a *focal-centric* scene-to-scene similarity, which builds on the rooted walk graph kernels of Fisher et al. [23] and assigns higher weights to walks which originate from the representative focals of that cluster. Secondly, the scene organization is given by the clustering of scenes based on the representative focals extracted. Some scenes may contain multiple focals, thus belong to multiple clusters. Such scenes, typically of a hybrid nature, provide linkages or gateways between scene clusters, allowing an exploration of the scene organization to naturally transition between meaningful scene categories, as illustrated in Figure 3.3.

Our main contribution is a focal-driven analysis and organization of heterogeneous data collections. While we only consider 3D indoor scenes in this work and we are not aware

of previous works on co-analysis and organization of heterogeneous scene collections, the analysis is general and not confined to scene data. Important characteristics of our work which set it apart from previous approaches to organizing data collections include:

- Data are not compared holistically without discrimination. We develop a focal-centric scene descriptor for scene comparison, which supports scene analysis in *perspective*.
- Similarity distance between two scenes may be *non-unique*, i.e., it is based on the focals designated for comparison.
- Multiple views on scene data depend on focal points, leading to overlapping clustering of a scene collection, rather than a partition. The resulting organization is particularly suited for retrieving and exploring complex and hybrid scenes.

We show advantages of focal-centric scene comparison and organization over existing approaches, particularly in dealing with hybrid scenes. We also demonstrate new capabilities offered by the new data organization for scene retrieval and exploration.

3.2 Related work

Background. At a conceptual level, our work can be seen as a realization of the notion of "family resemblances" from the seminal work of Wittgenstein [99]. A scene collection forms the "family", and the extracted focals represent the resemblances which "overlap and criss-cross" among the scenes. Works from cognitive psychology, in particular those by Rosch [73], provided evidences that *perceptual and semantic categories are naturally formed in terms of focal points or prototypes* (see account in [94]), though the so-called "cognitive reference points" in her work referred to *whole* representatives of a category instead of featured substructures. The role of context in measuring data similarity has long been studied in various fields, e.g., [7, 41]. Our work presents an algorithm for identifying conceptual focals which serve as reference points for comparing scenes in a heterogeneous collection.

Scene analysis. As the most familiar environments to humans, indoor scenes are ubiquitous in graphics applications such as virtual reality, gaming, and design. Much research in vision and graphics has been devoted to recognizing, classifying, and retrieving indoor scenes, e.g., [70, 69, 23, 44, 107, 115], among others. Our work recognizes the difficulty in comparing complex scenes globally, e.g., via the classic graph kernels [23]. We propose extracting and utilizing focal substructures for scene analysis. Of relevance are works which extract distinctive regions [86, 44] that are representative of a semantic category. The focals we extract are not meant for scene recognition but organization; one focal may be shared by scenes from different categories.

Object collections and co-analysis. There have been a growing body of work on unsupervised co-analysis [105, 34, 95, 33, 116] and organization of 3D object collections [68, 40, 47, 38]. Similar works exist on image collections, e.g., for image co-salience detection [15]. In most cases, co-analysis operates on objects belonging to the *same* semantic category. An exception is the recent work of Huang et al. [38] which performs qualitative analysis on heterogeneous object collections. However, their object comparison employs global shape descriptors while still resulting in unique qualitative distances, in terms of number of "hops" in a tree representation, between objects.

Another recent work, the co-hierarchical analysis of van Kaick et al. [95], also employs a clustering approach and the clustering partitions a set of shapes into different modes of structural variation. While hierarchical models offer the flexibility to account for structural variations, they still provide only a *single* view on each shape. Our representation allows multiple views of a scene model, each of which may be seen as from the perspective of a particular focal point. Moreover, our analysis produces overlapping clusters which characterize the underlying data with larger granularity.

Contextual analysis. Part-in-whole or object-in-scene types of retrievals have been studied in semantic analysis of 3D objects or indoor scenes. Shapira et al. [83] define the context for a shape part within an extracted part hierarchy. The series of work from Fisher et al. rely on spatial and semantic relations among the scene objects for context-based object search [20, 23] or object replacement for scene synthesis [22]. In all of these works, substructures in a scene provide the contexts for characterizing individual objects therein. We treat the substructures as explicit scene features, i.e., potential focals, and perform contextual analysis in a larger scope.

One possible way to find salient substructures in a scene collection is to extract object groups based on co-occurrences of object *categories*, like in the work of Xu et al. [107]. In contrast, we group scene objects, rather than object categories, to form focals. Furthermore, the grouping in Xu et al. [107] is based on frequency analysis only, while we perform both frequent pattern mining and subspace clustering for focal point extraction. Singh et al. [89] detect mid-level discriminative patches from a set of unlabeled images by alternating between clustering and training discriminative classifiers. A similar idea is then applied to extract, from a large repository of geo-tagged imagery, visual features which are both frequently occurring and geographically distinctive under weak supervision [19]. Our co-analysis is unsupervised, driven by a novel cluster compactness objective for both focal selection and focal-induced clustering.

Frequent pattern mining. Frequent pattern mining has been an extensively studied topic in data mining [29]. The most relevant works are those designed for frequent subgraph mining, e.g., [108], which are primarily based on subgraph isomorphism testing. Directly adapting these methods to our problem setting is infeasible since the relations among objects

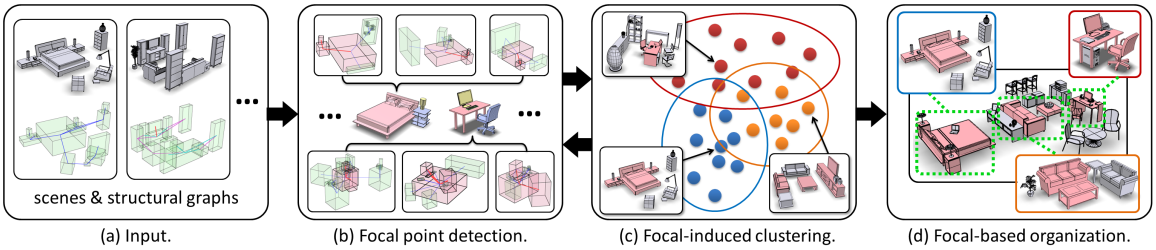


Figure 3.4: An overview of our algorithm. The input is a heterogeneous collection of 3D indoor scenes. We represent each scene by a structural graph (a). The co-analysis algorithm is iterative, between (b) and (c). Each iteration involves an interleaving optimization consisting of focal point detection (b) and focal-induced scene clustering (c). After the set of contextual focals are obtained, the entire scene collection can be organized with the focals serving as the interlinks between scenes from various clusters (d).

in our input graphs are loose and possibly uncertain. We adopt inexact subgraph matching formulated by graph edit distances [71] where the edit cost is defined based on spatial arrangements between scene objects. It is also worth noting that frequency of occurrence is not the only criterion for focal point selection. The subsequent cluster analysis further adjusts the extracted focals.

Subspace clustering. Subspace clustering clusters high-dimensional data into multiple subspaces, each modeled by a subset of features [96]. At a high level, the clustering problem we face has a similar setting as subspace clustering, where focals act as the feature subsets and characterize the subspaces that contain the clusters of scenes. Subspace analysis via spectral clustering has been one of the most effective approaches to subspace clustering [97]. However, spectral clustering always produces a partition. In our work, we perform cluster attachment to reveal cluster overlap based on their representative focals, making the obtained clusters better reflect the complexity and heterogeneity of the data collection.

3.3 Overview

The input to our algorithm is a heterogeneous collection of 3D indoor scenes collected from public repositories. Such scenes typically come with semantic labels for the objects and the scenes themselves. Our analysis uses the object labels but never the scene labels. Our goal is to extract a set of contextual focals, as well as a clustering of the scenes based on these focals; see Figure 3.4.

For each scene, a structural graph is constructed which encodes two types of relationships between scene objects: support and proximity. Our main algorithm consists of a coupled optimization whose objective is to maximize the overall compactness of the scene clusters while ensuring that the focals represent their respective clusters effectively. A key is that each representative focal is sufficiently *discriminative* so that it is frequent only within the

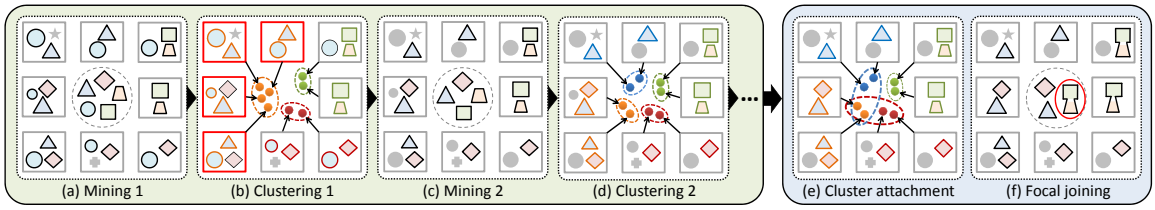


Figure 3.5: Illustration of iterative optimization pipeline. A scene is depicted with a grey box enclosing several substructures represented by circles, squares, diamonds, triangles, etc. To initialize the interleaving optimization, we first detect a set frequent substructure shown in the middle in (a). Based on that, subspace clustering leads to incorrect clusters (marked in red) due to the trivial substructure (circle) occurring in most scenes. Then we perform cluster-guided weighted mining which eliminates the trivial substructure. Following that, a more accurate clustering result is obtained in (d) based on the new set of discriminant substructures in (c). Finally, we perform cluster attachment to reveal overlapping clusters (the red and yellow clusters in (e)), as well as focal joining to discover non-local focal points (marked in red in (f)).

cluster it represents or characterizes. The optimization is *iterative*, where each iteration interleaves between cluster-guided focal point mining and focal-induced subspace clustering of the scenes; see Figure 3.5.

The first and initial phase of the optimization is to extract frequent substructures as focals from the input structural graphs, via subgraph mining (Section 3.4.1). Rather than relying on subgraph isomorphism, we perform inexact graph matching which insists on consistency of node labeling but not edge connection. The latter is to account for loose relations between corresponding objects across a large heterogeneous scene collection. The matching of such relations is based on a layout similarity measure between spatial arrangements of objects. This matching is confined by scene grouping resulting from the most recent clustering phase. Specifically, the subgraph matching is *weighted* so that the substructures found are frequent only within the clusters they characterize.

In the second phase, based on the extracted focals, we perform subspace clustering (Section 3.4.2) on the scenes. The structural graphs are clustered so that each cluster is characterized by a subset of current focals. Generally, the representative focals for a cluster are not unique. The clustering step seeks to maximize the compactness of all clusters, where compactness is defined by a scene-to-scene similarity based on *focal-centric graph kernels* (FCGK). We define FCGK based on the work of Fisher et al. [23] which utilizes rooted walk graph kernels. However, instead of weighting equally walks from all sources, we weigh more heavily those walks which originate from representative focals in the graphs. The maximization is based an *iteratively reweighted subspace clustering* scheme we develop, which gradually increases cluster compactness.

Finally, once the clusters and focals are determined by the optimization, we perform *cluster attachment* and *focal joining* (Section 3.4.3). Some clusters share scenes containing

Algorithm 1: Structural Graph Construction

Input : scene $C = \{O_i\}_i$
Output: structural graph $G = \langle V, E \rangle$

$\forall O_i \in C, V \leftarrow V \cup \{v_i\}$; // *vertices*
$E \leftarrow E \cup$ `SupportEdge(`C`)` ; // *support edges*
$E \leftarrow E \cup$ `ProximityEdge(`C, E`)` ; // *proximity edges*
$\mathcal{U} \leftarrow$ `DetectSymGroup(`C`)`;
foreach $U \in \mathcal{U}$ **do** // *for each group of symmetric objects*
  foreach $v \in V - U$ **do** // *for each outside object*
    if $\exists s, t \in U; \langle v, s \rangle, \langle v, t \rangle \in E; d_{\text{arr}}(\langle v, s \rangle, \langle v, t \rangle) < 0.1$ **then**
      foreach $u \in U$ **do** // *do symmetric connection*
        $E \leftarrow E \cup \{\langle u, v \rangle\}$;
$E \leftarrow E \cup$ `ConnectComponents(`V, E`)`;
return G;

multiple focals, each characterizing a different cluster. These clusters are naturally attached at the shared scenes. Within a cluster, multiple local substructures may occur concurrently across all or most scenes. These substructures are naturally joined to form non-local focals. Note that such non-local focals could not be detected via subgraph mining since only spatially close objects are connected in the graphs.

3.4 Focal-driven scene co-analysis

For each input scene, we construct a structural graph (Figure 3.6(b)) whose nodes are scene objects and edges encode spatial relationship, support and proximity, between objects; see Algorithm 1. Both nodes and edges are labeled, by object semantic labels and relationship types (support or proximity), respectively.

We first detect all support relationship between objects by testing vertical contacts between their shape geometries. Second, we add a proximity edge from any object that is not connected by a support edge, to the object which has the strongest connection with it, where connection strength (Equation 3.3) is defined as a part of layout similarity. Third, we ensure that any group of symmetric objects has symmetric connections to other objects, if any.

We detect all groups of mutually symmetric objects and examine for each group all outside objects connecting to that group. If more than two symmetric objects in the group have similar spatial arrangement (Equation 3.5) with respect to an outside object, we ensure they all connect to the outside object with edges of the same type, depending on their relationship against the outside object. To detect mutually symmetric objects, i.e., objects possessing similar geometry, we adopt the registration method described in [98]. Finally, we

detect the connected components in the current graph, and connect the components with proximity edges to make sure the entire scene is represented by a connected graph.

Our co-analysis operates on these structural graphs. The main algorithm involves a coupled optimization for both focal point mining and scene clustering. The objective of the optimization is

$$\max_{\mathcal{F},\Omega} \sum_{\ell=1}^{c} n_\ell \kappa_\ell(\mathcal{F}, \Omega) \tag{3.1}$$

where $\mathcal{F} = \{F_k\}_{k=1}^{n}$ are the set of focal points, and $\Omega = \{\mathcal{C}_\ell\}_{\ell=1}^{c}$ the set of clusters. κ_ℓ denotes the compactness of cluster $\mathcal{C}_\ell$ based on FCGK, and n_ℓ is the size of cluster $\mathcal{C}_\ell$. We optimize iteratively with the iterations continuing until the overall compactness of the clusters converges, specifically, when the change of the objective function is less than 1.0×10^{-6}. In the following sections, we detail our co-analysis algorithm.

3.4.1 Focal extraction via graph mining

A substructure of a scene consists of a group of nearby objects along with their spatial arrangement; it is a subgraph. We could define focals as substructures that occur frequently across a large number of semantically related scenes, e.g., bedrooms. However, since scene labels can be unknown or ambiguous, especially for hybrid scenes, we do not use them. Instead, we couple focal detection with the identification of meaningful clusters. If a substructure occurs in a scene, we say that the scene *supports* that substructure. The notion of occurrence will be quickly relaxed by inexact graph matching, which is enabled by a similarity measure of spatial layout between substructures of scenes.

Layout similarity. We define a layout similarity between two substructures by examining the pair-wise spatial arrangement of oriented bounding boxes (OBBs) of the objects in the substructures. Suppose we are given two substructures represented by two subgraphs in the structural graphs of two scenes: $S_a \subset G_A$ and $S_b \subset G_B$. The layout dissimilarity between them is defined as:

$$D_{\text{layout}}(S_a, S_b) = \sum_{\substack{\{p,q\} \in S_a, \\ \{\theta(p),\theta(q)\} \in S_b}} d_{\text{arr}}(\langle p, q\rangle, \langle \theta(p), \theta(q)\rangle), \tag{3.2}$$

where $\theta(p) \in G_B$ is the corresponding object of $p \in G_A$. Such correspondences can be determined during subgraph mining, as described below. d_{arr} measures the *spatial arrangement* dissimilarity between two pairs of objects which is defined based on two factors. The first is the *connection strength* between objects p and q:

$$\gamma(p, q) = \frac{d_{\text{H}}(obb(p), obb(q))}{dl(p) + dl(q)}, \tag{3.3}$$

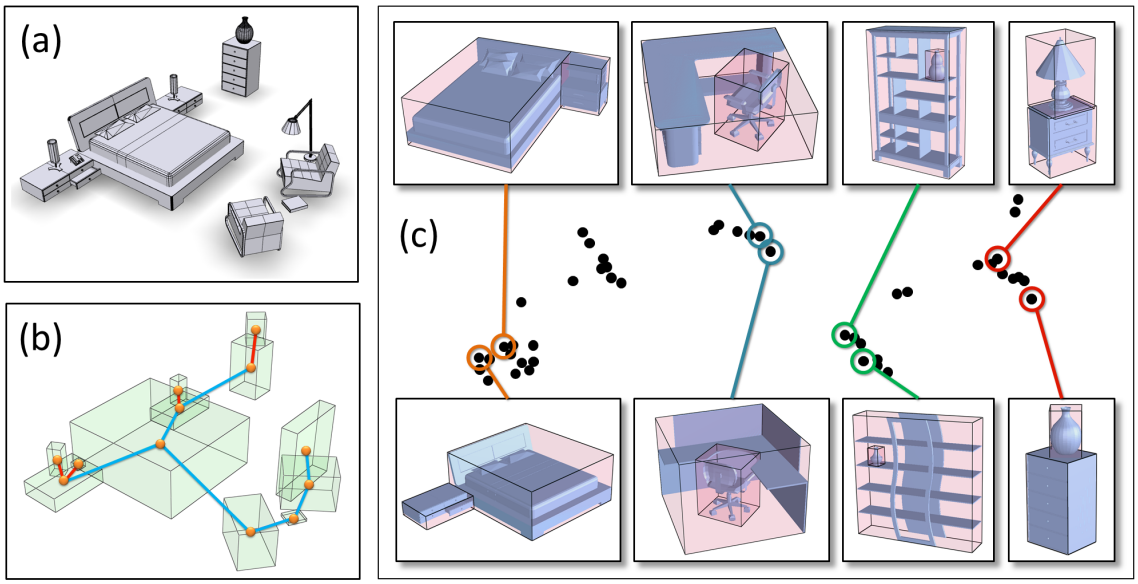


Figure 3.6: The structural graph (b) of the input scene (a) encodes two types of relationship: support (red) and proximity (blue). (c) plots the layout similarity of object pairs after spectral embedding.

where d_{H} is Hausdorff distance, $obb(p)$ the OBB of object p, and $dl(p)$ the diagonal length of $obb(p)$. The second factor is the angle between the upright vector and the vector between p and q:

$$\rho(p,q) = angle(\mathbf{v}_{\mathrm{dir}}(p,q), \mathbf{v}_{\mathrm{upright}}), \tag{3.4}$$

where $\mathbf{v}_{\mathrm{dir}}(p,q)$ is the vector from the larger object of the two to the smaller one and $\mathbf{v}_{\mathrm{upright}}$ the upright vector. The dissimilarity of spatial arrangement between two object pairs $\langle p,q\rangle$ and $\langle s,t\rangle$ is then defined as:

$$\begin{aligned} &d_{\mathrm{arr}}(\langle p,q\rangle, \langle s,t\rangle) \\ &= \alpha|\tilde{\gamma}(p,q) - \tilde{\gamma}(s,t)| + (1-\alpha)|\tilde{\rho}(p,q) - \tilde{\rho}(s,t)|. \end{aligned} \tag{3.5}$$

$\tilde{\gamma} = e^{-\gamma^2/(\sigma\gamma_{\max})^2}$ is normalized connection strength where $\sigma = 0.4$ and the maximum value $\gamma_{\max}$ is found for all pairs of objects. ρ is normalized similarly. We use $\alpha = 0.6$ in our implementation. Figure 3.6(c) shows a few examples of similar layouts.

Frequent substructure mining. Frequent subgraph mining extracts from a set of input graphs $\mathcal{G} = \{G_i\}_{i=1}^n$, a set of subgraphs $\mathcal{F} = \{F_k\}_{k=1}^d$, which frequently occur (more than a given threshold value $s_{\min}$) in the input graphs based on subgraph isomorphism. We define:

$$\mathcal{F} = \{F_k \mid |\mathcal{S}_k| = \sum_{i=1}^{n} x_{ik} > s_{\min}\} \tag{3.6}$$

Algorithm 2: Extended Frequent Substructure Mining

Input : structural graphs $\mathcal{G} = \{G_i\}_i$, minimal support $s_{\min}$
Output: frequent substructures $\mathcal{F} = \{\langle F_k, \mathcal{S}_k \rangle\}_k$

1 $\mathcal{F} = \{\langle F_k, \mathcal{S}_k \rangle\}_k \leftarrow$ `MineSubgraph(`$\mathcal{G}$, $s_{\min}$`);`
2 **foreach** $G_i \in \mathcal{G}$ **do** // *expand support*
3 **foreach** $\langle F_k, \mathcal{S}_k \rangle \in \mathcal{F}$ **do**
4 $\delta(G_i, F_k) \leftarrow$ `ErrorCorrectMatch(`F_k, G_i`);`
5 **if** $\delta(G_i, F_k) < \delta^{\mathrm{t}}$ **then**
6 $\mathcal{S}_k \leftarrow \mathcal{S}_k \cup \{G_i\}$;
7 **foreach** $\langle F_k, \mathcal{S}_k \rangle \in \mathcal{F}$ **do** // *filter support*
8 **foreach** $G_i \in \mathcal{S}_k$ **do**
9 **if** $\varphi(G_i, F_k) > \varphi^{\mathrm{t}}$ **then**
10 $\mathcal{S}_k \leftarrow \mathcal{S}_k - \{G_i\}$;
11 **if** $|\mathcal{S}_k| < s_{\min}$ **then**
12 $\mathcal{F} \leftarrow \mathcal{F} - \{\langle F_k, \mathcal{S}_k \rangle\}$;
13 **return** $\mathcal{F}$;

where $x_{ik} = I(F_k \subseteq G_i)$ is an indicator function for subgraph isomorphism and $\mathcal{S}_k = \{G_i \mid x_{ik} = 1\}$ is the *supporter set* of F_k.

Directly applying frequent subgraph mining to structural graphs is ineffective since the the proximity relationships are not necessarily consistent across different scenes, e.g., see Figure 3.7(a,b). One may then resort to inexact graph matching, e.g., based on graph edit distance [71]. However, the large search space of inexact subgraph mining makes such approaches prohibitive.

We propose a two-step scheme for frequent substructure mining (Algorithm 2) which carries out inexact graph matching efficiently. We first perform frequent subgraph mining based on exact subgraph isomorphism, using gSpan [108], with a relatively low minimal support threshold (Line 1 in Algorithm 2). Then, in the second step, we employ inexact subgraph matching [71] to match the frequent subgraphs mined in the previous step against all graphs in the set, to expand their support (Lines 2-6). Note that in both steps, the matching of graph nodes is exact and based only on node labels.

To create tolerance for different proximity connection graph structure, we use *error correction* of the subgraphs by introducing three edit operations on graph edges: insertion and deletion of proximity-type edges, as well as substitution between two proximity edges. The edit cost of each operation is defined as the spatial arrangement dissimilarity (Equation 3.5) between the two pairs of objects involved. If the total edit cost $\delta(G_i, F_k)$ for matching F_k and G_i is less than $\delta^{\mathrm{t}} = 0.1$, we add G_i to F_k's supporter set.

For a frequent subgraph F_k, we have obtained its embedding in any of its supporter graphs during the mining step, denoted as $G_i(F_k) \subseteq G_i$, $G_i \in \mathcal{S}_k$. However, the em-

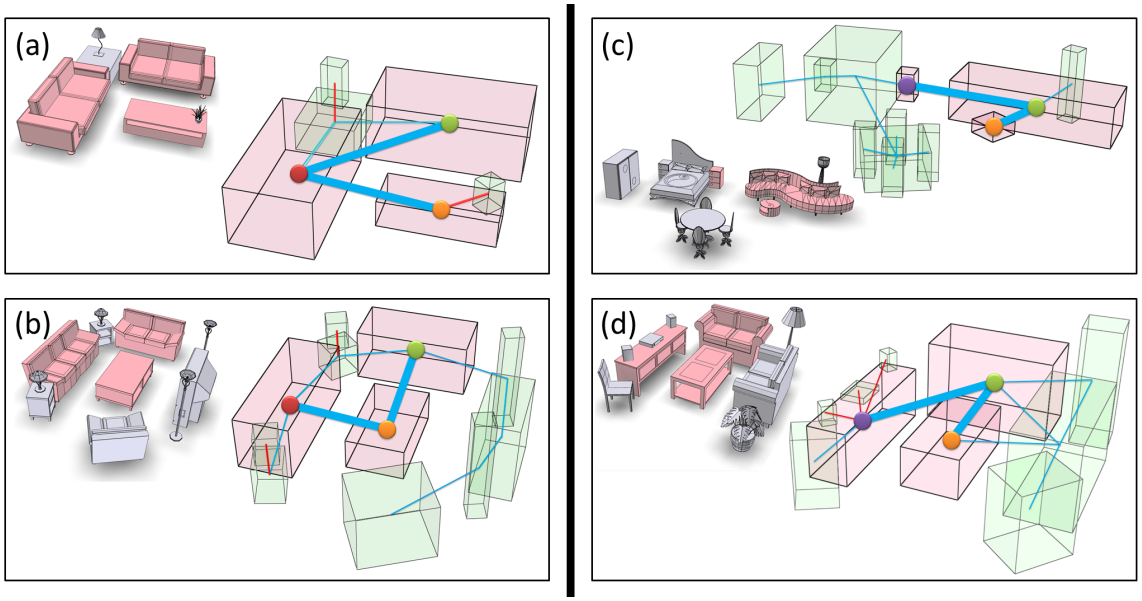


Figure 3.7: Scenes (a) and (b) have the same sub-scenes represented with different subgraphs. (c) and (d) have the same subgraphs while the layouts of the corresponding sub-scenes are different.

bedding of F_k in its supporters may have different layouts since the exact mining step is layout-oblivious, e.g., as shown in Figure 3.7(c,d). We locate and remove weak (or outlier) supporters in which the embedded subgraph has significantly different layout from those in the other supporters (Lines 7-10). Specifically, given a supporter $G_i \in \mathcal{S}_k$ of F_k, we compute the average dissimilarity between its corresponding embedding and those in all other supporters,

$$\varphi(G_i, F_k) = \sum_{G_j \in \mathcal{S}_k, i \neq j} D_{\text{layout}}(G_i(F_k), G_j(F_k)),$$

and filter out this supporter if the value exceeds a threshold $\varphi^{\text{t}} = 0.3|\mathcal{S}_k|$. Finally, we remove those subgraphs whose number of supporters falls below the minimal support threshold $s_{\min}$ (Line 12).

Cluster-guided weighted mining. Our goal is to detect representative focal points characterizing a meaningful clustering of the input scenes, and not substructures which are frequent over the entire collection. Therefore, instead of relying on the frequency criterion in Equation (3.6), we base our substructure mining on the *current clusters* and perform weighted subgraph mining [93]. For each cluster $\mathcal{C}_\ell$, we define *supporting weights* $(\varpi_{\ell i})_{i=1}^n$ as a measure of support of G_i to any substructure. A substructure is detected as frequent if its weighted sum of support, denoted by *discriminant score* $\eta_{\ell k}$, is greater than a threshold η_ℓ^{t}:

$$\mathcal{F}_\ell = \{F_k \mid \eta_{\ell k} > \eta_\ell^{\text{t}}\} \text{ where } \eta_{\ell k} = \left| \sum_{i=1}^{n} \varpi_{\ell i}(2x_{ik} - 1) \right|. \tag{3.7}$$

By using positive weights $\varpi_{\ell i}$, if G_i belongs to $\mathcal{C}_\ell$, and negative otherwise, the discriminant score favors a substructure which is frequent in cluster $\mathcal{C}_\ell$ and penalizes its frequency in other clusters. Therefore, the mined substructures in $\mathcal{F}_\ell$ are frequent mainly within cluster $\mathcal{C}_\ell$. Specifically, we set $\varpi_{\ell i} = x_{\ell i}/n_\ell - 1/n$, where $x_{\ell i} = I(G_i \in \mathcal{C}_\ell)$, and $\eta_\ell^{\mathrm{t}} = \mu n/n_\ell$. We fix $\mu = 0.1$ in our algorithm. The final set of focal points takes the union of per-cluster discriminant substructures: $\mathcal{F} = \bigcup_{\ell=1}^{c} \mathcal{F}_\ell$, where c is the number of clusters. To achieve weighted mining, we evaluate the discriminant score of the individual substructures, which are efficiently enumerated by gSpan, and identify the discriminative ones based on the current clusters. Then we perform support expanding and filtering for the extracted substructures. In the first iteration, when clustering is missing, we use unweighted frequent substructure mining.

3.4.2 Focal-induced scene clustering

With the focals extracted, we perform subspace clustering to group the input scenes according to the extracted focals that they "share", i.e., the scenes contain and support the same focal. For each scene, we build a high-dimensional feature vector for clustering. The feature is defined by the set of all extracted focals in the most current focal mining step (Section 3.4.1). Each entry of the feature vector is an indicator of support of the scene to the corresponding focal, forming a Bag-of-Words (BoW) feature: $\mathbf{x}_i = (x_{ik})_{k=1}^{d}$. Subspace clustering is then performed over all input data represented in the feature space, $\mathbf{X} = [\mathbf{x}_i]_{i=1}^{n} \in \mathbb{R}^{d \times n}$, to extract clusters characterized by a low-dimensional subspace.

For subspace clustering, we adopt the method of Wang et al. [97] on subspace segmentation via quadratic programming (SSQP), a state-of-the-art spectral clustering based approach. The basic idea of SSQP is to express each datum $\mathbf{x}_i$ as a linear combination of all other data in the dataset, $\mathbf{x}_i = \sum_{j \neq i} z_{ij}\mathbf{x}_j$, while implicitly enforcing the coefficients z_{ij} to be zero for all $\mathbf{x}_j$ which belongs to different subspace from $\mathbf{x}_i$. To learn such a coefficient matrix $\mathbf{Z} \in \mathbb{R}^{n \times n}$, it solves the following constrained optimization problem:

$$\begin{aligned} \min_{\mathbf{Z}} & f(\mathbf{Z}) = \|\mathbf{X}\mathbf{Z} - \mathbf{X}\|_F^2 + \beta\|\mathbf{Z}^T\mathbf{Z}\|_1 \\ \text{s.t.} \quad & \mathbf{Z} > \mathbf{0}; diag(\mathbf{Z}) = \mathbf{0}, \end{aligned} \tag{3.8}$$

where $\|\cdot\|_F$ is the Frobenius norm and $diag(\mathbf{Z})$ the diagonal vector of matrix $\mathbf{Z}$. The ℓ_1-regularization term enforces sparsity of the solution, leading to feature selection for subspace clustering. The problem is a linear constrained quadratic programming which can be solved efficiently. The resulting coefficient matrix then forms an affinity matrix, $|\mathbf{Z} + \mathbf{Z}^T|/2$, based on which spectral clustering is applied to obtain the clustering result. To automatically determine the number of clusters, we employ self-tuning spectral clustering [112]. In practice, the cluster count is relatively stable throughout the iterations since the structure of the BOW feature matrix does not change significantly.

Besides the clustering result, we need to identify the representative focals which characterize the clusters. For each cluster $\mathcal{C}_\ell$, we identify a set of representative focals, denoted as $\mathcal{R}_\ell$. We rank the importance of all focals supported by any structural graph in the cluster based on their discriminant score $\eta_{\ell k}$; see Equation (3.7). The top ranked focal is selected as the representative one. We select the top focals from the list until the i-th one, when there are over $p^{\mathrm{c}} = 80\%$ of the structural graphs in the cluster which support these top i focals simultaneously.

Our ultimate goal is to maximize the compactness of all clusters based on a scene-to-scene similarity emphasizing their representative focal points. The subspace clustering above is based on indicator features, which capture the *occurrence* of the focals but are not sufficiently informative to reflect the actual scene similarity. Directly incorporating focal-centric scene similarity into the subspace clustering is infeasible since the representative focals are unknown before the feature selective clustering is performed. Therefore, we propose an *iteratively reweighted subspace clustering* process to gradually produce more compact clusters where the compactness is measured based on the focal-centric graph kernel (FCGK).

Focal-centric graph kernel. Given a cluster $\mathcal{C}_\ell$, its compactness is defined as the average distance between all pairs of structural graphs belonging to it, measured by the FCGK:

$$\kappa_\ell = \frac{1}{n_\ell^2} \sum_{G_i, G_j \in \mathcal{C}_\ell} k_G^p(G_i, G_j), \tag{3.9}$$

where $k_G^p(\cdot, \cdot)$ is the *weighted* p-th order walk graph kernel:

$$k_G^p(G_i, G_j) = \sum_{r \in G_i, s \in G_j} \lambda_{r,s} k_R^p(G_i, G_j, r, s). \tag{3.10}$$

$k_R^p(G_i, G_j, r, s)$ is the p-th order rooted-walk graph kernel [23] which we briefly review below for completeness. It compares nodes r and s, in graphs G_i and G_j, respectively, by comparing all walks of length p whose first node is r against all walks of length p whose first node is s:

$$k_R^p(G_i, G_j, r, s) = \sum_{\substack{(r_1, e_1, \ldots, e_{p-1}, r_p) \in W_{G_i}^p(r) \\ (s_1, f_1, \ldots, f_{p-1}, s_p) \in W_{G_j}^p(s)}} k_{\mathrm{n}}(r_p, s_p) \prod_{i=1}^{p-1} k_{\mathrm{n}}(r_i, s_i) k_{\mathrm{e}}(e_i, f_i),$$

where $W_G^p(r)$ is the set of all walks of length p originated from r in graph G. The node kernel k_{n} takes both geometry and label comparison into account, similar to [23], except that we used a single label for each object, instead of a series of semantic tags. For edge

kernel k_e, we use the similarity of spatial arrangement (Equation 5.8), instead of a binary comparison of edge types.

For the walk kernel κ_ℓ to be focal-centric, we set higher weight for those rooted walks which originates from a node in a representative focal of cluster $\mathcal{C}_\ell$:

$$\lambda_{r,s} = \begin{cases} 1 + \lambda \cdot \eta_{\ell k} & \text{if } r \in G_i(F_k), s \in G_j(F_k) \text{ and } F_k \in \mathcal{R}_\ell \\ 1 & \text{otherwise} \end{cases}$$

where λ is a scaling factor. In our algorithm, we set $\lambda = 100$ which is fairly high and emphasizes more the role of focals in scene characterization than the overall scene similarity.

Iteratively reweighted subspace clustering. For a structural graph G_i, we weight the individual dimensions of its BoW feature vector by a weight vector $\mathbf{w}_i = (w_{ik})_{k=1}^d$ and solve a weighed subspace clustering which minimizes the error of linear approximation in Equation (3.8) under a *weighted* Frobenius norm. Specifically, we replace the first term in Equation (3.8) by:

$$\|\mathbf{XZ} - \mathbf{X}\|_{W,F}^2 = \sum_{i=1}^{n} \sum_{k=1}^{d} w_{ik}^2 [(\mathbf{XZ})_{ik} - \mathbf{X}_{ik}]^2. \tag{3.11}$$

The weights allow us to tune the importance of the individual dimensions when seeking subspaces and can be utilized to iteratively shift clustering results. For example, one can increase the weights corresponding to the dimensions spanning the subspace of a cluster obtained in the last round, to reinforce the cluster in the current clustering. In our case, we encourage the reoccurrence of the compact clusters in the next iteration by increasing the weights of the dimensions corresponding to its representative focal points, and deprecate incompact clusters by decreasing their corresponding weights.

Initially, the weights in $\mathbf{w}_i$ are set uniformly to 1. In each iteration, we perform the weighted subspace clustering and then update $\mathbf{w}_i$ based on the compactness of the cluster to which G_i belongs; see Algorithm 3. For each member of a cluster, we compute the weights of the dimensions corresponding to the representative focals of the cluster based on cluster compactness and focal point discriminant score (Line 10). If a focal is not a representative one for any cluster, we set a 0 for the corresponding dimension of the weight vector for all structural graphs (Line 11-14). The stopping criteria for this iterative process is the same as the one used during the interleaving optimization, i.e., the change of overall cluster compactness.

Figure 3.8 demonstrates the process of reweighted subspace clustering with a mini-experiment on 8 structural graphs with 5 focals. In the experiment, after obtaining the subspace clustering along with the representative focals, the weights corresponding to focal point F_3 and F_4 are decreased, due to low discriminant score and low cluster compactness, respectively. With the updated weights, G_2, which was originally clustered into the blue

Algorithm 3: Iteratively Reweighted Subspace Clustering

Input : structural graphs $\mathcal{G} = \{G_i\}_{i=1}^n$,
BoW features: $\mathbf{X} = [\mathbf{x}_i]_{i=1}^n$, $(\mathbf{x}_i = (x_{ik})_{k=1}^d)$
weights: $\mathbf{W} = [\mathbf{w}_i]_{i=1}^n$, $(\mathbf{w}_i = (w_{ik})_{k=1}^d)$

Output: subspace clusters $\{\mathcal{C}_\ell\}_{\ell=1}^c$

```
for i = 1 to n do
    w_i ← 1;
repeat
    {C_ℓ}_{ℓ=1}^c ← SubspaceClustering(G, X, W);
    for ℓ = 1 to c do                                   // update weights
        R_ℓ ← RepresentativeFocalSet(C_ℓ);
        κ_ℓ ← Compactness(C_ℓ, R_ℓ);
        foreach G_i ∈ C_ℓ do
            foreach F_k ∈ R_ℓ do
                w_ik ← n_ℓ · κ_ℓ · η_ℓk;
    for k = 1 to d do
        if F_k ∉ ⋃_{ℓ=1}^c R_ℓ then
            for i = 1 to n do
                w_ik ← 0;
until the overall compactness Σ_{ℓ=1}^c n_ℓ κ_ℓ does not improve;
return {C_ℓ}_{ℓ=1}^c;
```

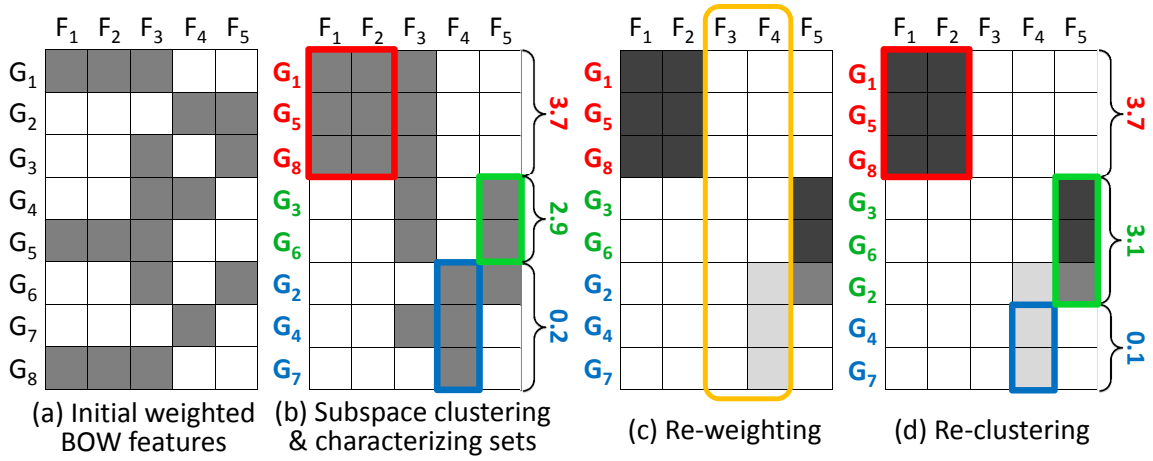


Figure 3.8: A mini-experiment on reweighted subspace clustering. The weighted BoW features are shaded in grey level (dark=large; light=small). From the initial BoW features (a), subspace clustering produces three clusters (colored) along with their representative focals (marked in corresponding color). The colored numbers indicate the compactness values of clusters. F_3 is not discriminant as it appears across three clusters (b) so in (c) the corresponding weights are set to 0. The weights for F_4 are decreased due to the low compactness of the blue cluster. The next clustering groups G_2 into the green cluster with F_5 as the representative focal point (d).

cluster due to F_4, is now grouped into the green one characterized by F_5. This is because F_5 plays the major role in clustering G_2 after F_4 is deprecated. After reweighting, the weighted feature vector of some structural graphs may decrease to (or close to) $\mathbf{0}$ vector (e.g., G_4 and G_7 in Figure 3.8). Since the clustering of these structural graphs is quite unpredictable, we choose to leave them out when their weight vector vanishes, to make the iterative clustering converge faster. These structural graphs are later introduced back in the beginning of the next round of interleaving optimization.

3.4.3 Cluster attachment and focal joining

Cluster attachment. Spectral clustering produces a partition of an input dataset, which does not reflect potential cluster overlapping due to scenes which exist in multiple clusters. In general, a structural graph for an input scene which support multiple focals may belong to multiple clusters that have other different representative focals. We simply attach such clusters with respect to the shared scenes, which can be easily identified, to reveal the overlap.

Focal joining. As subgraph mining is performed on structural graphs whose node connections only capture local proximity, it is unable to return large-scale and non-local substructures. This issue has been observed in the recent work of Xu et al. [107] which is based on structure group detection over the structural graphs. In our work, frequent substructure detection is coupled with subspace clustering. This enables us to combine the extracted focals to form a larger and non-local substructure, through analyzing the clusters they characterize. Suppose that F_1 and F_2 are both representative focals for some cluster $\mathcal{C}_\ell$. If their supporter sets in $\mathcal{C}_\ell$, denoted as $\mathcal{S}_{\ell 1}$ and $\mathcal{S}_{\ell 2}$, overlap sufficiently, i.e., $|\mathcal{S}_{\ell 1} \cap \mathcal{S}_{\ell 2}| > 0.9 \min\{|\mathcal{S}_{\ell 1}|, |\mathcal{S}_{\ell 2}|\}$, we join them, by a union of their nodes, to form a larger substructure F_{12} as a representative focal for $\mathcal{C}_\ell$.

3.5 Results

We present results obtained by our algorithm for focal point driven analysis of indoor scene collections. For scene retrieval, we compare our results to those obtained from state-of-the-art methods both through precision-recall curves and a preliminary user study, targeted for hybrid scenes.

Datasets. The datasets we experiment on were provided by the Stanford repository [22] and the Tsinghua repository [107]. Both datasets contain semantic tags with the objects originally collected from 3D Warehouse. Since the tags from the two datasets are inconsistent, we run our test on each dataset separately. For each scene, we remove the walls and focus only on the interior scene objects. The Stanford collection consists of 132 scenes

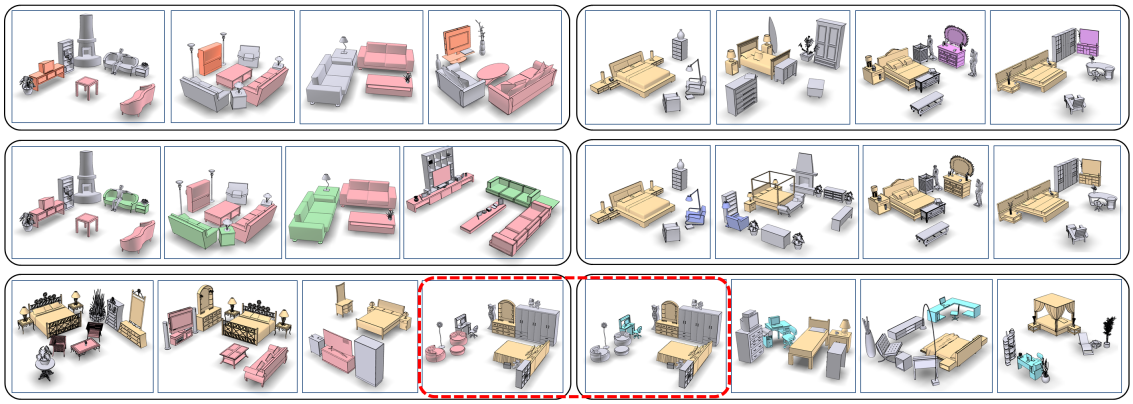


Figure 3.9: Several clusters and their representative focals (highlighted in colors) extracted from the Tsinghua scene collection. Top row shows an intermediate result for two clusters and the middle row shows the final result for the relevant clusters. Bottom rows show the final result for other clusters. Note multi-focal hybrid scenes, cluster overlap (marked with the red dashed box), and non-local focal points, such as the combos of {TV, TV-stand, table, sofa} and {bed, nightstands, dresser, mirror} in the last two rows.

and 3,461 objects, encompassing 78 object categories and five labeled scene categories. The Tsinghua dataset consists of 792 scenes and 13,365 objects, encompassing 119 object categories and six labeled scene categories. The Tsinghua dataset contains 102 hybrid scenes which is composed of many subscenes, each representing a room.

Parameters and statistics. The key parameters of our algorithm include: the minimum support s_{min} used for frequent substructure mining in the first iteration, and the rooted paths combination weights used in computing graph kernel. s_{min} controls how frequent the subgraphs should appear in scenes in order to be extracted as candidate focals. All the results reported in this work were obtained with the same parameter setting: $s_{min} = 40$ for Tsinghua dataset and $s_{min} = 20$ for the Stanford dataset. The parameters for graph kernel use the optimal ones available from the published work of Fisher et al. [23]. Values for all other parameters are fixed throughout and described in Section 3.4.

Statistics and timing. Table 3.1 shows some statistics from focal point extraction and scene clustering. Timing wise, it took 10.5 minutes to process the whole Tsinghua dataset (792 scenes) and 3.2 minutes for the Stanford scene collection (132 scenes). Over an iteration, compactness evaluation (including FCGK computation) takes ~60% of the time, with spectral clustering ~30%, and inexact frequent pattern mining ~5%. Note that the first two parts were both implemented in Matlab and could see significant speed-up if coded in C/C++. Timing is measured on a 4 quad-core 2.80GHz Intel Core CPU with 12GB RAM.

Focal point extraction. Figure 3.9 shows several clusters and their representative focal points extracted from the Tsinghua collection; the complete set of results for focal extraction

Collection	$\#f$	$\#nlf$	f_{min}	f_{avg}	f_{max}	$\%mf$
Stanford	24	4	2	3	6	50.4%
Tsinghua	34	7	2	3	5	46.1%

Table 3.1: Statistics for focal point extraction. #f denotes the total number of focals and #nlf that of non-local ones. The minimum, average, and maximum number of objects in an extracted focal is denoted by f_{min}, f_{avg}, and f_{max}. %mf is the percentage of multi-focal scenes over the whole collection.

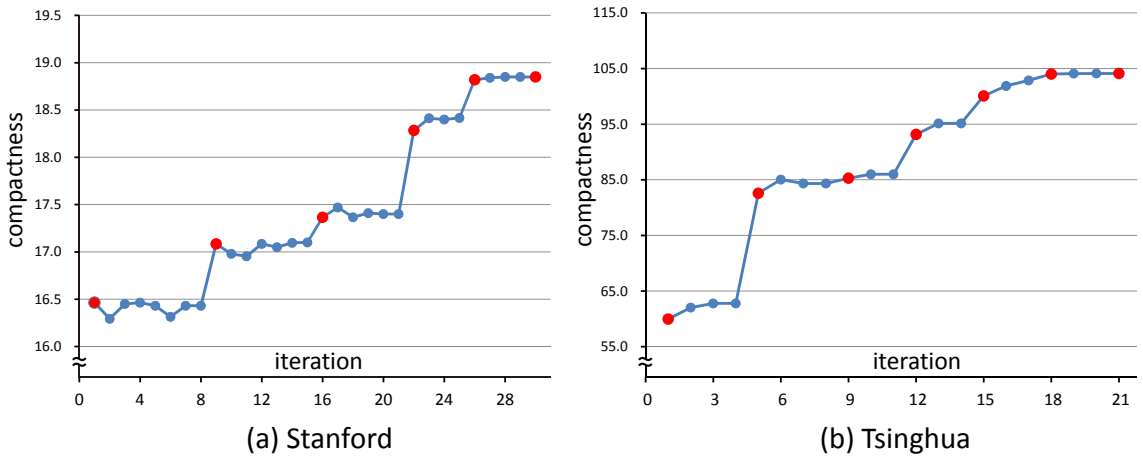


Figure 3.10: The plots show the change of the compactness of the clusters obtained as our interleaving optimization progresses, for Stanford and Tsinghua datasets respectively. The red dots represent the switching points from outer loop (mining) to inner loop (clustering). The optimization takes 5 and 6 interleaving iterations to converge on the two datasets, respectively.

can be found in the supplementary material of our paper [103]. We can observe hybrid scenes containing multiple focal points, which is fairly typical and results in cluster overlap. Also worth noting is the extraction of non-local focals, which are composed of relatively distant object groups, e.g., {*TV, TV-stand, table, sofa*}, etc. Table 3.1 gives the number of non-local focals extracted for both datasets. See also the last two rows in Figure 3.9 for the effect of focal joining.

Iterative clustering. Figure 3.10 plots how the normalized compactness of the clusters change as the iterative clustering algorithm progresses. While the change is not strictly monotone, it is evident that the iteration generally improves cluster quality over time. The final cluster counts for the two sets are 5 and 9, respectively.

Precision-recall on scene retrieval. Figure 3.11 compares our method to two other methods for scene retrieval:

1. **GK**: Graph kernels of Fisher et al. [23] to measure similarity between whole scenes. Since we were unable to obtain the authors' code, we coded up our own implementation

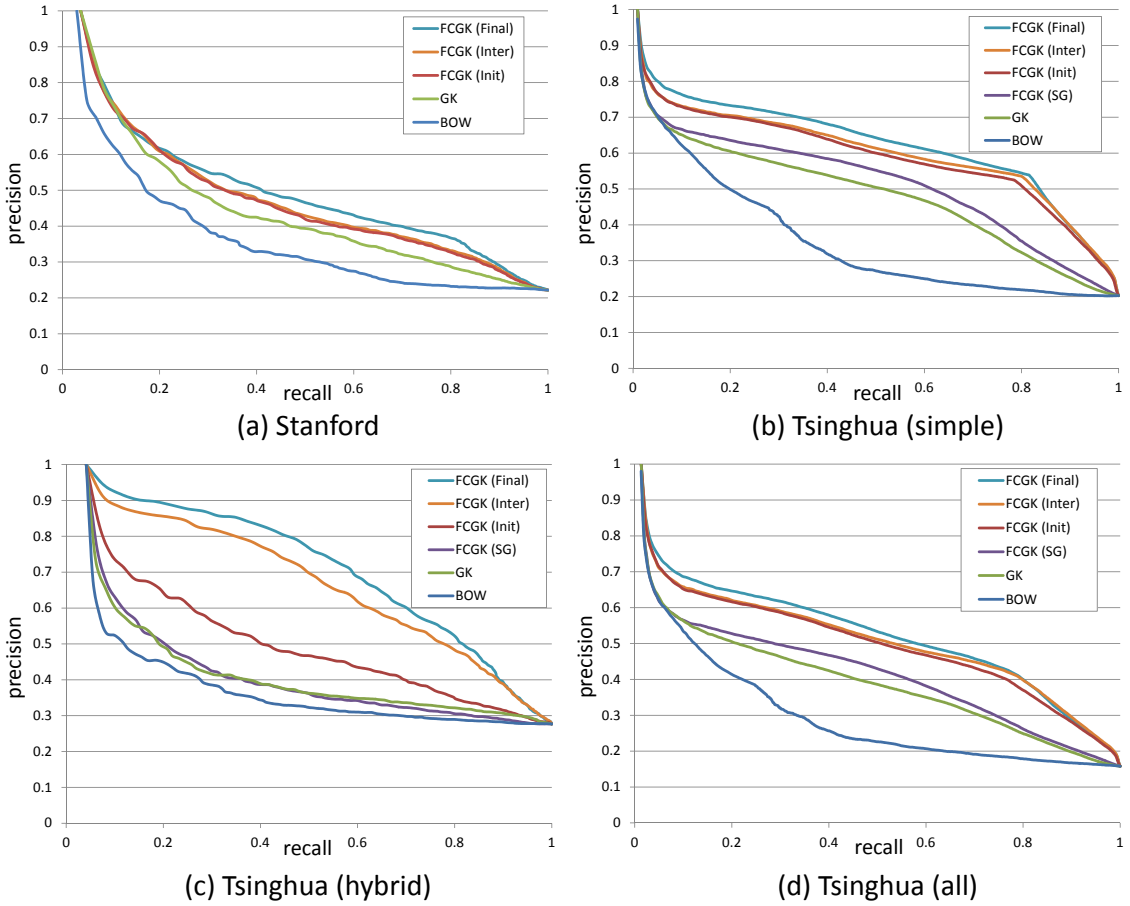


Figure 3.11: Precision-recall curves for scene retrieval. (a) Stanford scene collection. (b) Tsinghua collection, simple scenes. (c) Tsinghua, hybrid scenes. (d) Tsinghua, all scenes.

with two major differences to the original work. First, we use our structural graphs which only encode two types of relationships (support and proximity) and do not consider hierarchical scene graphs. Second, the computation of node and edge kernels are slightly different; see Section 3.4.2. For both GK and FCGK, the schemes for node and edge kernel estimation and graph kernel normalization, as well as all the parameters, are the same as the original work.

2. **BOW**: A baseline method where we use bag-of-words features on the focal points only as a scene-to-scene similarity.

3. **FCGK (SG)**: On the Tsinghua dataset, we also apply our FCGK similarity on the scenes where as focals, we use the 212 structural groups detected by Xu et al. [107].

When applying our method, which uses FCGK for scene similarity, we show results in three settings: 1) using the initial set of focals after only one step of frequent pattern mining; 2) using an intermediate set of focals; 3) using the final set of focals extracted.

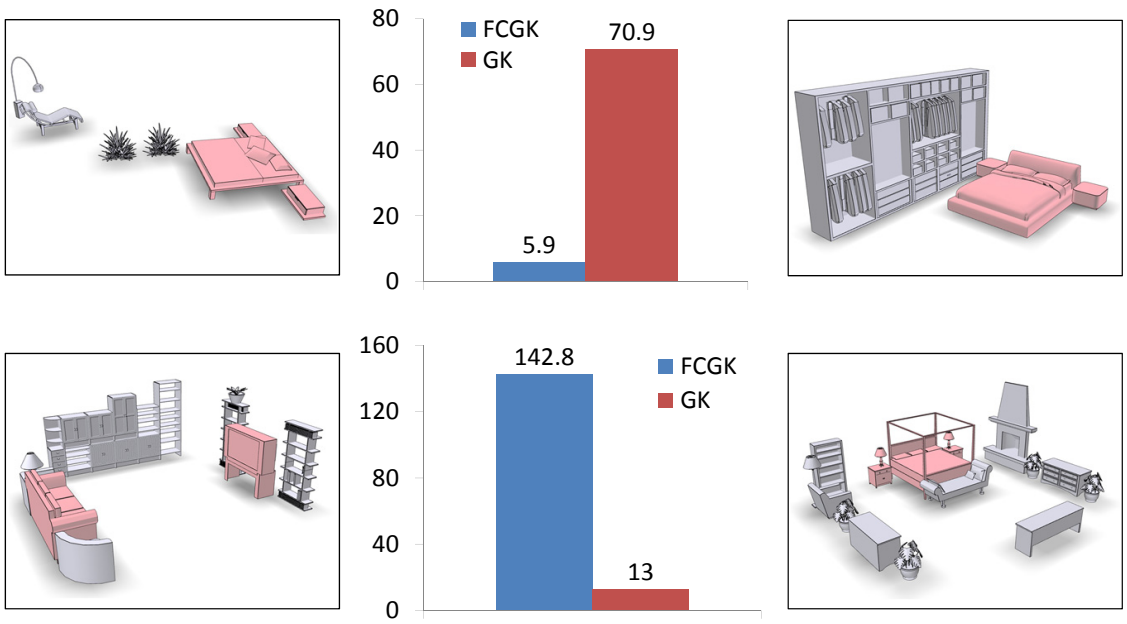


Figure 3.12: Comparing GK and FCGK on scene similarity. Top row: two scenes in the same category, but GK returns a large distance between them due to the dissimilar surrounding objects. Bottom row: two scenes belonging to different categories while GK returns a small distance also attributing to surrounding objects, e.g., the nearby bookshelves. In contrast, with a focal-centric view, our method gives more meaningful distances on the two pairs.

For the Tsinghua dataset, the ground truth for evaluating scene retrieval is given by the scene labels/categories which come with the dataset. Since this dataset contains many hybrid scenes, we separate it into a subset of simple scenes and the remaining hybrid (complex) scenes and report results on each and their combination. Since the Stanford collection does not come with scene labels, we provide our own labels obtained manually, which, admittedly, could introduce an evaluation bias. A potentially more reliable method, such as voting from multiple users, could be employed.

From the precision-recall curves, we see that our focal-centric similarity based on the final set of focals is the best in all four cases. Moreover, the performance gain is more prominent for hybrid scenes. These results demonstrate not only the merit of utilizing focals for scene comparison but also the merit of our focal extraction scheme, as it seems evident that retrieval performance improves as our iterative algorithm progresses.

Comparison to GK. Figure 3.12 shows an explicit comparison between GK and FCGK on scene similarity, attesting to the effectiveness of utilizing focals. In our experiment, we also observed that the matching performance of GK tends to be negatively affected by the presence of many small/trivial objects. For example, when a scene contains a shelf supporting many small objects, GK counts rooted walks from all these objects, which would

influence the similarity between more prominent objects. FCGK is more discriminative and trivial objects are less likely to have been chosen as focals.

User evaluation on retrieval. For a hybrid scene, it may be difficult to assign an unambiguous category label. The ground truth used for retrieval on such scenes may be unreliable. Thus instead of relying on scene categories as ground truth, we let human users judge scene similarity based on their prior knowledge. In this second comparative study on scene retrieval, we focus exclusively on retrieval where the query is a hybrid scene. We present a user with 10 queries. For each query, the top return from the three compared methods (GK, BOW and FCGK) are presented to the user and the user is asked to choose which of the three is most similar to the query. We repeat this for a total of 102 queries for the hybrid scenes in the Tsinghua dataset. Against GK, we obtain a winning percentage of **70.2%** and against BOW, we obtain **73.9%**. The results are statistically significant (with $p = 0.01$). In the studies, each scene has been rendered in three random bird's eye views and the images were presented randomly. Among the 43 participants, 80% are computer science researchers, with ages 20 to 50. The rest are frequent computer users with varying backgrounds.

3.6 Applications

Our scene organization allows classical scene queries and is thus suitable for any application which utilized retrieval results as before, e.g., [22, 107]. In this section, we discuss several new capabilities afforded by our focal-based data organization for scene retrieval and exploration.

Comprehensive retrieval. In classical retrieval, a single query would fetch a single ranked list of data items. With our focal-centric similarity and pre-computed set of focals, our scene organization supports such classical queries. It also supports part-in-whole type of queries, where the user specifies a region of interest (ROI) in the query scene. This is demonstrated with the exploration tool which we describe below. The interesting new feature enabled by our scene organization is what we call *comprehensive retrieval*. Here the query does not have a specified focal. However, the available focals in the organization are matched with the query scene. Instead of returning a single ranked list of scenes, the comprehensive retrieval returns *multiple* ranked lists, each of which corresponds to a well-matched focal. Figure 3.13 shows such a result. Note that the vertical order in the table has no clear meaning since the three (horizontal) lists are retrieved based on different sets of focals. If putting all the results together, however, one can expect that those retrieved with multiple focals should be ranked higher since they have more focal substructures receiving higher weights; refer to Equation (3.9).

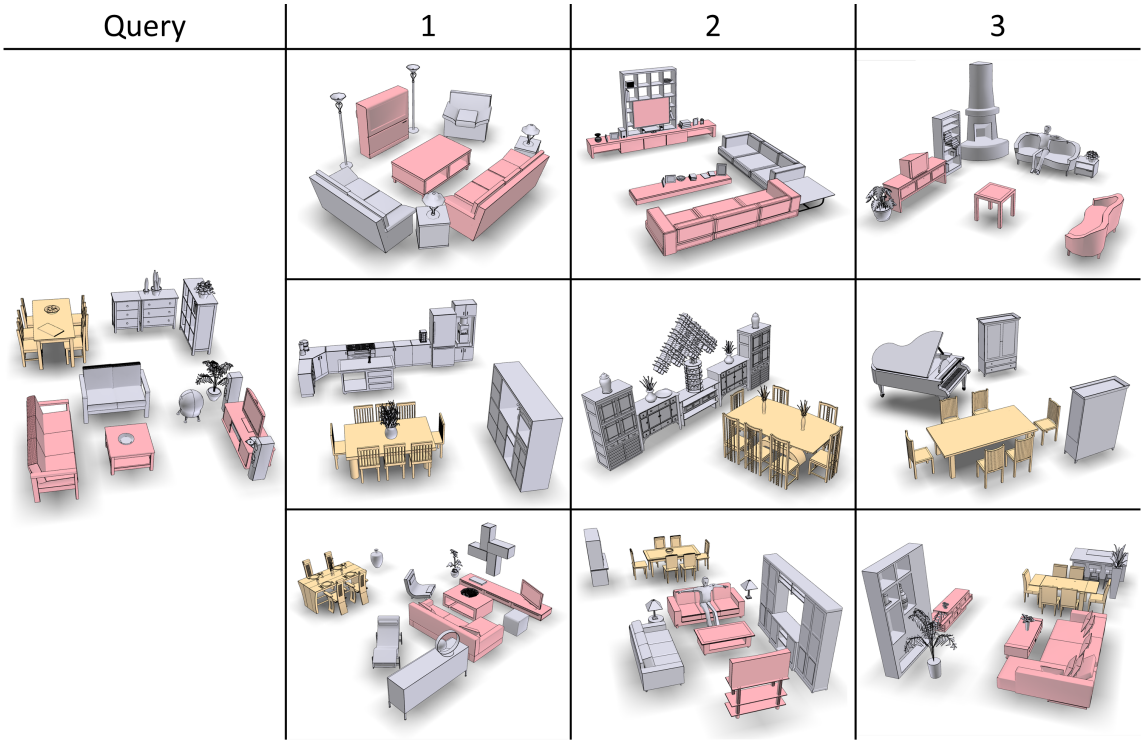


Figure 3.13: Comprehensive retrieval takes a query scene and returns scenes grouped by well-matched focals with the query. In each group, the returns are ranked by FCGK based on the corresponding focal. In this example, the query has two focals (colored yellow and red) matched from the scene organization. Three ranked lists of returns corresponding to the two focals (first two rows) and to the *joined* focal (bottom row) are shown.

For focal-to-scene matching, we utilize the efficient subgraph matching approach described in [71], by which the focal subgraphs are pre-compiled into a hierarchical representation to accelerate the online matching. The average query time is 960ms for the Tsinghua collection and 140ms for the Stanford set.

Multi-query retrieval. In applications such as example-based scene synthesis [22], one may form queries consisting of multiple *semantically related* scenes and wish to retrieve more scenes "of the same". Such *multi-query* retrievals are well-supported by our scene organization. Indeed, since the query scenes are related, they likely share meaningful substructures, making them suitable for focal-based scene comparisons.

Given a query set, we extract frequent substructures from the set and match them against the extracted focals in the scene organization. We then retrieve scenes from the organization using FCGK based on the matched focals. Figure 3.14 shows one such result with a query set of four hybrid scenes. For comparison, we also show a ranked list of returns based on GK similarity measured against any scene in the query set. As one would expect, the focal-based retrieval produces more discernable results, and more useful results. If the

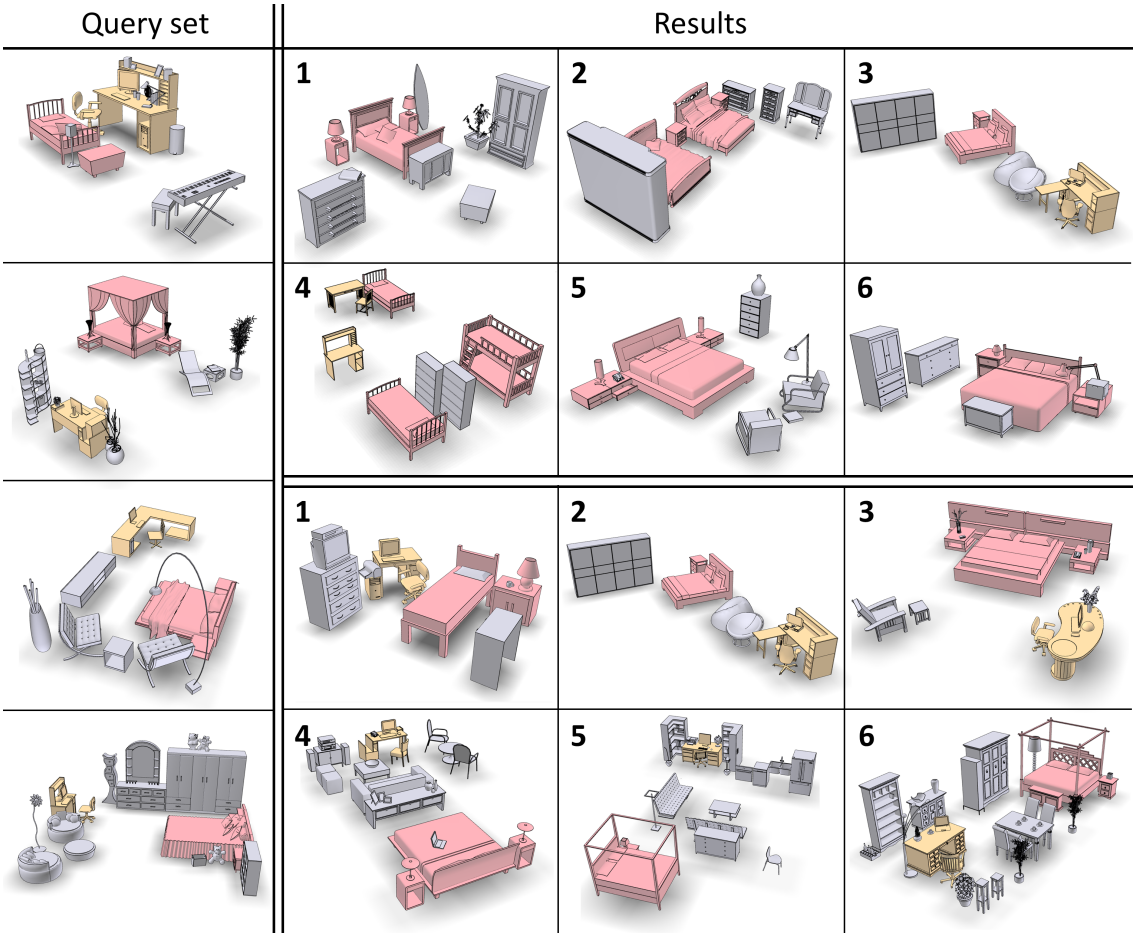


Figure 3.14: Multi-query retrieval takes a query set (left) and returns a ranked list of scenes (bottom-right) via focal-based scene comparison. FCGK similarity is used and measured based on focals (colored red and yellow) that well-match frequent substructures in the query set. Returns based on global scene similarity computed by GK are also shown (top-right). To not introduce a bias by coloring of the focals, in the GK returns, we also color any object whose tag matches that of an object in one of the focals.

user selected four query scenes all containing a bed-nightstand combo and a desk-chair combo, then it is likely that he/she was seeking scenes that contain similar substructures.

Scene exploration. We develop an exploration tool, based on the extracted focals, which enables a user to browse through a heterogeneous scene collection. Focal points are the primary means for search and navigation. Figure 3.15 shows the GUI of our tool. The user can select a few focals from the focal point list panel (bottom), and our tool automatically selects a set of scenes sharing similar focals and lists them in the scene list panel (right). The user can browse the list and view the scenes in the main viewer (middle). At any time, the user can click on a selected focal to view its embedding in the current scene. In terms

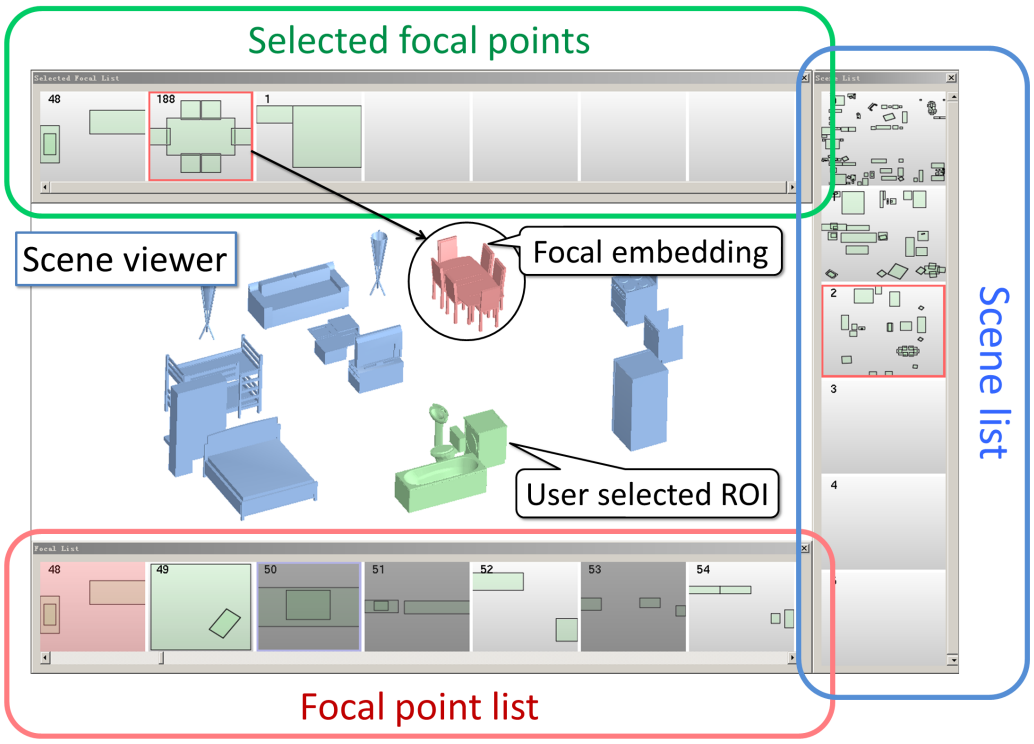


Figure 3.15: GUI for our exploration tool is composed of four parts: the focal point list panel (red box), the selected focal list panel (green), the scene list panel (blue), and the main scene viewer. The user can pick a selected focal to view its embedding in the current scene. She can also select a region of interest (ROI) in the viewer to explore more scenes via the focals around the ROI.

of navigation, as shown in Figure 3.3, the user can traverse from one scene to another, and one scene cluster to another, through focals which interlink them.

In addition, we provide an interface for the user to paint a region of interest (ROI) and search for scenes which contain sub-scenes that are similar to the surroundings of the ROI. When the user selects an ROI in a scene, our system first finds a focal point in the scene which overlaps most with the ROI and adds the focal to the selected list. It then retrieves a new list of scenes based on the updated list of selected focals. Exploring the database with focal points around an ROI, instead of with only the ROI, can provide more relevant results. For example, if the user selects only a chair model as ROI, naive partial matching would simply return all scenes containing a chair. In contrast, our tool searches for scenes sharing the same focal around the chair, returning results that are more context-aware.

Note that the rooted walk graph kernels of Fisher et al. [23] could also support contextual part-in-whole queries. However, performing subgraph search is likely too time consuming for online retrieval. With pre-analysis resulting a focal-based scene organization, our tool can support efficient context-aware partial matching over a large heterogeneous scene collection.

3.7 Discussion and future work

At the core of the data organization problem is the mechanism for comparing data. Traditional approaches rely on holistic data views and unique distances defined between data items for grouping or clustering. However, when the data become complex and multi-faceted, a fixed and global view on data similarity can hardly express the rich characteristics in the data.

We advocate the use of focal points for comparing and organizing complex and heterogeneous data and use 3D indoor scenes as a prototype to demonstrate its feasibility and performance gains, e.g., in retrieval. The new approach seems particularly apt at dealing with complex and hybrid scenes. Perhaps its most compelling feature is the ability to process large and heterogeneous collections of scenes and to organize them into an interlinked and well-connected cluster formation, which facilitates scene exploration.

FCGK vs. GK. While our retrieval experiment showed superior performance of FCGK over GK, one should realize that a direct comparison between the two is not exactly fair. GK is a standalone graph similarity measure, where only two graphs to compare are needed. FCGK-based comparison comes with a higher cost as it requires a set of graphs and a co-analysis for focal extraction. That said, if a scene collection is available, we would still suggest using FCGK for its better performance and modest processing costs.

Comparison to structural groups. In our work, a focal point consists of a group of scene objects and it is derived via structural scene analysis. By name alone, this suggests similarity to the structural groups computed by Xu et al. [107]. There are however major differences. First, their structural groups are *category* groups with objects of high co-occurrence frequency, while our focals are object groups corresponding to representative substructures. More importantly, their group extraction involves only frequent pattern mining through local proximity based search. The latter implies that their method is unlikely to return non-local structural groups. This is in part evidenced by the much higher number of groups (212) they obtain vs. the 34 focals we obtain, on the same scene collection (Tsinghua, 792 scenes). The retrieval results in Figure 3.11 seem to suggest that non-local focals extracted via mining and clustering provide the better perspectives for meaningful scene comparison.

Non-unique distance. The retrieval experiment using FCGK seems to suggest that our method assigns a unique distance between any two scenes. This is true once the set of focals is fixed and FCGK is to be computed based on those focals and the clustering result. However, the non-uniqueness of focal-centric distances is well utilized in other settings including comprehensive retrieval, multi-query retrieval, and ROI-driven scene exploration, where the relevant focals in the query scenes are all determined on-demand.

Limitations. Our current algorithm depends on semantic labeling of scene objects. It remains to be seen whether it works effectively with noisy or incomplete labels, based on pure geometry analysis. For example, it is interesting to test our method on inputs with various levels of label noise. However, it would be hard to quantitatively evaluate the robustness against noisy labels since it may be difficult to reproduce realistic labeling noise introduced by humans. Nevertheless, the two datasets we used do contain some incorrect labels, which did not seem to affect the overall performance. There are perhaps more than a desirable number of parameters in the algorithm, whose values were determined experimentally. From a technical stand point, improvements are possible in various components of the algorithm. For example, our layout similarity operates on OBBs only, which may be unsuitable for objects with complex geometry and spatial arrangements. The structural graphs model the scenes only as flat arrangements of objects. Hierarchical organization may be potentially advantageous.

Future work. One obvious pursuit is to apply our focal-driven approach to other datasets, e.g., large and heterogeneous collections of annotated images. An interesting technical question is whether our scene organization can be updated with an additional set of scenes without recomputing everything. Also, rather than replacing one object at a time for scene synthesis like in previous works, our scene organization and focal-based partial scene retrieval, may allow for substituting sub-scenes for the synthesis task.

We conclude this chapter with a question: "what is the best way to compare complex scenes?" This work, along with others before it, assume that comparing attributed graphs defined by semantic tags and object arrangements is the best way. However, we observe that visually, many retrieval results do not look so compelling even with the best method to date. If one takes away the colorings in Figure 3.14, then the contrast between GK and FCGK would not be as salient. Hence, the focal-centric view we advocate offers a perspective worth considering. The general question, also one that is attributed to complex data beyond those of indoor scenes, should perhaps be answered with user and application intent in mind.

Chapter 4

Action-Driven Scene Evolution

In this chapter, we introduce a framework for *action-driven evolution* of 3D indoor scenes. Our goal is to simulate how scenes are altered by human actions. To this end, we develop an *action model* with each type of action combining information about one or more human poses and a set of objects belonging to sub-scenes. We apply actions to trigger appropriate object placements, based on object co-occurrences and spatial configurations learned for the action model. We show results of our scene evolution that lead to realistic and messy 3D scenes, as well as quantitative evaluations by user studies which compare our method to manual scene creation and state-of-the-art, data-driven methods, in terms of scene plausibility and naturalness.

4.1 Introduction

We live in a 3D world and we constantly act on and interact with the 3D scene environments that surround us. The scenes evolve over time, driven by object movements resulting from human actions. It seems natural to ask whether digital 3D scenes can be processed in such an *action-driven* manner. With an increasing demand of 3D scene data, especially those of indoor environments, from emerging VR/AR applications to data-driven scene analysis, techniques for scene generation are drawing more attention in the graphics and vision communities. A method for action-driven scene evolution aims to replicate how indoor scenes evolve in real life, producing continuous series of realistic virtual 3D scenes.

Most indoor scenes available from public data repositories, e.g., the 3D warehouse, possess the organization and cleanness of a showroom; these scenes were mostly designed. In real life, our offices, labs, and bedrooms are often messier. So far, aside from scene construction from images [56, 21] and sketches [107], the predominant approach to realistic scene synthesis has been based on exemplar-based learning [22], where a scene is produced by sampling from a probabilistic distribution learned from 3D scene examples. In real life

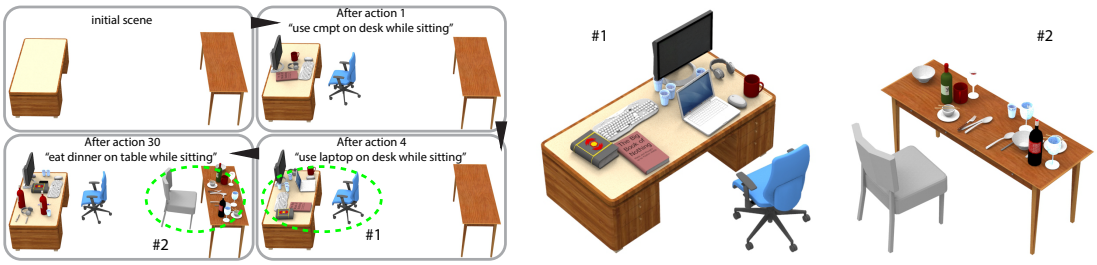


Figure 4.1: Action-driven scene evolution alters an initial scene consisting of a desk and a dining table. The initial scene and three intermediate evolution snapshots (after 1, 4, and 30 actions, respectively) are shown (left) with zoom-ins for better viewing on the right. Applied actions trigger both object relocation (e.g., keyboard and headphone) and insertion (e.g., laptop and books). Action selection and the resulting 3D scene evolution are all performed automatically based on action data learned from annotated photos.

however, a scene is not a random whole "event", but a snapshot from a continuous scene evolution.

Motivated by these observations, we introduce a framework for action-driven evolution of 3D indoor scenes, where the goal is indeed to simulate how scenes are altered by human actions, and specifically, by object *placements necessitated* by the actions. In our work, an action can involve one or more objects and one or more humans (e.g., a group meal). Object placements for a given scene can involve either *relocating* existing objects or *inserting* new objects into the scene. For instance, applying the action "use laptop on desk while sitting" to a scene without a chair near the desk would cause a chair to be moved there to support the action. Applying the action "eat dinner on table while sitting" would trigger the insertion of several objects, e.g., plates and forks, to an otherwise empty dining table.

We develop an *action model* which supports action-driven 3D scene evolution. The model is *data-driven* and learned from annotated scene data. Each type of action combines information about one or more human poses, one or more object categories, and spatial configurations of objects belonging to these categories which summarize the object-to-object and object-to-human (the pose in the action model) relations for the action. *Correlation* between the learned actions are analyzed to guide the construction of an *action graph*, whose nodes correspond to actions and edges encode correlations between actions in the form of transitional probabilities. Scene evolution starts with an initial 3D indoor scene. We probabilistically sample an action sequence from the action graph, where each action triggers appropriate object placements, which continuously evolve the scene; see Figures 4.1 and 4.4.

To account for the limitation of insufficient 3D indoor action data [21], we learn action models from a database of annotated photographs, i.e., Microsoft COCO [53]. First, we analyze the photo captions therein to collect photos related to certain actions. Then, photos of the same action are clustered based on object categories and human poses within the

photos. Each cluster defines an action node in our action graph. Object co-occurrences, object-human, and object-object spatial relations are learned within each action node. For the action graph, edges are added to all pairs of action nodes, and also a self loop to the node itself. We compute the transitional probabilities by examining the overlap of associated objects. From the action graph, plausible action sequences are generated automatically and stochastically. Each action applied triggers appropriate object relocations or insertions, based on object co-occurrences and spatial configurations learned for the action model.

Example-based synthesis [22] is also data-driven, but it takes a holistic view of scene generation: the produced scene must be similar to the exemplars overall and likely belonging to the same scene category. In contrast, action-driven evolution is a *procedural* and more atomic form of scene generation that is not tied to scene categories; the key criterion is what actions are applicable in a given scene context. For example, the action "read book on desk while sitting" is applicable in any scene with a desk. No less important is the fact that action data is more compact and more atomic than whole scene exemplars. Applying actions one at a time allows more local control and generates scenes with higher granularity.

To summarize, by taking an action-driven approach to indoor scene generation, our work offers a more atomic and fine-grained view of the problem. Our contributions include:

- A progressive approach to scene generation which leads to an evolving and granular set of 3D scenes exhibiting a higher level of scene complexity and messiness than previous works, without compromising plausibility and naturalness.
- Action learning from *annotated photos* rather than 3D scene exemplars in previous works. This enables us to tap into a much richer data source for action-driven scene processing.
- A more complete action model which accounts for group actions, as well as co-occurrences and joint placement of multiple objects, allowing both object relocation and insertion.

We show results of our scene evolution, leading to realistic and messy 3D scenes. Evaluations include user studies that compare our method to manual scene creation and state-of-the-art, data-driven methods, in terms of scene plausibility and naturalness.

4.2 Related work

Aside from serving VR/AR applications, large collections of 3D scenes are valuable both as training data to support machine learning for scene understanding and as model repositories for model-driven 3D scene modeling [47, 82, 107, 21]. Recently, there have been a great deal of work in computer graphics and computer vision on the processing and analysis of indoor scenes, e.g., reconstruction, understanding, and editing. In this section, we only focus on

works most closely related to ours, i.e., those on 3D scene generation as well as human- or action-oriented scene processing.

Scene modeling. Interactive, user-centric tools for 3D scene modeling exist commercially, e.g., *Autodesk Homestyler* [5], *Sweet Home 3D* [3], and in the research literature. The use of such tools requires advanced modeling skills and the modeling time is often quite long. Data-driven scene modeling or reconstruction from X has gained much interest lately where X could be a sketch [107], a photograph [56, 39], or a depth scan [47, 82, 14]. In these cases, the object arrangements inferrable from X are fixed and guide the retrieval and placement of suitable 3D objects from a model repository. In our work, we are interested in a more open-ended and less constrained synthesis, not modeling from X.

Furniture layout optimization. Several approaches have been proposed for the furniture layout optimization problem. Germer and Schwarz [27] arrange a room by letting each piece of furniture act as an agent in a multi-agent system, following manually specified room semantics and furniture layout rules. Merrell et al. [63] turn furniture layout guidelines into a probabilistic model and suggest sensible room layouts by sampling from the density function, as a user interactively moves furniture in a room. In contrast, Yu et al. [110] learn the layout rules from 3D scene exemplars. All of these solutions optimize the layout of a given room with a given set of furniture. In our work, we evolve a 3D scene by moving and inserting objects progressively.

Learning from 3D data. Existing 3D scene synthesis methods predominantly resort to probabilistic reasoning from 3D exemplars and 3D scene databases to drive the synthesis [22, 43, 21, 78, 74]. Some of these methods, e.g., Fisher et al. [22], take a holistic approach to scene generation, while others progressively alter an initial scene. These approaches could all be limited by the availability of well constructed and annotated 3D scenes. For reference, the state-of-the-art work by Fisher et al. [22] worked with about 130 user-constructed 3D scenes. The lack of data limits both the variability and the scale of the generated scenes. In our work, we utilize thousands of photos with action-related text annotations from the COCO database. The challenge is to recover the 3D scene layout and properly embed human poses into the photos.

Action-driven scene understanding. There has been a great deal of work in robotics and computer vision, and more recently in computer graphics, on utilizing human actions for various analysis tasks. After all, humans understand the world and function in it through their actions. Works that have been applied to 3D scenes include geometry estimation [25], object labeling [42], and affordance learning [77], to name just a select few. The key problem is to fit static human poses or pose sequences into various scene contexts to understand the structure and functionality of a scene and the objects therein. In our work, such a fitting task is necessary. But our focus is not on automated analysis, but on how to organize the

Figure 4.2: Given an initial scene (a), placing a laptop based on pairwise human-object relations would rely on a human pose predicted for the chair (b). In contrast, applying the action "use laptop on table while sitting" by our method simultaneously places the laptop and rotates the chair (c), owing to the joint positions learned of all three objects in the scene: table, chair, and laptop.

fitted human poses and their surroundings, learn a suitable action model, and then apply the model for 3D scene synthesis.

Human- or activity-centric scene modeling. Jiang et al. [43] propose an interesting, *human-centric* approach to arrange objects in a room, focusing more on human-object relations rather than object-object relations. Specifically, they learn, from 3D scene exemplars, density functions which characterize how each type of object is placed relative to a human pose. The role of the density functions is similar to that of our action model for scene generation. However, one distinction is that these density functions encode *pairwise* human-object relations while an action in our model can encode *joint* relations among multiple humans and objects; see Figure 4.2 for a comparison. When arranging a room, they first infer possible human poses and then place one object at a time, based on the predicted poses and the learned human-object or object-object relations. In contrast, our actions can trigger the placement of one or more objects at a time and the placement is not predicated only by pose estimation — it accounts for both pose fitting and object co-occurrence; see Figure 4.3 for a visual example.

Sharf et al. [84] study object mobilities in 3D scenes and edit scenes by altering object arrangements and configurations (e.g., drawers opening or closing) based on their mobilities. Mobilities of objects arise from their movements due to human actions. However, they learn mobilities, again from 3D scene exemplars, by analyzing only object-object relations between reoccurring objects.

Fisher et al. [21] propose an *activity-centric* approach to functional scene modeling, which generates 3D scenes that allow the same human activities as real environments captured through noisy and incomplete 3D scans. Given an input scan, affordance analysis [77] is first performed to detect potential activity regions and activity types. Then objects relevant to the activities are retrieved and fitted to the scan over the activity regions under

Figure 4.3: With only a desk in an initial scene (a), the action "use laptop on table while sitting" can insert a chair and a laptop along with other objects; see (b) and (c) for two possible results from our action-driven scene synthesis. In contrast, without a chair in (a), a sitting pose is unlikely to be predicted near the desk solely by pose estimation, hence a chair is unlikely to be placed or retrieved.

human-object interaction priors learned from 3D scene databases. While their work focuses on modeling functionally similar scenes conditioned on a coarse geometric input scan, the input scan is not a must. The synthesis problem we address in this work has a different input and different goal while learning priors from different data sources. Our action model is learned from annotated photos with only spatial constraints, while their activity model is learned from 3D scene exemplars with semantic annotations. More importantly, their synthesis is designed to serve functional scene modeling where object placements are mainly constrained by human activities inferred from a given scene; see Figure 4.3. In contrast, our action model evolves a scene with object placements conditioned on both human-object relations and object co-occurrences.

Comparing to Savva et al. [78] which synthesizes *interaction snapshots* by sampling prototypical interaction graphs learned from real-world observations of human-object interactions captured with commodity RGB-D sensors, we learn atomic actions from annotated photos for progressive scene synthesis. If we were to place a rigged human character in a single scene synthesized in our work, the result would be an interaction snapshot. However, our goal is not to sample a single scene instance, but to produce a continuously evolving sequence of snapshots. Furthermore, we aim to produce realistic and messy scenes populated with many objects, while [78] focuses on accurately depicting the interaction between a human and few key objects.

COCO+action database. A highly related recent development is the COCO-a database established by [72]. COCO-a enriches the Microsoft COCO database with comprehensive annotations of visual actions, designed to facilitate action discrimination and scene understanding. However, more than half of the annotated actions in COCO-a happen between people, e.g., talking and playing games, and occur during sports play or outdoors. As well, human representations remain as whole segments without pose embedding or joint labeling,

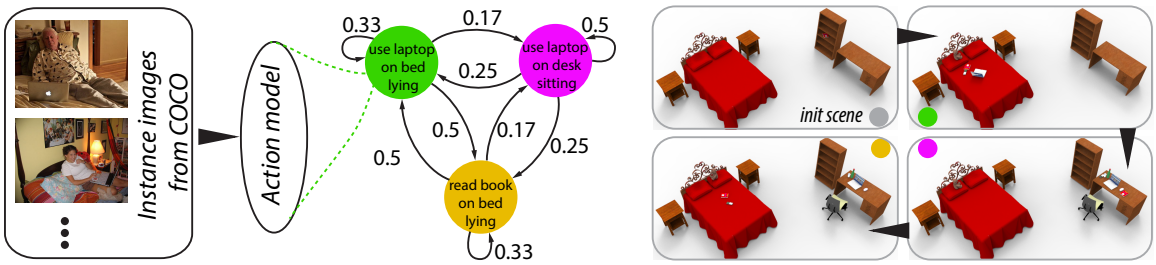


Figure 4.4: An overview of our action-driven 3D scene evolution. Action models are learned from annotated photos (left). An action graph (middle) is built, whose nodes are the learned actions, and edges represent transition probabilities between the actions. To drive the evolution of an initial 3D scene, we apply a sequence of actions sampled from the action graph onto the scene. Each sampled action triggers appropriate object placements, i.e., relocation or insertion, producing a continuous scene evolution (right).

which are necessary for action-driven object placement. For these reasons, we produced our own action-oriented annotations over photos from COCO and elsewhere that are designed to serve action-driven 3D scene evolution.

4.3 Overview

We first introduce the notations of our action model and action graph, with which we present an overview of our two-stage learning and synthesis framework; see Figure 4.4 for an illustration.

Action model. We describe an indoor scene action by the following action model $\mathcal{A} =< \mathcal{T}, \mathcal{K}, \mathcal{H}; \mathcal{C}, \mathcal{D} >$, where $\mathcal{T}$ is the action type, $\mathcal{K}$ is the key object specifying where the action happens, $\mathcal{H}$ is a representative 3D human pose, $\mathcal{C}$ stores the probability distribution of occurrence times for each object, and $\mathcal{D}$ specifies spatial configuration of constituent objects: for every object, we summarize its positional information relative to both the human pose and other objects. By definition an action model can be uniquely identified by the combination of $(\mathcal{T}, \mathcal{K}, \mathcal{H})$, which says *"what action ($\mathcal{T}$) is performed where ($\mathcal{K}$) with what pose ($\mathcal{H}$)"*. By taking all these five elements into consideration, our method can model actions occurring in rich contexts with human pose variations, e.g. *reading book lying on bed* vs. *reading book on desk while sitting*.

Action graph. This is a weighted graph $G = (V, E)$ over a set of nodes V, each of which is an action defined by the above action model. An edge $e_{i \to j} \in E$ is directed from an action node $a_i \in V$ to another node $a_j \in V$ or a node to itself, with weight $w_{i \to j}$ defining the transitional probability from a_i to a_j, i.e., how likely is action a_j going to happen after action a_i. A action graph is actually the state diagram of a Markov Chain, from which action sequences can be sampled to drive scene evolution.

System overview. As shown in Figure 4.4, our system consists of two stages: an offline action learning stage and an online scene synthesis stage. In the learning stage, we construct an action graph from action nodes learned from the Microsoft COCO database. First, we retrieve a set of instance images from the COCO database for each action type based on keyword searching, and infer 3D human pose and object layout information from each image (Section 4.4.1). We then cluster the instance images according to their associated actions – $(\mathcal{T}, \mathcal{K}, \mathcal{H})$. After that, we construct the action model for each cluster by analyzing the occurrence and spatial layouts of objects (Section 4.4.2). Lastly, we construct an action graph over the action nodes and compute edge transitional probability based on the *correlation* between action nodes (Section 4.4.3).

After the action graph is constructed, we use it to drive the evolution of a scene by applying a sequence of actions sampled from the graph – this is the online scene synthesis stage. Given an initial scene, we first adapt the action graph by disabling action nodes that cannot be applied to the scene (Section 4.5.1). Then we generate an action sequence by sampling the adapted action graph. Note that the action sequences are not learned in the learning stage; they are instances of Markov chains sampled from the action graph. The realization of an action involves placement of 3D human pose and synthesis (insertion and relocation) of corresponding objects (Section 4.5.2). In the end, we obtain a sequence of evolved scenes after applying a series of actions to a target scene.

4.4 Data-driven model construction

In this section, we describe the procedure for data-driven action learning and action graph construction. Given a set of COCO images with manually labeled human joints, our method automatically infer various action models and construct an action graph over all action nodes. As far as we know, this represents the first attempt at using 2D images to construct action models for 3D objects and scenes. Without loss of generality, we first introduce the procedure for learning actions performed by a single person (Sections 4.4.1-4.4.3). We then describe how to extend the procedure to learn group actions that involve multiple persons (Section 4.4.4).

4.4.1 Preparing action instances

The first step of our learning procedure is to collect a large number of action instances (exemplars), each of which describes the human pose, the key object, and the object-object and object-human relationships involved in the action. To this end, we take as input the Microsoft COCO database, which consists of a large set of pre-segmented, annotated photos providing the exact labels we need for extracting action instances: human-object segmentation, object category labeling, and five captions per photo that linguistically describes the scene; see Figure 4.5(a). Starting from the COCO database, our method first finds a set

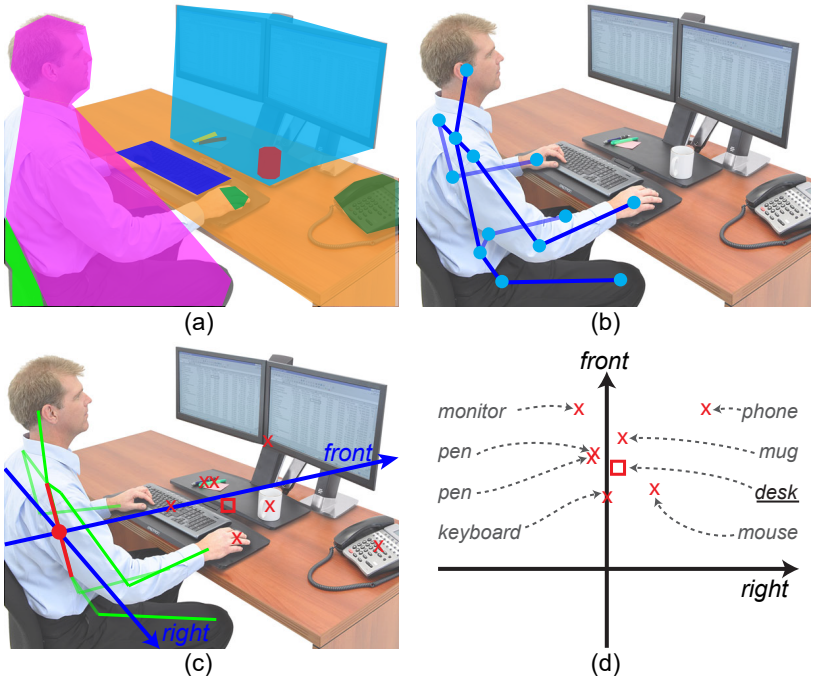


Figure 4.5: (a) A typical image from COCO with block annotation for each object. (b) The annotated 2D skeleton for 3D human pose recovery. (c) The recovered 3D pose (green) is projected onto the image, as well as its right and frontal directions (blue). The center of torso (red bar) is defined as body center. Placing the right and frontal vectors there forms a Cartesian coordinate system. Each object is reduced to a 2D point (red square for the key object and red crosses for others) located at its polygonal center. (d) All object locations are mapped into a standard Cartesian system and their polar coordinates are recorded for learning their spatial layout.

of instance images for each action type. Then, the 3D human pose is recovered for each instance with the help of manually labeled joints. After that, we infer the key object and 3D object layout in each instance with the help of recovered 3D human pose.

Instance extraction. Action type is the linguistic description of a typical activity performed by humans in an indoor environment. To start with, we predefine eight indoor action types for the experiment in this work: *use computer, use laptop, read book, prepare food, watch tv, eat snacks, eat dinner* and *eat dinner in group*. These action types are selected because the spatial layouts of their involved objects are anchored to the human poses, which is a necessary condition for our human-centric action model.

Given an action type, we retrieve relevant instance photos from the COCO database using their associated photo captions. To maximize recall, we collect synonyms and different tenses of the action type word and use them for keyword-based searching. For example, for action *use laptop*, the set of keywords we used include *use laptop, using laptop, work laptop, working laptop, operate laptop, operating laptop, utilize laptop, utilizing laptop*. Some of the returned photos may contain multiple irrelevant persons in the background, or too few

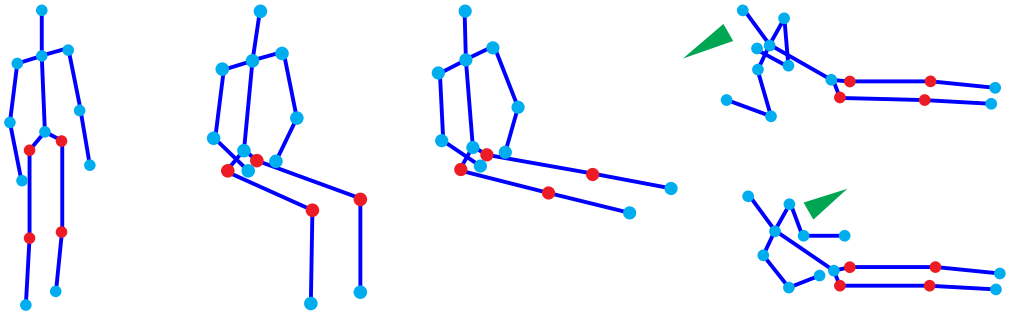


Figure 4.6: Five representative 3D human poses used for our action models. From left: standing, sitting, sitting with straight legs, lying with the face down, and lying with the face up. Angles at red skeletal joints are used for matching the recovered skeletons.

object categories to describe a meaningful action instance. We manually filter out these photos by examining the object category labels in the photo. We further remove photos that do not contain a visible human pose. This gives us a clean set of photo instances, of size 60 - 150, for each action type.

3D human pose recovery. The COCO database only provides a 2D image region of the human body (Figure 4.5(a)), while the human pose definition in our action model is 3D. Hence we need to recover the most plausible 3D human pose from the 2D image. To this end, we first manually annotate the human skeleton joints in each action instance image. Then we apply the method in [117] to find a 3D pose configuration $\mathcal{H}^p$ whose projection on the image plane matches the 2D joint annotations. To obtain a stable 3D pose estimation, we also need to manually provide the plausible locations of as many missing joints as possible, as in the case of partial occlusion; see Figure 4.5(b). The output is a 3D human skeleton and a weak camera projection matrix of the input image.

The resulting 3D human skeleton defines a 3D local frame in the scene, with origin at the center of the torso skeleton, the right direction determined by the vector between two shoulder joints, and the up direction defined by the torso. The frontal direction can be computed by the cross product of the up and right directions; see Figure 4.5(c). We use this coordinate frame for inferring and encoding all spatial layout information in the next step; see Figure 4.5(d).

However, the recovered 3D skeletons may have large variance and may be partial due to occlusions and thus cannot be directly used as the representative human pose for our action model. We thus follow the idea from [43] and introduce five representative human poses (see Figure 4.6) for our action model. For each action instance, we find one representative pose $\mathcal{H}^*$ that best matches the recovered partial skeleton $\mathcal{H}^p$ by minimizing

$$D(\mathcal{H}^p, \mathcal{H}^*) = \sum_i \omega_i \|\theta_i(\mathcal{H}^p) - \theta_i(\mathcal{H}^*)\|, \tag{4.1}$$

where $\theta_i \in \{\theta_{\text{hip}}^{\text{left}}, \theta_{\text{hip}}^{\text{right}}, \theta_{\text{knee}}^{\text{left}}, \theta_{\text{knee}}^{\text{right}}\}$ are angles at the hip (between torso and thighs) and knee joints (marked in red in Figure 4.6), and $\omega_i = 1$ if $\mathcal{H}^p$ contains the corresponding joint, with $\omega_i = 0$ otherwise. We assign the resulting representative 3D human pose to $\mathcal{H}$ of the current action instance. This representative pose $\mathcal{H}$ is only used to identify the action of the current instance, while the original recovered pose $\mathcal{H}^p$ is used to infer object layout.

Key objects. The key object in an action model specifies where the action is performed. Its functions are twofold: first, it is fixed in the scene and decides where and how to place the human pose; second, it provides a supporting surface for most objects involved in the action model. Since our work focuses on indoor scenes, we predefine *office desk, dining table, coffee table, kitchen island, bed,* and *couch* as the candidate key objects. The key object $\mathcal{K}$ in an instance is then automatically recognized by matching the object labels with the key object categories listed above. If several objects in the instance matches, we select the one whose image region is close to the human pose and has the largest region size.

Object layout. Given an action instance image with recovered 3D human pose and labeled object segmentations, we learn the object-human and object-object relationships in this step. Without a full reconstruction of the 3D scene, we encode an object's relative position to the human in a 2D polar coordinate system represented by the projected local human frame. Specifically, we project the local human frame onto the instance image and assume that the projected torso skeleton has unit length. The right axis is defined as the polar axis, as shown in Figure 4.5(c).

We assume that the key object is always in direct contact with the human pose: the pose is either (vertically) above the key object or (horizontally) around it. By assuming that all image instances are taken by cameras that are positioned in upright orientation, the on-relationship is true if the torso and thighs are entirely included in the convex hull of the key object. Otherwise, the human pose is deemed to be around the key object. For the around-relationship, we only calculate the angular coordinate $\psi_{\mathcal{K}}$ of the center of key object in the local human pose frame.

For each constituent object, we compute the polar coordinates $(r, \psi)_{o|\mathcal{H}}$ of its segment center with respect to the human pose. To infer the object-object relationship from the image, we translate the polar coordinate system described above to each object segment center o and record the polar coordinates of all other constituent objects o' as $(r, \psi)_{o'|o}$. In our experiment, we do not infer the relationship between the key object and constituent objects. Also, we do not record the layout of chairs because their positions can be well determined by human poses via a sitting relationship, but difficult to learn from 2D projections due to occlusions.

4.4.2 Generating action nodes

After collecting action instances, we group the instances with the same action $(\mathcal{T}, \mathcal{K}, \mathcal{H})$ and construct an action model for each group. By assuming the instances in a group cover sufficient statistical variance of the constituent objects in terms of their occurrence frequency and spatial layout, we learn an *occurrence* model $\mathcal{C}$ and a *spatial layout* model $\mathcal{D}$ for each action model.

Occurrence analysis. An action node usually involves many objects, which however might vary in their dependencies on the action model and their occurrence times in the action. For instance, both laptop and coffee mug can occur in the action node "use laptop on desk while sitting". While the laptop must be involved with one singe instance, the coffee mug might occur multiple times or not at all. Let $\mathcal{O}$ denote the set of objects that occur in at least one of the extracted image instances, excluding all key objects. Then the occurrence model $\mathcal{C}(o, n)$ of each object $o \in \mathcal{O}$ is computed by $\mathcal{C}(o, n) = N(o, n)/N_a$, where $N(o, n)$ is the number of instances that contain n copies of object o, and N_a is the total number of instances of the action model. If an object o has *occurrence frequency* of $\mathcal{C}(o) = \sum_{n>0} \mathcal{C}(o, n)$ lower than a certain threshold $\epsilon_{\mathcal{C}}$, we set $\mathcal{C}(o) = 0$, namely $\mathcal{C}(o, 0) = 1$ and $\mathcal{C}(o, n|n > 0) = 0$, which means the object o will never be involved in this action. This can remove most of the random objects appearing in the instance images that are irrelevant to the action node. In all experiments, we have $\epsilon_{\mathcal{C}} = 0.1$. The result occurrence model $\mathcal{C}(o, n)$ of object o is a probability distribution over its occurring times n.

Spatial layout analysis. To encode all possible correlations between object and human placements caused by an action, we model each object's spatial distribution relative to both human pose and other objects. This is different from our treatment of the occurrence model since spatial layout is a more critical factor for scene synthesis. As well, we have found that a single object-human spatial constraint is insufficient for satisfactory spatial layout.

From a set of input instances belonging to the same action node, we first learn the distribution of the human orientations with respect to the key object. We assume that it follows a von Mises distribution $\mathcal{P}(\mathcal{H}, \mathcal{K})$ and estimate its parameters from angular observations in the instance images, which has numerical solutions and can be done with a maximum likelihood estimation (MLE).

Then we learn how each object o is arranged with respect to the human pose $\mathcal{H}$. Ideally, we should directly model the spatial probability distribution over a two-dimensional space (r, ϕ), which defines the distance to the body center and the orientation angle with respect to the right direction of the human pose. In practice, we found that this 2D distribution can be well modeled by the product of two 1D distributions in distance and angle domain, which allows us to estimate the parameters of each distribution separately. We thus assume that the distance r follows a log-normal distribution, and the angle ϕ is a mixture of von

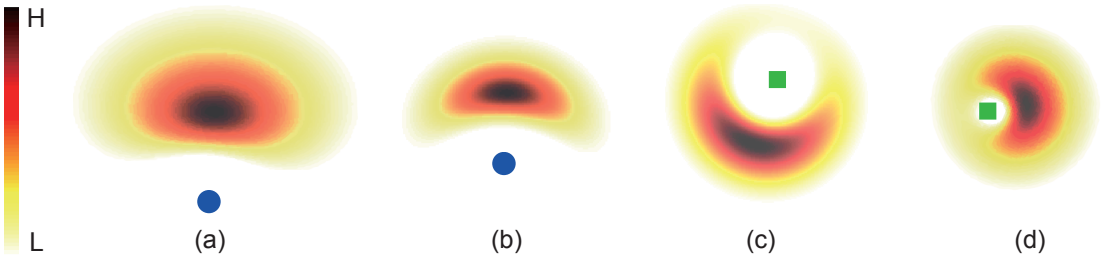


Figure 4.7: The learned spatial distributions of a few constituent objects in the action "use computer on desk while sitting": the object-human distribution for monitor (a) and keyboard (b) with the human marked as blue dot, and the keyboard-monitor (c) and mouse-keyboard (d) distributions with the reference objects (the second object in each pair) marked as green squares.

Mises distributions. Concisely, the object-human distribution is defined as

$$\mathcal{P}(o, \mathcal{H}; \Theta) = \mathcal{P}(r, \mathcal{H}; \Theta_r) * \mathcal{P}(\phi, \mathcal{H}; \Theta_\phi). \tag{4.2}$$

We apply MLE to estimate the parameters of the log-normal distribution $\mathcal{P}(r, \mathcal{H}; \Theta_r)$, which has closed form solution. The number of von Mises distributions in the mixture $\mathcal{P}(\phi, \mathcal{H}; \Theta_\phi)$ is a latent variable; we test values ranging from 1 to 4 and choose the one that maximizes the Akaike information criterion [4]. We then fix the number of von Mises distributions and utilize the numerical solution provided by [24] for estimating the parameters in the model. Figure 4.7(a-b) illustrate the learned distributions of monitor and keyboard with respect to human pose in the action node of "use computer on desk while sitting".

Finally, we formulate the object-object relationship $\mathcal{P}(o, o')$ in the same way as in Equation 4.2, with a similar learning procedure. The interdependencies between objects should not be treated equally; we would like to extract "reliable" object-object relationships that occur more frequently in the input instances. We compute the co-occurrence frequency $f(o, o')$ of two objects $o, o' \in O$, and object pairs with $f(o, o') < 0.5$ will be rejected from establishing the object-object relationships. Figure 4.7(c-d) illustrate the learned distributions for two pairs of frequent relationships in the action node of "use computer on desk while sitting".

4.4.3 Creating action graph

We prefer the action sequence that drives the scene evolution to be locally steady instead of totally random: two adjacent actions in the sequence should share certain constituent objects, which roughly approximates action progression in real life. For example, the action "use laptop on desk while sitting" is more likely followed by the action "read book on desk while sitting" (sharing desk) or "use laptop on sofa while lying" (sharing laptop).

We construct an action graph for modeling the transitional probability between actions. The action graph $G = (V, E)$ is a weighted directed graph, where V is a set of action nodes,

and E are edges connecting all pairs of action nodes, including action nodes to themselves. An edge $e_{i\to j} \in E$ directs from node a_i to a_j and carries the transition probability $w_{i\to j}$, i.e., how likely action a_j is to occur after action a_i. The transition probability is defined by

$$w_{i\to j} = \frac{\hat{s}_{ij}}{\sum_j \hat{s}_{ij}}, \tag{4.3}$$

where $\hat{s}_{ij}$ is the correlation between two action nodes that is computed by

$$\hat{s}_{ij} = \begin{cases} 1, & \text{if } i = j; \\ \max\{s_{ij}, 0.5\}, & \text{otherwise.} \end{cases} \tag{4.4}$$

Here s_{ij} measures the overlap of the constituent objects of two action nodes:

$$s_{ij} = \delta(\mathcal{K}_i - \mathcal{K}_j) + \frac{\sum_o \min\{\mathcal{C}_i(o), \mathcal{C}_j(o)\}}{\sum_o \max\{\mathcal{C}_i(o), \mathcal{C}_j(o)\}}. \tag{4.5}$$

The first term is a Dirac delta function encoding the overlap of key objects, which returns 1 if $\mathcal{K}_i = \mathcal{K}_j$, otherwise is of value 0. The second term measures the similarity between two occurrence frequencies $\mathcal{C}_i$ and $\mathcal{C}_j$; by considering $\mathcal{C}$ as a histogram over all constituent objects excluding the key object, the denominator measures the area of the "union" of two histograms, i.e., the sum of the maximum of each bin, while the numerator measures that of their "intersection" histogram, i.e., the sum of the minimum of each bin.

With this formulation, the two nodes with no object category intersection have the lowest transitional probability, whereas those with many shared object categories have higher transitional probability. The resulting graph is a directed state diagram, and the bi-directional transition probabilities are asymmetric (due to per node normalization in Equation 4.3). Sampling over the graph produces a Markov chain of human actions. The Markov assumption ("memoryless" transition probabilities) generates a new action based on the previous action instead of long-time causality relations – this simplifies the sampling process.

4.4.4 Group actions

The action model discussed so far is limited to describing actions involving a single person. Now we extend the model to handle actions involving multiple persons, i.e, *group actions*. To this end, we classify the objects in a scene into two classes: exclusive objects that are only affected by a single person action and *shared* objects affected by multiple persons' actions. We thus define a group action model as $\mathcal{A}_G =< \{\mathcal{A}_k\}, \mathcal{C}_s, \mathcal{D}_s >$, where $\{\mathcal{A}_k\}$ are constituent single action models, $\mathcal{C}_s$ specifies the probability distribution of occurring times for each shared object, and $\mathcal{D}_s$ describes the spatial distribution of shared objects and all human poses.

In this work, we focus on a common indoor group action – "group dining on table while sitting" with up to four people, denoted as $< \{\mathcal{A}, n\}, \mathcal{C}_s, \mathcal{D}_s >$, $2 \leq n \leq 4$, where all single action models $\{A\}$ are identical, i.e., "dining on table while sitting". We reuse the single action model already learned for this group action, thus the new tasks here are how to identify the shared objects from the image instance and learn $\mathcal{C}_s$ and $\mathcal{D}_s$ from all instances.

We extract image instances of group dining from the COCO database with group sizes of two to four. Ideally, we assume that a shared object is equally distant from all persons that have access to it. To automatically identify shared objects in each image, we first compute the maximal human-human distance d_h. Then for each object o, we compute its distance to all persons, where the maximal and minimal distances are denoted by $d_o^{\max}$ and $d_o^{\min}$ respectively. After that, we identify a object to be *shared* if it is neither too far away from nor too close to any person, which is quantified by the following two conditions:

$$\begin{cases} d_o^{\max}/d_h < 2/3, \\ d_o^{\max}/d_o^{\min} < 2. \end{cases} \tag{4.6}$$

After collecting all instances for the group action node, the occurrence model $\mathcal{C}_s$ is analyzed for all the shared objects in the same way as for constituent objects in the single-person action model.

We assume that the distances between modeled humans fall into a certain range so that they can share objects, i.e., $d_h^{i,j} = d(\mathcal{H}_i, \mathcal{H}_j) \in [a, b]$, where $a = 0.6m$ and $b = 2m$ for all experiments in this work. We further assume that the human-human distance follows a uniform distribution over $[a, b]$:

$$P(d_h^{i,j}) = \begin{cases} \frac{1}{b-a}, & a \leq d_h^{i,j} \leq b, \\ 0, & \text{otherwise.} \end{cases} \tag{4.7}$$

Given the configuration of multiple human poses, a shared object o must locate in the region defined by Equation 4.6, and its distribution over that region follows the following uniform potential:

$$\mathcal{P}(o, \{\mathcal{H}, n\}) \propto \frac{1}{d_o^{\max}}. \tag{4.8}$$

To construct the action graph with group action nodes, we use the involved single action model for computing the transitional probability from or to a group action node, thus the introduction of group actions demands no further change of edge weight computation in the action graph.

4.5 Action-driven scene synthesis

Given the action graph learned from the annotated COCO images, we are now ready to synthesize human actions to drive the evolution of a scene. Our solution follows recent work [22, 21] and models human actions including group activities to synthesize lively messy 3D indoor scenes.

Given an input scene with key objects, our method first adapts the action graph to the input scene. Then it generates an action sequence by traversing the graph based on the node-to-node transitional probability. The action sequence starts from a random action node and ends with a user specified length. Note that the action sequence driving the scene synthesis is not directly learned from photos. Instead, each action sequence is an instance of the Markov chain sampled from the action graph, which characterizes correlations between action nodes. Finally, the scene evolution is realized by exerting actions from the sequence in the scene one by one, which triggers the insertion or relocation of involved objects, and naturally leads to a messier scene at the end.

4.5.1 Graph adaptation

Our action graph is constructed from all types of actions learned from the COCO database and covers actions for various types of scenes: bedroom, kitchen, office room, etc. Applying it to a specific scene requires a preprocess of graph adaptation. That is, given an input scene with key objects, we need to prune certain action nodes if their key objects are missing in the scene. For example, if silverware is not present in the scene, the action node of dining could be on – the realization of this action will cause the insertion of silverware; but if the dining table (key object) is missing, dining should not be allowed to happen, because our action model relies solely on the key object to place the human pose and therefore all other constituent objects. After the graph adaptation, we also remove dangling edges connected to pruned nodes and update edge weights in the adapted graph according to Equation 4.3.

4.5.2 Action realization

After an action is performed, the involved objects retain in the scene. That means an action is performed in the context created by all its ancestors in the action sequence. To realize a new action $\mathcal{A} =< \mathcal{T}, \mathcal{K}, \mathcal{H}; \mathcal{C}, \mathcal{D} >$ from the sequence, we first place the human pose $\mathcal{H}$ into the scene w.r.t. the key object $\mathcal{K}$; then collect the set of active objects O (involved in $\mathcal{A}$) by sampling the occurrence model $\mathcal{C}$ and place them in the scene such that their spatial layouts follow the distributions $\mathcal{D}$; finally, we relocate non-active objects $\widetilde{O}$ (not involved in $\mathcal{A}$) so that they do not obstruct the current action.

Fitting human pose ($\mathcal{H}$). The human pose $\mathcal{H}$ of an action is always in contact with the key object $\mathcal{K}$ as per our assumption on their mutual relationships. If it is an on-

relationship, we randomly sample a location and orientation to place the human skeleton on the supporting plane of the key object, then locally adjust its location so that the pose can physically fit onto the key object. For the around-relationship, instead, we first randomly sample a location that is horizontally around the key object, and an orientation according to the learned angular distribution $\mathcal{P}(\mathcal{H}, \mathcal{K})$ of human pose w.r.t. the key object; then we place the human skeleton subject to collision rejection detection. If the scene contains multiple copies of $\mathcal{K}$, we randomly choose one for placing the human pose.

Placing active objects (O). Given the human pose and key object of the action, we need to figure out the set of constituent objects O to be inserted, as well as their locations and orientations in the scene. First, we generate the set of constituent objects according to the learned occurrence model $\mathcal{C}$. For the active objects that have been in the scene, we have two options – either reuse it or insert a new one. We predefine an upper bound for the occurrence times of each object category in the whole scene. For example, a scene can contain at most one monitor, one keyboard, but two coffee mugs, ten books. New objects are inserted into the scene before their time of occurrence reaches the upper bound; otherwise, we only allow reuse of objects for realizing new actions.

We then place these objects one at a time in order of descending occurrence probability and size in the second step. The placement of an object $o \in O$ follows the preference density function:

$$f^1(o) = \mathcal{L}(o) * \mathcal{S}(o) * \mathcal{P}(o). \tag{4.9}$$

The *collision penalty* term $\mathcal{L}(\cdot)$ enforces no physical collision between objects; $\mathcal{L}(o) = 0$ if o collides with any other objects in the scene, and is of value 1 otherwise. The *overhang penalty* term $\mathcal{S}(\cdot)$, similar to that in [22], prevents o from hanging off the edge of a supporting surface. We project the bounding box of o onto the supporting surface and compute the intersection area $A(o)$ between the projection and the supporting region. Precisely, $\mathcal{S}(o) = 1$ if $A(o) \geq 0.5$, otherwise $\mathcal{S}(o) = 0$. That says the placement of o is not plausible if more than half of its volume hangs off the supporting surface. The last term $\mathcal{P}(o)$ combines the spatial layouts of o w.r.t. both human pose and other objects:

$$\mathcal{P}(o) = \mathcal{P}(o, \mathcal{H}) + \sum_{o'} f(o, o') * \mathcal{P}(o, o'), \tag{4.10}$$

where $\mathcal{P}(o, \mathcal{H})$ and $\mathcal{P}(o, o')$ are the object-human and object-object distributions, respectively, $f(o, o')$ is the co-occurrence probability of (o, o'), and $o' \in O$ is the object that has been placed in the scene with $f(o, o') > 0.5$.

Note that the positioning of o so far is still in the human centric system, which might generate a 3D location floating in the air. To make the synthesized scene physically plausible, we move o vertically until it reaches the closest supporting surface of either a key object or the floor. A sampling strategy is utilized in the placement procedure to ensure a balance

between the diversity and the plausibility of object configuration. In our implement, we uniformly sample $k = 2,000$ locations and select the one that maximizes the score defined in Equation 4.9 as the final position for an object. To determine the orientation of an object o, we manually specify a *facing direction* for o w.r.t. human. Each placed object is rotated horizontally so that its facing direction pointing to the human center.

Placing non-active objects ($\widetilde{O}$). To relocate a non-active object $\tilde{o} \in \widetilde{O}$, we must consider two more constraints besides the collision and hanging-off rejections. First, $\tilde{o}$ must not obstruct the current action. We define a *working zone* for the current action and keep $\tilde{o}$ from that region. In our experiments, the working zone of an action is defined as the convex hull of its human pose and constituent objects with occurrence frequency greater than 0.5. Second, the new location of $\tilde{o}$ should be as close to its original location as possible.

Similar to active objects, we insert objects in $\widetilde{O}$ into the scene one at a time in the order of descending object sizes. For each object $\tilde{o} \in \widetilde{O}$, we uniformly sample $k = 10,000$ positions, and select the optimal location that maximizes the following score:

$$f^2(\tilde{o}) = \mathcal{L}(\tilde{o}) * \mathcal{S}(\tilde{o}) * \mathcal{W}(\tilde{o}) * \exp(-\Delta(\tilde{o})). \tag{4.11}$$

The first two terms, $\mathcal{L}(\cdot)$ and $\mathcal{S}(\cdot)$, are the same as that in Equation 4.9. The third term $\mathcal{W}(\cdot) = 1$ if $\tilde{o}$ falls outside of the working zone; otherwise, $\mathcal{W}(\cdot) = 0$. The $\Delta(\tilde{o})$ in the last term measures the distance of shift of $\tilde{o}$ from its original position.

We allow the placement algorithms for O and $\widetilde{O}$ to roll back to the previous object, modifying its placement in seeking of a relaxed solution, if the placement of the current object fails up to a prescribed number of times (set to 2,000 in our current implementation). Given the fact that single action models are rarely cluttered by constituent objects, the roll-back procedure always successfully places active objects O in all experiments. However, non-active objects $\widetilde{O}$ will keep accumulating as an action sequence proceeds. At a certain point, it becomes impossible to place all the non-active objects on a valid supporting surface. Further options in this scenario include stacking them vertically or placing them on the floor; both options occur naturally in messy scenes.

4.5.3 Realizing group action

The realization of group action $< \{\mathcal{A}, n\}, \mathcal{C}_s, \mathcal{D}_s >$ is a hybrid process. We first place multiple persons around the key object following the human distribution; second we generate a set of shared objects according to the occurrence model $\mathcal{C}_s$ and place them according to the spatial distribution of shared objects; third, we apply the single person action model $\mathcal{A}$ for each person to place exclusive objects belonging to each single action; lastly, to place non-active objects, we additionally include all persons and shared objects for computing the working zone of the current group action.

We place human poses into the scene one at a time. The first human pose is randomly placed around the key object. Suppose k human poses have been placed, the location of the $(k+1)$-th human pose must also be around the key object, as well as obey the distance constraint in Equation 4.7 to all other k human poses. This procedure stops until the group size reaches four or no further human pose can be placed, which leads to groups with size of 2 and 3.

Given the set of shared objects, we place them one at a time in order of decreasing object size. For each shared object o, we uniformly sample 2,000 locations and select the placing location that maximizes the following score:

$$f^3(o) = \mathcal{L}(o) * \mathcal{S}(o) * \mathcal{R}(o) * \mathcal{P}(o, \{\mathcal{H}, n\}), \tag{4.12}$$

where $\mathcal{L}(\cdot)$ is the collision penalty, $\mathcal{S}(\cdot)$ is the overhang penalty, $\mathcal{R}(o)$ is a binary indicator of whether o falls in the sharing region specified by Equation 4.6, and $\mathcal{P}(o, \{\mathcal{H}, n\})$ is the spatial distribution potential of o; see Equation 4.8.

4.6 Results and evaluation

In this section, we present results of action-driven scene evolution and compare the results, through user studies, to those created by an artist, and those by the most closely related methods for human- or activity-centric scene modeling [43, 22, 21]. We also demonstrate the capability of our method to synthesize messy scenes at larger scales.

Action data and graph. Action learning is conducted exclusively over 1,216 annotated photographs, 936 of which are from the Microsoft COCO database [53]. The remaining photos were collected on-line to enrich or complement action data extracted from COCO. The photos were all annotated with embedded human poses as well as action and object labels. It takes less than 30 seconds to mark all the joints for one human in a photo. An unlabeled on-line photo typically takes less than three minutes to annotate using LabelMe for a scene with 5-10 objects. For our experiments, the learned action graph is a complete graph (with self-loops) composed of 20 nodes and covering 8 types of actions. After annotating all the photos, action learning and graph construction take about 8 minutes in total to complete.

3D scene evolution. Figure 4.14 (also see Figures 4.1 and 4.8) shows a gallery of 3D scene evolution results that highlights the various features offered by our method. Timing-wise, sampling an action from our action graph takes on average 0.1 second and object placements take on average 1 second.

In each row of Figure 4.14, the action sequence is probabilistically sampled from the learned action graph and applied to the initial scene in order. Applied actions are indicated

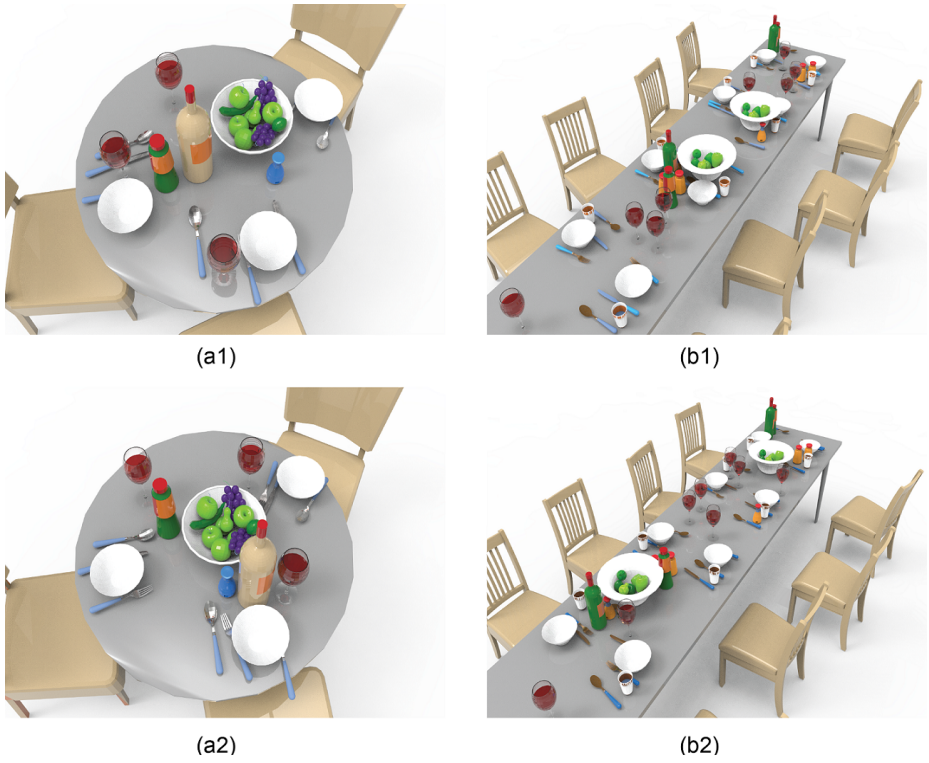


Figure 4.8: Two scenes synthesized from a group action "group dining" (bottom) vs. those by a baseline method (see text) mainly involving single-human actions for dining (top). Both syntheses start with the same initial scene and place the same set of objects. Notable distinctions can be observed for the placements of the large fruit bowl, which is a shared object.

in the figure, with zoom-in views to better visualize the insertion and relocation of objects. The two final rows in the figure show how actions involving working and dining learned from bedrooms and living rooms can be transferred to never-seen scene categories such as dining halls and computer labs to create quite a mess. It should be reiterated that the mess is purely the result of action-driven scene evolution, starting from a clean initial scene. The work of Fisher et al. [21] was able to show the modeling of messy 3D scenes when depth scans of cluttered scenes are given as input.

Instead of probabilistically sampling the action graph, our work easily supports user-guided scene evolution where a user drives the process by iteratively selecting from a set of probable actions suggested (in the order of decreasing probabilities) by the action graph.

Group actions. Figure 4.8 contrasts scenes synthesized via group actions to those generated by applying a "baseline" method mainly involving single-person actions. In both cases, we first sample from a distribution of persons based on the same initial scene. For group actions, we sample and place a set of shared objects learned from photos of group actions. Then for each person in the group, we apply a single-person action model to place

additional objects into the scene. For the baseline method, the shared objects are randomly assigned to persons in the group, then we apply a series of single-person actions to place all the objects including the shared ones.

It is interesting to observe that the placements of wine glasses appear to be more distant from the chairs (where persons using the wine glasses would sit), compared to the bowls and folks. Upon close examination of the data source, the wine glasses in the photos do generally appear relatively far from the chairs where people sit. This is likely due to the fact that people holding wine glasses tend to move around in the room.

To assess the plausibility of our learned group action model, we prepared 6 pairs of scenes like the ones shown in Figure 4.8, using our group action model and the baseline method described above. We then asked 24 human participants to select, one out of each pair, the scene they believe to "appear more like a scene during or after a group of people having a meal together". The participants selected the scenes generated by our group action model 75% of the times.

4.6.1 Plausibility tests against artist

A key evaluation for our method is whether the results from action learning and scene generation are plausible. Similar to previous works [21], we leave such judgments to human subjects. We ask users to give a score from 1 (least plausible) to 5 (most plausible) to a generated scene based on two criteria: *plausibility*, the scene is a plausible result after a given action is performed on an initial scene; and *naturalness*, the scene looks natural.

Object placement test (OPA). To compare our results to human-generated scenes, we hired a professional artist to manually create scenes based on given actions. When asking users to rate scene plausibility, we found that providing them with more contexts with which to make judgements is more likely to gather more reliable feedback. Therefore, for each initial scene and a given action, instead of providing only a pair of scenes to rate, we provide five scenes: three synthesized by our method with random initialization, and two scenes created by the artist. In each case, the same set of objects were inserted or relocated. The five scenes, randomly ordered, make up one query for the Object Placement test against Artist or OPA, for short. Our user study consists of 20 OPAs covering 10 actions. With 28 human participants working on the OPA test, we gathered a total of 140 user ratings.

Figure 4.9 (left) plots the overall average user rating for the artist's results and our results from the OPA test. Per-action average ratings are plotted in Figure 4.10. Our results received an average rating of 3.46, which approaches closely to that of the artist (3.83). If we take the best out of our three results from each OPA query, then the (best-of-3) average rating goes up to 4.29 and is higher than artist's average rating, indicating that our method is able to produce results no worse than an average artist in the best scenario.

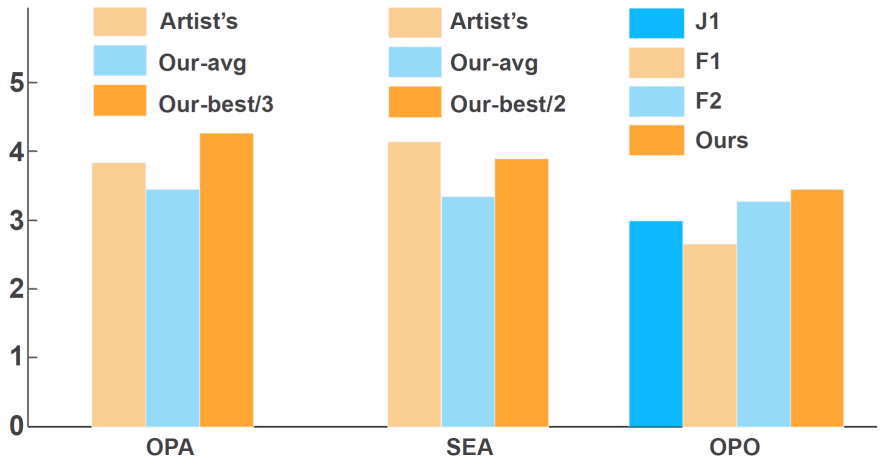


Figure 4.9: Overall average user ratings for scene plausibility test, when comparing our results to scenes created by an artist and to scenes created by closely related works.

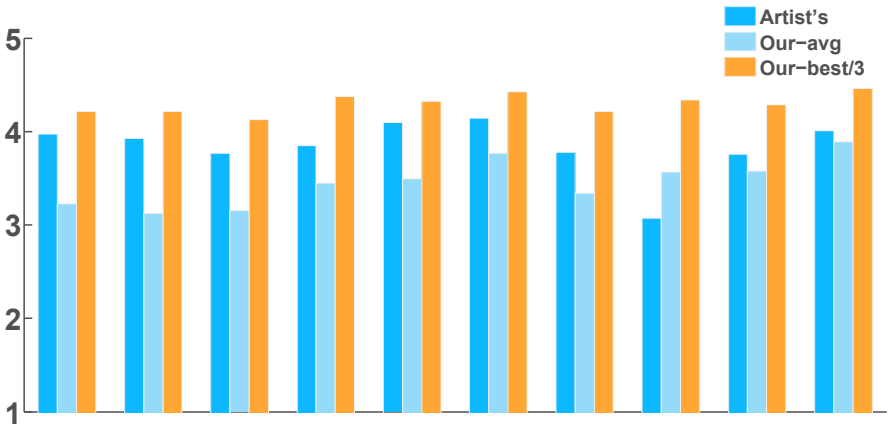


Figure 4.10: Per-action OPA test results: comparison to scenes created by an artist in terms of average user ratings. Along the x-axis, the 10 actions are: "use computer on desk while sitting", "use laptop on desk while sitting", "use laptop on coffee table while sitting", "use laptop on bed while lying", "read book on desk while sitting", "eat snacks on coffee table while sitting", "read book on coffee table while sitting", "read book on bed while lying", "eat dinner on table while sitting", "watch TV on sofa while sitting".

Scene evolution test (SEA). We scale up the OPA test to scene evolution involving multiple actions. Each Scene Evolution test against Artist, or SEA, consists of two sequences of scenes evolved using our method via probabilistic sampling of the action graph and one sequence of scenes created by the artist using the same action sequence, all with the same initial scene and same set of objects. Three evolution steps are applied for each initial scene and the SEA test covers 9 initial scenes. A total of 29 human participants rated for the SEA test. For each query, the participants were asked to rate the overall plausibility of each of three scene sequences. The overall average and per-action-sequence average ratings are shown in Figures 4.9 (middle) and 4.11, respectively.

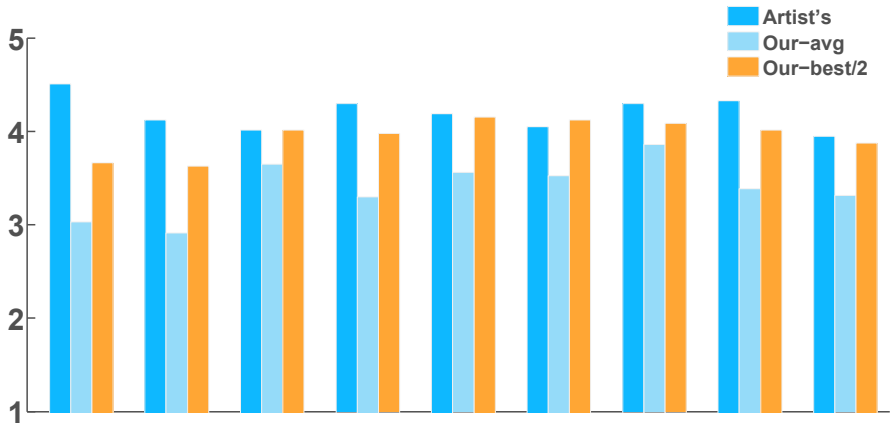


Figure 4.11: Per-action-sequence SEA test results: comparison to scene evolutions created by an artist, over 9 initial scenes and action sequences.

The average and best-of-2 average ratings for our scene evolution results are 3.35 and 3.90, respectively, while the average rating for the artist's results is the highest, at 4.14. Artist's performance in SEA appears to be superior than that in OPA. This may be partly attributed to the artist's ability to take into account of subtle object movements between action transitions – movements that our current action model does not accommodate. For example, we observed that the artist would *move a chair to the side* before having the human in action get up from the chair after dining to watch TV.

4.6.2 Comparisons to closely related works

Among all the previous works on 3D scene modeling, the example-based synthesis method (F1) of Fisher et al. [22] is the closest to our work in terms of end goal: both aim to develop an open-ended tool to synthesize 3D scenes, but take different routes to achieve the goal. Judging by modeling methodologies, the works by Jiang et al. [43] (J1) and Fisher et al. [21] (F2) come the closest as they both take a human- or activity-centric approach. All of these works, including ours, are data-driven. However, F1, J1, and F2 all learn from 3D scene exemplars, we learn from annotated photos.

In terms of scene synthesis capabilities, neither F1 nor F2 was designed to obtain a fine-grained scene evolution. Both J1 and our work can progressively alter a scene; their capabilities depend on the generative models developed and richness of data utilized. We present head-to-head comparisons to the three methods, via a user study to rate the plausibility of the generated scenes.

Since the compared methods perform learning from different data sources, which would influence object category occurrence, we only compare the plausibility of object placement where the set of objects placed are the same, as in the OPA test. Also, only single-frame object placements are compared since the other three methods were not all designed for

	CK-1	CK-2	LS-1	LS-2	D-1	D-2	Average
J1	2.43	2.33	3.53	4.13	2.77	2.77	3.00±0.09
F1	3.20	2.77	2.27	2.07	3.17	2.50	2.66±0.09
F2	3.73	3.77	3.27	3.40	2.87	2.60	3.27±0.10
Ours	3.08	2.95	3.30	4.07	3.97	3.38	3.46±0.06

Table 4.1: Numerical average user ratings for the OPO test, for three actions and two initial scenes and object replacements per action. The three actions are: CK = "use computer on desk while sitting"; LS = "use laptop on coffee table while sitting"; D = "eat dinner on table while sitting".

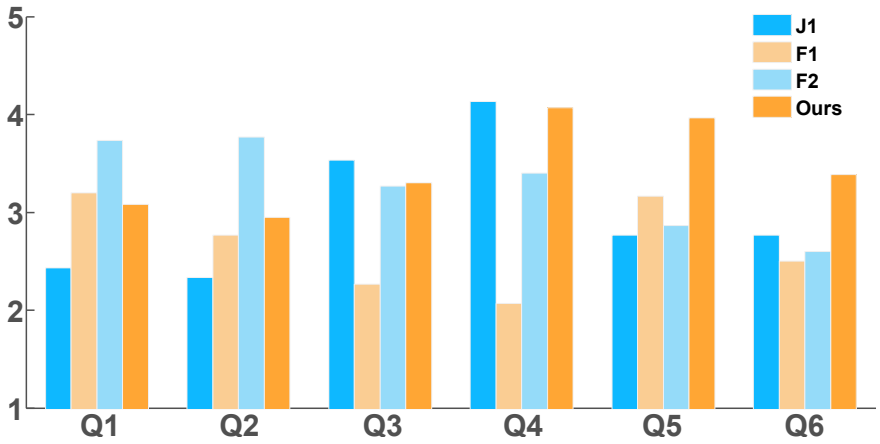


Figure 4.12: OPO test results on 6 queries: comparison to three related works in terms of scene plausibility. Along the x-axis, the actions are "use computer on desk while sitting"(Q1, Q2), "use laptop on coffee table while sitting"(Q3, Q4), and "eat dinner on table while sitting"(Q5, Q6).

scene evolution. Given an action applied to an initial scene, the four scenes resulting from J1, F1, F2, and our method make up one query in the Object Placement test against Other methods or OPO, for short. Human participants are asked to provide plausibility scores for these queries.

The actions selected for the OPO test are "*use computer on desk while sitting*", "*use laptop on coffee table while sitting*", and "*eat dinner on table while sitting*", which contain common object categories whose placements have been learned by all four methods. For each initial scene, the arrangements of essential furniture pieces are provided to assist J1 and F2 in pose or activity prediction. Then for J1, the same objects are placed with both human and object contexts learned from 3D example scenes in which the placements of a set of daily objects are labeled by users. Although F2 is designed for functional scene modeling from depth scans, its activity model is capable of placing objects without constraints from the scans. We asked the lead author of F2 to produce results for the tested actions and specified objects, conditioned only on given furniture pieces to ensure a fair comparison. For F1, we used the scenes generated by the artist from previous tests as the input examples for their

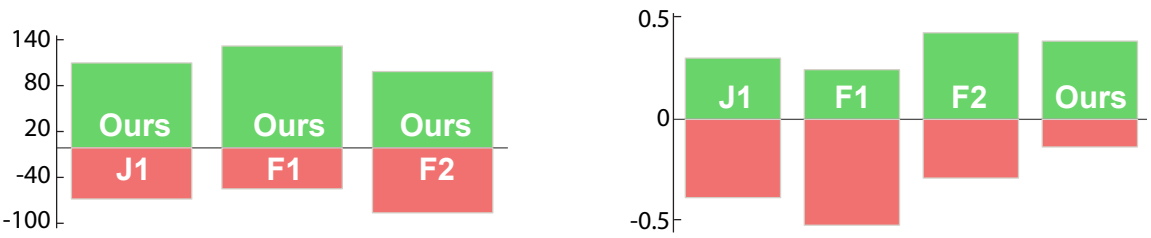


Figure 4.13: Additional comparisons among the four methods in terms of scene plausibility. Left: the number of times our method was rated higher (green) or lower (red) than other methods in the OPO test, respectively. Right: the percentage of each method being rated as the best (green) or the worst (red) among all.

example-based synthesis. The final scenes are synthesized by applying the necessary object placements, as guided by the input scene examples and scenes in the 3D scene database from the original paper.

For each of the three actions selected for the OPO test, two object placements are performed by all four methods. Thus, there are six OPO queries in total. The user study involved 30 human subjects to provide a total of 180 ratings. Table 4.1 summarizes the OPO test results, including average user ratings for each scene-action combination. The last column shows the overall averages and the standard deviations of the ratings. The numbers indicate that in the OPO test, our action-driven method outperforms all of the three methods compared. Significance tests also show that the performance gains are statistically significant. Figures 4.9 (right) and 4.12 visualize the numbers given in Table 4.1.

Test results in Figure 4.12 show that our method performed best for the action "eat dinner on table while sitting", which involves multiple small items. The likely reason is that our action model considers human-object relations as well as relations among multiple objects, e.g., the bowls and utensils, while both J1 and F2 focus on pairwise human-object relations. For the action "use computer on desk while sitting", F2 received the highest scores. This may be attributed to the *semantic* human-object relations that can be learned from their 3D scene data possessing semantic annotations, e.g., monitor is not blocked for visibility, keyboard and mouse need to be touchable. On the other hand, our action model learns the human-object distributions only with *spatial constraints* from photos.

In Figure 4.13, we show additional comparison results, including head-to-head ratings. These results consistently demonstrate the advantage of our method from different perspectives. Note that all the human subjects in our user studies are graduate students in the fields of computer science and engineering.

4.7 Discussion, limitation, and future work

Our ultimate goal is to automatically synthesize truly messy yet realistic 3D indoor scenes, a task we believe no current work has been able to come close to achieving. In reality, scenes

are messed up by our daily actions, not a probabilistic scene/sub-scene mixing. In addition, a mess is hardly something that can be properly learned from limited exemplars as the messiness is expected to be highly varied and unpredictable. In our work, we propose an action-driven approach with a progressive layout scheme, which we call a scene evolution. Furthermore, we believe that the data source ought to come from images since so far, only images possess the richness and variety to allow the synthesis of truly messy scenes.

The method we have presented only represents a first attempt at executing the above ideas. Yet, the action model we develop and apply, as well as action extraction and learning from annotated photographs, offers unique features which set our approach apart from previous works on human- or activity-centric 3D scene modeling [22, 43, 77, 21, 78]. Even with the various simplifying design choices in our work, comprehensive user study results positively support the action-driven approach in terms of its generative capabilities and plausibility of the generated scenes, as compared to human effort and state-of-the-art data-driven methods. That said, the action-driven approach should only *complement* and not replace other scene modeling paradigms. We also reiterate that the success of any data-driven approach relies heavily on the quantity and quality of its data. What our approach can accomplish is still limited by having only a handful of action types to work with.

Scene synthesis at two scales. An important lesson from our current pursuit is that indoor scene generation is inherently a process that ought to operate at *two* scales. At the coarse scale, one is concerned with how to layout large furniture pieces such as beds and sofas. How a human sits or lies on a sofa does not play a major role in its placement in a room. The layout problem is more suited for a rule-based approach [63], since in reality, it is mainly guided by design guidelines and functionality considerations. The movements or arrangements of small, mobile items are naturally tied to human actions. There are also *passive* actions such as watching TV, which do not involve moving an object but may trigger an object insertion. The action-driven approach is best suited for scene synthesis at such finer granularity level.

In our current implementation, the initial 3D scene is expected to contain large, fixed furniture pieces, which would serve to initialize applicable actions. However, this assumption could be lifted if we allow special handling of the first actions when the initial scene is completely empty. In general, we can allow certain actions to be entirely conditioned on the scene category. For example, watching TV is always applicable to a living room, empty or not.

Pattern-driven vs. action-driven. Arranging small objects over a relatively small workspace such as a shelf or desk *is* action-driven, but the relevant actions are difficult to capture and annotate from depth scans [77] or photographs. Since these kinds of arrangements typically exhibit predicable patterns, *pattern-driven* syntheses have been successful, where the patterns are rule- or style-based [59] and can be learned from examples [22]. On

the other hand, messed-up rooms after a party or kids play would hardly share common statistical properties or reveal predictable patterns. An action-driven, progressive approach is more befitting to the modeling of such scenes.

Static vs. dynamic action data. Our action model is learned from static photos showing snapshots of human-object and object-object relations. Without capturing dynamic transitions between actions, we currently assume that the transition probability from one action to others can be estimated by the correlation between actions; see Equation 4.3. Such a substitution of transition probabilities by correlations between action models is a limitation which could be lifted if dynamic action data, e.g., video or RGBD motion data [77, 78], can be utilized to study the transition probabilities. Then the learned model can be adapted to our action graph to drive the scene evolution.

Technical limitations. The main technical limitations arise from various difficulties in action learning from photos. Pose recovery from photos is challenging, especially from non-iconic photos of COCO. Even with manual annotations of joints for 3D pose recovery, the reconstructed 3D poses still may not be plausible. Similarly, object-human and object-object spatial relations learned from photos can be inaccurate due to erroneous camera projections and view angles. That said, these problems have all been intensely studied in the field of computer vision and there are many available techniques, e.g., those based on deep learning, that can be applied to alleviate the issues with our current implementation. Complementary to this, efforts can be made to acquire more meaningful photos which enrich the action data while facilitating action learning.

Future work. The set of action types we learn and apply in this work were hand-picked from the COCO database; they are clearly limited. To fully realize the potential of action-driven scene analysis and synthesis, a lot more action data should be acquired and prepared. To extend the effort to a much larger scale, action data should ideally be mined first from large text and image sources in a future pursuit. If we are able to obtain much more action data with text annotations, then we would not be far from achieving 3D scene scene evolution from textual instructions [78]. Also, our current object placement scheme can be enhanced with more advanced geometric and even functional analysis of 3D objects [31]. A final interesting thought would be to *reverse* the scene evolution: instead of going forward in time from an input scene, reconstruct a plausible scene sequence backward in time so that the sequence would lead to the input scene.

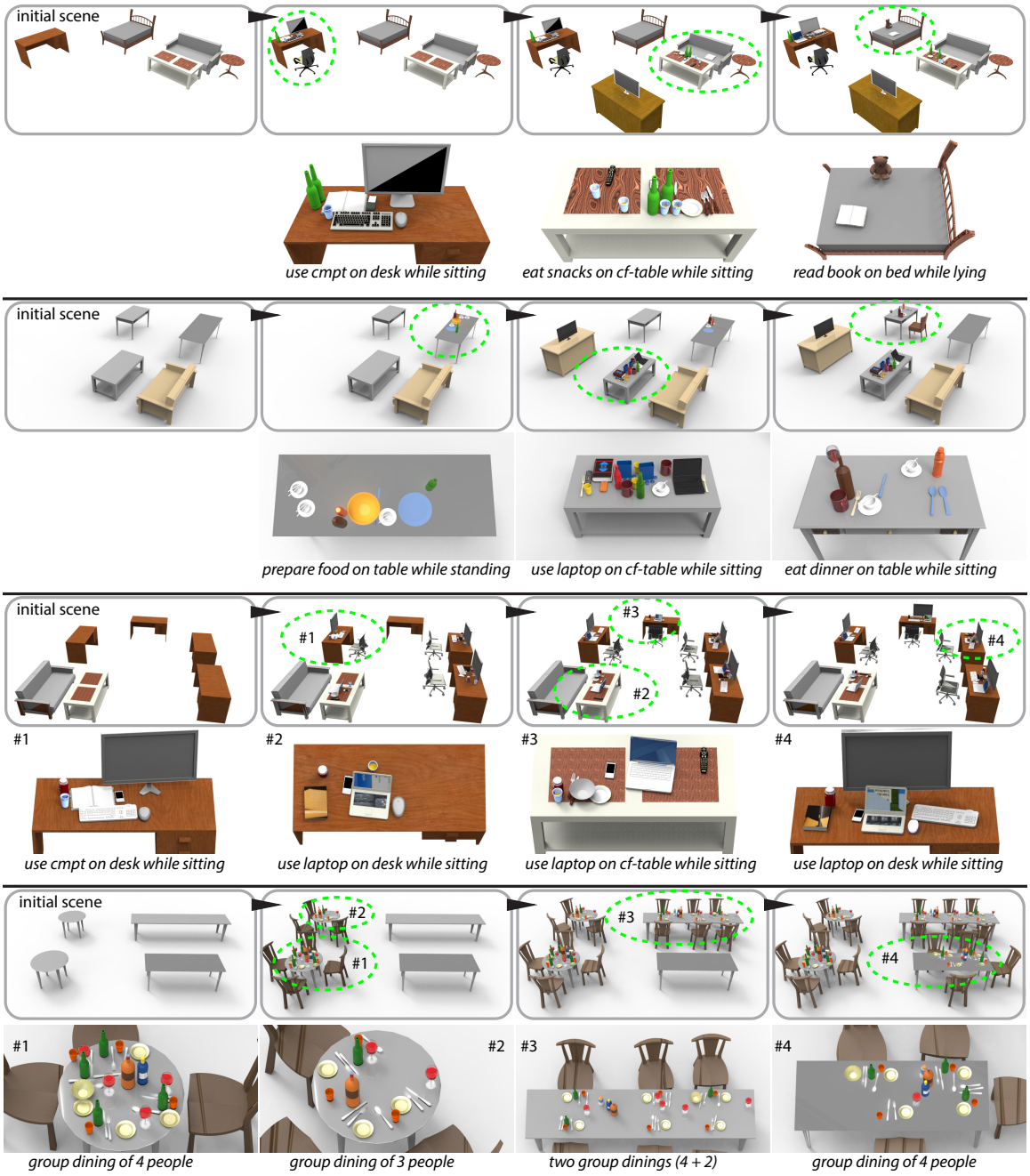


Figure 4.14: A gallery of our action-driven 3D scene evolution results. The figures and annotations should be self-explanatory. Sub-scenes enclosed in green ellipses are zoomed in below the scene sequence for better visualization. Last two rows show how actions involving laptop or computer use and dining, which were learned from small-scale bedrooms and living rooms from Microsoft COCO photos, can be transferred to never-seen scene categories such as a computer lab and a dining hall to create quite a mess.

Chapter 5

Language-Driven Scene Synthesis

This chapter introduces a novel framework for using natural language to generate and edit 3D indoor scenes. The advantage of natural language editing interfaces is strongest when performing semantic operations at the *sub-scene* level, acting on *groups* of objects. We learn how to manipulate these sub-scenes by analyzing existing 3D scenes. A *semantic scene graph* (SSG) is proposed to represent both the natural language scene description and the structure of 3D scenes. Our language-driven scene synthesis interactively retrieves sub-scenes from the database and accommodates the retrieved sub-scenes into the 3D scene-in-progress based on the SSGs. We conduct studies comparing our approach against both prior text-to-scene work and artist-made scenes and find that our method significantly outperforms prior work and is comparable to handmade scenes even when complex natural sentences are used.

5.1 Introduction

Recent advances in computational design, VR/AR, and robotics are placing an increasing demand for rich 3D content, especially in indoor environments [88, 91, 32]. While most efforts on 3D indoor scene modeling have been devoted to high-quality and interactive scene *reconstruction* from photographs or depth scans, there has also been a push for more open-ended, and often user-centric, approaches to synthesize and edit 3D scenes [22, 12, 78, 74, 58], aiming for improved richness and creativity of the generated content.

Natural language is arguably the most accessible input for content creation. With no modeling skills or training required, language- or text-driven modeling is fun, open-ended, and stimulates the imagination. Directly converting languages to 3D scenes has been a long and on-going pursuit since the pioneering work on WordsEye [17]. To date, most research has emerged from the natural language processing (NLP) community, e.g., [80, 16, 12], where the focus has been on mapping explicit scene arrangement languages, e.g., "a bed with three pillows and a bedside table next to it", to object arrangements. Clearly, such

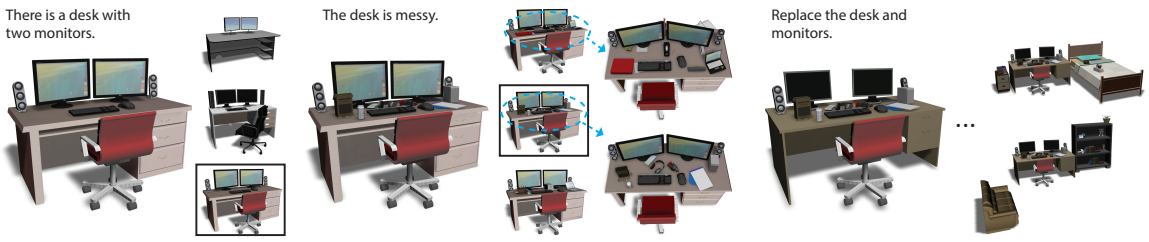


Figure 5.1: Our modeling tool uses compact and natural language to generate and edit 3D indoor scenes. Each language command triggers a scene evolution where the user can choose (marked by a box) from a list of suggested outcomes to alleviate ambiguities in natural language. One or groups of objects can be inserted, replaced, or re-arranged based on attributes or relations (e.g., "desk is messy") specified in the command.

explicit commands can be tedious and unnatural. Moreover, one may argue that precise, object-level controls are more effectively achieved by direct object manipulation using a mouse, e.g., to rotate a cup for its logo to face front. On the other hand, scene modifications at a semi-global or *sub-scene* scale or those involving changes in *group relations* (e.g., "make the tabletop arrangement messy or formal") represent situations where a language-based modeling interface can excel.

In this work, we introduce a framework for language-driven modeling of 3D indoor scenes, which is designed with two objectives in mind. Our foremost objective is to provide novice users the freedom to use compact natural language commands to generate and edit a 3D indoor scene. A key to reducing language redundancy and improving the efficiency of scene modeling is to relieve the user from having to provide explicit commands to affect every single object, like in most previous works. By strengthening the role of *implicit* or common-sense knowledge extracted from existing scene databases, our method supports generic user language expressions which drive the scene modeling at the *sub-scene*, rather than object, scale.

The second objective is to improve both the complexity and realism of the generated 3D scenes. As a scene increases in complexity, it provides increasingly richer spatial and semantic contexts beyond pairwise object relations [12]. To account for these, we need to encode and learn more complex object relations, particularly those involving *groups* of objects. With this in mind, we enhance existing 3D scene datasets such as SceneNN [32] with both finer-scale objects and annotations of group relations, e.g., "around", "aligned", "messy", etc., to support our modeling task.

To accomplish the goals set above, our key idea is to treat language-driven 3D scene modeling primarily as a combination of two tasks operating at the sub-scene level: language-driven sub-scene *retrieval* from a 3D scene database and scene *accommodation* which merges an appropriately retrieved sub-scene with the current 3D environment to *synthesize* a new 3D scene. Playing a central role in our modeling framework is a *semantic scene graph* representation which encodes geometric and semantic scene information and serves as the

bridge between user language commands and scene modeling operations which directly modify a 3D indoor scene. Specifically, a semantic scene graph is defined by object instances along with object-level attributes and pairwise, as well as group-wise, object relations. Both annotated 3D scene data and natural language commands are converted into semantic scene graphs.

We develop an interactive scene modeling tool which allows the creation of an initial 3D indoor scene followed by progressive editing to evolve the scene, where all the scene generation and editing commands are provided in natural languages; see Figure 5.1. Specifically, at each editing iteration, an input natural language statement is turned into a semantic scene graph to retrieve a suitable sub-scene via graph alignment from the 3D scene database. The retrieved sub-scene is augmented with additional objects based on the input text and scene context, and then semantically aligned with the current scene. Finally, a new scene is synthesized by splicing the augmented sub-scene into the current scene, possibly triggering appropriate adjustments to the existing scene arrangement. In addition, our tool also supports scene edits with *verb* commands, which may trigger, e.g., an object replacement, as shown in Figure 5.1. Overall, the accommodation of the sub-scene into the current scene is guided by object co-occurrences and relations learned from 3D scene databases. Since natural language is inherently imprecise, we develop a suggestive interface to provide multiple interpretations of user languages so that the user can select one or more scene options during the modeling process.

Compared to the holistic example-based scene synthesis approach via probabilistic sampling [22] and the fine-grained progressive synthesis by sampling human actions [43, 58] or executing texts on precise object-level controls [17, 80, 12], our method makes the following contributions:

- 3D indoor scene synthesis which operates at the sub-scene level and accentuate the power and utility of language-driven scene modeling on collections of objects.
- A novel semantic scene graph for unifying natural languages and 3D scenes. This common representation is used for editing 3D scenes by incorporating knowledge about object compositions present in a 3D scene database.
- Annotation, learning, and application of a *relational model*, which describes pairwise and group-wise object relations, for 3D scene analysis and synthesis.

The key impacts of our sub-scene-level "retrieve-and-accommodate" approach to data-driven scene synthesis are two-fold. First, it improves the efficiency of scene modeling, reducing language redundancy and allowing scene complexity to increase more quickly. Second, since the retrieved sub-scenes are from realistically captured and modeled 3D scenes, the common-sense knowledge and scene semantics that are reflected by the object co-occurrences and arrangements within these scenes would directly go into the new scene. They do not need to be reproduced or re-sampled from scratch.

We show results of our language-driven scene synthesis and evolution, leading to the generation of realistic 3D indoor scenes with much improved versatility and complexity than previous methods. We conduct studies comparing our approach against both prior text-to-scene work and artist-made scenes and find that our method significantly outperforms prior work and is comparable to handmade scenes even when complex natural sentences are used.

5.2 Related work

There has been a great deal of work in computer graphics and computer vision on 3D indoor scene modeling, and so far, most efforts have been invested into *reconstruction*. 3D scene reconstruction takes as input one or more images or depth scans and aims to reproduce the captured scene geometry and even semantics. There are surveys on the topic, e.g., [90, 79, 100], to name just a few, and most recent works are taking a data-driven approach, e.g., [47, 82, 104]. In this section, we mainly focus on works most closely related to ours, i.e., those on 3D scene synthesis where textual inputs are used.

3D indoor scene synthesis. One line of work for 3D scene synthesis focuses on furniture layout optimization [27, 63, 110] for a given room with a given set of furniture. Example-based approaches rely on probabilistic reasoning from 3D exemplars and 3D scene databases to drive the synthesis [22, 43, 21, 78, 74]. A few recent works aim to populate a given 3D scene with small objects to increase its realism. Majerowicz et al. [60] develop a method to fill a shelf-like environment based on styles learned from a photograph or a 3D exemplar. The ClutterPalette by Yu et al. [111] offers an interactive tool to allow a user to insert and position one object at a time to mess up an otherwise clean scene. As the user clicks on a region of the scene, the tool suggests a set of suitable objects to insert, where the suggestions are based on support and co-occurrence relations learned from data. In contrast to these works, our language-driven scene editing tool follows a pipeline of interpret (from texts), retrieve and synthesize (from scene databases), and disambiguate (via a suggestive interface).

In Chapter 4, we introduce a framework for action-driven 3D scene evolution [58], which progressively alters a scene by object placements necessitated by human actions. We learn action models from annotated photographs and probabilistically sample a sequence of actions conditioned on scene contexts to evolve a scene. While the actions applied in there are labelled textually, e.g., "group dining" or "use laptop", and the texts may not explicitly describe object presence or placement, the action texts themselves are pre-determined (only 8 action types) with each manually bound to an appropriate class of scene contexts retrieved from the photos. In contrast, this work aims to learn a mapping between an immensely richer text corpus to object perturbation. Furthermore, none of existing works have

considered learning, applying, or adjusting object group relations, e.g., "messy", "aligned", etc.

There is a slight resemblance between the group actions supported in Chapter 4 and our group relation model here. For example, applying the action "group dining on table while sitting" would trigger the insertion of a group of objects into a scene, while in our language-driven scene synthesis framework, the object relations and placements would have to be more explicitly specified by a language command. On the other hand, all scene affections in the action-driven scene evolution framework must be executed through human actions. For example, "making the table clean" and "moving the chairs apart from table" are well-supported by our synthesis tool through group relations, but would be quite involved to realize by the action-driven approach; it would have to capture and learn actions involving humans tidying up a table or moving chairs.

Text2Scene. One of the earliest systems for text-driven 3D scene synthesis is Words-Eye [17]. This work, along with other follow-ups [80, 16], is capable of generating 3D scenes directly from natural language descriptions, but rely on manual mapping between language and object placements in a scene. The rigidity of the mapping forces the users to issue unnatural commands, e.g., "the chair is three feet to the north of the window". More recently, Zitnick et al. [118] learn visual interpretation of sentences by focusing on mapping visual features to semantic phrases extracted from the sentences, where the phrases describe binary relations that are either spatial (e.g., "Mike is after Jen") or semantic (e.g., "Alice wants the ball"). However, the generated scenes are two-dimensional, composed of 2D clip art elements. Also relevant is the recent work by Savva et al. [78], which is able to convert single-sentence descriptions (e.g., "He is lying on the couch" or "He is sitting in bed and using a laptop") into scene prototypes which depict simple human-object interactions.

A series of papers by Chang et al. have provided improvements over the early systems. The key improvement [11, 12] was the utilization of spatial knowledge, learned from 3D scene data, to better constrain scene generations by unstated facts or common sense. The first work [11] also allowed interactivity, but with object movements directly controlled by the user. The more recent improvement [8] is on lexical grouding of textual terms to 3D model references, i.e., on selecting more appropriate objects, by combining a rule-based model [12] and lexical grounding learned from user annotation of 3D scenes. In their latest paper [9], improvements were made mainly on the UI side; the technical core on text parsing and Text2Scene generation remain the same as [12].

The languages supported by all of these systems are explicit about object presence, but possibly implicit about object spatial relations. In our work, we allow both to be implicit and enable scene annotation, retrieval, and synthesis at the sub-scene level, with the additional consideration of group relations. Moreover, in previous works, the learned spatial knowledge is quite basic, e.g., only binary object relations [118, 11, 12]. In our work,

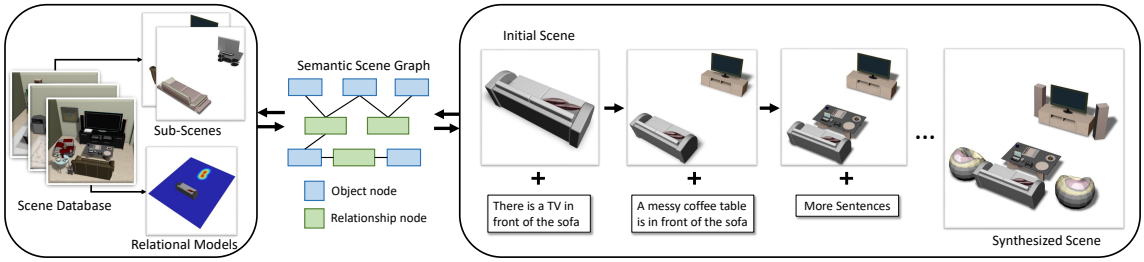


Figure 5.2: An overview of our language-driven scene evolution.

we learn and employ richer scene contexts for scene synthesis by setting up and utilizing a richer data source.

Robotics. To control a robot with textual commands is one of the central problems in robotics, e.g., [28, 92, 64]. Rather than learning to map language to object/scene arrangements, as for Text2Scene, the key problem in robotics is to map texts to robot motions, as well as robot-object interactions. However, both problems and their solutions share commonalities, including the need to utilize both rule-based models and data-driven learning of spatial and semantic relations, as well as the exploitation of common sense knowledge [92] to incorporate unstated facts.

Scene graphs. Fisher et al. [23] represent 3D scenes as object graphs with edges denoting pairwise semantic relationships, then use graph kernels to estimate scene similarity. In Chapter 3, we represent a 3D indoor scene by a graph of its constituent objects and detect focal points over a heterogeneous collection of 3D indoor scenes as representative sub-scenes. These focal points enable part-in-whole sub-scene retrievals and scene exploration, which could be adapted for our task. However, the object relations encoded in the scene graphs for focal extraction only include binary support, proximity, and symmetry groups. We convert natural language into a semantic scene graph representation that uses both pairwise relationships and multi-object relation annotations. We use partial graph matching to retrieve and align the query graph against a scene database, and use the retrieved environments for scene editing tasks.

5.3 Overview

Our overall framework for language-driven 3D scene evolution is composed of several components: 3D scene datasets serving as the knowledge base for scene processing, natural language processing which turns language commands to scene-related descriptions, and scene editing via language-driven synthesis; see an overview in 5.2. All of these components are strongly tied to a Semantic Scene Graph (SSG), the core data structure in our framework, as shown in Figure 5.3.

Semantic scene graph. The semantic scene graphs (SSG) encode the objects, as well as their attributes and relationships in a graph. By modeling both 3D scenes and text commands as semantic scene graphs, this uniform representation enables us to apply the edit specified by the command to the input 3D scene and update the scene with the help of the dataset.

3D scene dataset processing. We collect hundreds of annotated 3D scenes as the exemplars for scene editing. We construct the semantic scene graph for each 3D scene in the dataset. For later scene editing, we also learn relational models from all 3D scenes in the dataset by modeling the object co-occurrence and relative distribution. This database preprocessing step is executed only once.

Natural language processing. Our system evolves the scene with a sequence of natural language commands. Each command sentence can either specify a scene edit operation (e.g. "put plates on the table") or describe the changed parts of the scene after editing (e.g. "there are two plates on the table."). The language processing module transforms each sentence into a canonical semantic scene graph representation. For this purpose, we first extract the command and associated scene entities (i.e. objects, their attributes, and their relationships) from the sentence and then convert the command and entities into a scene semantic graph.

Language-driven scene synthesis. We edit the input scene according to the SSG parsed from natural language. If the language refers to a scene edit, we directly manipulate the target objects with the specified edit operations. Otherwise, we use the SSG of the input text to search the scene dataset and find a set of 3D sub-scene exemplars that best match the scene described in the text. We then update the SSG of the retrieved scene with the SSG of the input text and align the result SSG to the SSG of the input scene. After that, we insert new objects described in text into the input scene. The location of the new object is determined by the surrounding objects in the scene and the relational model learned from the scene database.

5.4 Semantic scene representation

To enable matching between a text description and a 3D indoor scene, we employ *Semantic Scene Graphs* (SSG), a semantic scene representation which contains rich information about objects, as well as their attributes and relationships. Both 3D scenes and natural language descriptions of scenes are converted into this uniform scene representation through scene processing (Section 5.5) and natural language processing (Section 5.6).

A semantic scene graph is an undirected graph with labeled edges and two node types: object nodes and relationship nodes. Object nodes represent objects in the scene and may

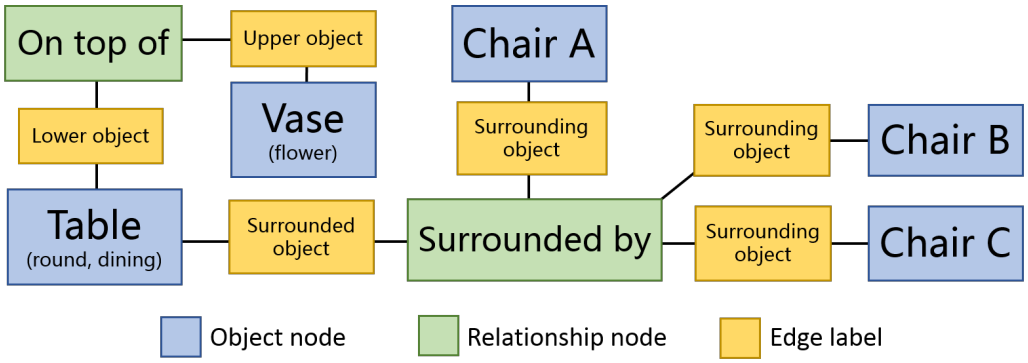


Figure 5.3: The semantic scene graph for the sentence "The round dining table is surrounded by three chairs and there is a flower vase on top of the table."

be annotated with a list of per-object attributes (e.g. "antique", "wooden"). Relationship nodes represent a specific instance of a relationship between two or more objects (e.g. "to the left of", "on each side of", "around"). Each edge connects an object node to a relationship node and is labeled with the type of connection between the object and relationship. Figure 5.3 shows an example of an SSG. Each relationship type has a pre-defined set of possible edge labels. Each instance of a relationship will have its own relationship node. In the rest of this chapter, edge labels are not shown for the simplicity of illustration.

Group relations may represent both spatial abstractions ("surrounded by") as well as abstractions over the object arrangement or composition ("messy", "organized"). This allows the SSG representation to capture many different ways that language encodes information about 3D environments and facilitates comparisons between scenes and natural language sentences.

5.5 3D scene processing

Our scene databases include 133 3D scenes from SceneSynth [22] and 80 scenes modeled from the real-world SceneNN dataset [32]. For each scene, we build a semantic scene graph to encode the object relationships. For each type of pairwise or group relationship, a relational model encoding object co-occurrences and relative distributions is learned from the database.

5.5.1 Database preparation

To enable semantic matching of objects between natural language and 3D models, consistent and meaningful semantic labels need to be annotated on each of the database models. We use annotations from ShapeNetSem [76, 10] for object labels such as category, attributes and front orientation. Each real-world scan from the SceneNN dataset is also converted

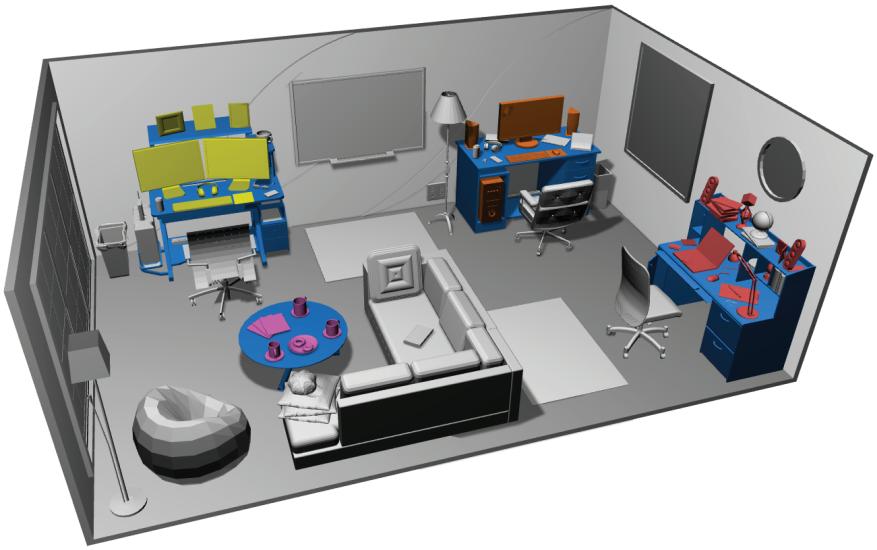


Figure 5.4: A database scene annotated with group relationships. Each object in blue corresponds to the anchor object in one relation group; other objects in other color are the active objects annotated for the corresponding group: yellow, orange - *organized*; red - *messy, disorganized*; magenta - *casual*. Note that yellow and orange objects belong to two independent *organized* group; the group of red objects are annotated with two types.

into a synthetic scene to facilitate the learning process. It is done by retrieving the best match 3D model from SceneSynth using the object tag in SceneNN.

Unlike previous methods which focus on pairwise relationships, we also consider group relationships, which enable more complex text to scene processing. We augment the scenes with high-level group relationships by annotating sub-scenes with semantic labels on corresponding objects for scenes in SceneSynth and SceneNN. In our current work, six high-level group relationship types including *messy, clean, organized, disorganized, formal, causal* and two spatial group relationships, *around, aligned* are considered (see Figure 5.4). These group relationships are extracted from a set of sentences that people used to describe the indoor environments (Section 5.6.3). As some relationships such as *messy* and *disorganized* have similar arrangements in the scene, we annotate the such groups with multiple types to provide more instances for each group.

5.5.2 Semantic graph from scene

To build the semantic graph, we first build an object node for each object in the scene and save its category as a node label. We also encode enriched object annotations from ShapeNetSem as node attributes to describe the refined properties of an object. Attributes may refer to many different properties of an object, including shape (ex. “round”, “square”), color, material, or usage (“dining”, “study”), which could increase the accuracy and control for retrieval of specific object instances as studied in Chang et al. [8].

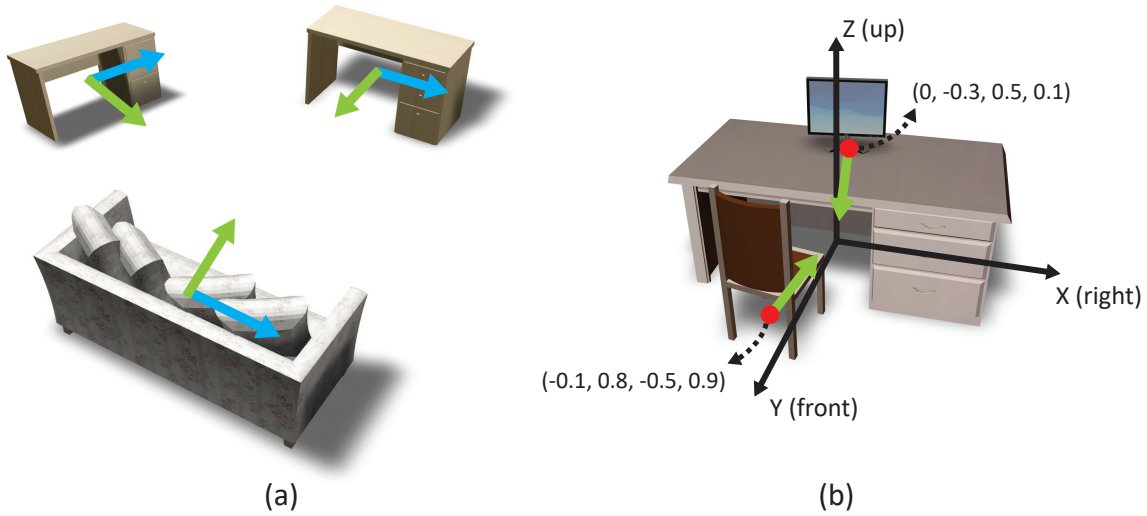


Figure 5.5: (a) a scene with two desks and a sofa annotated with adjusted object-centric frame: front (green), right (cyan); up is same as the scene's up and is not shown here; (b) position (red dot) and orientation (green arrow) of chair and monitor in the frame originated at the center of the desk; relative position is normalized by desk's bounding box size and orientation is normalized by pi, and together they are represented as (x, y, z, θ).

Next, we need to extract the relationships between objects and encode them into the graph. We use the similar approach as in [12] to extract a set of pre-defined spatial relations by examining arrangements of the bounding boxes of two objects. This list of spatial relations include: *on (vertical support), on left, on right, on center, under, left side, right side, front, back, near.*

As spatial relations (such as "left side" and "front") are ambiguous when using different reference frames, previous works [11, 12] use a view-centric coordinate frame to extract and apply such relationships. Unlike them, we employ an object-centric reference frame to extract spatial relationships between objects. This allows us to directly extract and retrieve view-oblivious object relationships from database scenes for future language-driven object layout. For each object, we first define a coordinate frame based on its front facing direction. Then, we adjust this object-centric frame based on the human-centric frame, in which the object would be interacted with by human (Figure 5.5(a)). Given a pair of objects, we define one object as the *anchor object* and the other as the *active object*, and record the relative position and orientation of the active object in anchor's frame for future learning (Figure 5.5(b)).

For each relation (pairwise or group), we create a relation node and connect it with the related objects. Labels are added to the edges based on whether the connected object is an anchor or active object. Unlike previous approaches that encode relations as edge labels, we add relation nodes to the semantic graph, which facilitates representing group relationships

more clearly, as all involved objects in a group relation will be linked to just one relation node.

5.5.3 Relational model

Given the scene database, we learn a *relational model* to encode object relationships (pairwise or group) based on a text description. The relational model contains an *arrangement model* $\mathcal{A}$, which records the spatial distribution of objects w.r.t the anchor object, and an *object occurrence model* $\mathcal{O}$, which describes the occurrence of individual objects and the co-occurrence of object pairs in a group relationship.

Given an object pair, we define the score of the pairwise arrangement model as

$$\mathcal{A}(o_{\mathbf{act}}, o_{\mathbf{anchor}}, r) = \mathcal{P}(x, y, z, \theta), \tag{5.1}$$

where $o_{\mathbf{act}}$ is the active object, $o_{\mathbf{anchor}}$ the anchor object, and r the relationship between them. $\mathcal{P}(x, y, z, \theta)$ is the probability distribution of relative position $[x, y, z]$ and θ the orientation between the active object and the anchor object. We first assume each arrangement model to be a Gaussian mixture and learn the parameters from observations in the database scenes similar to Fisher et al. [22].

As for some relations, observations of certain object categories may be limited and cannot be fitted with reliable distributions (e.g., a stapler on the desk), we record all observed (x, y, z, θ) tuples in the corresponding arrangement model when the number of observations is less than 15. In this case, the discrete arrangement probability for a new data point is defined as follows: $\mathcal{P} = 1$ if the point is close to any saved observation within a certain threshold (5 cm for the relative position and 15° for the relative orientation), otherwise $\mathcal{P} = 0$.

For a group relationship, we first define the occurrence model which characterizes the occurrence probability of an object in a group as

$$\mathcal{O}(o_i^m, r) = C(o_i^m, r)/C(s, r), \tag{5.2}$$

where $C(o_i^m, r)$ is the number of scene s annotated with relationship r that are observed with m instances of o_i. $C(s, r)$ is the total number of scenes annotated with r. We also compute the co-occurrence probability of an object pair in a group using

$$\mathcal{O}(o_i^m, o_j^n, r) = \frac{C(o_i^m, o_j^n, r)}{max\{C(o_i^m, r), C(o_i^n, r))\}}, \tag{5.3}$$

where $C(o_i^m, o_j^n, r)$ is the observation count of two objects with specified instance numbers co-occur in a group with relationship r.

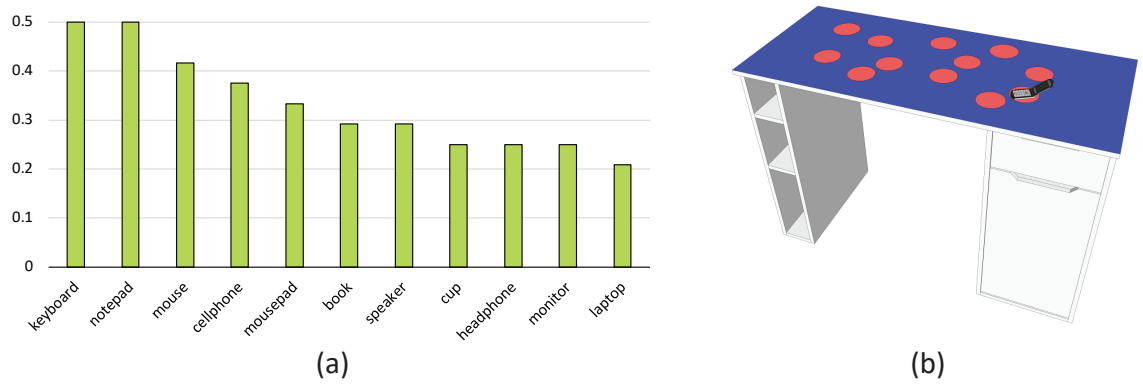


Figure 5.6: An example of learned group relational model *(desk, messy)*: (a) occurrence probability of active objects; (b) arrangement of cellphone on desk; as the observation number is not enough to fit a good distribution, we store all observations and use the discrete arrangement probability for placement of cellphone.

The arrangement model for a group relationship is represented as the sum of weighted pairwise arrangement scores:

$$\mathcal{A}(\mathbf{O}, r) = \sum_{o_i, o_j \in \mathbf{O}} \omega \mathcal{A}(o_i, o_j, r), \tag{5.4}$$

where $\mathbf{O}$ is the set of objects in the group; ω is the weight for the corresponding pairwise model $\mathcal{A}(o_i, o_j, r)$, which equals to 1, if one object in the pair is the anchor; otherwise ω is set to $\mathcal{O}(o_i, o_j, r)$, i.e., the co-occurrence probability of object instance o_i and o_j in the group. Figure 5.6 shows a learned model for a group relationship.

Relational model similarity. Some object categories do not have enough observations to learn a good distribution model. To enrich placements of such objects, we define the similarity between relational models and sample from similar relational models to find additional placements for an object when needed. For any two pairwise relational model with the same relation r, their similarity is defined similarly to the *graph kernel* in Fisher et al. [23]. Specifically, we define

$$Sim(\mathcal{A}_o, \mathcal{A}_{o'}) = k(o_{\mathbf{anchor}}, o'_{\mathbf{anchor}}, r) k(o_{\mathbf{act}}, o'_{\mathbf{act}}, r), \tag{5.5}$$

where $\mathcal{A}_o$ is short for $\mathcal{A}(o_{\mathbf{act}}, o_{\mathbf{anchor}}, r)$ and $\mathcal{A}_{o'}$ is defined similarly; $k(\cdot, \cdot, r)$ is the node kernel of object pair with relationship r:

$$k(o, o', r) = 0.5\delta_{\mathbf{cat}}(o, o') + 0.5k_{\mathbf{geo}}(o, o', r), \tag{5.6}$$

here $\delta_{\mathbf{cat}}(o, o')$ is a Dirac delta function which returns 1 if object o and o' are in the same category and 0 otherwise; $k_{\mathbf{geo}}(o, o', r)$ encodes the similarity of their geometry features

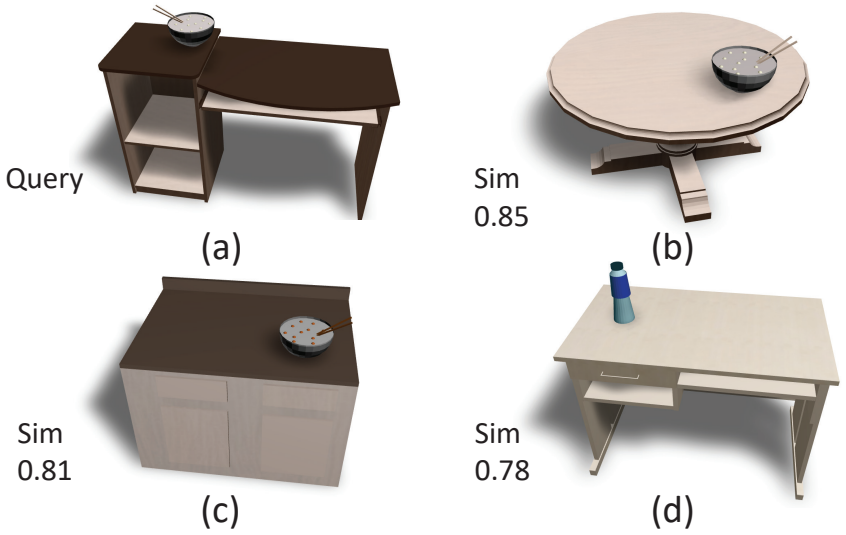


Figure 5.7: a query pairwise relational model (a) $\mathcal{A}(bowl, desk, on)$ and three retrieved models (b) $\mathcal{A}(bowl, table, on)$ (c) $\mathcal{A}(bowl, counter, on)$ (d) $\mathcal{A}(bottle, desk, on)$ ranked by the relation similarity.

including elevation to floor, bounding box height and volume. Figure 5.7 shows some similar relational models retrieved from the database using the query.

Relative relation priors. When generating scenes based on language, object placements must satisfy the *explicit* relations that are specified by the input sentence. Moreover, as there might be objects already in the scene, *implicit* relations imposed by existing objects must also be satisfied when placing a new object. Therefore, in addition to the specific pairwise or group relations, we also learn the *relative* relations, which encode arrangements between all pairs of objects in the dataset to provide prior knowledge of implicit constraints for object placements. The relative relation model is represented as $\mathcal{I}$ and defined similar to Equation 5.1, but without the specific relationship constraint r. The relative relation priors and the specified relational models are jointly used to provide plausible constraints when placing new objects (Section 5.7.2).

5.6 Natural language processing

People use many types of language to interact with scenes. Sentences such as "there are three plates on the table" are descriptive, indicating the way the user desires the scene to be. On the other hand, a sentence such as "put three plates on the table" are commands, indicating changes the program should make to the scene. In both cases, we attempt to transform the natural language input into an equivalent semantic scene graph as detailed in Section 5.4. Sometimes it is not be possible to represent a command as a scene graph;

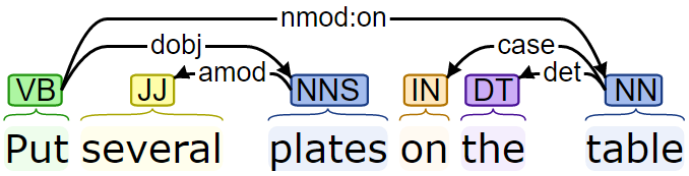


Figure 5.8: An example dependency parse using "Enhanced++ Dependencies" from Stanford CoreNLP. VB=verb, JJ=adjective, NN(S)=(plural) noun, IN=preposition, DT=determiner

ex. "remove the plates from the table". We represent such sentences as scene graphs with verb annotations that connect to entities in the graph.

5.6.1 Text parsing

We use the Stanford CoreNLP framework [61] to perform part-of-speech tagging and convert input statements into a dependency tree. This dependency tree assigns a parent token and annotation label to each token in the sentence; an example is shown in Figure 5.8. Parsing natural sentences is an inherently ambiguous process; reported on-corpus part-of-speech accuracy is 97% and dependency parsing accuracy is 91%. In practice we found our parsing accuracy to be lower as our corpus is dissimilar from the newspaper articles the Stanford CoreNLP model was trained on.

5.6.2 Entity-command representation

We seek to convert the low-level dependency representation shown in Figure 5.8 into a list of entities annotated with attributes and relationships, as well as a list of command verbs which operate over these entities. We call this the *entity-command representation* of the sentence. We use the term entity as opposed to object for cases where language refers to abstractions over groups of objects, ex. "the seating arrangement".

A scene entity consists of the following:

- **Category** — the base noun used to describe the object. Ex. "table", "plate".
- **Attributes** — a list of attributes, each of which may have a set of modifier words. Ex. "modern", "blue (very, dark)".
- **Count** — either an integer representing the number of entities in a group, or a qualitative descriptor. Ex. "many", "some".
- **Relationships** — a set of (string, entity) pairs that describe a connection to another specific entity in the sentence. Ex. "on:desk", "left-of:keyboard".
- **Determiners** — a list of determiners such as "a", "another", "the", "each". These are useful for estimating object binding (Section 5.7.2).

A command verb is defined by the following:

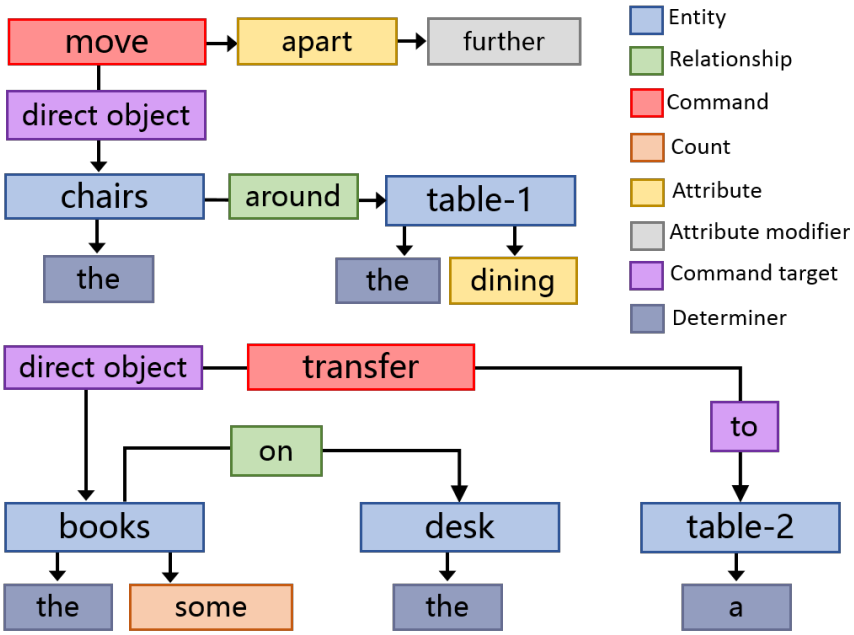


Figure 5.9: The sentence "Move the chairs around the dining table further apart and transfer some of the books on the desk to a table" in our entity-command representation.

- **Base verb** — the base verb used to describe the command. Ex. "move", "rearrange".
- **Attributes** — a list of attributes, each of which may have a set of modifier words. Ex. "closer", "dirty (more)".
- **Targets** — a list of (string, entity) pairs that represent different types of connections to specific entities. Ex. "direct object:laptop", "onto:table".

Figure 5.9 shows an example of this representation. At this stage, we only extract all explicitly specified information and do not make use of implied relationships, such as "there is a mouse to the left of the keyboard" implying the keyboard is to the right of the mouse.

Entity and command extraction. We take all nouns in the sentence to be entities unless one of the following conditions is met: the noun is in a compound dependency relationship with another noun (ex. "computer" in "computer desk"); the noun is an abstract concept (ex. "addition", "appeal"); the noun represents a spatial region (ex. "right", "side"). We take all verbs in base form to be commands.

Coreference. To dereference pronouns, we use the coreference information obtained from the CoreNLP framework. This assists with sentences such as "Add a dining room table and put plates on top of it", where we will not create a new entity called "it". However, we do not use coreference for non-pronoun scenarios such as the two table entities in Figure 5.9. In this case, it is possible that the two tables refer to different tables in the scene, and we will resolve this ambiguity when we align entities to objects in the user's scene (Section 5.7.2).

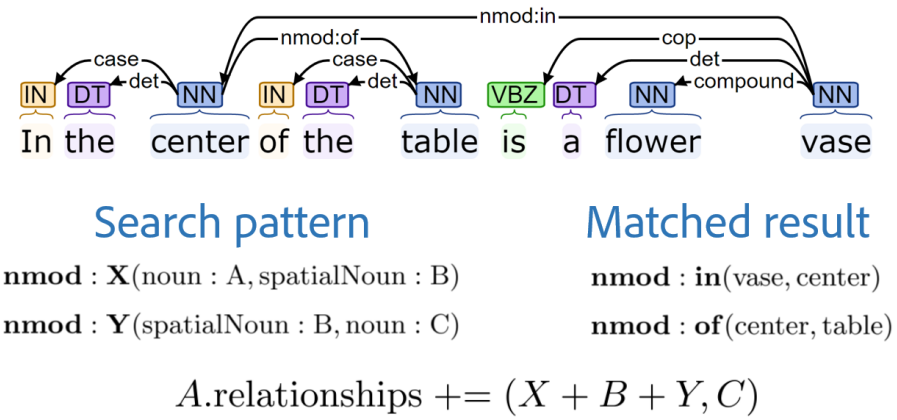


Figure 5.10: A pattern matching example using spatial nouns. The "in center of:table" relationship will be recorded for the vase entity.

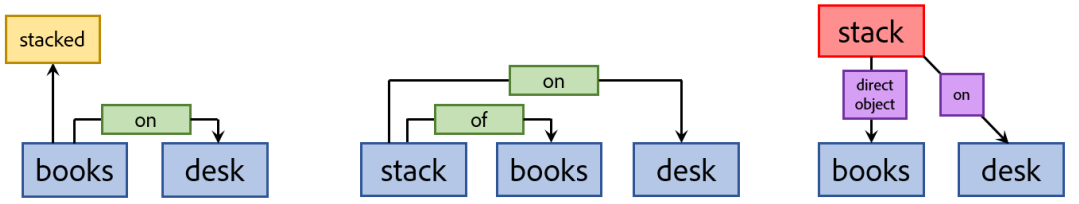
5.6.3 Pattern matching

After determining the seed tokens for all scene entities and commands, we use pattern matching over the dependency parse to assign all other properties. For some dependencies, this pattern matching is very simple; for example, **amod**(noun : **A**, adjective : **B**) assigns token **B** as either an attribute or a count of the entity seeded at **A**, if one exists (see Figure 5.8 for an example). When pattern matching, we augment the standard parts of speech used by our dependency parser with four special classes that are important for scene understanding:

- **Spatial nouns** — Spatial regions relative to entities. Ex. "right", "center", "side"
- **Counting adjectives** — Adjectives representing object count or general qualifiers. Ex. 'all", "many"
- **Group nouns** — Nouns that embody special meaning over a collection of objects. Ex. "stack", "arrangement"
- **Adjectival verbs** — Verbs whose effect can be modeled as an attribute modification over the direct object. Ex. "clean", "brighten"

An example of a more complicated pattern involving spatial nouns is shown in Figure 5.10.

To build our pattern matching database, we collected a set of twenty photographs of complex indoor environments and distributed them among ten different participants. One group was asked to describe the scene in such a way that another participant could reconstruct the important properties of the scene. Another group was told to provide a set of instructions for building the scene starting from an empty room, encouraging the use of command verbs. The participants were not aware that the sentences would be parsed by a computer program, encouraging the use of a wide range of language constructs.



(a) There are books stacked on the desk. (b) There is a stack of books on the desk. (c) Stack books on the desk.

Figure 5.11: Entity-command representation of three sentences representing the same underlying concept. We define the descriptive form, (a), to be canonical and transform (b) and (c) into (a) by group noun transformation and verb application, respectively.

In total we acquired 170 sentences in this way. We used the dependency parse from 100 such sentences to build our pattern matching database such that our system was able to correctly annotate all entities and commands in these training sentences (we skipped parts of sentences where the dependency parser failed to produce a correct parse tree). The remaining 70 sentences were reserved as a test set; the evaluation on these sentences is in Section 5.8.

5.6.4 Canonical entity-command representation

For a given sentence, assuming a correct parse there is a single corresponding entity-command representation. However, the same scene editing concept can be expressed in many different ways, resulting in different representations; see Figure 5.11 for an example. We define the descriptive form, which minimizes the number of commands and entities, to be canonical and when possible transform our input entity-command representation into the descriptive form. These transforms are also executed via a set of pattern matching rules. One such pattern detects group nouns and transforms them into adjectival form, transferring all relationships on the group to the compositional entity (Figure 5.11(b)). Another pattern detects adjectival verbs, converts them into adjective form, and applies this as an attribute to all targeted entities, also transferring any command targets to relationships on the direct object (Figure 5.11(c)). We also define transform rules for verbs such as “put”, “place”, and “add” that do not imbue any attributes on the targeted objects.

Not all commands can be applied as a graph transform. For commands such as “delete all the chairs around the table” or “rotate the monitor 90 degrees”, we leave these commands unchanged and use specialized functions to execute them on the scene (or inform the user that the command was not understood.)

5.6.5 Conversion to a semantic scene graph

Our entity-command representation transforms easily into a semantic scene graph as defined in Section 5.4.

Base noun to object category. Our model database uses a fixed set of object categories, and we start by mapping the base noun for each entity into a corresponding object category. We use equivalence sets derived from WordNet[1]. We discard entities that do not map to a category in our model database. Attributes and determiners are added as annotations to the corresponding object nodes.

Entity counts. When the ECR indicates multiple copies of an object exist, we instantiate a new object node in the SSG for each object instance. For integer counts, this is straightforward. For imprecise counts such as "some" and "many", we first obtain a frequency histogram for each object category by examining the scene database and counting the number of occurrences of 2 or more instance of the category. For each counting modifier, we use this distribution to obtain a lower and upper bound on the count implied by this modifier-category pair, then sample uniformly from this distribution. For example, "few" samples between the 0th and 25th percentiles, "some" between the 10th and 50th percentiles, and "many" between the 50th and 100th percentiles. Plural nouns without a modifier (ex. "there are chairs around the table") are taken to have an implied "some" modifier. Relationships, attributes, and determiners are duplicated across each new instance of an object.

Qualifiers. Some qualifiers such as "each" and "all" imply the presence of multiple object nodes, but are left as qualifiers over a single object node until the SSG is grounded (Section 5.7.2).

Relationships. Relationships in the ECR transfer directly into relationship nodes in the SSG. The edge label is determined by the directionality of the relationship in the ECR. Relationships that support multiple objects are grouped together into a single relationship node in the SSG; for example "the table is surrounded by two benches and some chairs" will create one "surrounded by" relationship node with appropriate edge labels shown in Figure 5.3.

5.7 Language-driven scene editing

Given a semantic scene graph constructed from an input sentence (Text-SSG, noted as $\mathcal{T}_g$), our system has two modes for evolving the scene given by the user (User-SSG, noted as $\mathcal{U}_g$). For graphs without a verb node (Figure 5.12(b)), our system starts by aligning $\mathcal{T}_g$ with semantic scene graphs of the database scenes (DB-SSG, noted as $\mathcal{D}_g$) and finds a subgraph (Sub-SSG, noted as $\mathcal{S}_g$) that best matches the given sentence (Figure 5.12(c)). Unaligned nodes from the sentence are added to $\mathcal{S}_g$ as synthesized nodes. For each updated

[1] http://wordnet.princeton.edu

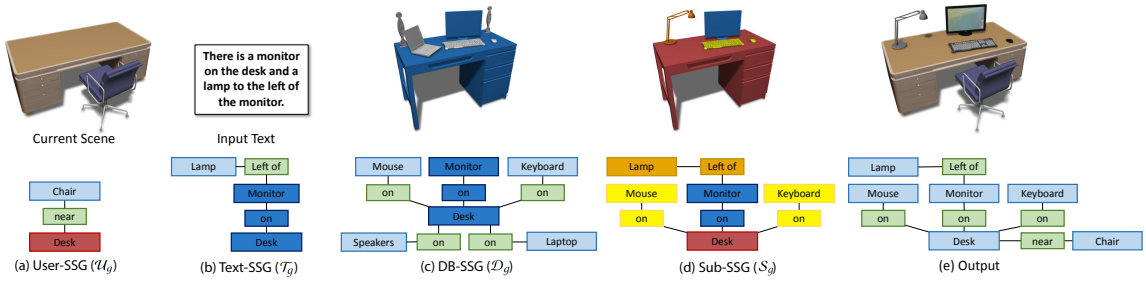


Figure 5.12: (a) an input scene and its graph $\mathcal{U}_g$; (b) an input text and its graph $\mathcal{T}_g$; (c) a retrieved database scene and $\mathcal{D}_g$ using the text with aligned nodes shown in blue; (d) a subscene $\mathcal{S}_g$ extracted from $\mathcal{D}_g$ with synthesized nodes (in orange) and nodes enriched by context (in yellow); desk is in red since it is aligned to the desk in (a); (d) updated scene by transforming objects from the subscene to current scene using desk as the anchor object.

$\mathcal{S}_g$, we align and bind its nodes to the semantic scene graph of the user's current scene and insert unaligned objects to current scene based on their relationship to the aligned objects (Figure 5.12(d, e)). For graphs with verb nodes, we directly align their object nodes to the objects in current scene, and execute the scene editing functions specified by the verb. By repeating this process, complex and realistic scenes can be generated by using a sequence of sentences.

5.7.1 Sub-scene retrieval and augmentation

Our goal is to extend the current scene by the obtained set of sub-scenes as well as the specified relationships in the text. One way to generate a candidate sub-scene is to synthesize it with object distributions learned from a 3D scene database (e.g. as proposed in Text2Scene [12]). However, synthesizing complex scenes by explicitly specifying each object is tedious due to the number of objects and the inherent ambiguity of language. To overcome these limitations, we employ the scene context from already existing scenes. We retrieve related sub-scenes by aligning the $\mathcal{T}_g$ to each $\mathcal{D}_g$ from our scene database. The output is a list of $\mathcal{S}_g$ ordered by the matching score to the $\mathcal{T}_g$.

Alignment metric. A node in the $\mathcal{T}_g$ represents an object or a relationship that the user explicitly wants to be present in the intended scene. We align these nodes with the nodes of every $\mathcal{D}_g$ to find the matches that best correspond to the candidate sub-scenes (blue nodes in Figure 5.12(c)). A node in the $\mathcal{D}_g$ can only be aligned with one node in $\mathcal{T}_g$, while already aligned nodes in $\mathcal{D}_g$ will be skipped. An object node is aligned when its category and associated attribute labels are all matched, and a pairwise relation node is aligned when its type and two connected object nodes as well as the edge labels are matched. For a group relation node, since the text may only mention the anchor object to represent a group, e.g., a messy table, we set the node to be aligned when its type and connected anchor object are

matched. Ideally, an exact match is expected from the database. However, it is more likely that some nodes of the $\mathcal{T}_g$ will not be aligned properly. To find the best match for a given graph, we rank the alignment based on following metric:

$$Score(\mathcal{T}_g, \mathcal{D}_g) = \sum_{N_i \in \mathcal{T}_g, N_j \in \mathcal{D}_g} M(N_i, N_j). \tag{5.7}$$

Here N_i and N_j are the nodes from $\mathcal{T}_g$ and $\mathcal{D}_g$, respectively; $M(N_i, N_j)$ equals to 1 if N_i and N_j is aligned, and 0 otherwise.

Language-driven graph augmentation. As the database is unlikely to store a $\mathcal{S}_g$ exactly as specified by the given text, some nodes in $\mathcal{T}_g$ may be unaligned because an object instance is missing, or some relationships are not satisfied by the database scene. Since every node from the input text is critical to produce the intended scene, we synthesize missing nodes in the subgraph to ensure exact alignment to the given text. New object or relation nodes are added to the original $\mathcal{S}_g$ so that each of them is matched with an unaligned node in $\mathcal{T}_g$. Next, we link the synthesized nodes to existing ones according to the edge connections specified in $\mathcal{T}_g$. By now, we get a $\mathcal{S}_g$ (the graph without the yellow nodes and related edges in Figure 5.12(d)) with all nodes and edges exactly aligned to the $\mathcal{T}_g$. Moreover, as the group relation node is only aligned based on the anchor object, we further synthesize nodes for active objects in the group based on the occurrence and co-occurrence model, and add their connections to the anchor object. Since a synthesized object node does not correspond to a concrete object instance, we use the same object if an object with same category appears in current scene, otherwise we randomly sample an object from the database according to its category.

Enriching $\mathcal{S}_g$ with scene context. Natural language does not conclusively describe objects and their relationships. Moreover, it is challenging to produce complex scenes and evolved object relationships by only specifying them with text. To enable the generation of scenes with a high level of detail and complexity, we exploit the stored scene context from the scene database to enrich the sub-scenes defined by the input text. Similar to Chang et al. [12], we use the scenes in the database to learn the support hierarchy of objects. Knowing about the support relationships can be used as a prior; e.g., if the most likely support parent of an object node is not present in a retrieved $\mathcal{S}_g$, we find its parent node in the original $\mathcal{D}_g$ and add related nodes to the subgraph. Moreover, we incorporate more scene context to enrich sub-scenes by introducing relevant objects based on their co-occurrence probability to the initial objects encoded in the $\mathcal{S}_g$. The co-occurrence probability of two object categories is defined in Equation 5.3, while the relationship r here means the object is supported by a parent object o_p. Objects with co-occurrence probability larger than the context control parameter α will be added to the subgraph.

5.7.2 Sub-scene accommodation

A $\mathcal{S}_g$ represents a subscene retrieved based on the input text. To edit a given scene, we further bind $\mathcal{S}_g$ to $\mathcal{U}_g$ (the scene the user is editing) by first merging the two graphs and then updating the scene layout accordingly.

Graph alignment and merging. During the scene evolution, sentences specified by the user commonly contain objects that already existing in the scene. For example, given the input "There is a desk to the right of the bed", a bed might already be in the scene. Such objects could be used as anchors for graph alignment and merging. Specifically, we align $\mathcal{S}_g$ and $\mathcal{U}_g$ using the similar way of aligning $\mathcal{T}_g$ and $\mathcal{D}_g$ (Section 5.7.1), but with consideration of determiners extracted from $\mathcal{T}_g$ of an object. Besides the category, two object nodes are aligned when the corresponding determiner of the object is *"the"*, with 50% likelihood if the determiner is absent or *"a"*, and are never aligned if the determiner is *"another"*. Then we merge the $\mathcal{S}_g$ to $\mathcal{U}_g$ by using the aligned object node in $\mathcal{U}_g$ as anchor and add all non-aligned nodes and related edges from $\mathcal{S}_g$ to $\mathcal{U}_g$. In the 3D scene, object instances and their arrangements in the original $\mathcal{U}_g$ will be kept; objects corresponding to newly added nodes are inserted into the current scene, while their relationships are resolved by layout adjustment in next stage.

Layout adjustment. Newly inserted objects come with positions as they are stored in the original database scenes. Therefore, we need to adjust their location to satisfy all relationships encoded in the updated $\mathcal{U}_g$. We compute a transformation matrix to align the anchor object in $\mathcal{S}_g$ and its correspondent in the $\mathcal{U}_g$, and set it as the initial transformation matrix for new objects. Since the anchor object and its correspondent may have different geometric features, applying the initial transformation to the new objects is likely to cause artifacts, e.g., intersection, object floating and implausible orientation.

We refine the layout of an object based on its specified relationship to the anchor object, as well as its implicit relations to other existing objects in the scene. For an object involved in a pairwise relationship, we define the layout score as follows:

$$Score(o) = \mathcal{L}(o) \cdot \mathcal{H}(o) \cdot \mathcal{R}(o). \tag{5.8}$$

Here $\mathcal{L}(o)$ is the *collision penalty* term which returns 0 if o intersects any object in the scene and 1 otherwise; $\mathcal{H}(o)$ is the *overhang penalty* term defined similar to that in [22] to prevent o hanging off the edge of a supporting surface. $\mathcal{R}(o)$ includes all relation constraints for o:

$$\mathcal{R}(o) = \omega \sum_{r \in \mathbf{E}} \mathcal{A}_o^r + (1 - \omega) \sum_{o_i \in \mathbf{O}_u} \mathcal{I}(o, o_i), \tag{5.9}$$

where $\mathbf{E}$ contains set of explicit relationships from the text and $\mathcal{A}_o^r$ represents the arrangement score for the relationship r of o; $\mathbf{O}_u$ is the set of objects in current scene and $\mathcal{I}(o, o_i)$ is relative relation prior between o and o_i; ω is set to be 0.7 in our current implementation to weight more on the explicit constraints. Layout scores for group relationships are extended by summing up the layout score of each object in the group.

Ideally, the placement of an object retrieved from a sub-scene would immediately produce the maximum layout score. In practice, the object arrangement may already violate the relation constraints after being aligned to the scene. Therefore, we define a layout quality threshold for an object based on the observed distribution for the explicit relation and its relative relations to existing objects. When the initial alignment causes an intersection or fails to pass the threshold, we optimize the above layout score by hill climbing and adjusting the layout of the object using the placement that produces the maximum score. New candidate locations are sampled depending on the distribution learned in the relational model. A location is blocked from future sampling if the layout score returns 0. In the case that there is no distribution learned and all observations in the current relational model have been tested and failed, we sample from the most similar relational models to find more candidate positions.

For placement of a group, object positions are adjusted in the order based on their level of support hierarchy and bounding box sizes. Thus, larger objects which provide support to other objects will be placed first. As the object number keeps increasing during the scene evolution, there might be no position to place a new object. Similar to the strategy used in Chapter 4, when the placement of an object fails up to a prescribed threshold, we allow the layout algorithm to roll back to the previous placed object, modifying its placement in seeking of a relaxed solution. If an object still cannot be placed after one roll-back step, we will skip placing this object and return a failure message to the user.

Scene editing by verb commands. If the $\mathcal{T}_g$ contains a verb node, we directly align its related nodes with the current $\mathcal{U}_g$ to find anchor and target objects. We define a set of functions based on commonly used verbs for scene modification: $Replace(A)$, $Move\_to(A, B)$, $Move\_on(A, B)$, $Move\_closer(A, B)$, $Move\_apart(A, B)$, $Delete(A)$, $Rotate(A, degree)$, $Scale(A, value)$ where A is the target object or objects for the verb command, and B is the anchor object. The effect of these functions is the same as indicated by their names. Although some of these editing functions can also be performed through a click-and-drag UI, defining these operations through verb commands allows to more efficiently editing of groups of objects. For example, replacing the chairs around a table using a normal 3D UI may involve inserting new instance of chairs, aligning them with each existing chair and delete the original chairs, which would need a larger number of operations.

5.7.3 Suggestive interface

Our system provides a suggestive interface to support two-way communication with a user working with our system (as shown in Figure 5.1). For each input sentence our system produces a set of suggestions (set to 5 when producing all results in this chapter). The user can interactively explore all suggested scenes and choose one she favors most for next iteration of text to scene generation. When ambiguities exist in input text, the system returns possible scene arrangements as suggestions. For example, when there are two desks in the current scene, and the textual command is "put a monitor on the desk", our system will return two possible results that the monitor is on either one of the desks.

The suggested scenes satisfy the constraints in the $\mathcal{T}_g$, while they also show possible variations in terms of objects as well as their arrangement. To rank the scenes shown in the suggestion list, we define a simple screen space visual similarity metric that compares pixel-wise color difference among the images rendered from 6 views of the result scenes. We sort the scenes in the order of their visual dissimilarity from high to low and show the suggestions to the user. To further improve the variation of the synthesis results, the user can always use the verb command *Replace* to change the instance of an existing objects. The layout of all related objects, e.g., children or neighbor of the changed objects will be automatically updated using our layout adjustment algorithm.

5.8 Results and evaluation

We first evaluate accuracy of converting natural language into our entity-command representation. Then, we present results of our language-driven scene synthesis method, evaluate its performance, and compare the results to those created by artists and those obtained by the state-of-the-art text-to-scene generation method of Chang et al. [12].

Entity-command representation evaluation To evaluate the accuracy of our method for converting natural language into ECR, we generated entity-command representations for the 70 natural language sentences described in Section 5.6.3. As users generated these sentences without the expectation that they would be parsed by a computer, they demonstrate many of the failures of our entity-command representation and evaluating on these sentences provides a rough lower-bound on the parsing accuracy that can be expected in a practical system.

Table 5.1 shows the accuracy of ECR conversion across test sentences, broken down by the token types delineated in Section 5.6. Many token types, such as attributes, counts, and determiners, are easily parsed. Difficult entities to parse include non-standard compound phrases such as "chest of drawers", which produces two objects in our system and results in hallucinated entities.

Token type	Recalled	Missed	Hallucinated
Entity category	98%	2%	11%
Entity attribute	100%	0%	0%
Entity count	100%	0%	0%
Entity relationship	76%	24%	18%
Entity determiner	100%	0%	0%
Verb base	84%	16%	11%
Verb attribute	100%	0%	8%
Verb target	85%	15%	23%

Table 5.1: Accuracy of converting natural language sentences into the entity-command representation. "Recalled" is the percent of ground-truth tokens that are correctly labeled as the given token. "Missed" are tokens that should have received this label but did not. "Hallucinated" are the ratio of tokens that are erroneously labeled to be of a particular type. When a base category/verb is not correctly isolated, tokens bound to them are discarded.

Scene synthesis results. Figure 5.16 shows a gallery of 3D scenes generated by our method, highlighting the various features offered. Taking advantage of sub-scene level scene synthesis, group relational models, as well as adding contexts from scene databases, we are able to achieve a much higher level of language efficiency for scene modeling than all previous attempts at text2scene generation. For example, a scene with 20 to 30 objects can be synthesized with only three sentences; see last column of Figure 5.16. More synthesis results can be found in Figure 5.1.

Parameter and timing. The main tunable parameter in our method is α, which controls the level at which context objects can be introduced to the scene. Throughout our experiments, unless otherwise noted, we set $\alpha = 0.5$. Timing-wise, conversion of input language to semantic scene graphs is instantaneous. Sub-scene retrieval and synthesis take 1-5 seconds to accomplish (on average) when the output scenes contain about 15 to 20 objects, respectively.

Verb commands. Verb commands serve as secondary editing options in our scene modeling tool to complement synthesis commands realized via sub-scene retrieve-and-accommodate operations. When a scene needs to be refined by object movements without the need for object or sub-scene retrieval, as shown in Figure 5.13(b), verb commands are appropriate. Object replacement is another scenario that is supported by verb commands, as shown in Figure 5.1 and Figure 5.13(c), allowing the user to more quickly refine the scene.

Comparison to [12]. The key evaluation for our method is whether the generated scenes are plausible and natural. Similar to previous works [21, 58], we leave such judgments to human participants, and first compare our results to those generated by the most closely related and state-of-the-art method of [12]. For the comparison, we used implementation

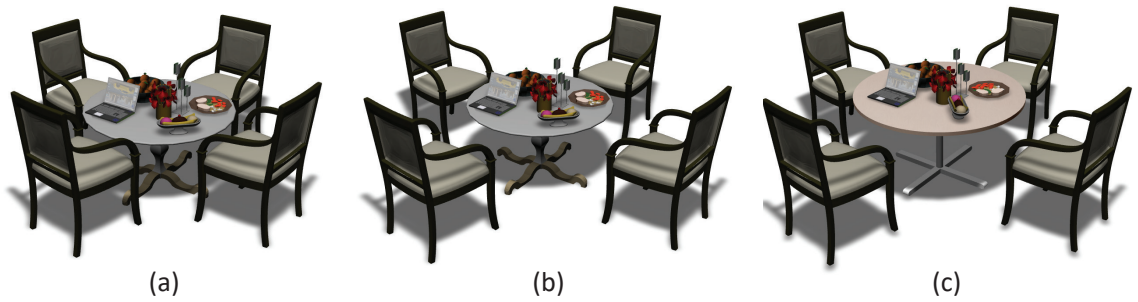


Figure 5.13: Verb commands applied to refine the current scene (a). (b): after "move the chairs apart from the table". (c): after applying the command "replace the table".

provided by the authors and executed it with the same underlying scene databases as our method.

We set up 10 editing scenarios, each described by three input sentences, with α set to 0 (i.e., no scene contexts). The editing commands cover bedroom, dining room, living room, and office scenes. Both pairwise and group relations were accounted for. However, since Chang et al. [12] only model pairwise relations, to ensure a fair comparison and help their tool obtain the best results, we split specifications of group relations into sets of pairwise constraints. We further chose the best result from Chang et al. out of five instances run through their provided implementation.

When applying our method, the first sentence in each editing scenario was used to generate five suggestions. For the next two sentences, two options are considered when selecting scenes from suggestions: *Our-random* corresponds to a random selection and *Our-user* corresponds to a user's selection as the most favourable result. To mitigate possible ambiguities caused by scene description sentences, we generate two scene variants independently for each editing scenario and each method/option: Our-user, Our-random, and Chang. Hence, there are a total of 20 scenes per method/option.

Our first user study is the Plausibility Test against Chang et al.[12] (PTC). We split the total of 60 scenes generated for this study into two sets. Each subject is shown three scenes from the three methods/options at a time. The subjects were asked to give a score from 1 (least favoured) to 5 (most favoured) to each scene based on two criteria: plausibility – whether the scene is plausible with respect to the given editing command, and naturalness – whether the scene and its object arrangement appear natural. We gathered feedback from 23 participants, resulting in 46 scores per test scenario per method/option and 460 scores per method/option.

Figure 5.14 plots the average subject scores and variances for each test scenario. Figure 5.15(a) plots the overall statistics. As can be seen, our method clearly outperforms Chang et al. [12] with average scores of 4.14 (Our-random) and 4.48 (Our-user), compared

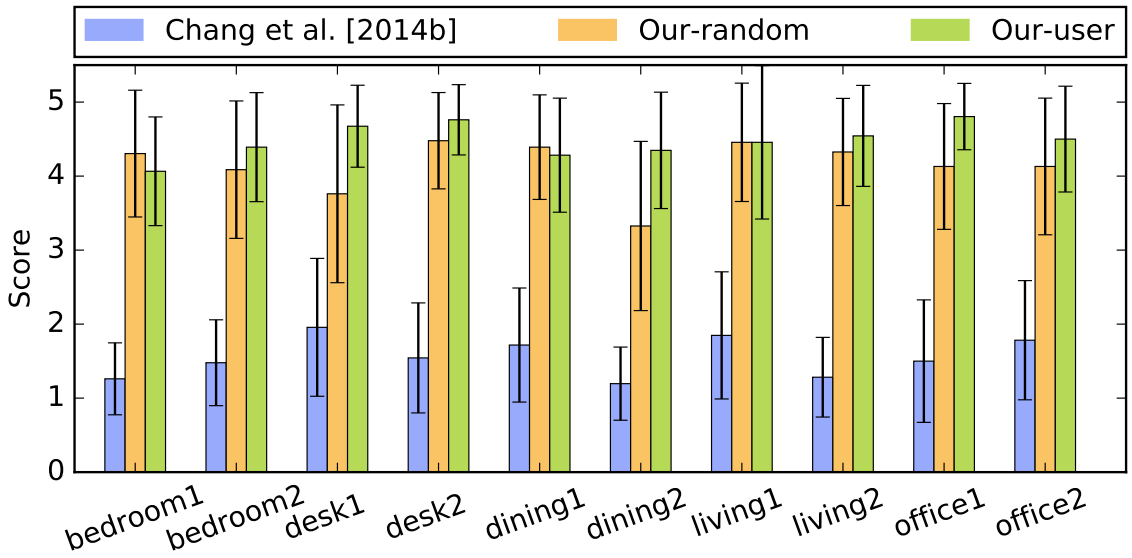
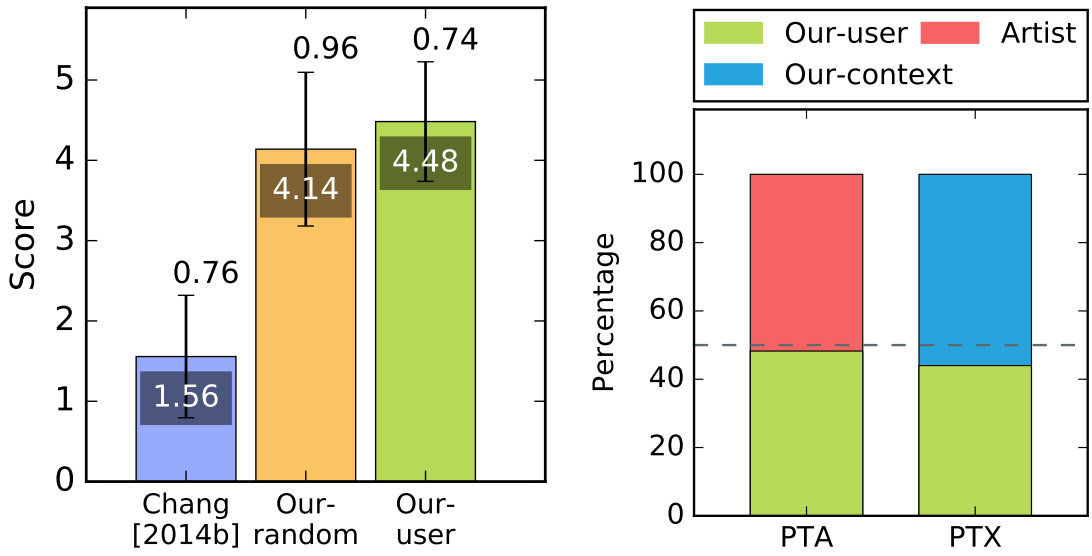


Figure 5.14: PTC study results: average plausibility-naturalness scores by subjects for all 10 scene editing scenarios.

to 1.56 for Chang. We attribute this to several improvements we made including our ability to handle group relations (Section 5.5.3), our transformation into a canonical graph representation (Section 5.6.4), and our sub-scene retrieval and alignment pipeline (Figure 5.12). Interestingly, Our-random is rated only slightly lower than Our-user, suggesting that our method has a good average performance.

Comparison to artist creations. Our second user study is the Plausibility Test against Artist (PTA), where we simply replaced the method of Chang et al. [12] by a professional artist we hired and essentially repeated the PTC study. When the artist created the scenes, she was given the natural language prompt, the same object database used by our tool, and was not limited by modeling time. At the end, the artist spent at least five minutes modeling each scene. Importantly, the artist had total freedom in choosing which objects to add to the scene, based on the language input, and how to place them using a modeling software. Hence, the comparison would cover the full spectrum of our scene synthesis method.

Instead of asking the subjects to score scenes, we asked them to *vote* between a pair of scenes, one created by the artist and the other an Our-user scene from PTC. The subjects were asked to vote based on their judgment of plausibility and naturalness of the scenes. We received feedback from 20 subjects, resulting in 400 subject votes over the 10 editing scenario since the artist also generated two scene variants per scenario. Figure 5.15(b) (first column) plots the average percentages of the artist and Our-user results being voted on by the subjects, respectively. Overall, our method received 48.25% of the votes, which is close to that of the artist (51.75%), indicating that our method is able to produce results that are comparable to average human performance.



(a) PTC average

(b) PTA and PTX average

Figure 5.15: Average PTC scores and PTA and PTX percentages.

Effect of adding context Our third user study is the Plausibility Test with Context (PTX), where we aim to evaluate the effect of adding scene contexts to the generated scenes. In this study, we set the context control parameter α to 0.5. For each query, two scenes were presented to the subjects, one generated with context added (Our-context) and the other is an Our-user result from the previous studies, without adding context. Feedback was received from 25 subjects, who selected for each editing scenario (from the previous two studies), which of the two scenes was preferred based on plausibility and naturalness, as before. As shown in Figure 5.15(b) (second column), Our-context scenes were slightly favoured with 54% of the total votes over all 10 editing scenarios. This validates our belief, but only weakly, that users tend to prefer more complex scenes (as a result of adding contexts), but only when the added objects do not violate scene semantics as a result.

5.9 Discussion, limitation, and future work

In this chapter, we present a tool which uses annotated 3D scene databases to support synthesis and editing of 3D indoor scenes using natural language. In designing such a tool, we contrast *selection* versus *affection* of scene objects, *object-level* versus *patch-level* scene manipulation, and emphasize that in each case, it is the latter option that accentuates the usefulness of language-driven scene modeling. With direct manipulation over a 3D scene, e.g., through the use of a mouse, attribute changes typically require more interactions than object selection, while affection on a group of objects is even more laborious. Language

commands, if interpreted and realized properly, can go a long way in saving user effort in these situations.

Our tool has been developed with the above thinking in mind and it separates itself from previous attempts at text-driven scene synthesis in several aspects. First, our tool supports scene editing at the sub-scene level which both accelerates scene evolution and improves the alignment and unification of natural language commands with 3D scenes. Second, we learn a relational model which enables change of relations between two or a group of objects during scene synthesis. Finally, the semantic scene graphs used in our text-driven scene retrieval and synthesis not only provides a grounding between text and 3D scenes, but also incorporates information about object arrangements and occurrence from 3D scene databases.

We regard our tool as a preliminary prototype for language-driven and data-driven 3D scene modeling. The current implementation only supports a limited set of group relations and a limited classes of commands for language-scene grounding. Enriching both would require expanded data annotations and language processing capabilities. Also, aside from learning and retrieving everything from available 3D scene databases, we could also extend knowledge acquisition to other sources such as ImageNet or KnowledgeNet.

We would also like to explore applying the techniques developed in our work in other contexts. For example, our scene alignment algorithm may hold the potential to enrich a set of synthetic scenes by aligning them with real scenes as a way to produce variations. Ultimately, an intelligent language-to-scene modeling tool should be able to learn and adapt *on-the-fly*. Examples of such intelligence include automatic requests for new or additional scene exemplars, annotations for unknown attributes, or ungrounded textual commands. With the additional data, the semantic scene graphs and underlying learning mechanism can be adjusted and enhanced on the fly.

Using natural language to assist creative tasks is a growing field. Many techniques we have explored in the context of editing scenes with language have applications to other creative media such as editing images and video. These systems must all answer challenging questions. How should voice or text input interleave with other input modalities? How do we deal with uncertainty in understanding the user's intent or the system's inability to fully execute the request? Our work has explored some of these issues in the context of 3D scenes and believe that language-powered interactive systems will soon be able to significantly lower the entry barrier for creative tools.

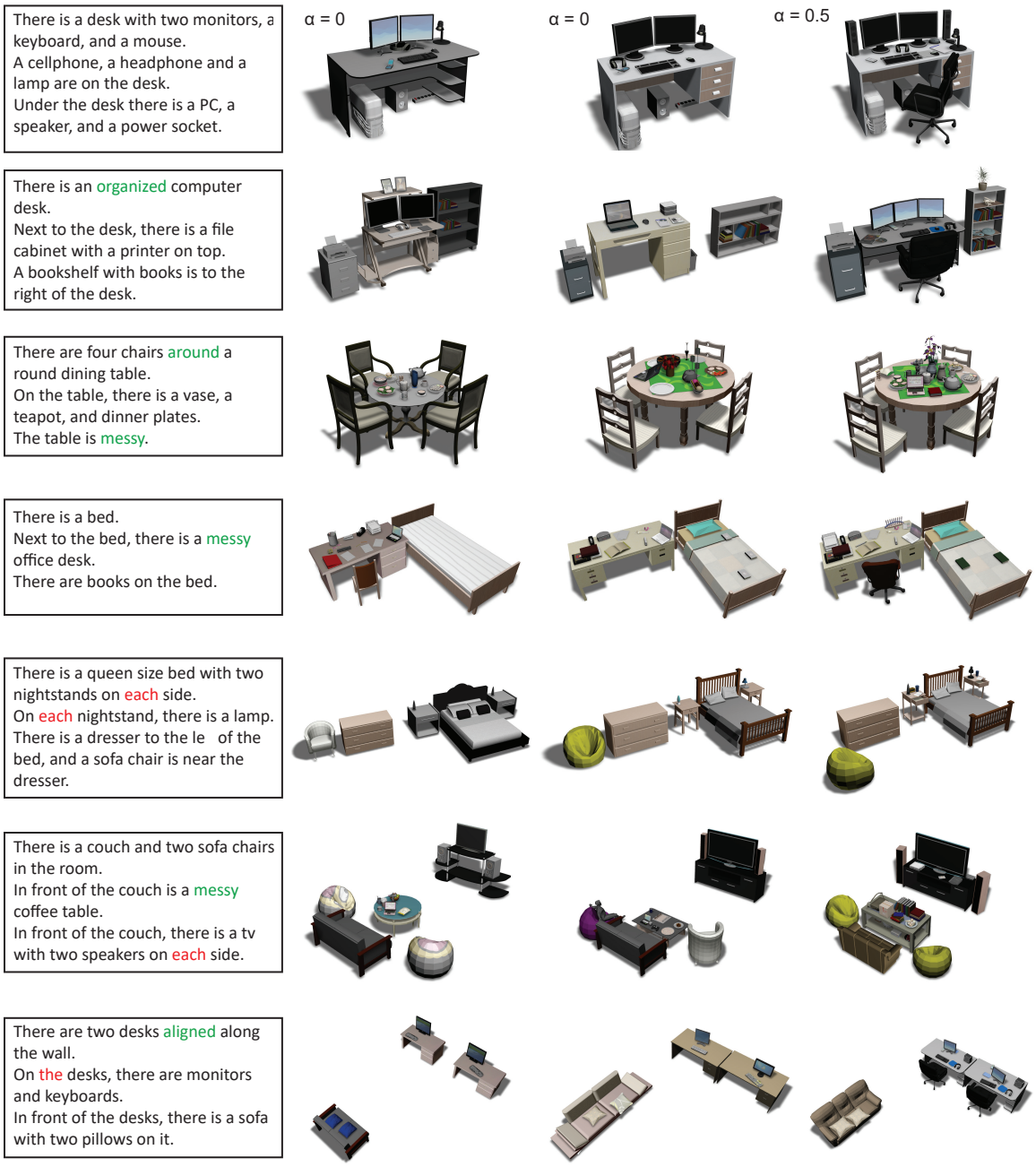


Figure 5.16: A gallery of our language-driven scene synthesis results. In each row, we show from left to right: input sentences; an output scene selected from 5 results ($\alpha = 0$); another output scene selected from 5 results ($\alpha = 0$); an output scene with context added ($\alpha = 0.5$). In the sentences, highlighted words map to the group relations (green) and some key determiners (red) that affect the scene binding.

Chapter 6

Conclusion and Future Work

In this chapter, we conclude this thesis with a summary of the main contributions of our sub-scene level processing for scene analysis and synthesis. Then, we discuss possible directions for future work.

6.1 Summary of contributions

In the thesis, we first proposed to use focal points, the representative sub-scenes, for characterizing, comparing and organizing collections of complex indoor scenes. Then we presented action-driven scene evolution to generate continuously evolving scenes by simulating how scenes are altered by human actions. Lastly, we introduced the language-driven scene synthesis for using natural languages to generate and edit 3D scenes. Both of the two proposed scene synthesis systems learned probabilistic models of sub-scenes, enabling efficient generation of complex 3D indoor scenes.

Mid-level scene analysis. Similar to the mid-level image understanding [89, 19, 44], the focal-centric scene analysis presented in Chapter 3 could be regarded as mid-level 3D scene analysis. In contrast to working on the discriminative image regions, we focus on the substructures of indoor scenes and learn contextually representative focal points by co-analyzing a scene collection. The focal points provide a novel perspective to understand and characterize complex and heterogeneous scenes, achieving better results for scene retrieval comparing to the method that work on the global scene level [22]. New applications such as scene organization and exploration are also enabled by the focal-based scene similarity.

To extract the representative substructures as focal points, we abstract 3D scenes as structure graphs and perform an interleaving optimization between frequent graph pattern mining and scene clustering. The context from scene clustering result is utilized to refine the focal mining while the refined focal points in turn lead to more compact scene clusters. The framework we proposed for extracting and applying the focal points are not limited to

be used for 3D scenes, but can be adapted to analysis of heterogeneous data in other forms, such as images and texts.

Probabilistic models of sub-scenes. The action model proposed in Chapter 4 and the relational model proposed in Chapter 5 are both probabilistic models of sub-scenes. As the probabilistic models are defined as joint distributions of objects, with the increasing number of objects, the models will become over-complicated and need large amount of data to train. Comparing to learning a holistic object distribution of a whole scene, learning the object distributions in sub-scenes requires less training data. The training process is also much efficient and the learned models are more reliable.

Comparing to the activity model in [21] and the PiGraph [78] which also learn probabilistic models for sub-scenes, our action model is learned from a much richer data source, i.e., photographs. To account for the challenge of 3D recovery from 2D, we embed human-object relationships in a human-centric coordinate frame, recovering possible 3D information from photographs with the help of reconstructed 3D human poses. We also construct an action graph by analyzing the correlation between different action models, so that possible sequences of plausible actions could be sampled from the graph to drive scene evolution.

Our relational model encodes semantic relationships of two or more objects, inducing a mapping from natural language to 3D scene arrangements. We are the first to learn object distributions related to high-level relationships of object groups such as *messy, organized*. Applying the relational models for text-to-scene generation reduces the redundancy of natural language, enabling generation of complex scenes with only a few compact scene editing commands.

Efficient progressive scene synthesis systems. With the action models and relational models, the two proposed scene synthesis systems efficiently generate or edit placements of multiple objects at the sub-scene level, both enriching the complexity of scenes in a progressive manner. In comparison to scene synthesis systems [22, 12] which sample the placement of every object in the whole scene from scratch, progressive scene synthesis systems that operate at the sub-scene level could produce large-scale scenes with much more complexity.

The action-driven scene evolution is able to automatically generate scenes with numerous variations due to the different positions where actions are applied and the different order of the actions. All of the resulting scenes are semantically and functionally meaningful, as they are inherently corresponding to the applied human actions. In the language-driven scene synthesis, we "retrieve-and-accommodate" sub-scenes based on the novel Semantic Scene Graph representation which enables the mapping between natural language and 3D scenes. Scene semantics from the 3D scene databases are efficiently utilized to generate complex scenes from language inputs.

6.2 Future work

This thesis presents frameworks of sub-scene level processing for analysis and synthesis of 3D indoor scenes. In the following, we discuss several possible directions of future work.

Analysis of large-scale online scene repositories. Our focal-centric scene analysis are evaluated with existing scene databases which consist of hundreds of scenes [22, 107]. The objects in these scenes are well segmented and annotated with semantic labels. It would be interesting to extend the framework in Chapter 3 to the large-scale online scene repositories, e.g., 3D Warehouse [1].

The first problem of dealing with the online scenes is the preprocessing, as they often contain over-segmented object parts whose labels may be missing or inconsistent. Liu et al. [55] make a first attempt to convert the initial scenes into consistent and semantic scene graphs, but their approach relies on the supervised learning of scene grammars from the limited set of annotated scenes. A more robust, precise and unsupervised scene preprocessing is needed to handle scenes with higher level of details and complexity. The second problem is to scale up our sub-scene analysis to the large number of online scenes. The interleaving optimization between the frequent subgraph mining and scene clustering may become slow when the scene number and the heterogeneity of the scene collection increase. A new coarse-to-fine and parallel processing framework will certainly improve the performance of focal extraction and scene clustering. The last problem is to design more easy-to-use interfaces for focal-centric scene retrieval and scene database exploration, so that the users will be able to browse the online scene repositories more efficiently and reduce the time of searching for their interested scenes.

Learning from more multifaceted training data. The key for data-driven approaches is the source of data, especially those with rich and semantic annotations. Learning from more multifaceted training data could improve our models of sub-scenes and lead to more realistic and complex scene synthesis results.

In Chapter 4, we learn action models of human-object relations from annotated photographs. The action types are limited to static actions or activities which correspond to stable object placements relative to the human poses. Our correlation model for computing the action transition probabilities is only an assumption and a rough simulation to generate possible action sequences. To fully model the dynamic nature of actions and the transitions between different actions, action models should be learned from dynamic human action data, e.g., long-term RGB-D videos of daily human actions. Moreover, the potential actions and their transitions may also be mined from the vast source of texts.

We utilize existing scene databases [22, 32] to build the relational models and suggest the candidate sub-scenes to create new scenes in Chapter 5. The SceneNN [32] database

provides the object arrangements from the real world and complements the SceneSynth database [22] with furniture-level object distributions. As more easy-to-use scene capture and annotation tools are created, large-scale richly-annotated scene database such as ScanNet [18], which contains thousands of scenes captured from real-world environments, is now available. Applying the learning framework in Chapter 5 to ScanNet and other similar scene databases will certainly construct more robust and more sophisticated relational models of sub-scenes which will in turn help generate more complex scenes.

Extending sub-level processing to other data forms. Our sub-scene level processing frameworks not only could inspire future work in 3D scene processing, but also hold the potential applications in 3D shape processing and other related fields such as image understanding and text processing.

Recall that there is an analogy between a 3D shape with the parts and a 3D scene with the objects. It is a straightforward idea to extend the sub-scene level processing frameworks presented in this thesis to the substructure level shape processing. The focal-centric analysis framework could be easily adapted to extract the representative substructures from a collection of shapes and then organize the shapes based on the focal points. For a 3D object, an analogy to the human actions we used for scene evolution is the human interaction. Discovering the substructures related to semantic human-object interactions and further exploiting the substructures by learned interactions will improve and motivate more functional-aware shape synthesis.

Our idea of using focal points to compare and organize complex data could also be applied to other data forms, such as images and texts. The key of extending our focal-centric analysis framework to such data is to build the graph representation of the underlying data, where the nodes represent the interested entities and the edges encode the spatial, structural or semantic relationships between the nodes. Furthermore, it is possible to extend our language-driven synthesis framework presented in Chapter 5 to creation and editing of other digital or visual content, including 3D shapes, images or videos. Data-specific adaptation will be needed to apply our techniques to build the text-to-content mapping and execute the "retrieval-and-accommodate" scheme to manipulate the related content.

Finally, this thesis proposed novel frameworks for sub-scene level processing of 3D indoor scenes. Results have shown that the performance of both scene analysis and scene synthesis is improved by operating at the sub-scene level. We believe that future developments on sub-scene level data-driven scene processing will bring us closer to creation of truly realistic and complex 3D scenes that are readily usable for applications such as 3D games and VR.

Bibliography

[1] 3D warehouse. `https://3dwarehouse.sketchup.com/`. [Online; accessed June-2017].

[2] Planner 5D. `https://planner5d.com`. [Online; accessed June-2017].

[3] Sweet Home 3D. `http://sweethome3d.com`. [Online; accessed June-2017].

[4] Hirotogu Akaike. Information theory and an extension of the maximum likelihood principle. In *Second International Symposium on Information Theory*, volume 1, pages 267–281, 1973.

[5] Autodesk. Homestyler. `https://www.homestyler.com`. [Online; accessed June-2017].

[6] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Analysis & Machine Intelligence*, 24(4):509–522, 2002.

[7] Yoram Biberman. A context similarity measure. *Machine Learning*, 784:49–63, 1994.

[8] Angel Chang, Will Monroe, Manolis Savva, Christopher Potts, and Christopher D. Manning. Text to 3D Scene Generation with Rich Lexical Grounding. In *Association for Computational Linguistics and International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, 2015.

[9] Angel X. Chang, Mihail Eric, Manolis Savva, and Christopher D. Manning. SceneSeer: 3D Scene Design with Natural Language. *CoRR*, abs/1703.00050, 2017.

[10] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An information-rich 3D model repository. 2015.

[11] Angel X. Chang, Manolis Savva, and Christopher D. Manning. Interactive learning of spatial knowledge for text to 3D scene generation. In *Proc. ACL Workshop on Interactive Language Learning, Visualization, and Interfaces (ILLVI)*, 2014.

[12] Angel X. Chang, Manolis Savva, and Christopher D. Manning. Learning spatial knowledge for text to 3D scene generation. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

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

[14] 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 Trans. on Graph.*, 33(6):208:1–12, 2014.

[15] Ming-Ming Cheng, Niloy J. Mitra, Xiaolei Huang, and Shi-Min Hu. SalientShape: group saliency in image collections. *The Visual Computer*, 30(4):443–453, 2014.

[16] Bob Coyne, Alex Klapheke, Masoud Rouhizadeh, Richard Sproat, and Daniel Bauer. Annotation tools and knowledge representation for a Text-To-Scene system. In *COLING*, pages 679–694, 2012.

[17] Bob Coyne and Richard Sproat. WordsEye: An automatic text-to-scene conversion system. In *Proc. of SIGGRAPH*, pages 487–496, 2001.

[18] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. IEEE CVPR*, 2017.

[19] Carl Doersch, Saurabh Singh, Abhinav Gupta, Josef Sivic, and Alexei A. Efros. What makes paris look like Paris? *ACM Trans. on Graph.*, 31(4):101:1–9, 2012.

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

[21] Matthew Fisher, Yangyan Li, Manolis Savva, Pat Hanrahan, and Matthias Nießner. Activity-centric scene synthesis for functional 3D scene modeling. *ACM Trans. on Graph.*, 34(6):212:1–10, 2015.

[22] Matthew Fisher, Daniel Ritchie, Manolis Savva, Thomas Funkhouser, and Pat Hanrahan. Example-based synthesis of 3D object arrangements. *ACM Trans. on Graph.*, 31(6):135:1–11, 2012.

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

[24] N. I. Fisher. *Statistical Analysis of Circular Data.* Cambridge University Press, Cambridge, 1993.

[25] David F. Fouhey, Vincent Delaitre, Abhinav Gupta, Alexei A. Efros, Ivan Laptev, and Josef Sivic. People watching: Human actions as a cue for single-view geometry. In *Proc. ECCV*, pages 732–745, 2012.

[26] Thomas Funkhouser, Michael Kazhdan, Philip Shilane, Patrick Min, William Kiefer, Ayellet Tal, Szymon Rusinkiewicz, and David Dobkin. Modeling by example. *ACM Trans. on Graph.*, 2004.

[27] T. Germer and M. Schwarz. Procedural arrangement of furniture for real-time walkthroughs. *Computer Graphics Forum*, 28(8):2068–2078, 2009.

[28] S. Guadarrama, L. Riano, D. Golland, D. Göhring, Y. Jia, D. Klein, P. Abbeel, and T. Darrell. Grounding Spatial Relations for Human-Robot Interaction. In *Proc. IEEE Int. Conf. on Intelligent Robots & Systems*, pages 1640–1647, 2013.

[29] Jiawei Han, Hong Cheng, Dong Xin, and Xifeng Yan. Frequent pattern mining: current status and future directions. *Data Mining and Knowledge Discovery*, 15(1):55–Ű86, 2007.

[30] Z. Harchaoui and F. Bach. Image classification with segmentation graph kernels. In *Proc. IEEE CVPR*, 2007.

[31] Ruizhen Hu, Oliver van Kaick, Bojian Wu, Hui Huang, Ariel Shamir, and Hao Zhang. Learning how objects function via co-analysis of interactions. *ACM Trans. on Graph.*, 35(4):47:1–12, 2016.

[32] Binh-Son Hua, Quang-Hieu Pham, Duc Thanh Nguyen, Minh-Khoi Tran, Lap-Fai Yu, and Sai-Kit Yeung. SceneNN: A Scene Meshes Dataset with aNNotations. In *Proc. of 3D Vision*, 2016.

[33] Qixing Huang, Hao Su, and Leonidas Guibas. Fine-grained semi-supervised labeling of large shape collections. *ACM Trans. on Graphics (Proc. of SIGGRAPH Asia)*, 32(6):190:1–10, 2013.

[34] Qixing Huang, Guoxin Zhang, Lin Gao, Shimin Hu, Adrian Bustcher, and Leonidas Guibas. An optimization approach for extracting and encoding consistent maps in a shape collection. *ACM Trans. on Graphics (Proc. of SIGGRAPH Asia)*, 31(6):167:1–11, 2012.

[35] Shi-Sheng Huang, Hongbo Fu, and Shi-Min Hu. Structure guided interior scene synthesis via graph matching. *Graphical Models*, 85(C):46–55, 2016.

[36] Shi-Sheng Huang, Hongbo Fu, and Shi-Min Hu. Structure guided interior scene synthesis via graph matching. *Graphical Models*, 85:46–55, 2016.

[37] Shi-Sheng Huang, Hongbo Fu, Ling-Yu Wei, and Shi-Min Hu. Support substructures: Support-induced part-level structural representation. *IEEE Trans. Visualization & Computer Graphics*, 22(8):2024–2036, 2016.

[38] Shi-Sheng Huang, Ariel Shamir, Chao-Hui Shen, Hao Zhang, Alla Sheffer, Shi-Min Hu, and Daniel Cohen-Or. Qualitative organization of collections of shapes via quartet analysis. *ACM Trans. on Graph.*, 32(4):71:1–10, 2013.

[39] Hamid Izadinia, Qi Shan, and Steven M Seitz. IM2CAD. In *Proc. IEEE CVPR*, 2017.

[40] Arjun Jain, Thorsten Thormählen, Tobias Ritschel, and Hans-Peter Seidel. Exploring shape variations by 3D-model decomposition and part-based recombination. *Computer Graphics Forum*, 31(2):631–640, 2012.

[41] G. Jeh and J. Widom. SimRank: a measure of structural-context similarity. In *Proc. of ACM SIGKDD*, pages 538–543, 2002.

[42] Yun Jiang, Hema Koppula, and Ashutosh Saxena. Hallucinated humans as the hidden context for labeling 3d scenes. In *Proc. IEEE CVPR*, pages 2993–3000, 2013.

[43] Yun Jiang, Marcus Lim, and Ashutosh Saxena. Learning object arrangements in 3d scenes using human context. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2012.

[44] M. Juneja, A. Vedaldi, C. V. Jawahar, and A. Zisserman. Blocks that shout: Distinctive parts for scene classification. In *Proc. IEEE CVPR*, pages 923–930, 2013.

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

[46] Andrej Karpathy, Stephen Miller, and Li Fei-Fei. Object discovery in 3d scenes via shape analysis. In *Proc. IEEE Int. Conf. on Robotics & Automation*, 2013.

[47] Young Min Kim, Niloy J. Mitra, Dong-Ming Yan, and Leonidas Guibas. Acquiring 3D indoor environments with variability and repetition. *ACM Trans. on Graph.*, 31(6):138:1–138:11, 2012.

[48] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. 2012.

[49] Hamid Laga, Michela Mortara, and Michela Spagnuolo. Geometry and context for semantic correspondence and functionality recognition in manmade 3d shapes. *ACM Trans. on Graph.*, 32(5), 2013.

[50] David C Lee, Martial Hebert, and Takeo Kanade. Geometric reasoning for single image structure recovery. In *Proc. IEEE CVPR*, 2009.

[51] Jeehyung Lee and Thomas Funkhouser. Sketch-based search and composition of 3d models. In *Proc. of the Fifth Eurographics Conference on Sketch-Based Interfaces and Modeling*, pages 97–104, 2008.

[52] Yangyan Li, Angela Dai, Leonidas Guibas, and Matthias Nießner. Database-assisted object retrieval for real-time 3d reconstruction. *Computer Graphics Forum*, 34(2), 2015.

[53] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *Proc. ECCV*, pages 740–755, 2014.

[54] Han Liu, Ulysse Vimont, Michael Wand, Marie-Paule Cani, Stefanie Hahmann, Damien Rohmer, and Niloy J. Mitra. Replaceable substructures for efficient part-based modeling. *Computer Graphics Forum*, 34(2):503–513, May 2015.

[55] 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. on Graph.*, 33(6), 2014.

[56] Zicheng Liu, Yan Zhang, Wentao Wu, Kai Liu, and Zhengxing Sun. Model-driven indoor scenes modeling from a single image. In *Proc. of Graphics Interface*, pages 25–32, 2015.

[57] Rui Ma. Analysis and modeling of 3d indoor scenes. *ArXiv e-prints*, 2017.

[58] Rui Ma, Honghua Li, Changqing Zou, Zicheng Liao, Xin Tong, and Hao Zhang. Action-driven 3D indoor scene evolution. *ACM Trans. on Graph.*, 35(6), 2016.

[59] Lucas Majerowicz, Ariel Shamir, Alla Sheffer, and Holger H. Hoos. Filling your shelves: Synthesizing diverse style-preserving artifact arrangements. *IEEE Trans. Visualization & Computer Graphics*, 20(11):1507–1518, 2014.

[60] Lucas Majerowicz, Ariel Shamir, Alla Sheffer, and Holger H. Hoos. Filling your shelves: Synthesizing diverse style-preserving artifact arrangements. *IEEE Trans. Visualization & Computer Graphics*, 20(11):1507–1518, 2014.

[61] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60, 2014.

[62] Oliver Mattausch, Daniele Panozzo, Claudio Mura, Olga Sorkine-Hornung, and Renato Pajarola. Object detection and classification from large-scale cluttered indoor scans. *Computer Graphics Forum (Proc. of Eurographics)*, 33(2), 2014.

[63] Paul Merrell, Eric Schkufza, Zeyang Li, Maneesh Agrawala, and Vladlen Koltun. Interactive furniture layout using interior design guidelines. *ACM Trans. on Graph.*, 30(4):87:1–10, 2011.

[64] Dipendra Kumar Misra, Jaeyong Sung, Kevin Lee, and Ashutosh Saxena. Tell me dave: Context-sensitive grounding of natural language to manipulation instructions. In *Proc. of Robotics: Science and Systems*, 2014.

[65] Niloy Mitra, Michael Wand, Hao Zhang, Daniel Cohen-Or, and Martin Bokeloh. Structure-aware shape processing. In *Eurographics State-of-the-art Report (STAR)*, 2013.

[66] Liangliang Nan, Ke Xie, and Andrei Sharf. A search-classify approach for cluttered indoor scene understanding. *ACM Trans. on Graph.*, 31(6):137:1–137:10, 2012.

[67] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger. Real-time 3D reconstruction at scale using voxel hashing. *ACM Trans. on Graph.*, 32(6):169:1–11, 2013.

[68] Maks Ovsjanikov, Wilmot Li, Leonidas Guibas, and Niloy J. Mitra. Exploration of continuous variability in collections of 3D shapes. *ACM Trans. on Graph.*, 30(4):33:1–10, 2011.

[69] Ariadna Quattoni and Antonio Torralba. Recognizing indoor scenes. In *Proc. IEEE CVPR*, pages 413–420, 2009.

[70] N. Rasiwasia and N. Vasconcelos. Scene classification with low-dimensional semantic spaces and weak supervision. In *Proc. IEEE CVPR*, pages 1–6, 2008.

[71] Kaspar Riesen, Xiaoyi Jiang, and Horst Bunke. Exact and inexact graph matching: Methodology and applications. *Managing and Mining Graph Data*, 40:217–247, 2010.

[72] Matteo Ruggero Ronchi and Pietro Perona. Describing common human visual actions in images. In *Proc. of the British Machine Vision Conference (BMVC)*, pages 52:1–12, 2015.

[73] Eleanor Rosch. Cognitive reference points. *Cognitive Psychology*, 7(4):532–Ű547, 1975.

[74] Zeinab Sadeghipour, Zicheng Liao, Ping Tan, and Hao Zhang. Learning 3D scene synthesis from annotated RGB-D images. *Computer Graphics Forum (Proc. Eurographics Symp. on Geometry Processing)*, 35(5), 2016.

[75] Manolis Savva, Angel X. Chang, and Maneesh Agrawala. Scenesuggest: Context-driven 3d scene design. *CoRR*, abs/1703.00061, 2017.

[76] Manolis Savva, Angel X. Chang, and Pat Hanrahan. Semantically-Enriched 3D Models for Common-sense Knowledge. *CVPR 2015 Workshop on Functionality, Physics, Intentionality and Causality*, 2015.

[77] Manolis Savva, Angel X. Chang, Pat Hanrahan, Matthew Fisher, and Matthias Nießner. SceneGrok: Inferring action maps in 3D environments. *ACM Trans. on Graph.*, 33(6):212:1–10, 2014.

[78] Manolis Savva, Angel X. Chang, Pat Hanrahan, Matthew Fisher, and Matthias Nießner. PiGraphs: Learning interaction snapshots from observations. *ACM Trans. on Graph.*, 35(4), 2016.

[79] Steven M. Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Proc. IEEE CVPR*, pages 519–528, 2006.

[80] Lee M. Seversky and Lijun Yin. Real-time automatic 3D scene generation from natural language voice and text descriptions. In *Proc. of ACM International Conference on Multimedia*, pages 61–64, 2006.

[81] Tianjia Shao, Aron Monszpart, Youyi Zheng, Bongjin Koo, Weiwei Xu, Kun Zhou, and Niloy J. Mitra. Imagining the unseen: Stability-based cuboid arrangements for scene understanding. *ACM Trans. on Graph.*, 33(6), 2014.

[82] Tianjia Shao, Weiwei Xu, Kun Zhou, Jingdong Wang, Dongping Li, and Baining Guo. An interactive approach to semantic modeling of indoor scenes with an RGBD camera. *ACM Trans. on Graph.*, 31(6):136:1–11, 2012.

[83] Lior Shapira, Shy Shalom, Ariel Shamir, Daniel Cohen-Or, and Hao Zhang. Contextual part analogies in 3D objects. *Int. J. Computer Vision*, 89(2-3):309–326, 2009.

[84] Andrei Sharf, Hui Huang, Cheng Liang, Jiapei Zhang, Baoquan Chen, and Minglun Gong. Mobility-trees for indoor scenes manipulation. *Computer Graphics Forum*, 33(1), 2014.

[85] Yifei Shi, Pinxin Long, Kai Xu, Hui Huang, and Yueshan Xiong. Data-driven contextual modeling for 3d scene understanding. *Computers and Graphics*, 55:55–67, 2016.

[86] Philip Shilane and Thomas Funkhouser. Distinctive regions of 3D surfaces. *ACM Trans. on Graph.*, 26(2):7:1–15, 2007.

[87] HyoJong Shin and Takeo Igarashi. Magic canvas: Interactive design of a 3-d scene prototype from freehand sketches. In *Proc. of Graphics Interface 2007*, pages 63–70, 2007.

[88] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *Proc. ECCV*, 2012.

[89] Saurabh Singh, Abhinav Gupta, and AlexeiA Efros. Unsupervised discovery of mid-level discriminative patches. In *Proc. ECCV*, pages 73–86, 2012.

[90] Greg Slabaugh, Bruce Culbertson, Tom Malzbender, and Ron Schafer. A survey of methods for volumetric scene reconstruction from photographs. In *Proc. of Eurographics Conference on Volume Graphics*, pages 81–101, 2001.

[91] Shuran Song, Samuel P. Lichtenberg, and Jianxiong Xiao. SUN RGB-D: A RGB-D scene understanding benchmark suite. In *Proc. IEEE CVPR*, pages 567–576, 2015.

[92] Moritz Tenorth and Michael Beetz. KnowRob: A knowledge processing infrastructure for cognition-enabled robots. *Int. J. Rob. Res.*, 32(5):566–590, 2013.

[93] Koji Tsuda and Taku Kudo. Clustering graphs by weighted substructure mining. In *Proc. Intl Conf on Machine Learning (ICML)*, pages 953Ű–960, 2006.

[94] Amos Tversky. Features of similarity. *Psychological Review*, 84(4):327–352, 1977.

[95] 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. on Graph.*, 32(4):69:1–10, 2013.

[96] René Vidal. Subspace clustering. *IEEE Signal Processing Magazine*, 28(3):52Ű–68, 2011.

[97] Shusen Wang, Xiaotong Yuan, Tiansheng Yao, Shuicheng Yan, and Jialie Shen. Efficient subspace segmentation via quadratic programming. In *AAAI*, pages 519–524, 2011.

[98] Yanzhen Wang, Kai Xu, Jun Li, Hao Zhang, Ariel Shamir, Ligang Liu, Zhiquan Cheng, and Yueshan Xiong. Symmetry hierarchy of man-made objects. *Computer Graphics Forum (Proc. of Eurographics)*, 30(2):287–296, 2011.

[99] Ludwig Wittgenstein. *Philosophical investigations*. New York: Macmillan, 1953.

[100] Jianxiong Xiao. 3D Reconstruction is Not Just a Low-level Task: Retrospect and Survey. Technical report, MIT9.S912: What is Intelligence?, 2012.

[101] Hualiang Xie, Wenzhuo Xu, and Bin Wang. Reshuffle-based interior scene synthesis. In *Proc. VRCAI '13*, pages 191–198, 2013.

[102] Kai Xu, Hui Huang, Yifei Shi, Hao Li, Pinxin Long, Jianong Caichen, Wei Sun, and Baoquan Chen. Autoscanning for coupled scene reconstruction and proactive object analysis. *ACM Trans. on Graph.*, 34(6), 2015.

[103] 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 Trans. on Graph.*, 33(4):35:1–12, 2014.

[104] Kai Xu, Yifei Shi, Lintao Zheng, Junyu Zhang, Min Liu, Hui Huang, Hao Su, Daniel Cohen-Or, and Baoquan Chen. 3D Attention-Driven Depth Acquisition for Object Identification. *ACM Trans. on Graph.*, 35(6), 2016.

[105] Kai Xu, Hao Zhang, Daniel Cohen-Or, and Baoquan Chen. Fit and diverse: Set evolution for inspiring 3D shape galleries. *ACM Trans. on Graph.*, 31(4):57:1–10, 2012.

[106] Ken Xu, James Stewart, and Eugene Fiume. Constraint-based automatic placement for scene composition. In *Graphics Interface*, pages 25–34, 2002.

[107] 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. on Graph.*, 32(4):123:1–10, 2013.

[108] X. Yan and J. Han. gSpan: graph-based substructure pattern mining. In *Proc. Int. Conf. on Data Mining*, pages 721Ű–724, 2002.

[109] 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 Trans. on Graph.*, 31(4):56:1–11, 2012.

[110] 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. on Graph.*, 30(4):86:1–12, 2011.

[111] Lap-Fai Yu, Sai Kit Yeung, and Demetri Terzopoulos. The clutterpalette: An interactive tool for detailing indoor scenes. *IEEE Trans. Visualization & Computer Graphics*, 22(2):1138–1148, 2016.

[112] L. Zelnik-Manor and P. Perona. Self-tuning spectral clustering. In *Advances in Neural Information Processing Systems (NIPS)*, volume 17, pages 1601–1608, 2004.

[113] Yizhong Zhang, Weiwei Xu, Yiying Tong, and Kun Zhou. Online structure analysis for real-time indoor scene reconstruction. *ACM Trans. on Graph.*, 34(5), 2015.

[114] Xi Zhao, Ruizhen Hu, Paul Guerrero, Niloy J. Mitra, and Taku Komura. Relationship templates for creating scene variations. *ACM Trans. on Graph.*, 35(6), 2016.

[115] Xi Zhao, He Wang, and Taku Komura. Indexing 3d scenes using the interaction bisector surface. *ACM Trans. on Graph.*, 33(3), 2014.

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

[117] X. Zhou, S. Leonardos, Xiaoyan Hu, and K. Daniilidis. 3D shape estimation from 2D landmarks: A convex relaxation approach. In *Proc. IEEE CVPR*, pages 4447–4455, 2015.

[118] C. Lawrence Zitnick, Devi Parikh, and Lucy Vanderwende. Learning the visual interpretation of sentences. In *Proc. ICCV*, pages 1681–1688, 2013.