TEXT TO 3D SCENE GENERATION

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Angel Xuan Chang
December 2015

This dissertation is online at: http://purl.stanford.edu/vg064sy5087

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Christopher Manning, Primary Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Pat Hanrahan

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Percy Liang

Approved for the Stanford University Committee on Graduate Studies.

Patricia J. Gumport, Vice Provost for Graduate Education

This signature page was generated electronically upon submission of this dissertation in electronic format. An original signed hard copy of the signature page is on file in University Archives.

Abstract

The ability to form a visual interpretation of the world from natural language is pivotal to human communication. Similarly, from a computational perspective, mapping descriptions of scenes to 3D geometric representations is useful in many areas such as robotics, interior design and even education.

Text to 3D scene generation is a task which addresses this problem space. A user provides natural language as input and the output is a plausible 3D scene interpretation. This is a challenging domain connecting NLP and computer graphics. The few existing systems for generating 3D scenes from text are severely restricted in scope and robustness. The key challenge, and focus of this dissertation, is in incorporating prior knowledge which is essential for successfully generating 3D scenes from highly under-specified natural scene descriptions. Prior systems do not leverage such priors, requiring explicit and verbose language.

This dissertation formalizes and decomposes the problem of text to 3D scene generation, and describes the implementation of a new text to scene framework that enables incorporation of priors learned from data. I propose viewing the problem as extracting a set of explicit constraints from input descriptions, combining them with learned common-sense priors for inferring implicit constraints, and then selecting objects and positioning them to satisfy the constraints and generate plausible scenes. To capture the basic semantics of a scene, I define the scene template representation which consists of the objects, their attributes, and relations between them. A given scene template, can be used to generate many matching scenes whose plausibility can be scored. I then define two subtasks: scene template parsing where templates are parsed from natural language, and scene inference

where templates are expanded with additional objects and spatial constraints. From the expanded scene templates, my system grounds object references by selecting appropriate 3D models, and then computationally arranges the selected objects to satisfy spatial constraints and maximize plausibility. I then demonstrate how to extend the text to scene system to allow iterative refinement of the generated scenes using natural language commands to add, remove, replace, and manipulate objects.

In building the text to scene framework presented here, I learn a set of common-sense priors using datasets of 3D models and scenes and evaluate their impact on the quality of generated 3D scenes. From the scene data, I collect several sets of priors: (1) object occurrence priors to determine what other objects should be present, (2) support and relative position priors to determine where objects are placed, and (3) attachment priors to determine how objects are attached. In addition, I collect a new dataset of 3D scenes corresponded with textual descriptions and use it to learn how to ground spatial relation language and object descriptions. I provide this dataset to the community and perform an empirical evaluation of the output of the system against manually designed scenes and simpler rule-based baselines. Using a perceptual evaluation study, I show that the system can generate high quality 3D scenes given natural language input. This initial step in connecting language with 3D geometry opens up many areas of research for bridging the gap between language, semantics and geometry.

To my parents, Qiu and Tianrong Zhang who instilled in me the love of learning.

Acknowledgements

I wouldn't be where I am today without the support of so many wonderful people. My advisor Chris patiently guided me and helped me grow through my years as a PhD student. He was kind, understanding, and encouraging when I needed it the most, and I'm happy that I can now talk with him about anything. My other reading committee members Pat and Percy were also invaluable to my growth as a student, and I benefited a lot from their wisdom whenever I talked with them.

If it weren't for Val, I wouldn't have decided to come back to school to do a PhD. I am also indebted to Don Knuth who has motivated and helped me to come to Stanford. He has always greeted me with a smile and remembers my name even though we only run into each other maybe once a year now.

I got the chance to work together with great collaborators and learn many different things from them: Heeyoung, Marta, Nate, Dan J, Chris P, Will, Sebastian, Ranjay, Gilbert, Matt, Matthias and others. I was also fortunate to work with many bright young students including Mihail, Arthur and Nikita. Their energy was very inspirational especially when I was starting to feel like I had been in school for too long.

In my time as a PhD student I was fortunate to have many friends to support me. All the NLP folks: Sonal, Gabor, Will, Richard, Danqi, and many others made it so much more fun to be a member of the group. Everybody together made the group a great, happy environment through fun events such as our regular *tea time*. Outside the NLP group, the graphics folks: Katherine, Matthias, Matt, Angie, Daniel, and others were always very welcoming and were my second home away from the NLP group. Many other friends in the department have all been fabulous people who made me feel truly a part of a great community.

Finally, my family was always there for me and helped me to get through the harder times. And of course, I am extremely grateful for Manolis, who has been by my side. You introduced me to WordsEye, the idea of text to scene, supported me through every step of the way, and helped me figure out my path.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

To understand language, we need an understanding of the world around us. Language describes the world and provides symbols with which we represent meaning. Still, much knowledge about the world is so obvious that it is rarely explicitly stated. It is uncommon for people to state that chairs are usually on the floor and upright, and that you usually eat a cake from a plate on a table. Knowledge of such common facts provides the context within which people communicate with language. Therefore, to create practical systems that can interact with the world and communicate with people, we need to leverage such knowledge to interpret language in context.

In this dissertation, I will focus on the task of text to 3D scene generation as a means to explore what type of knowledge is needed to understand common, everyday language describing the world around us. The task of generating 3D scene representations from natural language (i.e., text to 3D scene generation) pulls together many challenging strands of research. It needs to address several key questions in natural language understanding: what objects, attributes, and relationships are implied by text? To answer these questions we need to go beyond understanding language at the surface syntax level and instead focus on a grounded semantic interpretation. In this process, core problems in NLP such as coreference and grounding need to be addressed. Furthermore, the ability to leverage world knowledge is important as what can easily be assumed or inferred is frequently omitted

from natural language discourse. In scene descriptions, many objects that are likely to exist are rarely explicitly mentioned. Natural language also commonly omits the expected sizes, orientations and placements of mentioned objects. To enable scene generation we need to ground language to constraints on visual representations. Many challenging subproblems need to be tackled: estimation of object attributes such as size, extraction of spatial constraints, and placement of objects at appropriate relative positions and orientations. Finally, after connecting language to visual referents and reconciling them with prior knowledge, we need to fully specify the arrangement of all physical objects to satisfy constraints. This final step is the scene layout problem which has been addressed by prior work in computer graphics. Since this problem is only one part of the text to 3D scene generation task and not the focus of this thesis, we draw upon the prior work in graphics.

The above description should make it clear that the text-to-3D task is a broad and challenging endeavor. One key factor in the difficulty of this task is that though spatial knowledge is an important aspect of the world it is often not expressed explicitly in natural language. This is one of the biggest challenges in grounding language and enabling natural communication between people and intelligent systems. For instance, if we want a robot that can follow commands such as "bring me a piece of cake", it needs to be imparted with an understanding that cakes are likely found in kitchens, the specific locations for the cake in the kitchen, and that the cake should be placed on a plate.

Why should we try to tackle a problem such as text-to-3D scene generation when it subsumes many subproblems that are themselves largely unsolved? By focusing on an end-to-end task which requires a deep understanding of natural language and its connection to visual representations of the world, we can guide our contributions to a practical domain where we can naturally evaluate the performance of computational systems. Furthermore, we can establish a bridge between research in natural language understanding and computer graphics. By grounding language to concrete visual representations of the world we can see whether we can computationally interpret language richly enough to enable practical, pragmatically-charged communication with people.

While language can certainly be used to express more general concepts (thoughts, emotions, arguments) that are not easily visualizable, it is still rooted in our perception of the

Figure 1.1: Generated scene for "There is a room with a chair and a computer." Note that the system infers the presence of a desk and that the computer should be supported by the desk.

world and what we can visualize. We can see before we can talk, and our visual experiences are intertwined intimately with our knowledge of the world (Berger, 1972). Thus, the visualizable part of language is an important starting point in grounding language. In this dissertation, I will focus on one of the most fundamental aspects of visualizable language: spatial language. Spatial language is the basis that we use for higher level concepts. Time is viewed as an extension of space (Boroditsky, 2011). Through metaphors such as "high price", "close friendships" it is used for expressing high level concepts (Lakoff and Johnson, 2008). The embodied experience that is necessary to understand spatial language and its use in abstract expressions such as these can be represented as a set of spatial knowledge priors. Defining and learning these priors is one of the goals of this thesis.

The pioneering WordsEye system (Coyne and Sproat, 2001) addressed the text-to-3D task and is an inspiration for our work. However, there are many remaining gaps in this broad area. Among them, there is a need for research into learning spatial knowledge representations from data, and for connecting them to language. Representing unstated facts is a challenging problem unaddressed by prior work and one of our contributions. This problem is a counterpart to the image description problem (Kulkarni et al., 2011; Mitchell

et al., 2012; Elliott and Keller, 2013), which has so far remained largely unexplored by the community.

In this thesis, we present a text to 3D scene framework that incorporates prior knowledge learned from data. We start by defining the text to scene task, and formalizing several subtasks. We then present a representation for spatial knowledge that we learn from 3D scene data, and connect it to natural language. We show how this representation is useful for generating 3D scenes from natural language, inferring unstated facts and resolving spatial constraints. Using a parallel corpus of 3D scenes and natural language descriptions of the scenes, we learn groundings of lexical terms and show how they can improve 3D scene generation. Lastly, we extend our model to handle linguistic commands that interactively manipulate the state of generated 3D scenes.

By focusing on the text-to-3D task I will demonstrate that extracting spatial knowledge is possible and beneficial in a challenging scenario: one requiring the grounding of natural language and inference of rarely mentioned implicit pragmatics based on spatial facts. Figure 1.1 illustrates some of the inference challenges in generating 3D scenes from natural language: the desk was not explicitly mentioned in the input, but we need to infer that the computer is likely to be supported by a desk rather than directly placed on the floor. Without this inference, the user would need to be much more verbose with text such as "There is a room with a chair, a computer, and a desk. The computer is on the desk, and the desk is on the floor. The chair is on the floor."

My thesis is that linking language to concrete geometric representations and establishing a set of common-sense spatial priors are key steps towards understanding and interpreting natural language describing real environments. Though it might be possible to approximate such understanding without explicitly representing geometric structure, such an understanding is unlikely to map to human notions of space and geometric relations. In this thesis, I demonstrate that prior knowledge connecting language and geometry is critical for improving 3D scene generation from succinct natural language, and that we can learn such priors directly from data.

1.2 Challenges of grounding language

Grounding natural language to concrete representations of real-world structure is an incredibly challenging task. One of the assumptions of this thesis is that visually situated understanding can be helpful in addressing this challenge. In a visually-situated framework, we seek to improve natural language understanding by supporting a system with extra information about the physical world, mostly in the visual modality. We choose to tackle the task of generating 3D scenes from text as a specific case of visually-situated reasoning because in confronting this domain, we are forced to face the missing common-sense knowledge we need to connect language to everyday environments.

What are some of the most significant challenges of interpreting language? We might consider a "simple" model that would map noun phrases to objects with the head noun giving the category of the object, adjectives describing the object, and prepositions giving the spatial relationships between objects. However, this approach is not as straightforward as it might initially seem. For instance, we need to be able to identify the visualizable nouns representing concrete objects from non-visualizable ones corresponding to abstract concepts. In addition, often there is not necessarily a one-to-one mapping between a noun and a 3D model. Furthermore, 3D model databases typically lack semantic information making it hard to query with attributes and constraints. Moreover, ambiguity and synonymy have to be handled intelligently. For example, even within a fairly concrete and unambiguous reference such as "cat", there are different meanings that can map to disparate visual representations (kitty, feline, etc.) Finally, the interpretation of many relations and adjectives has to be performed in context. Tall for a person is different from tall for a building and both require notions of the typical height of a person or a building respectively. Similarly, spatial language such as "on" a wall and "on" the floor needs to be visualized very differently depending on the implied context.

In summary, the key challenges are as follows:

- Separation of visualizable and non-visualizable entities
- Handling of referential ambiguity and synonymy
- Grounding of attributes and relations is intrinsically context-based

Figure 1.2: Connection between 3D scene representation and image space Our scene representation is a full 3D data structure (top left) which can be projected and rendered from any viewpoint.

1.3 Why 3D?

Why should we aim to generate 3D scenes instead of images? Though images are by far the most common encoding of the perceived world, we live in a full 3D environment. Our interactions with the world take place in the full three dimensions of space (or even four dimensions if we include time as another axis). It is important to consider the constraints that a full 3D interpretation of the world imposes on the structure of real scenes. Moreover, a single 3D scene compactly describes the appearance of an environment from an arbitrary number of viewpoints (in contrast to 2D images which are described by a particular viewpoint).

Figure 1.2 demonstrates the connection between our 3D scene representation and images rendered from it. In addition to this connection between 3D environments and perceived 2D viewpoint images, we are interested in how language is used to succinctly capture and describe relevant aspects of the scene. Text represents this communication channel which often leaves out details and is open to many interpretations. However, text and language also capture many other ways to experience the world beyond visual sensing, including touch and smell. In addition, language can be used to express higher level concepts such as thoughts, opinions, arguments, and so forth.

In text to scene, we aim to take a natural language description, and generate a virtual representation that would fit it. Language is a low bandwidth means of communications. Only the essential, relevant parts are communicated (Clark, 1996). Common expected arrangements of objects in the world are left out. What is communicated also depends on context, both in what has been said before and what we are currently trying to achieve.

To take a broad view, we can consider the goal of *graphics* as representing and simulating the world in a virtual space. In contrast, *vision* focuses on the understanding of sensory data (mainly visual data), while *robotics* is about manipulating and interacting with the real world.

Traditional graphics has focused mainly on visual simulation (rendering), but also includes sound simulation, and physics simulations for animation. The best simulations take advantage of existing knowledge of how the world works: the physical laws of nature that generate what is observed. For instance, models of light propagation and light interaction with matter form the basis of realistic rendering algorithms. While the ultimate goal of vision research is to recover the three dimensional structure of the world from images, and there is a long line of work in modeling the world starting from very early vision research (Roberts, 1965; Besl and Jain, 1985), this remains a largely unsolved problem. In contrast, current state of the art techniques come from an alternative line of work that is concerned purely with the input sensory data (images) and desired output (labels or sequences of words). In this way, vision methods approximate the learning experience of an agent for a specific task: a learning algorithm that can take input observations and output appropriate responses given training data of such input/output pairs. There is not necessarily an explicit attempt to model the underlying process that generated the observations. For

example, we can think of deep convolutional neural networks (LeCun and Bengio, 1995) trained on images as assimilating sensory input and responses to create a model that can generate additional responses given similar input.

In this dissertation, we are interested in taking natural language scene descriptions and creating a virtual 3D representation of the scene. This implies that we are interested in having a world representation in full three dimensional space, and a model of how scenes can be generated in that space. From this virtual representation, we can then generate many images showing different viewpoints of the scene, and interactively manipulate and change the scene. Furthermore, a 3D representation implicitly encodes many important facts about the structure of scenes, including distances between objects, and object-to-object support relations.

Let us clarify the distinction we make between 3D scenes and images of those scenes. When we refer to images we mean the typical 2D representation of a grid of pixels with colors. More recently RGB-D sensor technology has allowed depth to be added to the color of each pixel giving a "2.5D" representation of a viewpoint. Fundamentally, we assume that a single image always represents a single viewpoint. Due to this, an image has necessarily been the subject of a projection operation, and has lost much information on the structure of a perceived scene.

In contrast, when we refer to 3D scenes, we mean a full 3D representation capturing the geometric structure and appearance of a scene from any viewpoint. Computer graphics has explored several methods of representation for 3D scenes. Two of the most common representations are surface mesh representations, and voxel space representations. We use a surface mesh representation for our scenes since the vast majority of 3D models of objects are designed using surface meshes. More concretely, surface mesh representations describe the surface boundaries of solid objects as collections of simple primitives (such as triangles in 3D space). 3D surface meshes with millions of primitives can be efficiently rendered at real-time rates with modern graphics hardware.

Going back to our task of text to 3D scene generation, there are many representations that we could choose as knowledge sources. Parallel corpora of text in different languages, corresponding images and text, or corresponding video and text, and finally corresponding 3D scenes and text. While there are many text corpora, and quite a few image or video

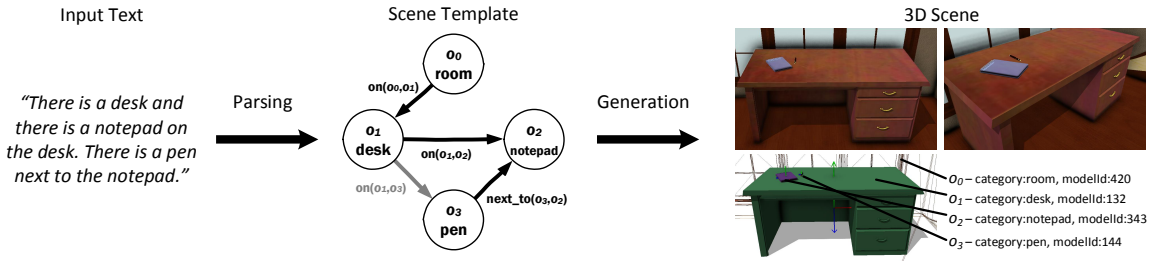


Figure 1.3: Illustration of the text to 3D scene generation pipeline. The input is text describing a scene (left), which we parse into an abstract scene template representation capturing objects and relations (middle). The scene template is then used to generate a concrete 3D scene visualizing the input description (right). The 3D scene is constructed by retrieving and arranging appropriate 3D models.

corpora, they lack the completeness of information of 3D scenes which can capture the geometric structure of real environments. A high-fidelity representation of the environment is critical in taking an embodied, agent-centric view of the world. An agent gets sensory information about the world and forms a mental model of the world based on the sensory input and priors on what the world is like. The priors can come from direct prior experience with the world or second hand information we glean from books and images. Humans use *language* to communicate information about the world. For instance, a speaker would say "cat on a table" in the hopes that it would trigger a listener into imagining a virtual world that is the same as the world the speaker is currently experiencing or imagining. However, all of these references require the original structure of the world to have meaning.

1.4 Task definition

We define text-to-scene generation as the task of taking text that describes a scene as input, and generating a plausible 3D scene described by that text as output.

More precisely, given an utterance u as input, the output is a scene s: an arrangement of 3D models representing objects at specified positions and orientations in space. To generate scene s, we select objects from a dataset of 3D models and arrange them in likely configurations based on the constraints imposed by the input text u.

There are several challenges in converting a natural language description to a 3D scene.

- What constraints are implied by the text?
- What is the prior knowledge required to represent a scene?
- How should we ground the objects and spatial relationships?

We address three main challenges (see Figure 1.3). First, we take natural language and extract physical and spatial constraints expressed which we will refer to as *scene template parsing*. We then take the explicitly specified constraints and expand them through *scene inference*. Finally, we transform the scene template into a physically realizable 3D scene through the process of *scene generation*.

For this to be possible, the system must be able to automatically specify the objects present and their position and orientation with respect to each other as constraints in 3D space. To do so, we need to have a representation of scenes (§ 3.3). We need good priors over the arrangements of objects in scenes (§ 4.2) and we need to be able to ground textual relations into spatial constraints (§ 6.1) and object presence constraints.

To re-iterate, we break down our task as follows (see Figure 3.3):

Scene template parsing (§ 5.1): Parse the textual description of a scene into a set of constraints on the objects present and spatial relations between them.

Scene inference (§ 5.2): Expand this set of constraints by accounting for implicit constraints not specified in the text using learned spatial priors.

Grounding (chapter 6): Given the constraints and priors on the spatial relations of objects, transform the scene template into a geometric 3D scene with a set of objects to be instantiated.

Scene layout (§ 5.3.2): Arrange the objects and optimize their placement based on priors on the relative positions of objects and explicitly provided spatial constraints.

To perform grounding and layout we need semantically annotated 3D models so as a part of this thesis, I contribute to ShapeNet[1]: a broader project aiming to construct a large dataset of 3D models with useful semantics.

[1] `http://www.shapenet.org`

1.5 Contributions

In this dissertation, I make the following contributions.

- I provide a formal problem description for text2scene and break down the task into well-defined subtasks. I introduce an overall probabilistic formulation that allows for incorporation of prior knowledge. I define a scene representation that can be connected to prior knowledge, and on which inference can be performed.
- I present a spatial knowledge representation that can be learned from 3D scenes and captures the statistics of what objects occur in different scene types, and their spatial positions relative to each other. Using this learned spatial knowledge representation, implicit constraints can be inferred, and plausible scenes can be generated from concise natural text input.
- I learn to ground both references to objects and spatial relations. I present a model to learn lexical groundings of textual terms to 3D model referents (i.e., choosing 3D models that represent objects referred to by terms in the input utterance u). In addition, I model spatial relations (left, on top of, etc.) and learn a mapping between language and the geometric constraints that spatial terms imply.
- I extend the presented system to handle parsing and grounding of natural language commands for manipulating scenes.
- Present a model that does scene template parsing learned from data
- Provide a framework and prototype system for the text-to-3D task, along with a dataset of 3D scenes and natural language descriptions provided by people. As part of this dissertation, we also created a semantically enriched database of 3D models. This dataset is provided to the public for research as part of the ShapeNet project.

For the work in this thesis, I created two frameworks. Most of the images in this dissertation were produced using a Java+Scala framework based on JMonkeyEngine, an open source Java based game engine. In addition, I also implemented a web based framework using THREE.js. Figure 1.4 shows a screenshot from the online system which is available as an online demo at `http://nlp.stanford.edu/projects/text2scene.shtml`.

Figure 1.4: Web based UI for the text to scene system.

1.6 Dissertation structure

Chapter 2 In this chapter, I will introduce prior work on text to scene generation as well as relevant work in the image space (from text to image and image to text). In addition, I will describe related work in graphics for automatic scene layout and grounding of spatial concepts in NLP. Finally, I will discuss prior work on using natural language commands and semantics for interactive scene design.

Chapter 3 I present a formal problem formulation for text to scene generation and define the representation that will be used in the remainder of this dissertation. I will also describe a probabilistic model formulation and the overall system architecture for addressing the problem.

Chapter 4 This chapter describes the representation of prior knowledge required in my system. Learning of spatial priors is also described. Contents of this chapter are drawn from Chang et al. (2014b) and Savva et al. (2015).

Chapter 5 I present the details of a basic Text2Scene system with a more in depth discussion of the individual components of scene template parsing, scene inference, and scene

generation. The system is largely similar to the system presented in Chang et al. (2014b).

Chapter 6 I describe work on learning lexical groundings of text to spatial relations (Chang et al., 2014b) and to objects (Chang et al., 2015).

Chapter 7 In this chapter, I give an concrete application for the text to system in interactive scene design. I describe how the system can be extended to handle interactive natural language commands. Ideas for this chapter are taken from Chang et al. (2014a).

Chapter 8 Finally, I conclude with some discussion and avenues for future research in text to scene generation.

Chapter 2

Related work

The importance of linking language and visual information has long been recognized, with early work such as Waltz (1980) arguing for the understanding language through the process of describing and picturing the world. There has been a progression of prior work addressing generation of visual output given some input text. Early systems have been limited to simple grammars and simple 2D outputs or primitive 3D shapes (Winograd, 1972; Boberg, 1973; Simmons, 1975; Kahn, 1979; Adorni et al., 1984). Much of the early work has focused more on establishing frameworks for reasoning about important concepts such as stability and gravitational support. However, due to the limits of technology at the time these explorations had to be done in very limited domains such as blocks world with fixed rules to handle the input grammar. The graphics technology at the time also severely restricted the complexity of the primitives to be visualized and manipulated for producing output imagery. Examples of such pioneer systems are the SHRDLU (Winograd, 1972) and PUT (Clay and Wilhelms, 1996). Srihari (1994) gives an overview of some early work linking language and visual information.

Since then, there have been significant advances in computing as a whole, and the individual fields of graphics, vision, NLP and AI. However, the focus has moved away from the integrative problem domains handled by the early systems, and into tasks that can be addressed in isolation within each field. Recent successes, particularly in the field of vision using neural networks, have relied on a methodology which bypasses the need for an explicit semantic representation to act as intermediary between input and output.

The task that is the focus of this thesis, text to 3D scene generation, is an integrative problem domain in the style of the early work, which I believe is very timely to explore after several decades of progress in the related fields of research. To generate scenes from text, we draw upon prior work in natural language processing for grounding text, and in graphics for automatically laying out 3D scenes.

In this chapter, I start with a broad overview of the problem of translating from one modality to another (§ 2.1) and a discussion of how the task of text to 3D scene generation fits in this space. In § 2.2, I summarize prior work on generating 3D graphics from text. As a comparison, I will also discuss efforts to create 2D images from text (§ 2.3.1) and some recent work that aims to generate text descriptions for images by performing automatic image captioning (§ 2.3.2). A good alternative summary of systems that go from text to graphics is given in Glass (2008).

I then describe in more detail relevant work for each of the specific components in the system. In Section 2.4, I discuss work in language processing for grounding language, focusing on the interpretation of language to spatial relations. In § 2.5.2, I discuss prior work that uses natural language to manipulate scenes and how they relate to the the concept of declarative modeling (§ 2.5.1). Recent advances in graphics for automatic scene layout are described in § 2.5.3. Finally, I conclude with prior work investigating the use of text and semantic knowledge in interactive scene editing system (§ 2.5).

2.1 Overview of cross-modal systems

When we consider methods that go from input in one modality to output in another modality, we can view them as translating from one space to another. As such, we can view the problem as a variant of the machine translation problem where we are translating from one language to another. We can categorize research in this area by the input and output modalities, and also by the intermediate representations and method they use.

2.1.1 Choice of modality

In this thesis, I focus on going from text to a high-level semantic representation, to a 3D scene representation, and finally to an image. This pipeline allows us to utilize graphics methods in the last steps to render realistic 2D images from the 3D scenes. The inverse of this process—i.e., 2D to 3D reconstruction is an open research problem in the vision community with many challenging subproblems. In this section, I will categorize prior work that deals with converting text to other modalities, or converting other modalities to text. As a broad split, we'll consider systems that deal with 2D vs 3D visual representations, and also static vs dynamic visuals.

Typically a text to scene system ultimately generates an image, since that is how we visually perceive the scene. However, we will consider systems that only target generation of images separately from systems that generate a 3D representation. Systems that go from text to 2D images fall in the top left quadrant of Table 2.1. The closest prior work in this space is the generative part of the clip-art image system of Zitnick et al. (2013). This work brought to the forefront a high-level view of scene structure by using simple 2D characters and cartoon scenes to explore how language is connected to the semantics of the scenes and the actions.

There has been much recent work in generating textual descriptions for image input (and to a lesser extent, for video input). Most approaches either take advantage of linguistic knowledge and structure (Kulkarni et al., 2011), or they leverage purely statistical methods that treat text as a sequence of output symbols to be generated given the observed visual input (Karpathy and Fei-Fei, 2015). This work falls in the top left part of Table 2.2. The recent abundance of video data has made it possible to train systems with similar principles to the ones working with images. Some prominent examples of this work are given in the top right part of Table 2.2.

In the same table, we note that there is a conspicuous absence of work aiming to generate text given 3D scenes or 3D animations. This observation can perhaps be explained by contrasting the abundance of data used for image2text and video2text with the scarcity of publicly available 3D scene and 3D animation data. This is changing in recent times as there is an increasing amount of openly available 3D data on the web. Generating text

	static	dynamic
2D	**text2image** Zitnick et al., 2013	**text2video** Lin et al., 2014
3D	**text2scene** WordsEye (Coyne and Sproat, 2001) *This work*	**text2anim** Swan (Lu and Zhang, 2002) CarSim (Åkerberg et al., 2003) CONFUCIUS (Ma, 2006)

Table 2.1: Summary of prior work in image/scene generation from text.

	static	dynamic
2D	**image2text** BabyTalk (Kulkarni et al., 2011) NeuralTalk (Karpathy and Fei-Fei, 2015)	**video2text** Krishnamoorthy et al., 2013 Venugopalan et al., 2015b
3D	**scene2text**	**anim2text**

Table 2.2: Summary of prior work in describing images/scenes with text.

captions for images or videos is a challenging vision problem since it typically involves object recognition. While the problem may seem trivial in the context of 3D scenes where the segmentation and positions objects are known, it is still challenging due to the lack of semantic annotation of objects. In addition, it is extremely challenging to decide what is pragmatically relevant and salient to a natural description.

We have investigated systems that generate either text or visuals but we can also consider systems that interactively manipulate visuals using text. This class of systems is summarized in Table 2.3. The most well-known examples of such systems are early pioneers of grounded AI research such as SHRDLU (Winograd, 1972) and Put (Clay and Wilhelms, 1996). Although there are textual interfaces for searching images and videos, there are no well-known text-based systems for manipulating existing images and videos. To the best of my knowledge, there are no text-based systems for manipulating existing 3D animations.

	static	dynamic
2D		
3D	SHRDLU (Winograd, 1972) Put (Clay and Wilhelms, 1996) *This work*	

Table 2.3: Summary of prior work in using interactive commands to edit images/scenes.

2.1.2 Choice of representation

As with translating from one language to another, we can view the methods and the representations they choose as belonging to a spectrum similar to Vauquois' triangle (see Figure 2.1).

There are three important legs of this cross-modal translation process: the representation used as an intermediary, and the methods used to perform the analysis and generation steps. Most systems that go from text to 3D scenes or animations use a full semantic representation as an intermediary. This allows the system to leverage existing NLP approaches for augmenting the analysis stage with text understanding. For text-to-scene, the generation step leverages graphics systems and existing 3D model data to drive the synthesis of the scenes. This is in contrast to image-space methods where one could imagine using a similar scheme, but synthesis of images from first principles is not at all studied. Furthermore, recent work in image-to-text methods has largely avoided the need for an intermediate semantic representation (other than mapping between token-level entities in language and images). Instead, the intermediate representation is a continuous vector space (in contrast to the structured semantic representation in text-to-scene systems). This current trend and the different approaches associated with it are discussed in more detail in § 2.3.2.

Looking at the generation step in more detail, there is a spectrum of retrieval-based vs composition-based approaches: options on this continuum differ in the level of the basic unit: from characters, to words, to phrases, to whole descriptions; pixels, to image segments, to clip art, to entire images; triangles, to meshes representing object parts, to 3D objects, to entire 3D scenes. The necessity to specify rules for putting these basic units together is what makes the generation problems challenging. Language is in many ways the

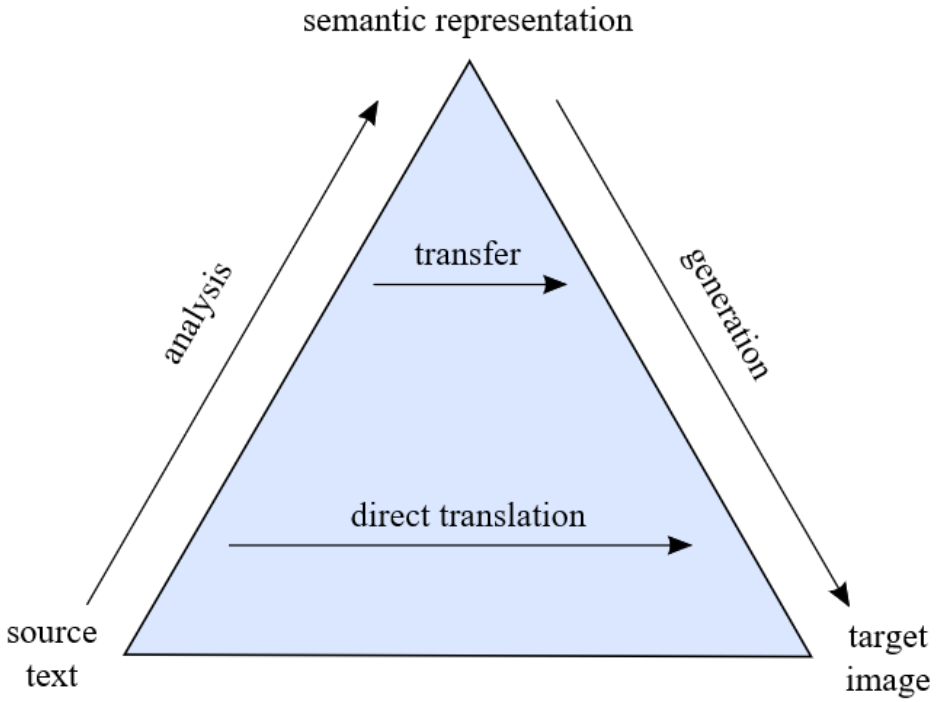


Figure 2.1: Vauquois' triangle adapted for translating between modalities.

simpler problem of the three: it is linear, discrete, with a long history of study on its units and structure (linguistics). In contrast, composition rules are less well studied in 2D images and 3D models—this is perhaps more the domain of the fine arts, and less of science. There is also a spectrum of choices for intermediate representations. The intermediate representation can be a distributed vector representation, or a explicit structured semantic representations, of which there are many choices ((logical forms, frames, or maps).

2.2 Text to 3D

Text to 3D scene generation systems presented by prior work require various kinds of world knowledge in order to operate. The language parsing components need world knowledge to disambiguate and enrich the intermediate representation. The geometric 3D models representing real-world objects also need to be annotated with various attributes so that priors on their relations to each other can be used. The spatial relation priors themselves are also a form world knowledge which is necessary to evaluate how well different arrangements of objects depict the input language. Prior work has assumed that either this world knowledge is provided through laborious manual annotation, or it has restricted the domain of discourse and output capabilities of the text to scene system to reduce the required world knowledge.

This thesis focuses on showing that we can learn priors encoding common-sense world knowledge directly from scene and language data. It then illustrates how these automatically learned priors alleviate constraints on the acceptable input language, and the complexity of the output scenes.

2.2.1 Text to 3D scene

The main prior work in text-to-3D scene generation is WordsEye (Coyne and Sproat, 2001), a pioneering prototype system that was one of the first systems to demonstrate how textual descriptions of scenes could be converted to 3D graphics (see Figure 2.2). WordsEye was based on a pipeline that transformed text to a semantic representation that could then be depicted by selecting models from a model database, and arranging them to form a 3D

Figure 2.2: Example of scene generated using WordsEye. This is an impressive, albeit fairly specialized, demonstration of WordsEye's capabilities.

scene. The earlier Words Into Pictures system Olivier et al. (1994), viewed constraints as placing potential (i.e., unnormalized probability mass) over positions and investigated ambiguity in frame of reference. However, it did not present an end-to-end pipeline for text-to-scene generation in the way that WordsEye did. Later work (including our own) follows a similar pipeline. The authors of WordsEye have continued to develop the system, with subsequent papers to be discussed later providing more details on how the system works. WordsEye is also now available online at `https://www.wordseye.com`.

The 2001 WordsEye system used a model database of 2000 3D objects. Most importantly, the authors of WordsEye identified key aspects of semantics and attributes of graphics representations that were needed for this task. The key contribution of WordsEye was to provide a comprehensive set of the most important semantics of objects that are necessary for text to scene generation: sizes, orientation, parts (what parts should be colored, what parts are transparent), spatial tags (shape, base, where are objects placed for "on", or "under" relations), skeletal poses, functional tags (what object are used for what verb, where objects are gripped for a verb). These attributes were connected to semantic representations of the input text formulated with entities, actions, attributes, and relations. WordsEye used a semantic frames representation, which was later formalized as the concept of "vignettes" (Coyne et al., 2011) —an extension of FrameNet (Baker et al., 1998)

frames with knowledge necessary to visualize actions and locations. The appropriate vignettes and related 3D models are identified using table lookup from given words. The 3D models are mapped to WordNet synsets so that they can take advantage of the taxonomy to represent references to the same concrete objects from different levels of abstraction (e.g., cat, feline, mammal can reference the same model).

WordsEye was implemented in LISP. The input text was tagged and parsed, and converted to a dependency structure. It was then semantically interpreted and converted into a high-level representation capturing semantics (later developed as the "vignettes" concept). Depiction rules were used to convert the semantic representation to a set of low-level descriptors representing 3D objects, poses, spatial relations, and color attributes. In this process, WordsEye identified the need to resolve conflicts and add implicit constraints. It used transduction rules to address part of this issue but was limited in the types of inference it could perform. For example, given the sentences "The lamp is on the table. The glass is next to the lamp." it could infer that the glass is also on the table. Another example is the sentence "The dog and cat are on the rug" from which it could infer that the dog and cat are not in the exact same position, and are instead next to each other. The types of spatial relations used in WordsEye and how they are interpreted is described in Coyne et al. (2010). The system also focused on supporting some richer concepts such as agents and actions by agents, and symbolically depicting these concepts (see Figure 2.3)

In many ways, WordsEye was far ahead of its time. It handled complex scenes such as the scene in Figure 2.4. The authors demonstrated the promise of text to scene generation systems but they also pointed out some fundamental issues which restrict the success of their system. Much spatial knowledge is required in interpreting text—knowledge which is hard to obtain and thus had to be manually specified in a laborious effort, or was omitted thus requiring users to use unnatural language to specify it. This meant that users had to come up with convoluted, unnatural sentences such as "the stool is 1 feet [sic] to the south of the table") to express their intent (see Figure 2.5).

In an earlier effort, Sproat (2001) investigated inferring aspects of the environment (room type, time of day, season) based on co-occurrences from text. However, this work seems to have never been integrated into the WordsEye system. Follow up work by Coyne et al. (2012) has attempted to collect spatial knowledge through crowd-sourcing. This effort

The blue daisy is not in the army boot.

John rides the bicycle.
John plays the trumpet.

Figure 2.3: Example of generated scenes from WordsEye (Coyne and Sproat, 2001). WordsEye can depict symbolic representations of abstract ideas and also posed agents. However, WordsEye required explicit, detailed descriptions of all objects and relations.

was also followed by other groups as well (Hodhod et al., 2014). However, these efforts do not address directly the root cause of WordsEye's limitations—the inability to learn a rich spatial knowledge base from data. This is precisely the focus of my thesis, and as I will show, doing so allows us to interpret more natural, and concise language describing scenes.

Following WordsEye, there have been several systems that attempted to generate static scenes from text (see Figure 2.6 and summary in Table 2.4). These systems did not have the scope of WordsEye, and several focused on how to interpret various spatial relations and the automatic placement of objects to satisfy all constraints in the input text. I present a brief description of these systems below. Later in this chapter, I will review prior work focusing specifically on these aspects. In § 2.4.1, the complexities of grounding spatial language is described in more detail, and Sections § 2.5.1 and § 2.5.3 discusses the role of constraints for scene modeling and recent work in automatic scene layout.

- Seversky and Yin (2006) attempted to clarify the interpretation of spatial relations such as "left" or "on" and to perform automatic layout. The authors describe how to automatically extract the six surfaces (top, bottom, front, back, left, right) and nine spatial regions of an object (center, left, right, front, back, upper right, upper left, lower right, lower left) using a voxel representation. These were then used to

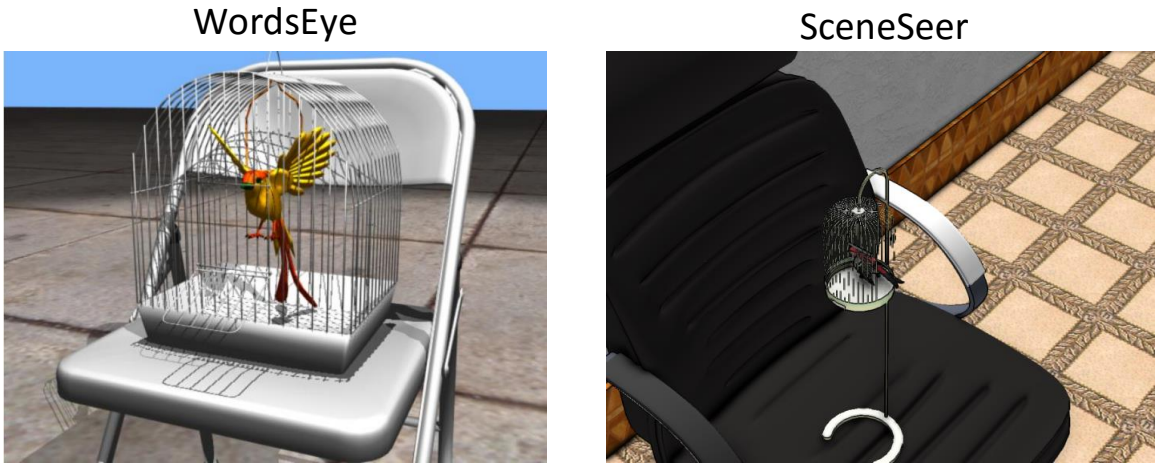


Figure 2.4: Example of generated scene from WordsEye and our system for *The bird is in the bird cage. The bird cage is on the chair.*

automatically position objects so as to satisfy spatial constraints.

- Tappan (2004) and Tappan (2008) partitioned space using rings/wedges and mapped spatial relations to a circular, two-dimensional field of 100 rings and 32 sectors encoding a probability distribution over space. The authors were concerned with how context impact the interpretation of spatial language, and selected a zoo scenario to explore the influence of context. For instance, they considered how the interpretation of "in" differs for "a hippo in a lake" vs "a raft in a lake" vs "a hippo in a raft". This work was also one of the first to use Monte Carlo sampling to lay out the objects in order to satisfy spatial constraints. While limited to 2D layouts, the system could handle complex combinations of constraints.

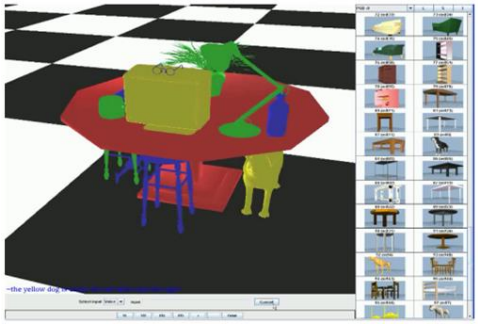
- Lu et al. (2010) used the same ring/wedge based spatial representation as Tappan (2004). They claimed that semantic information about objects and spatial relations are easily extracted from input text description to give a graph based semantic representation with constraints on the position of objects. The system iteratively placed objects, in order of most constrained to least, until all objects are placed. Feasible locations are randomly sampled and bounding boxes were used for collision detection.

- Unal (2014): presents a basic end-to-end text to scene system with actual source

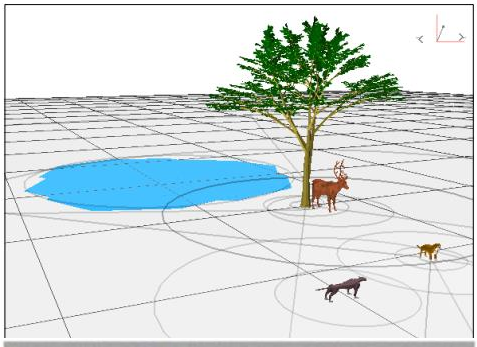
Our System
There is a desk with a chair. On the desk, there is a red stapler.

WordsEye
There is a desk with a chair. On the desk, there is a red stapler.

Figure 2.5: Example of generated scene from WordsEye and our system. Our system generates complex scenes and fills in additional object detail with much more succinct natural language input.



Seversky and Lin 2006
A green plant is on the red table. The blue water bottle is next to the green lamp. The blue stool is under the red table towards the front.



Tappan 2008
The dog is south of the tree and near the panther; the panther is to the right of the dog; and the elk is near the maple tree and midrange from and facing away from the pond

Lu et al 2010
The desk is in the middle of the house. A fireplace is to the north of the desk and facing the desk. It is adjacent to the desk. A shelf is to the west of the fireplace and adjacent to the fireplace. The shelf is facing south. There is a sill to the west of the desk and directly facing the desk. A potted plant is to the south of the sill. A cupboard is to the east of the desk and facing directly west. A flower is to the south of the cupboard. A pumpkin is in the shelf and a fruit is on the fireplace. To the north of the cupboard there is also a floor lamp.

Unal 2014
There is a room. A sofa is in the room. A table is in front of the sofa. A man is behind the sofa. A toy is on the table. A car is behind the room.

Figure 2.6: Some example results from prior work in text to scene generation following WordsEye. These systems all required explicit mentions of all objects present in the scene along with detailed descriptions of their relative positions.

System	Additional Priors	Layout	Relation Mappings	Spatial Reasoning
WordsEye (Coyne and Sproat, 2001)	attachment, spatial tags	rules	many (manual,context)	minimal
(Tappan, 2004)	containment, scale map	MCMC sampling	some (manual,context)	some
(Seversky and Yin, 2006)	relation voxel regions	voxel regions	some (manual)	none
(Lu et al., 2010)	none	random sampling	some (manual)	none
(Unal, 2014)	scale map	rules	some (manual)	some
This thesis	learned from corpus	probabilistic score	some (learned,context)	some

Table 2.4: Summary table of text to static 3D systems. Overview and categorization of systems presented by prior work. Most of the systems use rule-based methods for language processing. All prior work systems require a manually annotated corpus of 3D model with categories, sizes, and orientations as well as the listed additional priors.

code.[1] The system also includes a basic question answering module which can answer questions about spatial relations of the objects in the scene (e.g. where is X, is X to the right of Y, can X see Y). The authors demonstrate that with a 3D representation, it is possible to perform simple spatial reasoning.

Both Tappan (2008) and Lu et al. (2010) used a semantic network representation similar to our scene templates for representing the scene. While some of these systems reiterated the need for common sense knowledge in text-to-scene generation, these systems did not attempt to encode any spatial priors on how objects are placed. The knowledge they included was in the form of object-level annotations such as object name, size, front side, and taxonomic organization (similar to WordNet). As shown in Figure 2.6, these systems were able to take text input with complex constraints and generate appropriate scenes. However, they required all objects to be explicitly mentioned, and the placement of objects to be explicitly stated. Without constraints on placement, only uniform priors were used for placing under-constrained objects.

As we discussed, WordsEye and later follow up work have introduced the first full text-to-scene pipeline systems but have not addressed the learning of spatial priors, leaving several fundamental challenges unanswered:

- Limited inference of implicit constraints can be performed
- Users need to state all objects and relationship between objects.
- Everything is manually coded and annotated.

[1] https://github.com/ai-ku/langvis

Our contribution is an attempt to help mitigate some of these issues through the use of probabilistic scene priors. We learn these spatial priors from data, and show how they can be used in going from text to semantic representations of scenes.

2.2.2 Text to animation

In addition to work on text to static 3D scenes, there have also been a number of systems that attempt to transform text to 3D animation. Interestingly, there is actually significantly more work that attempts to address text to animation than to static scenes. This is likely due to the broader set of scenarios in which simple illustrative animations are useful—in contrast to static 3D scenes.

Text to animation is considerably more complex. Some of these systems attempt not just to provide visual animation but include generation of music and narration. Other than the problem of parsing natural language to a semantic representation, there are additional challenges in story-telling such as selection of camera viewpoints, lighting, etc. Furthermore, planning for temporally-extended sequences of actions introduces an even more complicated setting for resolving ambiguities and constraints. This increased complexity often necessitates a different focus than the work in this thesis.

In this section, I will describe some of the relevant work in generating animation from text. While generation of animation is outside the scope of this dissertation, we view text to animation as a natural next step and we believe that improvement in static scene generation (or "snapshotting") is essential to improving generated animation. Conventional animation is frequently first storyboarded with sequences of key static scenes, so the ability to generate scenes, even though they may be static , is still directly applicable to the realm of animation.

Many of the challenges of text to scene generation are also found in text to animation (also called fiction to animation). The key steps of parsing and grounding natural language text to semantic representations, inferring implicit unstated constraints with priors, mapping the semantic forms to graphics primitives and visualizing the primitives exist in the context of animation generation as well.

Grounding and interpretation of spatial language are at least as challenging in scenes with animation as they are in static scenes. In addition, animation generation has some

unique challenges. One of these challenges unique to animation is the temporal sequencing of events. Input language has to be interpreted to identify ordered constraints on the environment, the agents (characters), and the events taking place. Issues such as coreference and referent resolution are even more challenging with longer sequences of text. Other NLP challenges such as identifying the location and setting of an event are also compounded by the temporally-extended descriptions of animations.

Another task that is made more difficult is the interpretation of verbs and actions. There are more implicit statements and unmentioned common sense knowledge relating to the implications and constraints of common actions. For example, even generating a static snapshot for "John is washing the dishes" requires understanding that John should be standing in front of a sink, with dishes in the sink. By and large, current NLP techniques are not yet robust enough for automatic annotation of realistic scripts describing stories to be animated.

On the graphics side, the automatic posing of characters and simulation of realistic human movements are much more challenging than the automatic layout of static scenes. Consequently, posing and animation of characters is predominantly done through the manual effort of experienced 3D artists at great expense in the entertainment industry. Automated systems are rarely entrusted with the task of depicting convincing human emotions in virtual characters without human assistance or verification. Chaining together sequences of animations and planning for how they will be realized in a specific geometric environment is also challenging. Even a simple "The man grabs the mug on the table" could be hard if the man is sitting in a couch on the other side of the room. It is necessary to plan a sequence of actions that makes the man stand up, walk over to the mug while realistically navigating around furniture, and then grab the mug with a natural pose. Moreover, if we wanted to render some part of the wide range of human emotions, we would struggle to do so with current graphics systems since much manual editing of character animations would be required.

Much of the work in text to animation focuses on the above challenges which are specific to the animation context: grounding of objects/agents, action sequencing, and path planning. Prior work for text to animation can be broadly grouped into either work that generates animation for a short scenario, or work that attempts to generate an entire sequence of events for a story. Depending on the scope of the generated animation, different

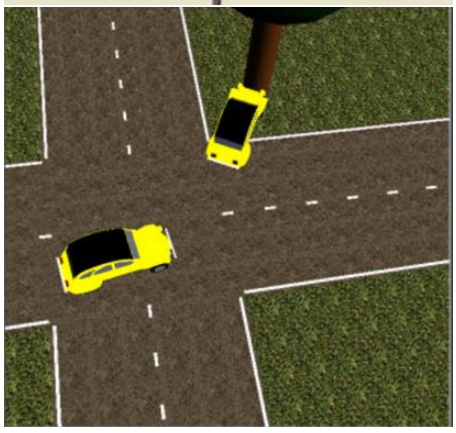
compositional units are picked.

Ani (Kahn, 1979), was one of the earliest systems which was created for animating Cinderella (and only Cinderella). It did not really take text as input. Instead it took parenthesized formal language for describing characters and actions. There was also no 3D animation as all characters were represented by 2D polygons. However, it did explore key ideas such as viewing film as a program, and defining high level semantics for controlling animation. It only addressed certain limited aspects of the problem due to "incomplete bodies of knowledge." The Story Driven Animation System (Takashima et al., 1987) was another early system that explored the idea of generating animation from natural language. Both of these early works were focusing on generating entire stories.

Swan (Lu and Zhang, 2002) was a sophisticated, rule-based system for generating animation from a restricted form of natural language. It used five levels of representations, each with a corresponding language with well defined syntax. As input, the system takes a limited form of Chinese natural language, which is transformed to a frame-based semantics description language. This language included the environment, characters, objects, and a sequence of actions extracted from the text. Using knowledge-based and pragmatic reasoning, the system constructs a qualitative script consisting of a sequence of temporally ordered scenes. Camera movements and lighting for each scene were also planned. The high-level qualitative script is then transformed into a quantitative script that specifies the series of static image frames to be generated, and includes the concrete specifications of the objects, movements, camera, lighting, and actions. Finally, the quantitative script is converted into a low-level animation scripting language that can be used to render the final animation. It is important to note that the extensive knowledge base used by the system was hand-constructed and is not accessible to future work. While it is an extremely impressive effort, it took a lot of manual effort (over 30 graduate students contributed to the project over a course of nearly 10 years).

Some other early systems focused on generating animations of simple interactions: CONFUCIUS (Ma, 2006) generated animation for single interactions, while SceneMaker (Hanser et al., 2010) is a proposed follow-up that will handle screenplays. ScriptViz (Liu and Leung, 2006) handles object resolution, and action sequence planning for grounding to a frame representation called PAR (Badler et al., 2000). BEAT (Cassell et al., 2001) is a

Confucius 2006
"John put a cup on the table."



CarSim 2003
I was driving on a crossroad with a slow speed, approximately 40 km/h. Vehicle B arrived from my left, ignored the priority from the right and collided with my vehicle. On the first impact, my rear fender on the left side was hit and because of the slippery road, I lost control of my vehicle and hit the metallic protection of a tree, hence a second frontal collision.

Figure 2.7: Some example results from prior work in text to animation generation

"behavior expression animation toolkit" that allows animators to input text that they wish to be spoken by an animated human character. The system then generated synchronized gestural behavior for the speech.

Table 2.5 provides a summary of text to animation systems from prior work. The many different approaches even for small aspects of this problem domain are a testament to the breadth of the technical challenges. We briefly discuss some systems below:

- CarSim (Åkerberg et al., 2003) is a system which generates simulations of car accidents from input text. It exhibits more advanced text processing and temporal reasoning than seen in many other systems. It uses classifiers for semantic role labeling (Gildea and Jurafsky, 2002) to identify events and arguments, and decision trees for temporal ordering, and also tries to predict the environment using a classifier. The specific domain allowed for fairly impressive parsing of natural language to animation.

- CONFUCIUS (Ma, 2006) proposes a Lexical Visual Semantic Representation (LVSR): a representation between 3D animation and syntactic information. Adapted from Jackendoff's lexical conceptual structure (Jackendoff, 1983; Jackendoff, 1992), it introduces finer ontological categories and adds basic human actions as EVENT predicates. It also decomposes actions into subactions and discusses extensively how to visualize actions, nouns, adjectives, and prepositions. It focuses on single interactions.

- Glass, 2008 present a system using hierarchical rule learning for text annotation. It incorporates automatic world generation (of terrain, cities, rooms) using graphics techniques from prior work. The authors describe the method as a "knowledge poor" approach which sometimes necessitates manual intervention. They also describe how text can be converted to annotations (associated with geometric models from a database, and trajectories). The presented system encodes several spatial relations such as NEAR, INSIDE, NO_COLLIDE, BEHIND, IN_FRONT_OF, TO_LEFT_OF and TO_RIGHT_OF. Explicit constraints are derived directly from transition and relation annotations, while implicit constraints are created using an automated heuristic approach to enforce world constraints. Constraint solving and path planning is

System	Domain	Language Complexity	Temporal extent	Scene Layout
SWAN (Lu and Zhang, 2002)	general, requires coded knowledge	restricted Chinese	long	complex
CarSim (Åkerberg et al., 2003)	car accidents	complex English	short	complex
CONFUCIUS (Ma, 2006)	single event	simple English	short	simple
(Glass, 2008)	restricted to positioning	simple English	short	simple
(Ye and Baldwin, 2008)	short activities	pre-annotated script	short	simple
(Oshita, 2010)	general, requires motion DB	simple English	long	none

Table 2.5: Summary table of text to 3D animation systems. Overview and categorization of systems presented by prior work.

used to find trajectories using a quantified interval-based constraint optimization algorithm. Keywords are used to retrieve matching models. This system handles short interaction stories.

- Ye and Baldwin, 2008 viewed the problem as translating between natural language and programs consisting of 21 graphics commands that directed the generation of the animation. They assumed an input script annotated using semantic role labeling. Then they used a maximum entropy classifier to ground verbs to a sequence of graphics commands. However, all anaphora is resolved by hand. It focused on visualizing short interaction snippets.

- Oshita, 2010: Developed a motion database that stores many motion clips for different characters. When the input text is given, the system retrieves an appropriate motion clip from the database for each verb. Temporal constraints between verbs are also extracted from the input text. The searched motion clips are scheduled based on the temporal constraints. In addition, when necessary, some automatic motions such as locomotion, taking an instrument, and changing posture are synthesized. This work focuses on longer sequences of interactions.

As was discussed, most of these systems focus on animation of characters and sequencing of event scenes, and less so on the spatial relations and placements of objects. The most notable prior work is SWAN which was used to create a complete short animation film, illustrating that it is possible to handle very complex scenarios. However, it required a lot of knowledge and manual encoding which made the task of building up the system incredibly laborious.

In contrast, Glass (2008) argues for a system that is knowledge-poor and fully automated. It is the first system to try to learn from text an annotation that is appropriate for animation synthesis. He also provides an analysis of how well different components of the system perform. However, despite the claim that the system is "knowledge-poor", it still relies on a manual mapping of a database of models. The grounding problem is not addressed at all, since a pre-specified set of rules is used to handle a small set of spatial relations, and models are looked up purely by keyword search. We believe the opposite claim is more likely to be true: knowledge is essential for text to animation, and instead the pertinent question is how to learn this knowledge without hand-coding. This was partly exemplified by Ye and Baldwin (2008) who attempted to ground verbs to graphical commands.

To summarize, in contrast to this prior work, we directly address the challenge of handling natural language for static scene generation. We believe this is an important step before the more complex animation scenario can be handled better. In order to handle general scenes without requiring unnatural language or manual annotation, we learn spatial knowledge from 3D scene data and use it to infer unstated implicit constraints. Our work is similar in spirit to recent work on generating 2D clipart for sentences using probabilistic models learned from data Zitnick et al., 2013. In the next sections we will discuss in more detail the relevant work in text to image, and image to text generation.

2.3 Other cross-modal systems

2.3.1 Text to image

Text to image generation was first addressed by Env 1 (Boberg, 1973), CLOWNS (Simmons, 1975) and the Natural Language Image Generation System (NALIG) from Adorni et al. (1984). This is some of the earliest work that attempted to generate 2D scenes from text using simple compositional approaches predicated on specific problem domain models being given as part of the input.

Most work addressing the "text to image" problem takes a retrieval approach. The given text input is used to retrieve existing images (e.g., photographs) to represent scenes in stories. The fundamental limitation of the image retrieval approach is that it can only return

Figure 2.8: Some example results from prior work in text to image generation

images that already exist in an available database of images. Furthermore, for generation of coherent stories, consistency of characters and objects is required which is challenging to achieve purely through retrieval methods.

These issues have been the topic of recent work in vision. For example, Joshi et al. (2006) retrieve and arrange photographs from a large corpus of images to generate desired scenes. Similarly in Schwarz et al. (2010), text is automatically segmented into parts and images are retrieved from Flickr for each part, and compiled together with user-in-the-loop assistance, producing an animated slide show in the end.

Naturally, we can expect that improvements in vision systems will lead to improved retrieval of images. These improvements can be used for better retrieval of 2D clipart, or appropriate background images to generate novel scenes. Furthermore, some approaches can use a semantic representation for retrieval (Liu et al., 2007; Elliott et al., 2014; Johnson et al., 2015; Schuster et al., 2015), which can lead to even more impressive retrieval of photographs from complex textual descriptions.

Systems such as Zhang et al. (2012) demonstrate that a convenient interface for retrieving images representing sequences of scenes and sequencing them appropriately can be of great value. Current systems are constrained since their output is only sequences of images. As expected, there is almost no work in the text to video task, and existing methods are based on retrieval of videos only.

There is a recent example of work that queries for video using text in the context of autonomous driving (Lin et al., 2014). This work first converts the text to a semantic graph

that is then used for querying. In this way, it follows the concept-based querying for video described in Snoek and Worring (2008). There is also some work on alignment of text and video (Tapaswi et al., 2015; Zhu et al., 2015c) though again there is no generation of video.

A recent line of research exemplified by Zitnick et al. (2013) and Alcantara et al. (2014) takes the approach most similar to this thesis. It learns priors from a dataset of 2D clipart and 2D background images (instead of a 3D model and 3D scene dataset). Due to the 2D image domain, the interpretations of spatial relationships and layouts are all constrained to 2D image space. Despite this difference, the work by Zitnick et al. (2013) is the most similar to our approach. Much like our system, it involves learning of priors and grounding of lexical terms. However, it cannot be directly used in the task of text to 3D scene generation.

In the last year, there has also been some work that attempts to generate images from text using neural networks (Gregor et al., 2015; Lazaridou et al., 2015; Mansimov et al., 2015). These efforts are still in their infancy and can only generate simple low resolution suggestive blobs. In contrast, these deep learning methods has been proven to be extremely effective at the inverse task of going from image to text, or image captioning. Though conceptually the text to image problem is closest to the text to scene task, the inverse image to text problem is also quite relevant to consider and I will describe it in the next section.

2.3.2 Image to text

Very recently, a lot of progress has been made to create vision systems that generate natural language descriptions for given input images. The typical approach used to generate text from images first maps the input image to a semantic representation, and then produces the text from the resulting higher-level representation. The alternative to this approach, is to go directly from an image input to a textual description. Most recent successes in image to text generation have utilized the latter approach. In large part, the success of such recent systems is due to taking advantage of large datasets of image/text pairs and learning a reflexive agent that can generate text given a new input image. By combining deep learning with large amounts of data, these methods have yielded impressive results (Vinyals et al., 2014; Karpathy et al., 2014; Karpathy and Fei-Fei, 2015; Donahue et al., 2015; Xu et al., 2015; Kiros et al., 2014; Mao et al., 2014; Fang et al., 2015; Chen and

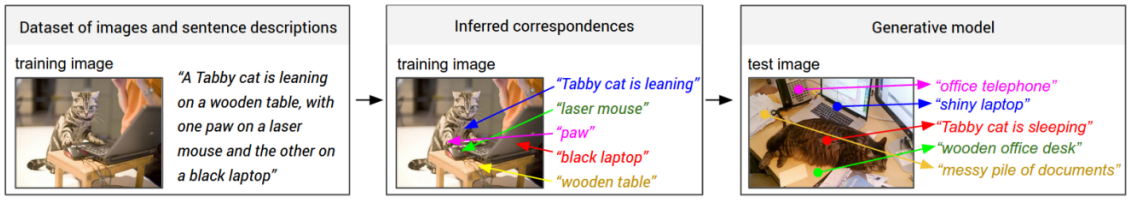


Figure 2.9: Image to textual description pipeline from Karpathy and Fei-Fei (2015).

Zitnick, 2015). Much more work along these lines has been carried out recently, and a more thorough description is given by Karpathy and Fei-Fei (2015).

In Figure 2.9, we show the image description system from Karpathy and Fei-Fei (2015). At a high level, this system uses a large corpus of image and text pairs to learn an alignment representing the latent correspondences between visual and language data. The key in doing so is to have common multi-modal embedding where both image regions and words are represented as high-dimensional vectors. For the text, a bidirectional RNN is used to learn a representation for words that includes the left and right context. For the image, a Region-CNN pretrained on ImageNet is used to detect the top 19 image regions giving 20 image regions in total when the whole image is included. Then a maximal alignment between regions and words is learned through an appropriate optimization objective. The basic alignment of sentence snippets to image regions is used to generate additional training data for a generative multi-modal RNN which is given a sequence of words and can take an image as input, model the context in its hidden layers, and predicts the next word. At generation time, it can either pick the most likely word, or sample the distribution of possible words at that step. A special end token is used to terminate sentences.

Despite the representational simplicity, these image description approaches give impressive results. More advanced models incorporate attention (Xu et al., 2015), and are capable of visualizing as well as captioning (Chen and Zitnick, 2015). While these methods seem to generate much more impressive results than simple retrieval-based methods, a recent analysis by Devlin et al. (2015) suggests that the performance of deep neural methods is much more similar to retrieval methods than one would imagine at first glance. Retrieval-based methods use a dataset of images and a nearest neighbor based approach to return the most compatible caption in the training set (Farhadi et al., 2010), decompose

and recombine language fragments, or generate descriptions using templates or generative grammars. Karpathy and Fei-Fei (2015) argues that these approaches are all limited in the variety of possible outputs and that they have not worked as well as the recent work using deep learning. However, Devlin et al., 2015 suggests the critical difference lies more in the power of the features learned by neural networks.

An alternative line of work has generated sentences that describe 2D images by first generating a semantic representation (Herzog and Wazinski, 1994; Roy, 2002), followed up more recently by work in Farhadi et al. (2010), Kulkarni et al. (2011), Mitchell et al. (2012), Elliott and Keller (2013), Elliott and Vries (2015), and Aditya et al. (2015). When we consider the task of converting from images to semantic representations, we are faced with similar challenges as taking text to semantic representations (e.g., coreference resolution (Kong et al., 2014)). Given the recent successes of image-to-text systems, there have also been some attempts at video-to-text (Barbu et al., 2012; Rohrbach et al., 2013; Krishnamoorthy et al., 2013; Venugopalan et al., 2015b; Venugopalan et al., 2015a; Yao et al., 2015).

Despite the remarkable recent progress in image to text generation, generating scenes is currently out of reach for purely image-based approaches since much of the structure of 3D scenes is unavailable in image data. In contrast, 3D scene representations serve as a more appropriate intermediate level of structure between raw image pixels and simpler symbolic microcosms (e.g., grid and block worlds). This level of structure is amenable to the generation task, but still realistic enough to present a variety of challenges associated with natural scenes.

In order to bridge the gap between visual and text input, we need to have accurate grounding of language to physical representations. A good example of this is given by work in understanding or generating referring expressions for specific objects in images (FitzGerald et al., 2013; Kazemzadeh et al., 2014).

2.4 Grounding language

An important part of generating scenes from text is to be able to correctly ground text to concrete representations. There has been extensive prior work in the grounding of language, though the vast majority of work grounds (or aligns) to either formal representations or image-based representations.

Grounding in the NLP community is typically used for generating and resolving referring expression (Krahmer and Van Deemter, 2012). More specifically, recent work in NLP has explored grounding text to physical attributes and relations (Matuszek et al., 2012; Krishnamurthy and Kollar, 2013), generating text for referring to objects (FitzGerald et al., 2013) and connecting language to spatial relationships (Vogel and Jurafsky, 2010; Golland et al., 2010; Artzi and Zettlemoyer, 2013).

There is also a line of relevant work in robotics where grounding of instructions to robotic systems is the focus (Guadarrama et al., 2013; Matuszek et al., 2013; Misra et al., 2014; Tellex et al., 2014). Naturally, in these systems the central entity is the robot and so grounding is mostly in relation to its viewpoint and capabilities to interact with the environment. A related line of work focuses on grounding referring expressions to referents in 3D worlds with simple colored geometric shapes (Gorniak and Roy, 2004; Gorniak and Roy, 2005).

Most of this work focuses on learning a mapping from text to formal representations, and does not model implicit spatial knowledge. Many priors on real world spatial facts are typically unstated in text and remain largely unaddressed. The focus of this thesis lies in the domain of natural language discourse about common environments, where implicit spatial facts are of critical importance. However, since being able to correctly interpret and ground language to spatial relations and objects is so essential for the problem of text to scene generation, I will also discuss how we learn groundings in Chapter 6.

2.4.1 Spatial language

Cognitive psychology research has explored how spatial relationships are expressed in text (Freeman, 1975; Pick and Acredolo, 1983) and how spatial concepts are mapped to different languages (e.g., the topological relations picture series by Bowerman and Pederson

(1992)). In general, there has been much investigation of the variability of human spatial language, in terms of propensities for using different frames of reference (such as egocentric versus object-centric (Levinson, 2003) as one example).

Prior work that required modeling spatial knowledge has typically defined representations specific to the task addressed. Such knowledge is usually manually provided. The most relevant work in this line is the investigation by Gapp (1994) of how language can be mapped to 3D space and relations within it. Kelleher and Genabith (2004) show how using visual saliency cues it is possible to identify the most likely referent for an ambiguous mention. More recent work by Kelleher and Costello (2009) presented a computational model of proximity and directional prepositions. Depending on the context, the exact grounding of the spatial expression can map to very different interpretations (e.g., a person near a house vs a fork near a plate have very different scales).

Collection of this sort of knowledge has also been crowdsourced. For instance, more recent work on WordsEye uses a set of manually specified relations and collects related priors from crowds of annotators (Coyne et al., 2010; Coyne et al., 2012). However, it is unclear whether they have attempted to learn any priors from the collected data, which is one of the goals of this thesis.

There have been other learning approaches that learn groundings directly from data. For instance, Dawson et al., 2013 present a generative probabilistic framework for learning spatial language in a 2D table-top environment. A closer domain is investigated by Golland et al. (2010) who ground spatial relationship language in 3D scenes (e.g., *to the left of*, *behind*) by learning from pairwise object relations selected from a list by crowd-workers.

2.4.2 Visual attributes

Other recent work has the more specific focus of grounding text to concrete object attributes such as color and shape in images (Matuszek et al., 2012; Krishnamurthy and Kollar, 2013). This work again comes from a robotics perspective, where grounding of references to objects through their perceived attributes can be extremely helpful to the competence of robotic systems. A study of grounding color terms in a more general discourse context using Bayesian methods is performed by McMahan and Stone (2015).

2.5 Semantics for scene generation

Designing virtual worlds, particularly if they need to be interactive is a challenging task. Much of the challenge stems from the need to connect the virtual representation with various kinds of real-world semantics (Tutenel et al., 2008). The meaning of virtual scenes is tied to the perceived semantics of the objects, and semantic relations rely on maintaining constraints. If we don't know anything about the usefulness of chairs, we cannot say why keeping them stacked on top of a table is a bad idea when we want to have dinner, but an excellent idea when we want to clean the floor.

This thesis views natural language text as the main source of semantic constraints on the structure of scenes. In addition, we view text as the natural medium through which intended interactions with a scene, or manipulations of its state can be conveyed. This is in contrast to most work in interactive design of scenes, which predominantly focuses on methods to make direct manipulation more efficient. This can either be done by using simple physics and rules to establish associations between objects as exemplified by Bukowski and Séquin (1995), or by learning object occurrence priors and leveraging to make automatic suggestions during interactive scene design, as presented more recently by Yu et al. (2015).

The focus of this thesis includes the representation and learning from data of spatial knowledge priors, but always in service of a scene generation system driven mainly by language. The following sections discuss work on several types of systems that were created to support 3D scene design, including declarative scene modeling, language-based systems, automatic scene layout systems, and finally some general supporting work in learning spatial knowledge priors.

2.5.1 Declarative scene modeling

One line of work in scene design systems focuses on using declarative modeling as the main mechanism. A thorough description of declarative scene modeling systems is given by a variety of prior work (Colin et al., 1998; Gaildrat, 2007; Plemenos and Miaoulis, 2009). At the highest level, declarative modeling generates scenes by maintaining a set of given constraints.

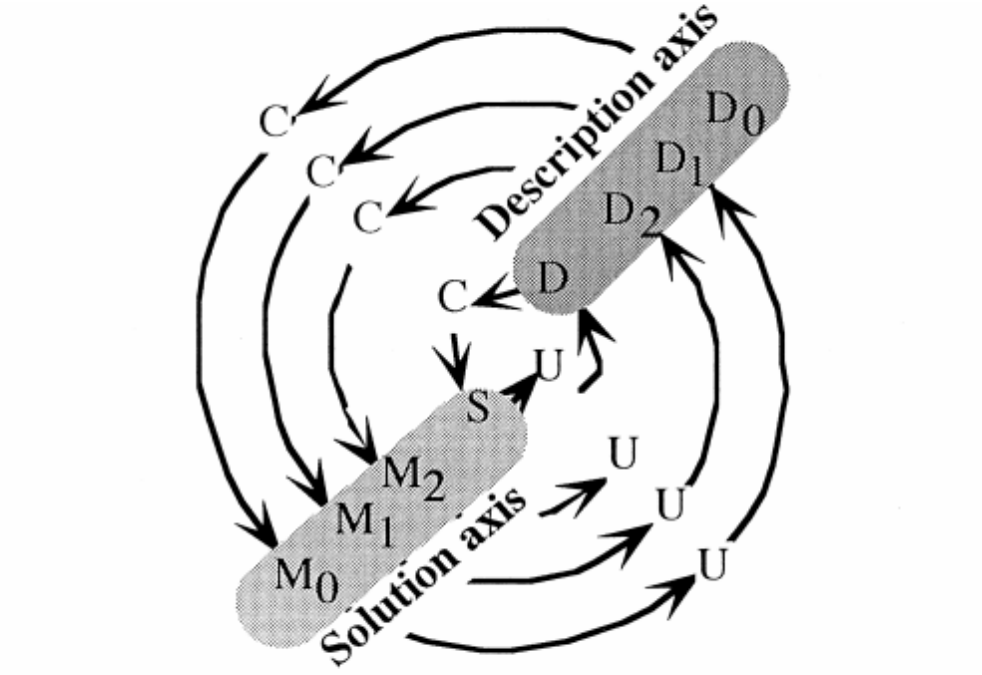
The scene design process is split into three phases: description, generation, and understanding. In the description phase, the designer provides a broad description of the scene, typically in the form of constraints. Given the description, the computer explores the search space and generates scenes that match the provided description (and provide a partial solution to the constraints). During the understanding phase, the scenes are presented to the designer, and the designer judges how well each scene matches what they had in mind. Depending on whether the designer is satisfied or not, they can accept a presented scene as the final solution, or provide more constraints to refine the solution. This iterative process is visualized in Figure 2.10.

Relationship to text to scene: typically, declarative systems for scene design do not use natural language, but instead a domain specific design language (Desmontils, 2000; Ruchaud and Plemenos, 2002; Zaragoza Rios, 2009). In some systems, constraints can also be provided through a GUI. In this way, scene generation can be viewed as a constraint satisfaction problem (CSP) and can be solved in a variety of ways (Le Roux et al., 2004).

We can think of a text to scene system as a declarative modeling system that uses natural language as input. In § 2.5.2, we describe some early systems that used natural language for interactive manipulation of a virtual scene.

2.5.2 Natural language for interactive scene design

Early work on the SHRDLU system (Winograd, 1972) gives a good formalization of the linguistic manipulation of objects in 3D scenes. By restricting the discourse domain to a micro-world with simple geometric shapes, the SHRDLU system demonstrated parsing of natural language input for manipulating scenes. However, generalization to more complex objects and spatial relations is still very hard to attain. Another similar early system was Put by Clay and Wilhelms (1996) which also handled simple geometric primitives in constrained scene scenarios. The challenge of using language to interact with realistic environments populated by objects occurring naturally in the real-world remains largely unexplored. This thesis will discuss an approach to parsing natural language as interaction commands within the context of a broader text to scene generation system in Chapter 7.



D_i	:	Description
M_i	:	Solutions or partial model according to D_i
C	:	Computation
U	:	Understanding
D	:	Description of the final solution set
S	:	Final solution set

Figure 2.10: Declarative scene modeling process (Colin et al., 1998).

2.5.3 Automatic scene layout

To generate scenes from text, I leverage recent advances in graphics for automatic scene layout. Prior work on scene layout has focused on determining good furniture layouts by optimizing energy functions that capture the quality of a proposed layout. These energy functions are encoded from design guidelines (Merrell et al., 2011) or learned from scene data (Fisher et al., 2012). Knowledge of object co-occurrences and spatial relations is represented by simple models such as mixtures of Gaussians on pairwise object positions and orientations. We leverage several ideas from this body of prior work, noting however that this work has not focused on linking spatial knowledge to language.

Xu et al. (2002) presents a system which uses the constraint-based formulation found in declarative scene modeling for the task of automatic scene layout. This makes it much easier to create scene layouts with precise deterministic relations and symmetries compared to the approaches using optimization and sampling.

Regardless, most work in scene layout relies on partial specifications of desired scene structure through either energy functions or constraints motivated by specific rules or common guidelines, avoiding a more holistic model for the structure of 3D scenes. An exception is the contextual maximum entropy-based framework of Dema and Sari-Sarraf (2014) which was used for data-driven 3D scene generation.

2.5.4 Spatial knowledge priors

Spatial knowledge priors are a critical component for text to scene generation systems that can handle the underspecified descriptions of scenes given in natural language. Much prior work has focused on the construction and expansion of ontologies and knowledge bases such as Cyc (Lenat, 1995), WordNet (Miller, 1995), OpenMind common sense (Singh et al., 2002) which was the basis for ConceptNet (Liu and Singh, 2004; Speer and Havasi, 2013), and others. These ontologies contain definitions for entities, and relational predicates between them. However, spatial knowledge is not well represented in ungrounded predicate form, and is not well covered by existing ontologies.

More recently, there is a resurgence of interest in common sense knowledge for vision and robotics that focuses more on grounded knowledge bases. Work from vision

researchers has demonstrated the usefulness of visual common sense for visual question answering (Zhu et al., 2014; Zhu et al., 2015a), identifying actions, and generating more accurate image captions (Vedantam et al., 2015; Aditya et al., 2015). There is also recent effort in robotics to collect common sense knowledge for indoor environments (Gupta and Kochenderfer, 2004; Saxena et al., 2015). For robots to operate in the world, it is necessary for them to have a sense of how objects should be positioned. To this end, the robotics community is starting to explore the learning of spatial priors either directly from 3D scenes, or by hypothesizing interactions that people can perform with the objects that are present (Rosman and Ramamoorthy, 2011; Jiang et al., 2012; Jiang and Saxena, 2013).

Follow up work by the authors of WordsEye has attempted to collect annotations from crowds of people and encode them into predefined rules to be used in the semantic frames of WordsEye (Coyne et al., 2010; Coyne et al., 2012). Some work has been done in extracting numerical attributes for objects such as heights and weights from web text (Davidov and Rappoport, 2010). Despite these efforts, there is still comparatively little work on automatic extraction of spatial knowledge since it is typically implied in natural language and challenging to extract from text.

In this thesis we will automatically learn and leverage spatial knowledge priors for the task of text to scene generation. A more thorough description of how these priors are learned and used in our model is provided in Chapter 4.

Chapter 3

Problem formulation

In this chapter, I formalize the problem of text to scene generation and introduce the representation and notation that will be used for the rest of the dissertation. I propose a probabilistic formulation of the problem and a pipeline system for text to scene generation that incorporates spatial knowledge priors.

3.1 Problem statement

I define the text to scene generation problem as follows: given an utterance u as input, the output is a scene s where s is an arrangement of 3D models representing objects at specified positions and orientations in space. More concretely $s = (m_i, T_i)$ is a set of pairs of 3D model m_i and 3D transformation matrix T_i placing each model within a scene coordinate frame to visually represent the scene. To generate scene s, objects are selected from a database of 3D models M and arranged in likely configurations based on the constraints imposed by the input text u.

To represent the constraints expressed by utterance u, I use a graph based semantic representation that I will call a *scene template*. A scene template represents the objects to be visualized as nodes, and constraints between the objects as edges (see § 3.3.1 for more details). The scene template is a template from which many concrete geometric scenes can be instantiated.

In this thesis, I will focus on a fairly literal domain of utterances describing common

Figure 3.1: Generated scene for “There is a sandwich on a plate.” Note that the room, table and other objects are not explicitly stated, but are inferred by the system.

everyday room interiors. For example, the utterance "there is a computer in a living room" is within this scope. We do not handle style or more abstract concepts which require a better understanding of pragmatics (e.g., the utterance "I want a messy child's bedroom"). Though the presented model is a first step towards inferring unstated, implicit facts, it does not perform iterative Bayesian pragmatic reasoning about the communicative goals of the speaker.

One of the main goals of this dissertation is to be able to handle underspecified natural language input. For instance, in Figure 3.1, the system is able to generate a full scene with inferred kitchen table and chair for the short sentence "there is a sandwich on a plate". To accomplish this goal, I break down the problem into the following subproblems:

Scene template parsing (§ 5.1): Parse the textual description of a scene u into a scene template t that represents the explicitly stated set of constraints on the objects present and spatial relations between them.

Scene inference (§ 5.2): Expand the literal scene template t into a complete scene template t' by accounting for implicit constraints not specified in the text using learned spatial priors.

Scene generation (§ 5.3): Given a completed scene template t with the constraints and priors on the spatial relations of objects, transform the scene template into a geometric 3D scene with a set of objects to be instantiated.

The subproblem of scene generation is further decomposed into:

Object selection (§ 5.3.1): Select a set of models from the model database to represent the objects in the scene. For this we need grounding of text to objects (see § 6.2).

Scene layout (§ 5.3.2): Arrange the objects and optimize their placement based on priors on the relative positions of objects and explicitly provided spatial constraints. For this we need grounding of text to spatial constraints (see § 6.1).

Each of the subproblems above requires specific inputs which we mention briefly here and discuss in more detail in the relevant sections. In order to carry out the object selection and scene layout steps, our database of 3D models is annotated with semantic information such as the category, size, and natural orientation (as described in § 4.1). In addition to the database of 3D models, a knowledge base of spatial priors is required for the earlier steps. The spatial priors are used to infer implicit unstated constraints and likely positions and

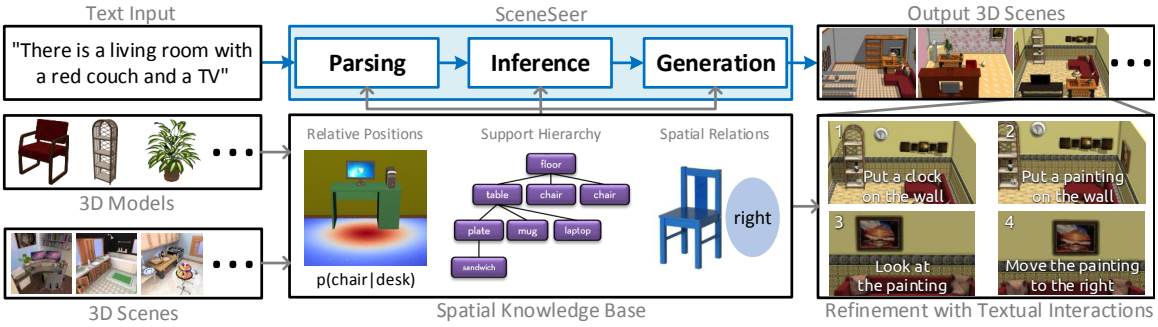


Figure 3.2: Illustration of the our system architecture. Input text is parsed into a scene representation and expanded through inference using knowledge learned from a corpus of 3D scenes and 3D models. The representation is then used to generate a 3D scene that can be rendered and manipulated iteratively through textual commands.

orientations for the objects. These spatial priors are learned from a dataset of 3D scenes (see § 3.5.2). Conceptually, it is also possible to learn spatial priors from real world observations such as scans from RGB-D sensors. The choice of 3D scenes versus alternative sources for spatial priors is discussed more in § 4.2.

3.2 System architecture

The components of our system map directly to the subproblems that we have identified above. Figure 3.2 illustrates the architecture of our system as a whole. Explicitly defining a modular system architecture in this way is beneficial for having well-defined, modular components that address the conceptually distinct subproblems we identified earlier. In the first transition from the left side of the figure to the middle we perform scene template parsing of the input text, leveraging a spatial knowledge base that was constructed from the 3D model and 3D scene corpora. Once an initial scene template has been parsed, we again use the spatial knowledge base in the scene inference stage to expand the template. We then generate a geometric 3D scene given the expanded scene template, and allow a user to interactively view and refine the scene using textual commands (again requiring the use of the spatial knowledge base for grounding and disambiguation). Below we provide a more detailed overview of each of the main components of the system architecture.

3.2.1 Scene template parsing

During scene template parsing we take the input text and create a scene template that identifies the objects and the relations between them. We first process the input text using the Stanford CoreNLP pipeline[1], identify visualizable objects, physical properties of the objects, and extract dependency patterns for spatial relations between the objects. The parsed text is then deterministically mapped to the scene template or scene operations. An interesting avenue for future research is to automatically learn how to map text using more advanced semantic parsing methods.

3.2.2 Scene inference

After we obtain a scene template with the explicitly stated constraints, we expand it to include inferred objects and implicit constraints. As an example, for each object in the scene, we use the support hierarchy prior $P_{support}$ to find the most likely supporting parent object category. If there is no such parent object already in the scene, we add it to the scene. For instance, given the objects "computer" and "room", we infer that there should also be a "desk" supporting the "computer".

In addition to inferring missing objects, we infer the static support hierarchy for the objects in the scene. For each object, we first identify support constraints that must exist based on the scene template. For any objects that do not have a supporting parent, we sample $P_{support}(C_p|C_c)$ restricted to the set of available object categories. In addition, we can infer missing objects based on the location (for instance, a bedroom typically contains a bed and a dresser). Future work would involve enhancing this component with improved inference by identifying what objects must exist for each location, instead of sampling a random selection of objects. We can also consider additional contextual details such as the distinction between a messy desk with a variety of items versus a tidy desk with few items.

The system parses this text to a set of explicitly provided constraints on what objects should be present, and how they are arranged. This set of constraints is automatically expanded using our prior knowledge so that "common sense" facts are reflected in the general scene.

[1] http://nlp.stanford.edu/software/corenlp.shtml

3.2.3 Scene generation and interaction

Once an expanded scene template is created using scene inference, we can proceed to scene generation. We first need to select 3D models as representatives for each object in the scene. In doing so we are addressing a text to 3D model grounding problem, as described in more detail in Section 6.2. When the set of 3D models has been selected, we then perform scene layout to determine the placement of each of the objects within the scene. This component is based primarily on prior work in automatic scene layout, as we discuss more fully in Section 5.3.2. After these two steps, the system generates a candidate geometric scene that the user is free to view and interactively manipulate either by direct control or through textual commands. The final output given by this stage is a 3D scene that can be viewed from any position and rendered by a graphics engine. When presenting the scene to a user, we select an initial viewpoint such that objects are in the frame and view-based spatial relations are satisfied.

3.3 Scene representation

There is a big gap between the representations typically used for 3D object and scene rendering, and the high-level semantics that people assign to scenes. Here we define a scene template representation to make explicit the information necessary for connecting higher-level semantics to the lower-level geometric representations. Since natural language usually expresses high-level semantics, we can view the text-to-scene task as a problem of first extracting a higher-level scene template representation and then mapping into a concrete geometric representation.

To capture the objects present and their arrangement, we represent scenes as graphs where nodes are objects in the scene, and edges are semantic relationships between the objects.

We represent the semantics of a scene using a *scene template* and the geometric properties using a *geometric scene*. One critical property which is captured by our scene graph representation is that of a static support hierarchy, i.e., the order in which bigger objects physically support smaller ones: the floor supports tables, which support plates, which can

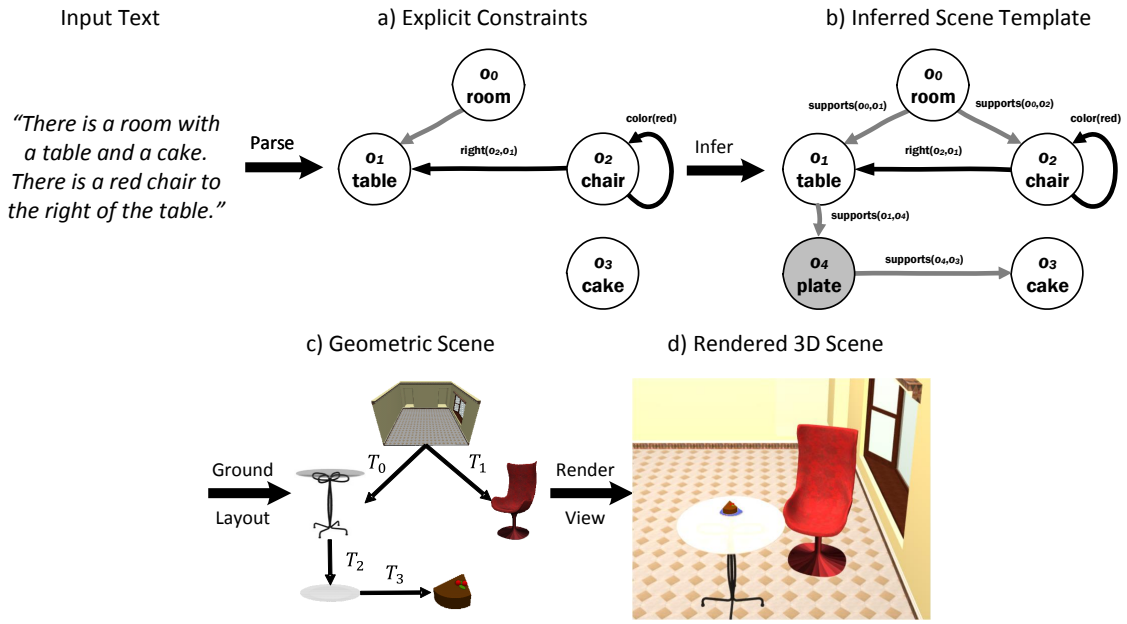


Figure 3.3: Overview of our spatial knowledge representation for text-to-3D scene generation. We parse input text into a scene template and infer implicit spatial constraints from learned priors. We then ground the template to a geometric scene, choose 3D models to instantiate and arrange them into a final 3D scene.

support cakes. Static support and other constraints on relationships between objects are represented as edges in the scene graph.

3.3.1 Scene template

A scene template $\mathcal{T} = (C_s, \mathcal{O}, \mathcal{C})$ consists of a scene type C_s, along with a set of object descriptions $\mathcal{O} = \{o_1, \ldots, o_n\}$ and constraints $\mathcal{C} = \{c_1, \ldots, c_k\}$ on the relationships between the objects. This provides a high-level yet unambiguous representation of scene structure.

The scene type C_s is used during inference (see § 5.2) to help determine what additional, unstated objects should be in the scene. Each object o_i, has properties associated with it such as category label, basic attributes such as color and material, and number of occurrences in the scene. For constraints, we focus on spatial relations between objects, expressed as predicates of the form *left*(o_i, o_j) or *supported_by*(o_i, o_j) where o_i and o_j are recognized objects.[2] Figure 3.3a shows an example scene template. This is a fairly simple, easy to understand representation that captures important information about the structure of the scene. Due to its simplicity, it is also easy to establish a direct correspondence to a rendered scene.

3.3.2 Scene tree

We want to use the scene template as a source of constraints to guide the generation of a 3D scene. To do so, we flesh out the scene template by inferring additional objects and constraints. To infer implicit constraints on objects and spatial support we learn priors on object occurrences in 3D scenes (§ 4.2.1) and their support hierarchies (§ 4.2.2).

We further transform the scene template into a *scene tree* (using the *supported_by* and *subgroup_of* relationships to form a backbone of the scene tree). The scene tree allows us to have a hierarchical traversal of the scene so we can more easily perform scene layout. Using the scene tree we instantiate concrete geometric 3D scenes. We will describe how we extract the scene tree from the static support relations and priors in 3D scenes in Section 5.2 and use it perform scene layout in Section 5.3.2.

[2] Our representation can also support other relationships such as $larger(o_i, o_j)$.

3.3.3 Geometric scene

We refer to the concrete geometric representation of a scene as a *geometric scene*. A geometric scene s consists of a set of 3D model instances $\{i_1, \ldots, i_n\}$ where each model instance $i_j = (m_j, T_j)$ is represented by model m_j in the model database and the transformation matrix T_j. The model represents the physical appearance (geometry and texture) of the object while the transformation matrix encodes the position, orientation, and scaling of the object. The transformation matrix is typically represented as a 4×4 matrix. The geometric scene corresponds to a traditional scene graph representation in graphics.

Working directly with such low-level representations of scenes is unnatural for people, which is a factor in the difficulty of learning current 3D scene design interfaces. We generate a geometric scene from a scene template by selecting appropriate models from a 3D model database and determining transformations that optimize their layout to satisfy spatial constraints. To inform geometric arrangement we learn priors on the types of support surfaces (§ 4.2.2) and the relative positions of objects (§ 4.2.4).

3.3.4 Relationship to standard representations

Graphics systems often define a scene graph for the purposes of rendering and manipulating a scene composed of a hierarchy of simpler primitives. This form of rendering scene graph is primarily used for specifying hierarchical groupings and positioning of primitives such that they can more easily be rendered and manipulated. Therefore, the traditional scene graph representation is only implicitly concerned with the semantics of scene structure, depending on the degree to which the designer of the scene has decided to encode a semantically meaningful hierarchy into the scene graph. Tobler (2011) describes the need to separate the semantic notion of a scene graph from the more typical rendering scene graph used in graphics pipelines. This indicates that it is useful to more precisely define distinct representations and what they are intended to capture. This thesis uses the following set of representations:

- The *scene template* captures the objects and constraints implied by the utterance. This is similar to the semantic scene graph used by Johnson et al. (2015) for image retrieval. The term template is appropriate because it specifies the semantics of

entities and relations which in general can be satisfied by a broad set of concrete geometric scenes, each of which can said to be a sample or interpretation of the scene template.

- The *scene tree* establishes a traversal order for the scene and captures important dependencies between objects. It is particularly useful when performing scene layout since an ordering according to salient relations makes layout optimization much more efficient.

- The *geometric scene* captures the low-level appearance of the scene (geometry modeling of objects and transforms of position, scale, and orientation). This corresponds to the traditional rendering scene graph in graphics and is the representation that fully specifies the intended appearance of the scene.

We have focused on contrasting our representations with traditional rendering scene graphs. There are many other representations used in graphics and vision that can be used to represent real-world scenes including point clouds, dense voxel grids specifying volume occupancy, and implicit functions defining the surface boundaries of shapes. Each of these representations is advantageous to use in particular scenarios, though the scene graph of triangle mesh 3D models is common, compact, and used pervasively in real-time graphics engines.

3.4 Model formulation

We take a probabilistic view and model the task of generating scene s from an input utterance u as that of estimating the distribution of possible scenes $P(s|u)$ given the input utterance u. This allows us to incorporate prior knowledge to perform inference, and to leverage recent advances in machine learning for learning from data. To generate plausible scenes to present to the user, we sample from the distribution $P(s|u)$.

We decompose $P(s|u)$ into:

$$P(s|u) = \sum_{t,t'} P(s,t,t'|u) = \sum_{t,t'} P(t|u)P(t'|t,u)P(s|t',t,u)$$

where $P(t|u)$ is the probability of a scene template t given a utterance u, and t' is a completed scene template. In our system, we will assume that s is independent of t, u and that t' is independent of u to get

$$P(s|u) = \sum_{t,t'} P(t|u)P(t'|t)P(s|t')$$

To generate a scene given an utterance, we can then sample from $P(s|u)$ by sampling from each of the component distributions. To generate more likely scenes, we will select $t^* = \arg\max P(t|u)$ and $t'^* = \arg\max P(t'|t^*)$ and sample $P(s|t'^*)$. In our framework, **scene template parsing** corresponds to estimating $P(t|u)$ and picking the most likely scene template $t^* = \arg\max P(t|u)$, **scene inference** to estimating $P(t'|t)$ and selection of $t'^* = \arg\max P(t'|t^*)$, and **scene generation** to estimating and sampling from $P(s|t')$.

In this dissertation, I present a pipelined system. Future work can consider a more sophisticated system where $P(s|u)$ is jointly estimated and sampled. We note here that a deterministic parsing model represents $P(t|u)$ as a delta probability at the selected t.

Since the scene s is represented as a set of models $\{m_j\}$ and their transforms $\{T_j\}$, scene generation can be decomposed into two parts: **object selection** and **scene layout**. Object selection consists of identifying a likely set of models $\{m_j\}$ from the model database given a complete scene template t (which corresponds to teh t' from above). Scene layout is the problem of arranging the objects and finding appropriate transforms $\{T_j\}$ for the models $\{m_j\}$ given the constraints expressed by the scene template t.

$$P(s|t) = P(\{m_j, T_j\}|t) = P(\{m_j\}|t)P(\{m_j, T_j\}|\{m_j\}, t)$$

Again, a more sophisticated system may choose to estimate $P(s|t)$ jointly. Note that in the equation above, $\{m_j\}$ denotes the set of models in the scene and $\{m_j, T_j\}$ denotes the set of model transformation matrix tuples (m_j, T_j). To further simplify the problem, we assume that we can select the model for each object independently. Thus, we have

$$P(\{m_j\}|t) = \prod_j P(m_j|o_j)$$

This assumption clearly does not hold for many scenes and scene templates (e.g., matching furniture sets).

For scene layout, we estimate $P(\{m_j, T_j\}|\{m_j\}, t)$ using a layout score $\mathcal{L}$ described in more detail in § 5.3.2.

To support **scene interactions**, we follow a similar model, except now we are given a starting scene s and an utterance u, and we want to determine the desired scene s'. We can model this as estimating $P(s'|s, u)$. For simplicity, we will assume that the utterance consist of one scene operation O that takes s and transforms it to s'. Thus we can break down $P(s'|s, u) = P(s'|O, s, u)P(O|s, u)$. Here, given a specific starting scene s and the utterance u expressing a desired operation, we want to find the operation O that can be applied on s to yield s'. This problem can be decomposed into parsing, inference and generation steps that we use for regular scene generation as well. The difference is that we now estimate an operation O.

Now that we have defined a model for our problem, we will consider the input data and its properties.

3.5 Data

3.5.1 Model database

Unfortunately, most commonly available 3D representations of objects and scenes used in computer graphics specify only the geometry and appearance of objects, and rarely include high-level semantic information. Prior work in text to 3D scene generation has focused on collecting manual annotations of object properties and relations (Rouhizadeh et al., 2011; Coyne et al., 2012), which are then used to drive rule-based generation systems.

Though prediction and automatic annotation of high-level semantics for raw 3D model data is an interesting problem, it is a concern within the graphics community dealing with geometric analysis, and is beyond the focus of this thesis. In order to enable the our system to ground references to specific 3D models, we take a corpus of 3D models used by prior work in 3D scene synthesis and augment it with semantic attributes (Fisher et al., 2012).

This effort is part of a larger umbrella project (ShapeNet) which aims to construct a large-scale corpus of semantically annotated 3D models. In § 4.1, I will describe in more detail the semantic annotations we need for text to scene generation.

The model database contains roughly 12 thousand models belonging to more than 200 categories which we correspond to the WordNet taxonomy. Most of the models belong to categories of common indoor objects such as chairs, tables, lamps, beds, TVs, etc. This bias in the distribution over object categories is to be expected for 3D models downloaded from public web repositories, as these classes of objects are the easiest to model with existing modeling software such as SketchUp.

3.5.2 Scene database

The scene database that we use to learn priors on object occurrences and on spatial relations between objects is the Stanford 3D Scene Database which was constructed by Fisher et al. (2012) for their scene layout system. This scene database contains approximately 130 scenes of indoor environments such as bedrooms, offices, living rooms, and kitchens. Though these scenes are restricted in scope and complexity, this dataset is one of the largest scene corpora available for research. As a whole, the dataset contains 3461 model instances (corresponding to objects appearing in a scene), using 1723 distinct 3D models.

3.5.3 Scene and description corpus

We introduce a new dataset of 1128 scenes and 4284 free-form natural language descriptions of these scenes.[3] To create this training set, we used a simple online scene design interface that allows users to assemble scenes using available 3D models of common household objects (each model is annotated with a category label and has a unique ID). We used a set of 60 seed sentences (see Appendix A) describing simple configurations of interior scenes as prompts and asked workers on the Amazon Mechanical Turk crowdsourcing platform to create scenes corresponding to these seed descriptions. Figure 3.4 shows a screenshot of the interfaces the workers used to create scenes. To obtain more varied descriptions for each scene, we asked other workers to describe each scene. Figure 3.5 shows examples

[3] Available at `http://nlp.stanford.edu/data/text2scene.shtml`.

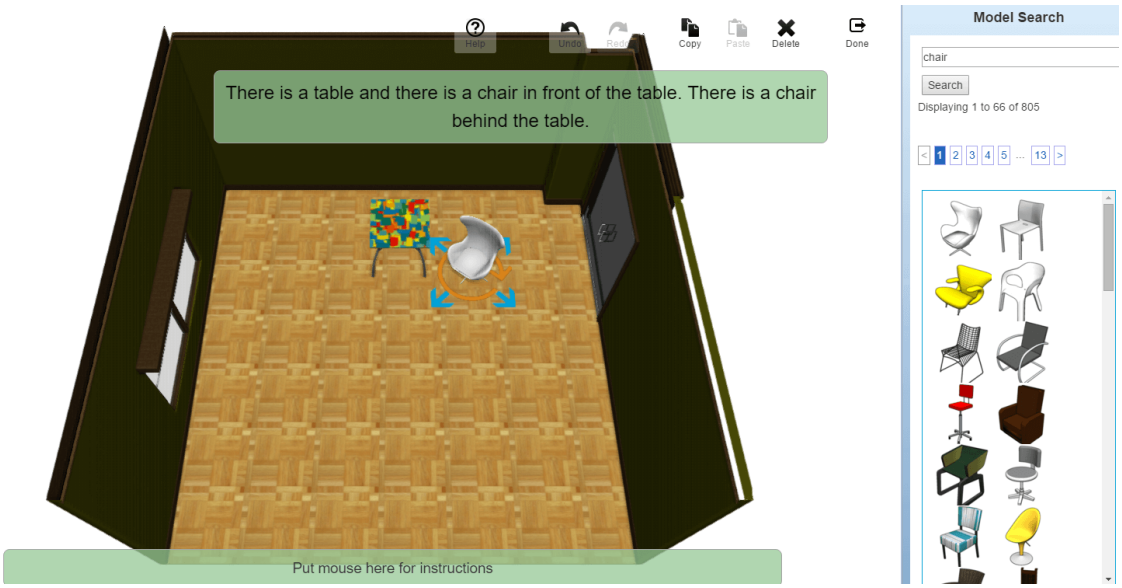


Figure 3.4: Online Scene editor UI used for creating description to scene corpus

Figure 3.5: Dataset of scenes with textual descriptions. Scenes created by participants from seed description sentences (**top**). Additional descriptions provided by other participants from the created scene (**bottom**). Our dataset contains around 19 scenes per seed sentence, for a total of 1129 scenes. Scenes exhibit variation in the specific objects chosen and their placement. Each scene is described by 3 or 4 other people, for a total of 4358 descriptions.

of seed description sentences, 3D scenes created by people given those descriptions, and new descriptions provided by others viewing the created scenes.

We manually examined a random subset of the descriptions (approximately 10%) to eliminate spam and unacceptably poor descriptions. When we identified an unacceptable description, we also examined all other descriptions by the same worker, as most poor descriptions came from a small number of workers. From our sample, we estimate that less than 3% of descriptions were spam or unacceptably incoherent. To reflect natural use, we retained minor typographical and grammatical errors.

Despite the small set of seed sentences, the Turker-provided scenes exhibit much variety in the specific objects used and their placements within the scene. Over 600 distinct 3D models appear in at least one scene, and more than 40% of non-room objects are rotated from their default orientation, despite the fact that this requires an extra manipulation in the scene-building interface. The descriptions collected for these scenes are similarly diverse and usually differ substantially in length and content from the seed sentences.[4]

3.6 Conclusion

In this chapter we first formalized the text to 3D scene problem. We then described the overall architecture of the system presented in this thesis, defined the representation that we will use for scenes, and formulated a probabilistic model for the text to scene task that can be decomposed into subtasks. Finally, we discussed the various forms of input data that we will use for our system. The next chapter discusses the approach we will use to extract common sense priors on objects and spatial relationships from the available input data.

[4]Mean 26.2 words, SD 17.4; versus mean 16.6, SD 7.2 for the seed sentences. If one considers seed sentences to be the "reference," the macro-averaged BLEU score (Papineni et al., 2002) of the Turker descriptions is 12.0.

Chapter 4

Modeling prior knowledge

To generate scenes from concise natural text, we need to have a model of what the world is expected to look like. We learn prior knowledge about the occurrences of objects in the world and common spatial relationships between them from the available 3D scene data.

Why do we need to model prior knowledge? We need this prior knowledge to answer basic questions about the structure of the world which cannot be answered by just considering what is explicitly communicated in natural language. These basic questions revolve around the attributes and relations of physical objects in scenes. We separate these questions and corresponding priors into three types:

- *Object-level questions:* What is this object? What shape is it? How big is it? Does it have a front? Which side is the front?
- *Object in context questions:* Where is x typically found? (In a kitchen, or on a desk?) How is x typically positioned with respect to y?
- *Object support questions:* What part of the object is used for support, what part for attachment?

In this chapter we will specify the spatial knowledge required for text to scene generation and describe how we encode it in a set of priors. Most importantly, our goal is to be able to generalize beyond specific instances of objects in previously seen scenarios. For example, if we have seen lamp A positioned on desk B, we want to be able to create other

scenes with lamp C on desk D or lamp E on the floor. While this form of generalization is challenging in general, and we certainly don't handle all cases robustly, this is still our goal.

In order to make progress towards this goal we annotate our model database through a combination of semi-automatic and manual methods. Given these annotated models, we can then leverage observations within 3D scenes composed of the models to learn the set of desired spatial priors at a level that generalizes across model instances. Generalization beyond the instance level for most of the 3D model knowledge and scene layout problems that we consider is a big open research problem in graphics and a bottleneck for automating 3D content creation. As an example from prior work, WordsEye used a model database that was spatially tagged for each model instance. The model database is proprietary so it cannot be shared as a resource, making the annotation effort not very useful for future work. In contrast, we use open-sourced models from the public 3D repositories, and specify our annotations on 3D models in a shareable, open format. This effort is part of a larger collaborative effort under the ShapeNet project, which aims to annotate a large corpus of 3D models with rich semantics[1].

Key to our representation is the notion of categories. We assume that both objects and scenes are categorized within a taxonomy. Work in psychology has shown that basic-level categories are a fundamental cognitive model used by people that derives from a desire to efficiently capture and communicate prevalent correlations in the attributes that objects possess in the real world (Rosch and Lloyd, 1978).

Categories deeply influence perception and communication. When used in natural language, categories carve out the space of things we interact with into discrete sets. Categories can also organize the world in a taxonomy of *is-a* relations that give increasingly abstracted and generalized views of the world at higher levels. These category taxonomies are constructed by people and also more rigorously defined in academia, but in both scenarios they allow inferences about the nature of particular entities under consideration. In other words, category taxonomies are useful for generalizing beyond a single instance and for establishing priors for unobserved or less observed entities.

The prototype theory developed in cognitive science, is based on the fact that categories

[1] www.shapenet.org

possess gradated, continuous membership (Rosch and Mervis, 1975). Though the concept of a category is a complex phenomenon that is still the subject of much research, for the purposes of this thesis we will consider categories to be discrete, semantically meaningful grouping of entities (primarily partitioning physical objects and scenes into distinct sets). A more continuous category membership can be captured with feature vectors (representing geometric or image features), or with dense representations of categories. How that can be used for text to scene generation is beyond the scope of this thesis. In our context, we will use categories primarily to aggregate priors and encapsulate attributes such as the shape, appearance and functionality of objects.

The following sections describe the semantic attributes we need to have for our 3D models, and the spatial knowledge that we collect from our 3D scene data. The split between these two kinds of knowledge makes the contrast between object-specific semantic attributes, and contextual spatial knowledge (how an object relates to other objects in the context of a scene).

4.1 Semantic attributes

In this section we discuss semantic attributes, i.e., the annotations that we assign to the models in our 3D model database. Most of the attributes that we describe here have been independently addressed in prior work (e.g., orientation information, basic category labels, absolute size information). WordsEye in particular has collected even more complex information such as canonical grips and poses for each object. Our focus is not in presenting each of this attributes as a novel contribution but rather in showing how many of them can be learned from data through automated approaches, which was not the case in prior work. In some cases we still use manual annotations since many of the associated problems are individually challenging areas of research, instead focusing on leveraging the attributes in conjunction to enable text to 3D scene generation.

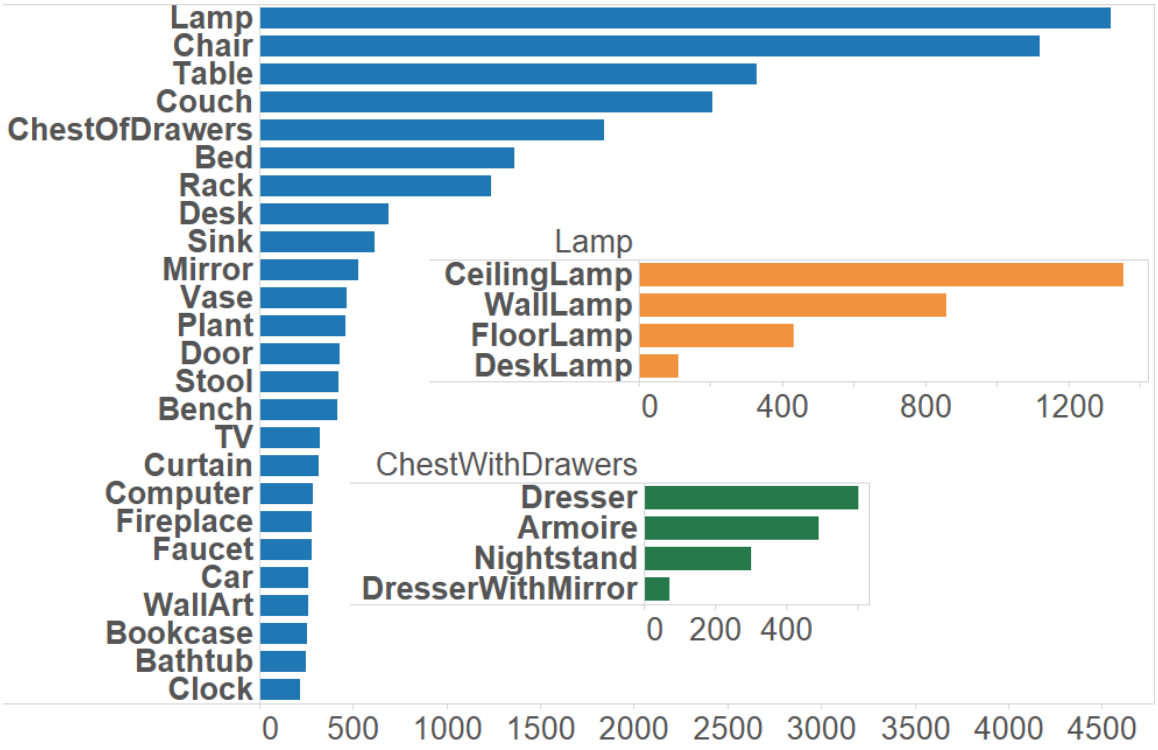


Figure 4.1: Distribution of 3D models in our corpus over categories at different taxonomy levels Inner distributions are over lamp and drawer furniture categories respectively. Our dataset is based on a 3D scene synthesis dataset from prior work (Fisher et al., 2012) and consists of 12000 object models in total over about 200 basic categories.

4.1.1 Category taxonomy

We define a manual taxonomy of categories for our dataset of 3D models. Since we focus on indoor scene design, our taxonomy mainly consists of furniture, common household objects, and electronics. Using a taxonomy is important, as it allows for generalization from fine-to-coarse grained categories (see Figure 4.1). We break up basic categories into subcategories mainly by geometric variation and functionality. For example, the *lamp* basic category is subcategorized into *table lamp*, *desk lamp*, *floor lamp*, *wall lamp*, and *ceiling lamp*. The key distinction is the typical location and the type of static support surface for the lamp. For the contrast between table and desk lamps the difference is between radially symmetric and focused spotlights for desk tasks.

Nodes in our taxonomy correspond to synsets in WordNet (Miller, 1995). WordNet is a taxonomy over concepts, where each concept is identified by a set of words or phrases ("synonym set" or "synset") that map to the same concept or sense. Mapping to an established, well constructed hierarchy of concepts such as WordNet offers the big advantage of establishing a hierarchical categorization and connecting the categorized models to useful semantic information such as typical part names and descriptions of object function. Furthermore, WordNet establishes links to other knowledge bases and ontologies such as ImageNet, ConceptNet, Freebase, and Wikipedia with additional information.

However, the taxonomy used by WordNet is not always ideal for categorizing physical objects. Since WordNet is a hierarchy over all English words and their senses, it includes many abstract concepts (e.g., fairness and truth), and extremely fine grained senses (some of which are not relevant for 3D models, e.g., synsets for "ball", "table"). Moreover, despite the fine granularity of senses for some words, the WordNet taxonomy does not have finer subcategorization based on shape and lacks coverage for modern products and electronics (e.g., USB sticks, video game consoles).

In addition to an object category taxonomy, we also employ a scene category taxonomy (see Appendix B) where simple categories such as "living room", "kitchen", and "bedroom" are given to our 3D scenes. This scene category is used as a conditioning scene type variable for establishing different priors for different contexts, as we will describe in § 4.2. Since both scene and 3D model classification are challenging and open research

Figure 4.2: Examples of three basic shape types: 1D (thin), 2D (flat), 3D

problems, we manually assign categories to our models and scenes. There is much prior work in 3D model retrieval and categorization that could be leveraged to automatically assign categories to 3D models—a recent survey is given by (Li et al., 2014).

4.1.2 Basic shape types

In order to estimate an overall shape type for objects we use a simple geometric processing approach. We encode the vertex positions of the surface mesh representation of a given 3D model into a $3 \times n$ matrix where n is the number of vertices. We then perform a principal components analysis (PCA) of the matrix and obtain the eigenvalues of the given linear system. The eigenvalues roughly represent the overall variation along each axis. Objects where the largest eigenvalue is more than ten times larger than the second eigenvalue, are annotated as thin (i.e., roughly 1D) objects. Typically, objects such as pens and flag poles will belong to this category. If the second eigenvalue is more than a tenth of the first eigenvalue but the third eigenvalue is less than a tenth of the second one, we classify the object as flat (2D). Examples of this type of object would be notebooks, posters, and carpets. The remaining objects are classified as 3D objects. See Figure 4.2 for examples of each of these basic shape types.

These three basic shape types allow us to define a fall back method when deciding how to place objects in categories that have sparse observations. Section 4.2.3 discusses how we use the basic shape to determine which side of the object should be the attachment side (e.g., how to put a piece of paper on a table). Furthermore, these basic shape types allow us to perform an easy first-pass automatic classification of model instances. Of course, a simple

approach like this has many error cases such as confusions between pencils and floor lamps. WordsEye identifies even more shape types such as bowl-shaped, long and thin, ring, etc. However, a better approach would use prior work in shape classification to automatically group the 3D models into finer classes—a recent example of such an approach is given by Tabia and Vu (2014).

4.1.3 Absolute sizes

Another critical attribute of objects is their physical size. Unfortunately, most commonly available 3D model datasets have incorrect or missing physical scale information. Prior work has looked at propagating priors on 3D model physical sizes through observations of the models in scenes, and predicting the size for new model instances (Savva et al., 2014a).

This approach first acquires category-level priors by crawling product catalog websites for furniture dimensions. To mitigate issues with incosistent alignment between 3D models and the quoted dimensions, it uses the diagonal size as a relatively stable measure. The authors found that in general the distinction between width, height and length is not made consistently across instances or product catalogs and therefore restricted their data collection to two furniture websites that were relatively reliable. We use this approach to establish category-level size priors for all models observed within our 3D scene corpus and then propagate these priors to all models within each category. See Figure 4.3 for an illustration of a consistently sized set of 3D models.

Sizes are particularly important for text to scene generation since they are one of the most salient properties of physical objects in the context of a scene. Figure 4.4 shows the contrast between using 3D models in their original virtual units vs using models rescaled with the above approach. Even though this approach uses category-level priors and is not in general accurate for specific instances, in the context of generating plausible 3D scenes, a rough size estimate is usually acceptable. A common failure case of this category-level size prior approach is that it ignores contextual cues and distinctions within a category that have a bearing on physical size (e.g., a toy airplane vs a real airplane).

Figure 4.3: Example models in our corpus spanning a range of physical sizes.

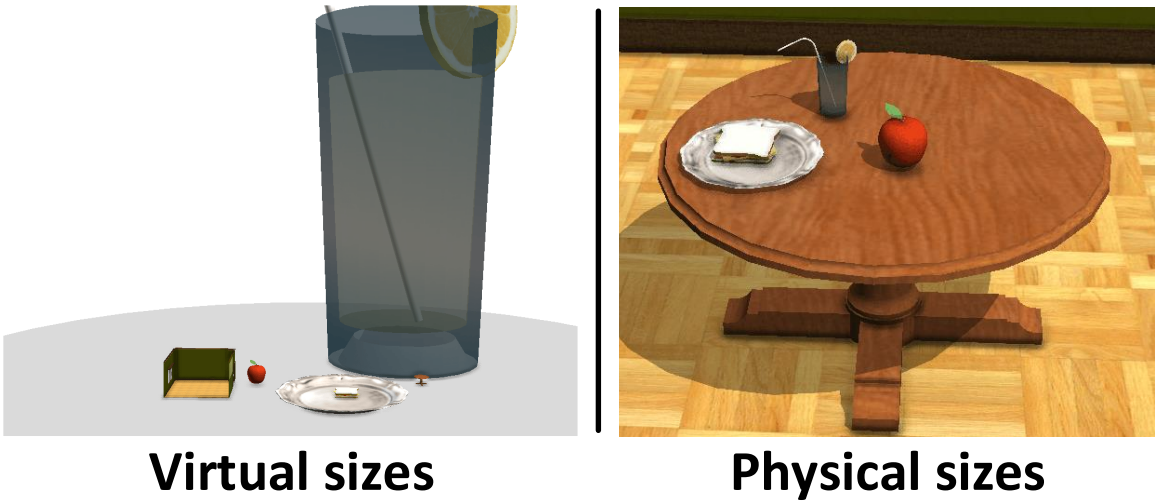


Figure 4.4: Usefulness of absolute size information for 3D models Without absolute size information, the original virtual dimensions of 3D models are often implausibly large or small (left). We resize all the 3D models used in our system to physically plausible sizes.

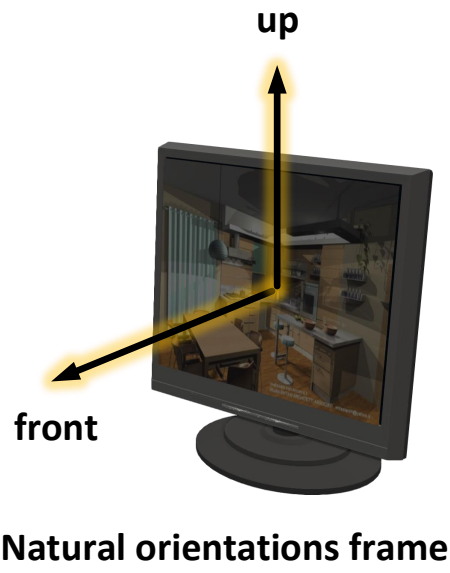


Figure 4.5: Illustration of natural orientations for objects. The semantic "up" and "front" directions in the local coordinate system of a model define a natural orientation frame for positioning that model.

Figure 4.6: Some examples of consistently oriented categories of models: chairs, monitors, desk lamps, and cars.

4.1.4 Natural orientations

Consistent alignments within each category of objects are extremely useful in a variety of applications. There has been some prior work in predicting the upright orientations of 3D models (Fu et al., 2008). However, since most models retrieved from web repositories already have a consistent upright orientation, we just manually verify each model. During this verification, we also specify a *front* side, in addition to the *upright* orientation, to provide a ground truth *natural orientation* for each object (see Figure 4.5). This specification of both up and front directions establishes a common reference frame for all 3D models (see Figure 4.6).

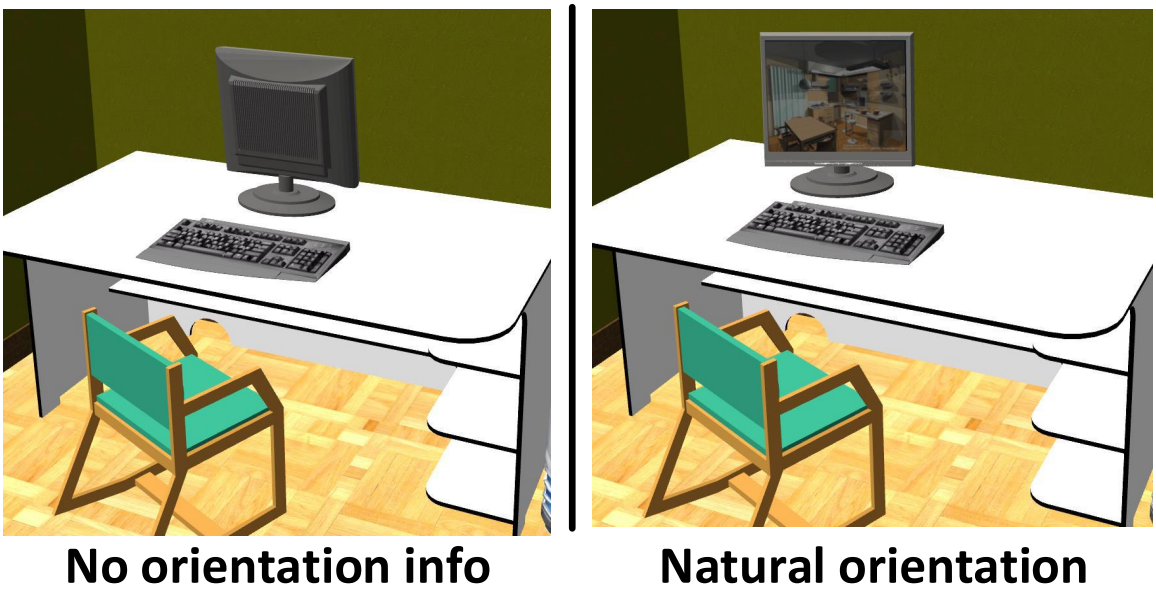


Figure 4.7: Usefulness of orientation information for 3D models. Without semantic orientations, it is hard to reason about how the monitor should be placed (left). When we know the semantic "front" side of the monitor we can place it such that it is facing in the same direction as the "front" of the desk.

Though most object categories have a common upright orientation, some categories may not have a well-defined front side (e.g., bowls, round tables). In these cases, the front side is assumed to be given by the original orientation in which the 3D model was designed. The presence of rotational symmetries can indicate such cases, so an interesting avenue for future work is to use geometric analysis to predict whether a semantic front exists for a given model and if it does, identify it.

Since this thesis does not make a contribution in automatic alignment of 3D models, we have manually verified the natural orientation of our 3D models. There is much prior work in computer graphics aiming to establish correspondences between 3D models and thus put models into alignment. A survey is provided by Van Kaick et al. (2011). In addition to these approaches, there is also a method to automatically align models based on the scene context within which they are observed (Fisher et al., 2012). However, such alignments don't necessarily correspond to the semantic notion of "front" or "up". A good compromise between manual effort and algorithmic complexity is to manually annotate the

semantic orientation of representative objects for each category and then automatically find correspondences between the representatives and other instances within the category to put them into a consistent alignment.

4.1.5 Discussion

In this section we have only discussed some of the most basic and relevant attributes of objects which we annotated for our 3D models. There are many other object level annotations that are useful for text to scene generation. For instance, object-level attributes such as materials, and colors can help us identify appropriate 3D models to select when the input text refers to these attributes (see § 5.3.1). In order to handle this additional information we currently only make use of textual tags and description associated with each model to extract relevant keywords conveying material, color, etc. This is a simple approach which allows us to refine our 3D model selection when the information is provided in the input text (see § 6.2).

Another class of object attributes that we do not address is defined at the object part level. Object part labels and part-level attributes are critical for allowing manipulation of the appearance of specific objects. For example, if we want to make a black office chair into a red office chair, information on what parts of the model are typically colored in black or red is important (we would likely only change the color of the seat and backrest, not the legs, or wheels). Another example where a part-level decomposition of models is important would be in modifying a desk to be "taller" or "wider". These shape deformation and material manipulation problems are challenging research questions in computer graphics which we do not address in this thesis.

4.2 Spatial knowledge

Our model of spatial knowledge relies on the idea of abstract scene types describing the occurrence and arrangement of different categories of objects within scenes of that type. For example, kitchens typically contain kitchen counters on which plates and cups are likely to be found. The type of the scene and categories of the objects present condition the

spatial relationships that can exist in the scene.

What do we mean by spatial knowledge? With this term we refer to the knowledge that is necessary to answer several types of basic questions:

- *Object occurrences:* What is typically found in an office?
- *Relative positions:* How is a chair typically positioned with respect to a desk in an office?
- *Support priors:* What is a chair typically placed on? What about a desk lamp? (Also, since we are considering 3D data: what surface of a desk can support a lamp? What part of a lamp is attached to the desk)

In WordsEye and prior work investigating "Smart Objects" (Kallmann and Thalmann, 1999), these forms of spatial knowledge are manually annotated. Our goal is to derive this information directly from data. We learn spatial knowledge from 3D scene data, basing our approach on that of Fisher et al. (2012) and using their dataset of 133 small indoor scenes created with 1723 Trimble 3D Warehouse models (Fisher et al., 2012). Though the original work also extracted spatial priors from this scene dataset, they did not attempt to condition it on scene type, did not connect it to language, and did not empirically evaluate any of the learned priors independently outside the very specific context of automatically synthesizing plausible 3D scenes. Furthermore, the original work does not attempt to generalize any of the priors beyond specific instances of observed support surfaces—i.e., books cannot be placed on desks that have not been previously observed to support books. In addition, we also focus on connecting the learned priors to language which enables us to perform language-based semantic queries as discussed in Section 7.3.

This 3D scene dataset includes a static support tree hierarchy for each scene (giving the larger parent object supporting each child object—e.g., desk supporting each of the books on it). We use this support hierarchy to extract child-parent pairs of statically supported and supporting objects. For each such pair, we identify the surfaces on the supporting parent by a proximity threshold to the midpoint of each bounding box face around the child object. Given an identified pair of parent support surface and child bounding box plane, we also retrieve the attachment surfaces of the child object that are within a small threshold ($1\,\mathrm{cm}$) of the attachment plane.

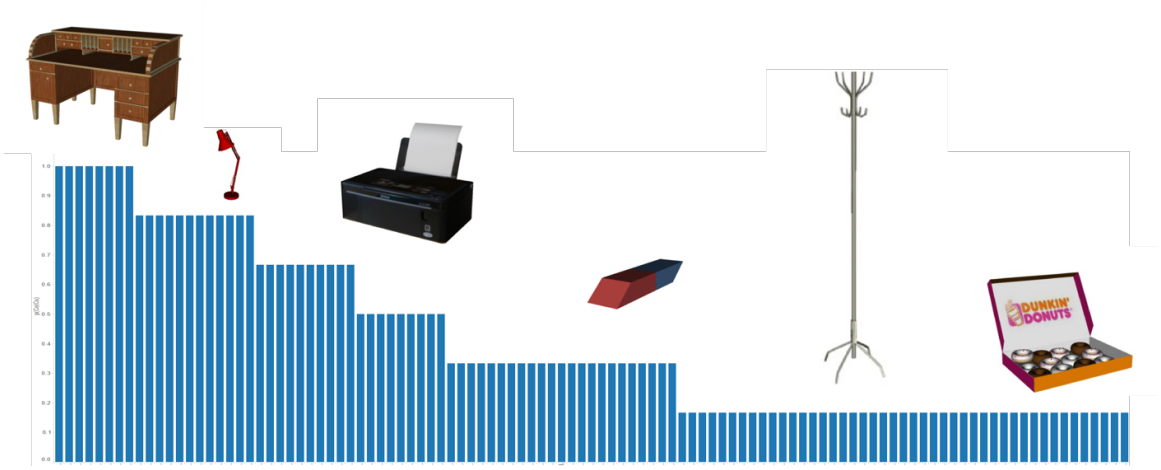


Figure 4.8: Object categories that are likely to be found in an office.

Spatial knowledge priors are estimated using observations in scenes. To handle data sparsity we utilize our category taxonomy. If there are fewer than $k = 5$ support observations we back off to a parent category in the taxonomy for more informative priors.

4.2.1 Object occurrence priors

We learn priors for the occurrence of an object with category C_o given a scene type C_s (e.g., kitchen, office, bedroom) by using a simple counting approach:

$$P_{occ}(C_o|C_s) = \frac{count(C_o \text{ in } C_s)}{count(C_s)}$$

Though this is a rather simplistic approach, it still captures meaningful priors on the likelihood of occurrence for objects in different scenes. Figure 4.8 shows an example of likely objects that can be found in an office.

Using Bayes' rule, we can evaluate the probability of different scene types given what objects are present:

$$P(C_s|\{C_o\}) \propto P_{occ}(\{C_o\}|C_s)P(C_s)$$

We can use this to infer the most likely scene type given a set of objects with given categories. Figure 4.9 shows the likely scene types given the presence of a knife and a

Figure 4.9: Probabilities of different scene types given the presence of "knife" and "table".

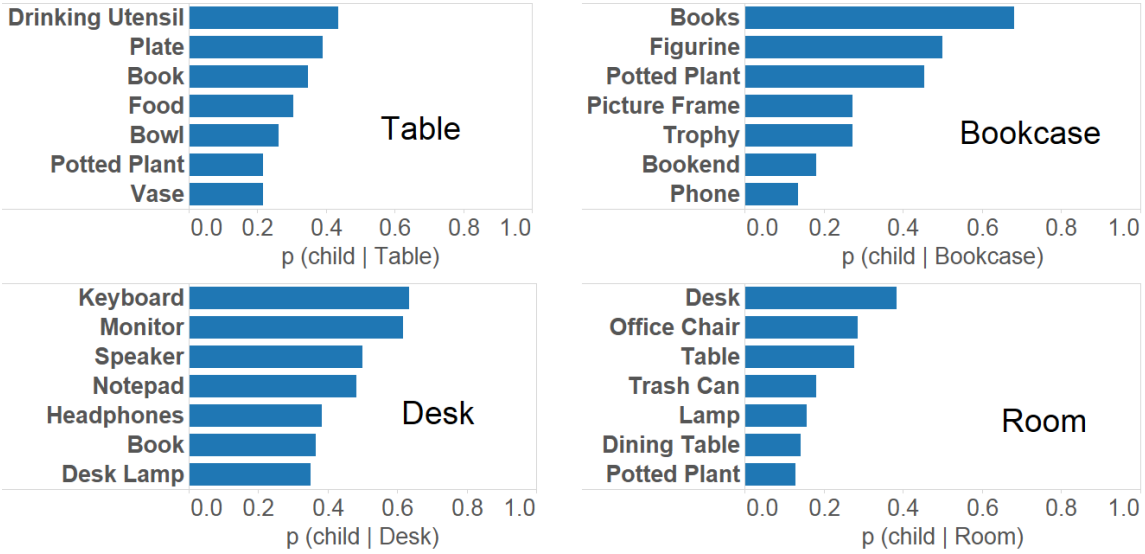


Figure 4.10: Probabilities for likely child object categories given four different parent support object categories From top left clockwise: dining table, bookcase, room, desk.

table. Now, given input of the form "there is a knife on the table" we are likely to generate a scene with a dining table and other related objects.

4.2.2 Support hierarchy priors

We observe the static support relations of objects in existing scenes to establish a prior over what objects go on top of what other objects. As an example, by observing plates and forks on tables most of the time, we establish that tables are more likely to support plates and forks than chairs. We estimate the probability of a parent category C_p supporting a given child category C_c as a simple conditional probability based on normalized observation counts[2].

$$P_{support}(C_p|C_c) = \frac{count(C_c \text{ on } C_p)}{count(C_c)}$$

[2]The support hierarchy is explicitly modeled in the scene dataset we use.

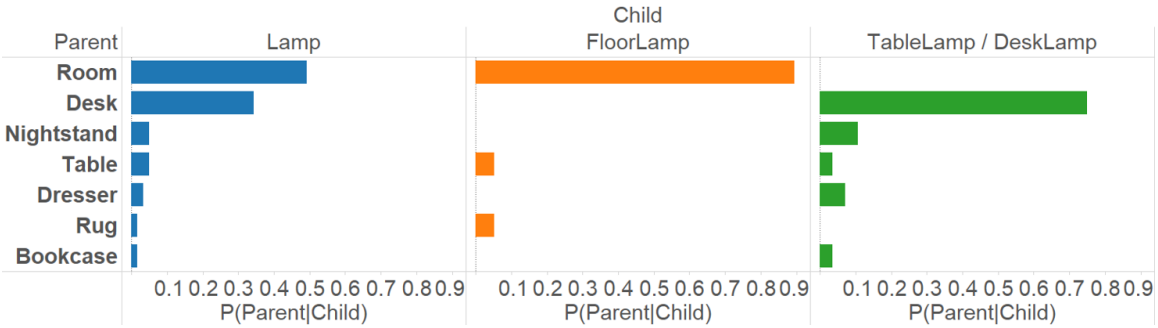


Figure 4.11: Probabilities of likely parent support object categories given different child object categories: lamp, floor lamp, table/desk lamp.

We show a few of the priors we learn in Figure 4.10 and Figure 4.11 as likelihoods of categories of child objects being statically supported by a parent category object.

Note that a floor lamp is more likely to be directly supported by the room, while a table or desk lamp is most likely to be supported by a desk. These probabilities can also be conditioned on the scene type by restricting the above count formulation to only consider observations within a given scene category.

4.2.3 Surface priors

Since we are working with scenes in full 3D, we need to precisely determine the positioning of objects on the surfaces of other objects. In other words, it is not enough to specify that a desk lamp is "on" a table, we need to specify the exact horizontal surface of the desk and position on that surface at which the lamp's base surface is attached. In order for this level of reasoning to be possible, models need to be decomposed into roughly planar surfaces. Furthermore, we need to identify likely attachment points and supporting surfaces for each 3D model. In the general case, this is a fairly challenging geometric analysis problem but for simplicity, we assume that objects are attached on one of the six sides of their bounding box (i.e., that attachment surfaces are planar axis aligned surfaces at the minimum or maximum extent of the object geometry).

The surfaces on which objects are statically supported determine many other object attributes, and critically the likely placements of objects within scenes. In order to establish a set of simple Bayesian priors for static support surfaces and object attachment points, we

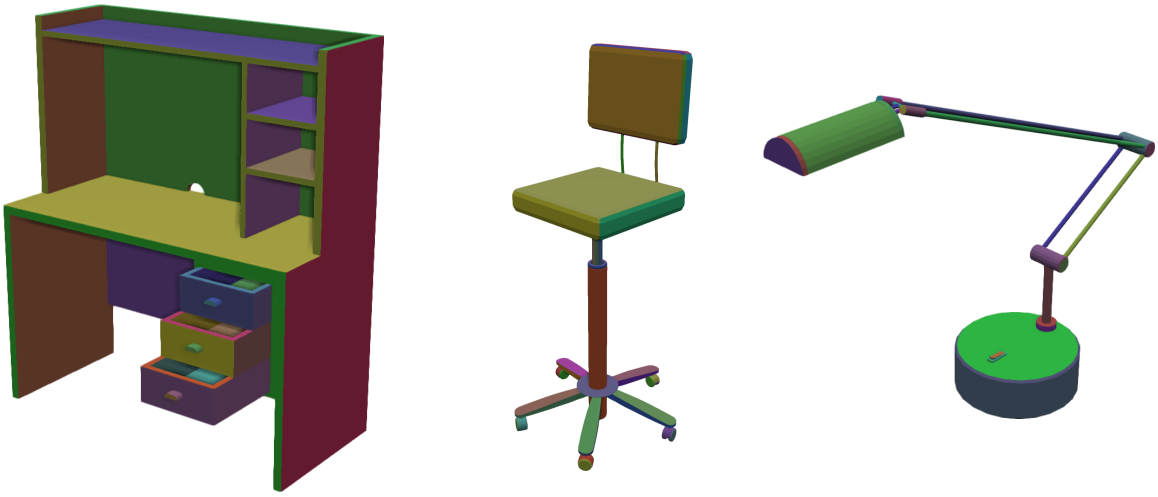


Figure 4.12: Some example planar surface segmentations of 3D models.

first segment our 3D models using the SuperFace algorithm (Kalvin and Taylor, 1996) to obtain a set of mostly planar surfaces (see Figure 4.12). More details about the segmentation algorithm are given in Appendix C. Given a 3D scene dataset we now extract priors on the support surface attributes and object attachment surfaces by observing how the surfaces of each model instance support other model instances in each scene.

We aggregate the above detected support pairs onto the parent and child object categories to establish a set of priors on the supporting surfaces and attachment surfaces:

$$P_{surf}(s|C_c) = \frac{\text{count}(C_c \text{ on surface with } s)}{\text{count}(C_c)}$$

where C_p and C_c refer to the parent and child object categories, and s is a surface descriptor. We then use these priors to evaluate the likelihood of support and most likely supporting surface attributes in new instances of objects in unlabeled scenes through a simple featurization of the supporting and attachment surfaces s.

With these learned Bayesian priors, we can now predict the the support probability for each surface of a candidate parent object within a 3D scene (Figure 4.13), and the static attachment probability for a model's surface (Figure 4.15).

To handle data sparsity we utilize our category taxonomy. If there are fewer than $k = 5$ support observations for a given category, we back off to a parent category in the taxonomy

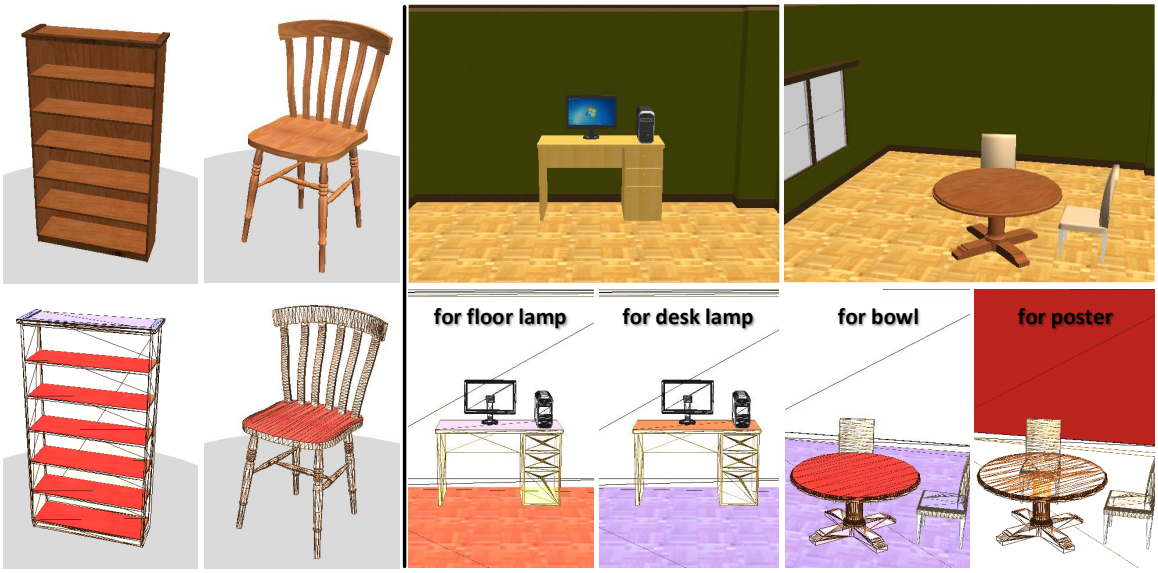


Figure 4.13: Predicted support surfaces. **Left**: predicted high likelihood support surfaces on a bookcase model and a chair model (red indicates surface with high probability of statically supporting other objects, magenta is low probability). **Right**: Likelihoods of static support for some object categories on surfaces in two different rooms.

for more informative priors. If there are no observations available we use the geometry of the model instance to make a decision. Similarly to the case of static support priors, the support surface priors can be conditioned on the scene type by restricting the counts to only the relevant scene category.

Features for support surface priors

We featurize the parent supporting surface depending on the direction of the surface normal (*up, down, horizontally*) and whether the surface is *interior* (facing into the bounding box of the supporting object) or *exterior* (facing out of the bounding box). For instance, a room has a floor which is an upwards interior supporting surface, roof (upwards exterior), ceiling (downwards interior), and inside walls (horizontally interior).

Given this featurization, we now learn from observations in scenes the distribution of

supporting surface features for each category of object:

$$P_{surf_{sup}}(s|C_c) = \frac{\text{count}(C_c \text{ on surface with } s)}{\text{count}(C_c)}$$

This formulation is identical to the one used for the static support priors except in this case, s represents the feature vector of a given support surface. As an example, posters are typically found on walls so their support normal vectors are in the horizontal plane (parallel to the ground). Any unobserved child categories are assumed to have $P_{surf}(S_n = up|C_c) = 1$ since most things rest on a horizontal surface (e.g., floor).

Figure 4.13 illustrates through a variety of examples how we can use these static support priors to predict which surfaces are likely to be support surfaces for a bookcase and a chair. In addition, we can combine these static support priors with support hierarchy priors to get an overall support likelihood for a given object across all available support surfaces in a scene (illustrated in the right part of Figure 4.13).

Child attachments surfaces

To featurize the supported object attachment surfaces we use the bounding box side of the object which attaches to a parent: *top, bottom, front, back, left*, or *right*. For instance, posters are attached on their back side to walls, while rugs are attached on their bottom side to floors. Once again, we use a similar formulation, except now the attachment surface features are used for a given surface s:

$$P_{surf_{att}}(s|C_c) = \frac{\text{count}(C_c \text{ attached at surface } s)}{\text{count}(C_c)}$$

For sparsely observed child categories, we assume 3D (blocky) objects are attached on the bottom, flat (2D) objects are attached on their back or bottom, and thin (1D) objects are attached on one of their sides (which are all assumed to be equivalent). This heuristic results in the placements seen in Figure 4.14 for each of the three object types (pencils representing 1D objects, an iPad representing 2D objects, and the hourglass representing 3D objects).

Figure 4.15 shows how different varieties of lamps and lighting fixtures vary drastically

Figure 4.14: Inferred placements of three different object categories on bookcases. Left: "pencil on bookcase", middle: "iPad on bookcase", right: "clock on bookcase".

in terms of their attachment points. The colored surfaces on each model represent the highest likelihood attachment. Desk and floor lamps are attached by the bottom of their base while the wall lamp is attached by its side plate and the ceiling lamp is attached by its topmost surface.

4.2.4 Relative position priors

We model the relative positions of objects based on their object categories and current scene type: i.e., the relative position of an object of category C_{obj} is with respect to another object of category C_{ref} and for a scene type C_s. We condition on the relationship R between the two objects, whether they are siblings ($R = Sibling$) or child-parent ($R = ChildParent$).

$$P_{relpos}(x, y, \theta | C_{obj}, C_{ref}, C_s, R)$$

When positioning objects, we restrict the search space to points on the selected support surface. The position (x, y) is the centroid of the target object projected onto the support surface of its parent in the semantic frame of the reference object. The θ is the angle between the front orientations of the two objects. We represent these relative position and orientation priors by performing kernel density estimation on the observed samples.

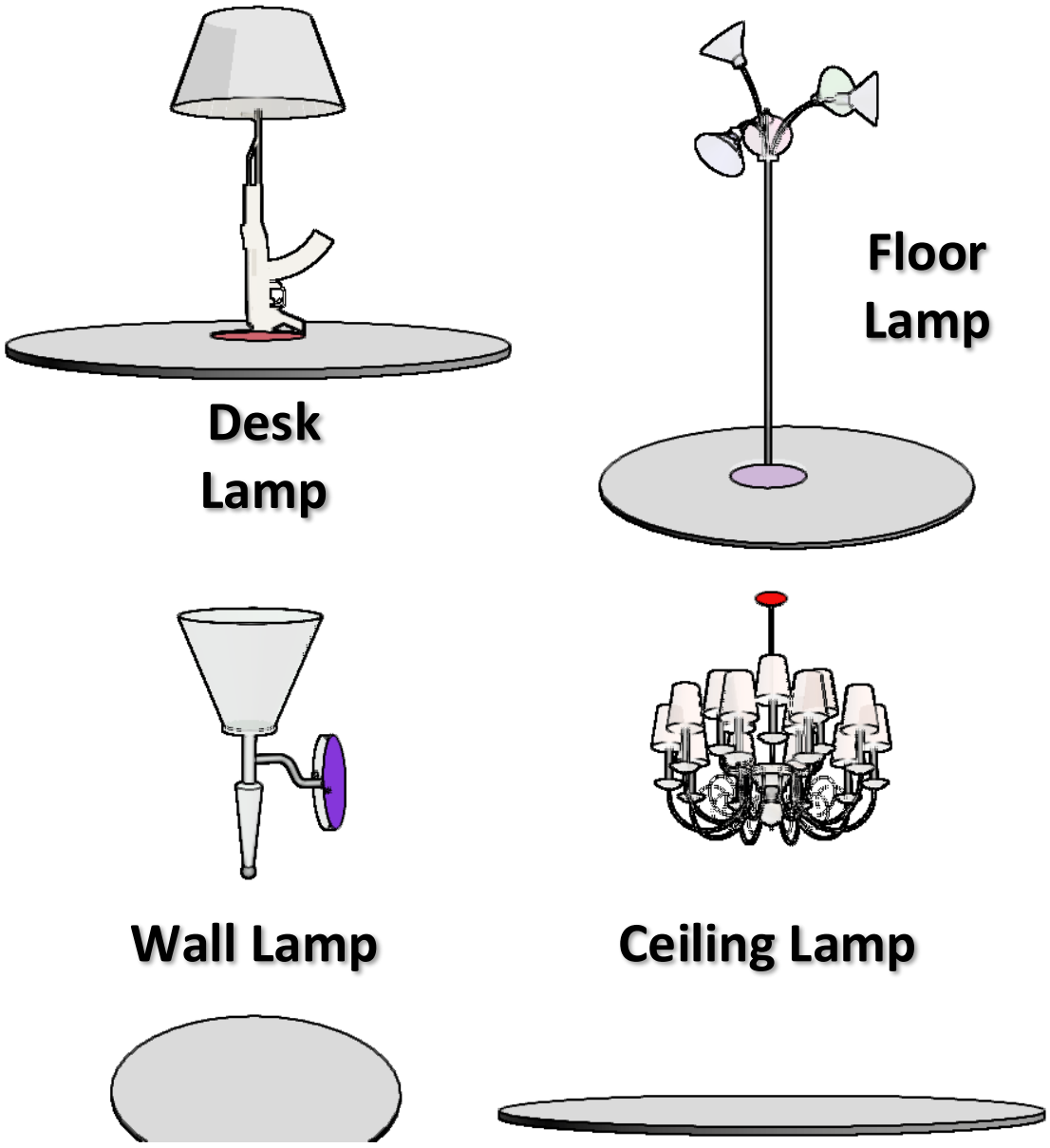


Figure 4.15: Predictions of attachment surfaces for several types of lamp fixtures The highest likelihood attachment surfaces are shown as colored regions on the 3D models.

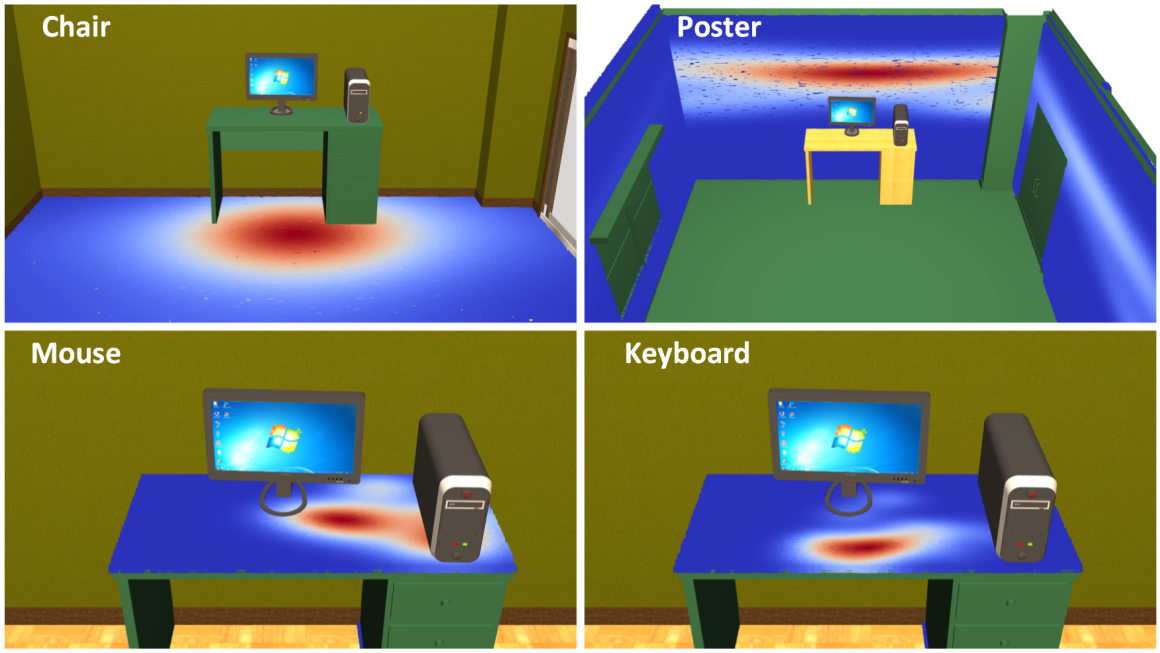


Figure 4.16: Predicted positions using learned relative position priors Clockwise from top left: *chair* given *desk* (top left), *poster-room* (top right), *mouse-desk* (bottom left), *keyboard-desk* (bottom right).

Figure 4.16 shows predicted positions of objects using the learned priors. In case we do not have a sufficient number of observations for the given scene type C_s (we require at least $k = 5$ observations), we backoff up the scene hierarchy to a more generic scene type.

4.2.5 Discussion

In addition to the spatial priors we discussed there are several other kinds of knowledge that can be useful for improving text to scene generation. For example, knowledge of the global and local symmetries of arrangements of objects can allow us to intelligently reposition chairs around a table, or simultaneously relocate a couch, coffee table, and TV arrangement while preserving the function of the (i.e., watching TV from the couch, or placing cups on the coffee table while sitting in the couch). Similarly, capturing the color and style coordination between different objects in an arrangement is critical for generating aesthetically pleasing scenes. For the same example of couch, coffee table and TV, our current system will independently choose the couch, coffee table and TV models with no regard for clashes between colors and styles. A promising avenue for capturing such relations in larger-scale arrangements of objects is to learn a hierarchical grammar of scene components. Some early work along this line has attempted to learn 3D scene grammars for labeling scene parts (Liu et al., 2014) but there is much work that remains to be done in order to obtain hierarchical, multi-scale decompositions of 3D scenes.

4.3 Conclusion

In this section we discussed the forms of common sense world knowledge that we need for text to scene generation: object-level semantic attribute annotations, and spatial priors learned from 3D scenes. In the next chapter, I will present the text to scene system that I have implemented, and describe how each type of world knowledge is incorporated into the system. Later, in Chapter 7, I will show how the same types of world knowledge can improve a natural language–driven interactive scene design system.

Chapter 5

Text to scene

In this chapter we will go over the details of our entire text to 3D scene pipeline in a stage by stage fashion. We will first discuss how the input text is parsed into scene templates. Then we will describe how we can perform inference to flesh out the parsed scene templates by including implicit constraints and priors. Finally, we will address the generation of 3D scenes given the completed scene templates. The basic system was published in Chang et al. (2014b).

5.1 Scene template parsing

The first task that we have to address in our text to scene pipeline is that of taking input text and parsing it into the scene template representation. In chapter 3, we presented scene template parsing as the task of taking an utterance u and extracting the set of constraints expressed by u as a scene template. More precisely, we want to be able to estimate $P(t|u)$ and to pick the most likely scene template $t^* = \arg\max P(t|u)$.

As we defined in § 3.3.1, a scene template $t = (C_s, \mathcal{O}, \mathcal{C})$ consists of a scene type C_s, along with a set of object descriptions $\mathcal{O} = \{o_1, \ldots, o_n\}$ and constraints $\mathcal{C} = \{c_1, \ldots, c_k\}$ on the relationships between the objects. During scene template parsing we identify the scene type C_s, the set of objects $\mathcal{O}$ that must be present in the scene along with their attributes, and the relations between them $\mathcal{C}$. During scene template parsing, we are primarily concerned with extracting the explicitly stated information from the utterance u. Figure 5.1

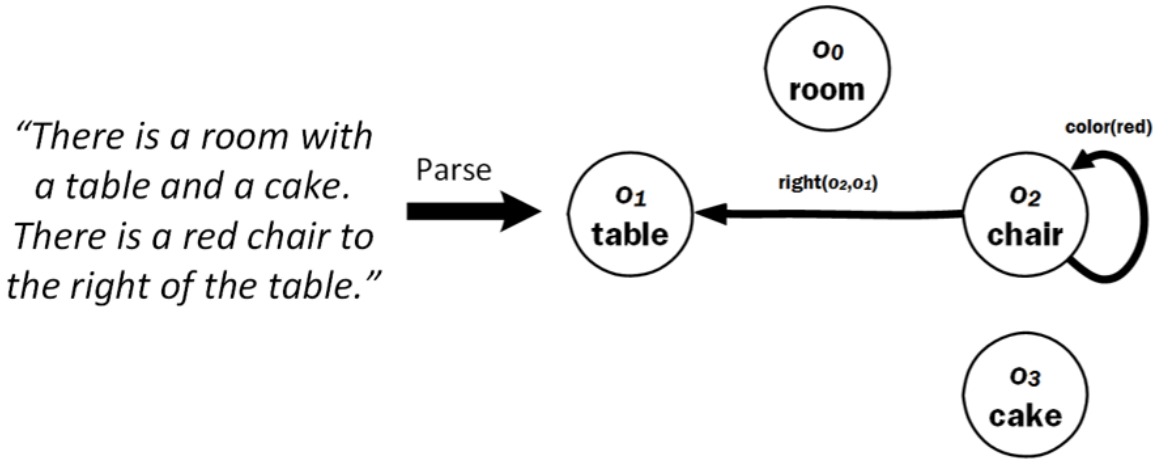


Figure 5.1: Example parsed scene template graph. Given the input sentence on the left, we parse a scene template which represents the objects as nodes and the relations and attributes as edges in a graph.

shows a example of what we would like to extract during scene template parsing.

This task is by itself a challenging problem and in this thesis we will take a fairly straight-forward approach where we first process the input utterance u using the Stanford CoreNLP pipeline (Manning et al., 2014). The Stanford CoreNLP pipeline provides a syntactic annotation of the input text, including tokenization, sentence splitting, lemmatization, part-of-speech tagging, parsing, and coreference resolution. Figure 5.2 shows example output from the Stanford CoreNLP pipeline.

As a utterance u for a scene description may contain multiple sentences, the scene template t is formed by considering each sentence in turn, and aggregating the information across sentences. We note that given a syntactic dependency parse of a sentence, it is fairly straightforward to transform it into a partial scene template via a sequence of rules. Though using rules might seem like a simplistic approach to take and contrary to the broader goal in this thesis of learning from data, rules are particularly well-suited to the syntactically annotated output of existing NLP systems. In particular, many common sense spatial relations and the attachment of properties to objects in descriptions can be specified as fairly simple heuristic rules over dependency parses. By using the output of existing NLP systems, we are able leverage the large amount of research and advancement in NLP. These state-of-the

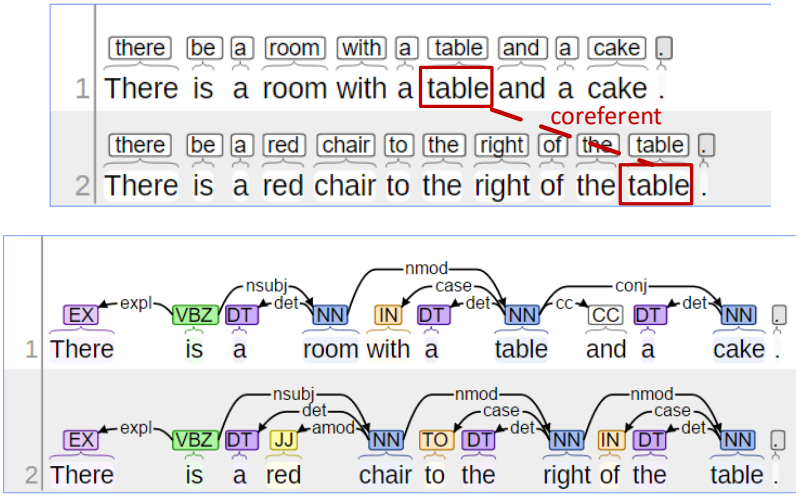


Figure 5.2: Example output from the Stanford CoreNLP pipeline. **Top** Lemma and coreference. **Bottom** dependency parse and part-of-speech tags.

art systems are typically statistically trained using large of amounts of text data and can cope with the ambiguities that are challenging for deterministic rules.

Now, let's consider how we determine the components of the scene template. First, the scene type C_s, if stated in the utterance, is determined by matching the words in the utterance against a list of known scene types from the scene dataset (see Appendix B).

To identify objects, we look for noun phrases and use the head word as the category, filtering with WordNet (Miller, 1995) to determine which objects are *visualizable*. As an example, for the phrase "a piece of cake", the noun "piece" is not a visualizable object while the noun "cake" is. Words that are hyponyms under the "physical object" noun synset, excluding locations, are considered visualizable. Locations are excluded since they are interpreted as scene types for which different objects are composed together to create the scene. To determine when the same object is being referred to across different mentions, the Stanford coreference system (Lee et al., 2011) is used.

To identify properties of the objects, we extract other adjectives and nouns in the noun phrase. Each word in the noun phrase is checked to see if it identifies an attribute of the object. Attribute types are determined using word lists for colors, materials, patterns, and shapes derived from WordNet (see Appendix D for the word lists). Using the word lists, the system identifies the appropriate attribute type to use (e.g., *attribute*(red)=color,

attribute(round)=shape). Additional attributes are further extracted using dependency patterns such as "X is made of Y". Based on the object category and attributes, and other words in the noun phrase mentioning the object, the system also identifies a set of associated keywords to be used later for querying the 3D model database. In addition to the attributes associated with each object, we also extract a count of how many times the object should occur in the scene.

rule	`{tag:VBN}=verb >nsubjpass {}=nsubj >prep ({}=prep >pobj {}=pobj)`
text	The chair`[nsubj]` is made`[verb]` of`[prep]` wood`[pobj]`.
	attribute`(verb,pobj)(nsubj,pobj)`
	`material(chair,wood)`
rule	`{}=dobj >cop {} >nsubj {}=nsubj`
text	The chair`[nsubj]` is red`[dobj]`.
	attribute`(dobj)(nsubj,dobj)`
	`color(chair,red)`
rule	`{}=dobj >cop {} >nsubj {}=nsubj >prep ({}=prep >pobj {}=pobj)`
text	The table`[nsubj]` is next`[dobj]` to`[prep]` the chair`[pobj]`.
	spatial`(dobj)(nsubj, pobj)`
	`next_to(table,chair)`
rule	`{}=nsubj >advmod ({}=advmod >prep ({}=prep >pobj {}=pobj))`
text	There is a table`[nsubj]` next`[advmod]` to`[prep]` a chair`[pobj]`.
	spatial`(advmod)(nsubj, pobj)`
	`next_to(table,chair)`

Table 5.1: Example dependency patterns for extracting attributes and spatial relations The parsed text and extracted attribute or spatial relation is given below each rule.

Dependency patterns are also used to extract spatial relations between objects. Spatial relations are looked up using the learned map of keywords to spatial relations. Semgrex patterns are used to to match the input text to dependencies (Chambers et al., 2007). Semgrex provides pattern matching for dependency graphs. In Semgrex, {} are used to denote word notes and `<label` and `>label` are used to denote paths with the specified label connecting words in the dependency tree. For convenience, matches can be named by using `=name`. Table 5.1 shows some example dependency patterns (the full list of patterns is shown in Appendix E).

As an example, given the input "There is a room with a table and a cake. There is a red

chair to the right of the table." we extract the following objects and spatial relations:

Objects	category	attributes	keywords	quantity
o_0	room		room	1
o_1	table		table	1
o_2	chair	*color*:red	chair, red	1
o_3	cake		cake	1

Relations: right(o_2, o_1)

Note that the Stanford coreference system is able to identify the two mentions of the word "table" as referring to the same object (which we identify as o_1).

The illustration in Figure 5.1 visualizes this example showing the resulting scene template structure as a graph. Note that this scene template only includes explicitly stated objects and relations. As a result, it is unclear how the room and the cake relate to the other objects (so they appear as disconnected nodes in the graph). In the next section we will discuss how we can take an initial scene template and perform inference to determine how it should be completed to account for implicitly present objects and unmentioned constraints.

Even within this seemingly simple task of extracting all present objects in the scene, there are several challenging subproblems. For one, it is not trivial to determine which entities in the text are visualizable and likely to refer to concrete objects in the scene. In addition, correctly identifying coreferent mentions remains an open challenge. While current systems provide reasonable coreference in some cases, their performance is still far below that of part-of-speech taggers or parsers, and thus a significant source of error as we shall see in § 5.7.

Recent work by Schuster et al. (2015) has addressed a similar problem of constructing *scene graph* representations from image descriptions to do better image retrieval. In that work, a trained classifier is used to identify objects and relations. While the trained classifier does improves over a rule-based approach, our rule-based approach is still very competitive. In § 5.4.3, we show a qualitative comparison of the scene graph parser of Schuster et al. (2015) versus the rule based parser presented here. In the future, more advanced approaches such as this one can easily be used to improve upon the method we presented here. In general the problem of taking text and identifying a semantic graph of

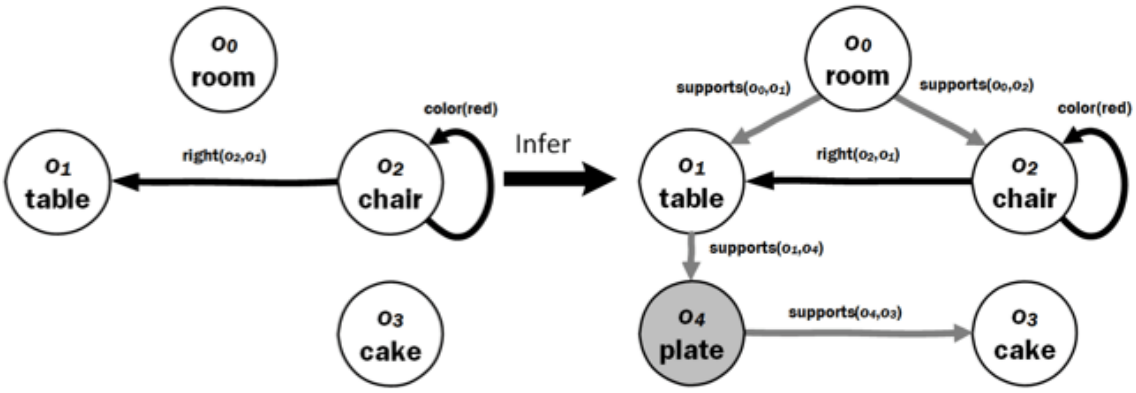


Figure 5.3: Example of inference process on scene template graphs. Given the basic scene template for the sentence "There is a room with a table and a cake. There is a red chair to the right of the table." (left), we infer additional nodes and edges that represent common sense constraints between objects in the scene to complete the scene teplate (right). Gray nodes represent objects that were inferred to be implicitly present.

the objects and relations expressed in the text is a challenging problem domain that is open for much research in NLP and other fields of AI.

5.2 Scene inference

The next step in the text to scene pipeline is to take an initial parsed scene template t and to expand it through scene inference. During this inference process we will expand the set of objects with additional objects that are likely to be present given the scene we are generating. We will then infer additional support constraints between the added objects and existing objects. This step corresponds to the estimation of the probability of a completed scene template t' given the basic template t: $P(t'|t)$. We can then either sample from the distribution or directly maximize such that $t'^* = \arg\max P(t'|t^*)$.

Inferring scene type: In order to infer additional objects for a scene, we need to be aware of the type of scene we are considering. If the scene type is unknown, we use the presence of known object categories to predict the most likely scene type by using Bayes' rule on our object occurrence priors P_{occ} to get $P(C_s|\{C_o\}) \propto P_{occ}(\{C_o\}|C_s)P(C_s)$.

Inferring objects: Given the initial parsed scene template, we infer the presence of additional objects and support constraints (see Figure 5.3). We can do this by using object occurrence probabilities conditioned on the scene type. Once we have a scene type C_s, we sample P_{occ} to find objects that are likely to occur in the scene. We also sample a random number of objects to decide how many extra objects should be inferred (the distribution is set to have $n = 7$ as the average). We also use the support hierarchy priors $P_{support}$ to infer implicit objects. For instance, for each object o_i if the object is not already supported by a parent, we find the most likely supporting object category and add it to our scene if not already present. We only add new objects if they are compatible with our scene type and can support the object under consideration.

Inferring support hierarchies: After inferring implicit objects, we infer the support constraints. Using the learned text to predefined relation mapping from § 6.1.2, we can map the keywords "on" and "top" to the *supported_by* relation. We infer the rest of the support hierarchy by selecting for each object o_i the parent object o_j that maximizes $P_{support}(C_{o_j}|C_{o_i})$.

5.3 Scene generation

Once we have a complete scene template we need to select 3D models to represent the objects, trying to match any described attributes, and to arrange the models in the scene based on constraints. During this step we aim to find the most likely scene given the scene template and prior spatial knowledge. This step corresponds to the estimation of the probability of a concrete scene s given the completed scene template t': $P(s|t')$ and subsequently sampling from the distribution to obtain an instance of a concrete scene. Figure 5.4 gives an overview of this scene generation step.

As we described earlier in § 3.4, the scene s is represented as a set of models $\{m_j\}$ and their transforms $\{T_j\}$. Thus scene generation can be decomposed into two parts: **object selection** and **scene layout** which we now describe in detail.

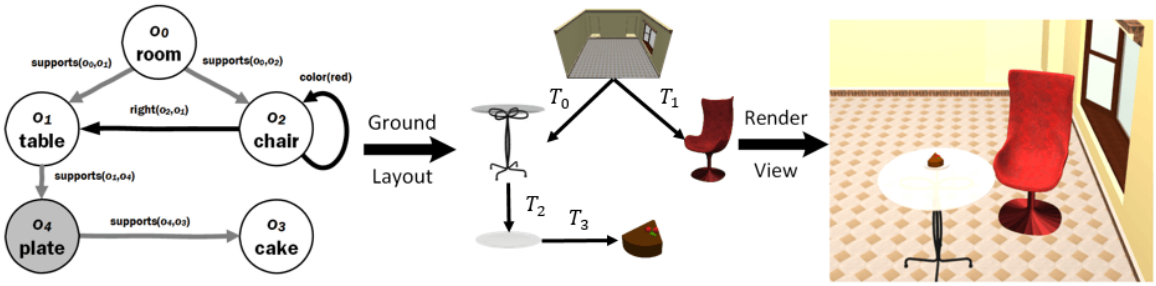


Figure 5.4: Example of generating a geometric scene from a scene template. Given the completed scene template from the output of scene inference in Figure 5.3 (left), we ground the nodes to specific 3D models and lay them out by specifying 3D transform matrices T_i for each model that position the model such that the specified constraints between the objects are satisfied (middle). The resulting geometric scene is one of potentially several plausible interpretations of the given scene template and can be rendered and viewed interactively (right).

5.3.1 Object selection

Object selection consists of identifying the set of models $\{m_j\}$ from the model database given a complete scene template t. As we mentioned, we simplify the problem by assuming that we can select each model independently. Thus, we have $P(\{m_j\}|t) = \prod_j P(m_j|o_j)$ for $o_j \in t$. To select a model for each object, we use the keywords associated with that object to query the model database. Figure 5.5 shows some examples of object selection. We select randomly from the top 10 results for variety and to allow the user to regenerate the scene with different models. This step can be enhanced to take into account correlations between objects (e.g., a lamp on a table should not be a floor lamp model). We use a widely used full text search engine Lucene[1] index to index all our models which allows us to search over specific fields such as the basic category labels and allows us to use additional keywords to rank the returned models. We describe how this simple approach can be improved upon by learning how to ground object references to models in § 6.2.

[1] `https://lucene.apache.org/`

Figure 5.5: Example of selecting models from the database. Selection of "blue office chair" (top) and "wooden desk" (bottom).

5.3.2 Scene layout

Scene layout is the task of arranging the objects by finding appropriate transforms $\{T_j\}$ for each model $\{m_j\}$ given the constraints expressed by the scene template t. Given the selected models, the source scene template, and priors on spatial relations, we find an arrangement of the objects within the scene that maximizes the probability of the layout under the given scene template. We estimate $P(\{m_j, T_j\}|\{m_j\}, t)$ using a layout score $\mathcal{L}$.

In order to compute how well the layout satisfies the constraints given by the scene template, we parameterize the layout of each object using semantically meaningful attributes: support parent $parent_{sup}$, support surface $surf_{sup}$, attachment surface $surf_{att}$, position on support surface pos_{sup}, orientation θ, size σ. These semantic attributes allow us to do higher level reasoning and interactions with the object.

We use a sampling approach to determine the position, orientation, and size of objects within the scene. We first traverse the support hierarchy in depth-first order, positioning the largest available child node and recursing. Child nodes are positioned by first selecting a supporting surface $surf_{sup}$ on a candidate parent object through sampling of $P_{surf_{sup}}$. Using

$P_{surf_{att}}$, we find the most appropriate child attachment surface $surf_{att}$ and orient the child node accordingly. We then sample possible placements $(pos_{sup}, \theta, \sigma)$ on $surf_{sup}$, ensuring that the node is not overhanging and there are no collisions with other objects. Each sampled placement is scored with an overall layout score $\mathcal{L} = \lambda_{obj}\mathcal{L}_{obj} + \lambda_{rel}\mathcal{L}_{rel}$ that is a weighted sum of the object arrangement $\mathcal{L}_{obj}$ score and constraint satisfaction $\mathcal{L}_{rel}$ score:

$$\begin{aligned} \mathcal{L}_{obj} &= \sum_{o_i} P_{surf}(S_n|C_{o_i}) \sum_{o_j \in F(o_i)} P_{relpos}(\cdot) \\ \mathcal{L}_{rel} &= \sum_{c_i} P_{rel}(c_i) \end{aligned}$$

where $F(o_i)$ are the sibling objects and parent object of o_i. We use $\lambda_{obj} = 0.25$ and $\lambda_{rel} = 0.75$ for the results we present.

To ensure that constraints specified explicitly by the utterance are satisfied, we also remove and randomly reposition the objects violating the constraints, and iterate to improve the layout. This sampling-based iterative layout process is illustrated in Figure 5.6. The resulting scene is rendered and presented to the user. In this thesis we use a fairly naïve sampling method, but more advanced approaches such as MCMC sampling can be explored to assist in generating better scene layouts.

5.4 Scene generation results

We show examples of generated scenes, and compare against naive baselines to demonstrate learned priors are essential for scene generation. We also discuss interesting aspects of using spatial knowledge in disambiguating geometric interpretations of "on" (§ 5.4.2). In Chapter 7, we discuss how spatial knowledge can be used to answer semantic queries about the generated scenes and perform view-based object referent resolution.

5.4.1 Generated scenes

Support hierarchy Figure 5.7 shows a generated scene along with the input text and support hierarchy. Even though the spatial relation between the lamp and desk was not

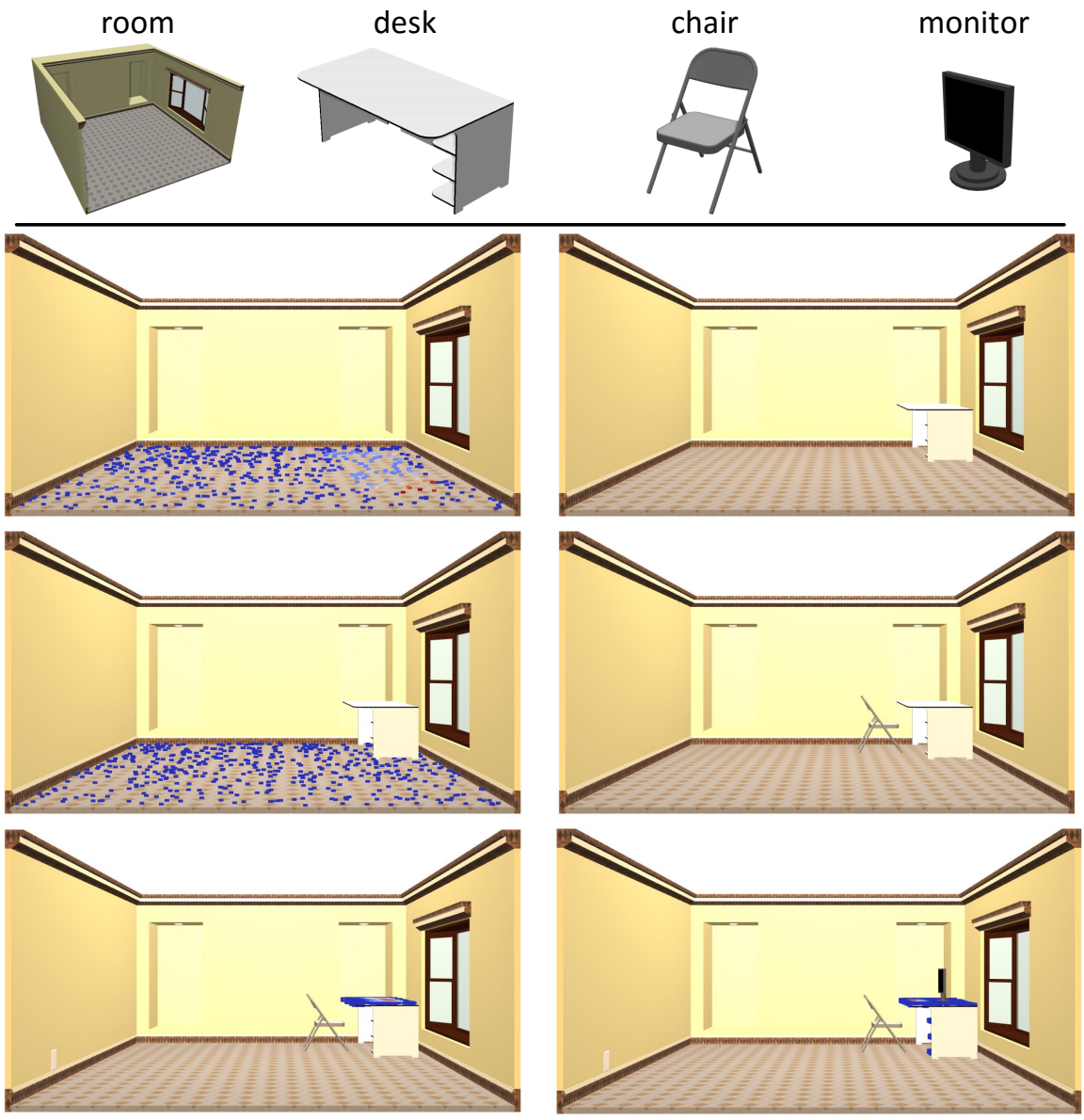


Figure 5.6: Sampling-based scene layout process. Given the input sentence "there is a desk and a chair" the set of 3D models at the top is selected for layout. Using the ordering of the inferred support hierarchy, we place each object in order (from left to right). When placing an object, we sample for high likelihood placements of each model (sample values range from blue for low to red for high probability). The sequence of images proceeds from top left rightwards and then down: first we position the desk, then the chair, and finally the monitor.

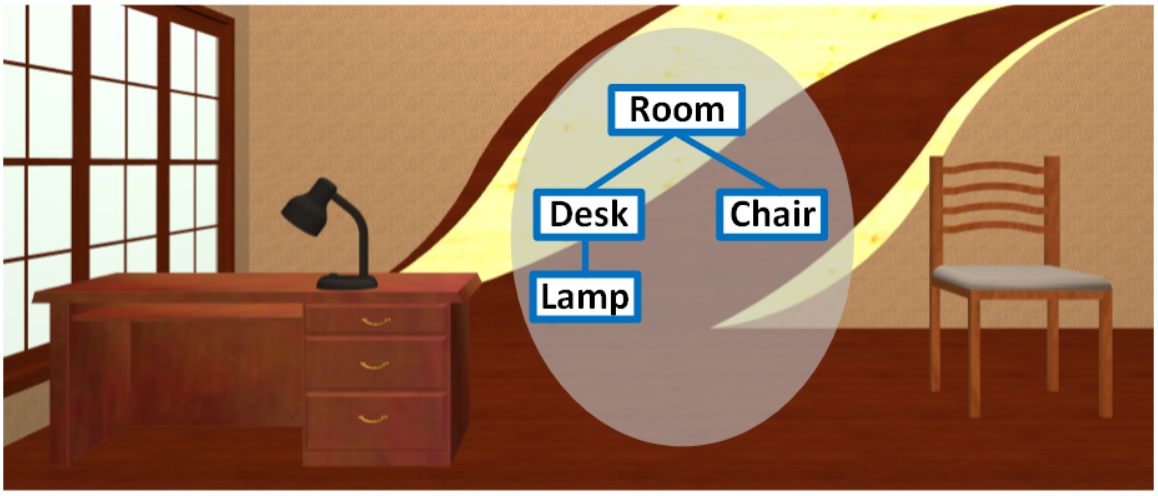


Figure 5.7: Generated scene for "There is a room with a desk and a lamp. There is a chair to the right of the desk." The inferred scene hierarchy is overlaid in the center.

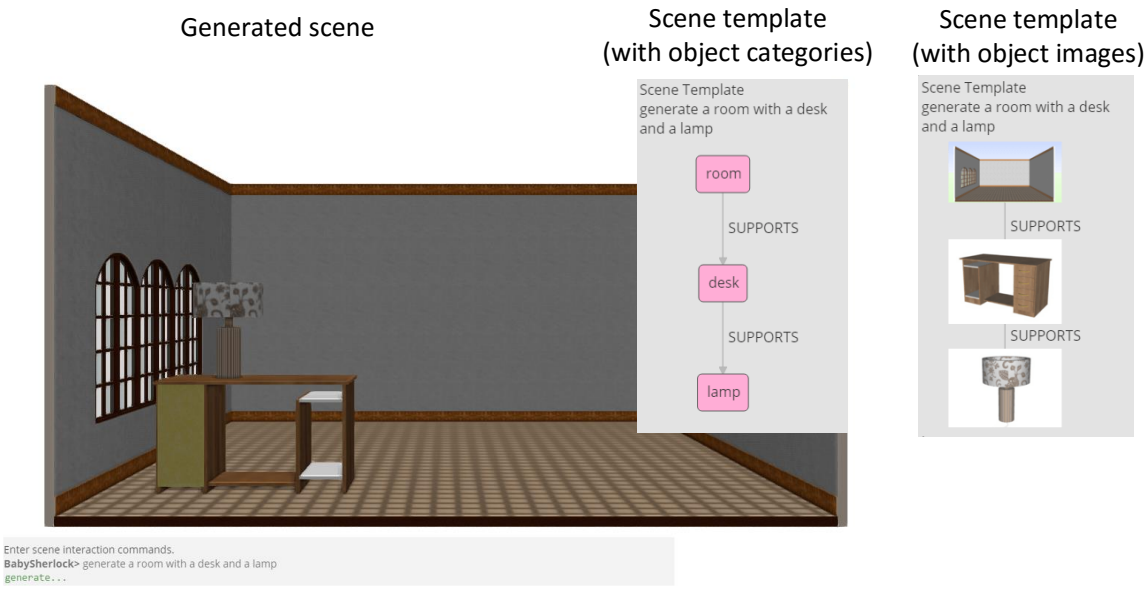


Figure 5.8: Generated scene for "There is a room with a desk and a lamp" using the online demo without inferring extra objects.

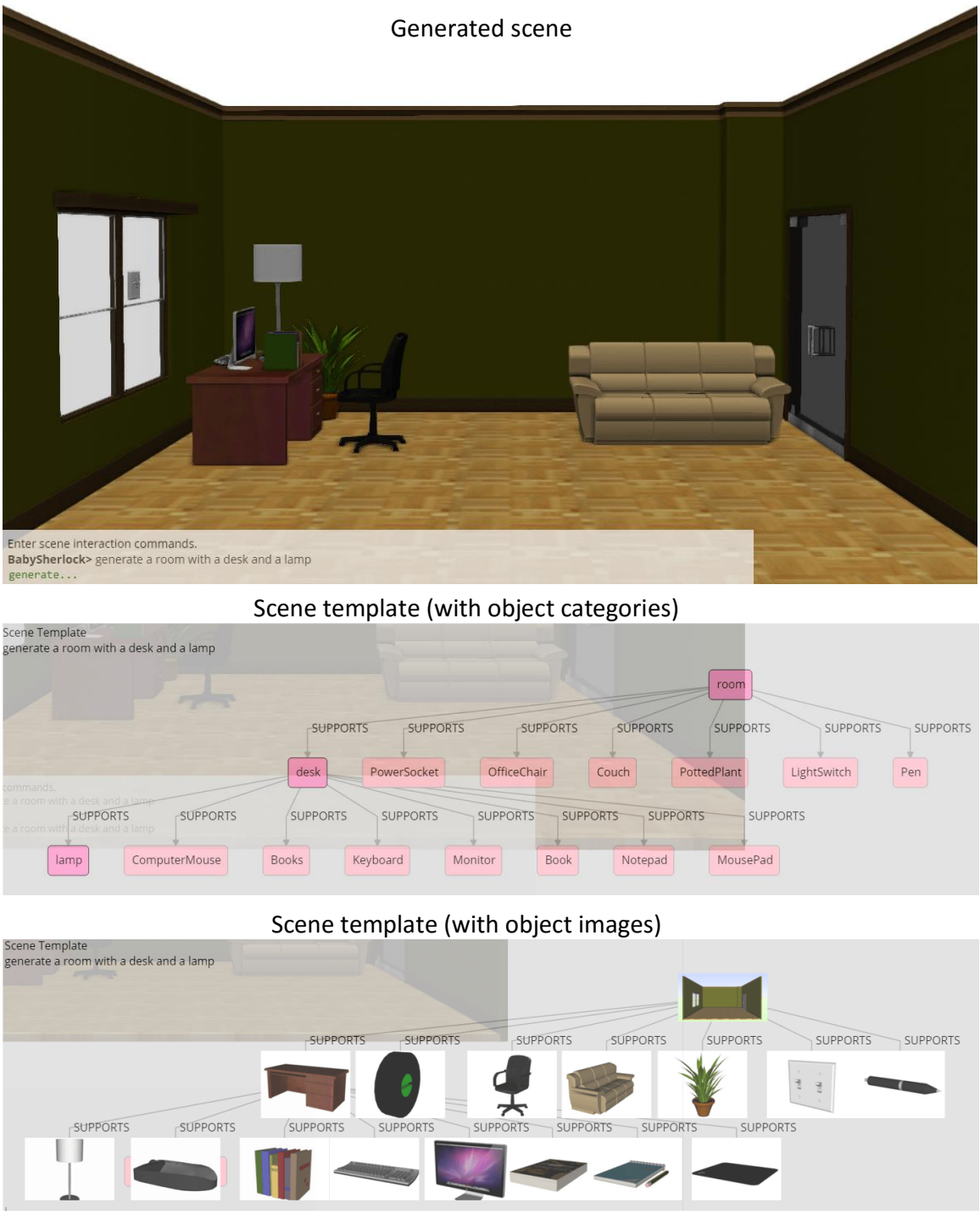


Figure 5.9: Generated scene for “There is a room with a desk and a lamp” using the online demo inferring extra objects.

Figure 5.10: Generated scene for “There is a room with a poster bed and a poster.”

mentioned, we infer that the lamp is supported by the top surface of the desk. Figure 5.8 shows a similar generated scene for “there is a desk and a lamp” using the online demo. The online interface also provides a visualization of the scene template illustrating the inferred support hierarchy.

Inferring extra objects Figure 5.9 shows a generated scene from “there is a desk and a lamp”, this time allowing for inference of extra objects, using the online demo. The different objects imagined by the system is shown in the visualized scene template. As we saw earlier in Chapter 3, Figure 3.1 shows another generated scene along with the support hierarchy and input text. Even though the room, table, and other objects were not explicitly mentioned in the input, we infer that the plate is likely to be supported by a table and that there are likely to be other objects on the table. Without this inference, the user would need to be much more verbose with text such as “There is a room with a table, a plate, and a sandwich. The sandwich is on the plate, and the plate is on the table.”

Disambiguation Figure 5.10 shows a generated scene for the input “There is a room with a poster bed and a poster”. Note that the system differentiates between a “poster” and a

Figure 5.11: Generated scene for the input text “bedroom”.

Figure 5.12: Generated scene for “living room”.

“poster bed” – it correctly selects and places the bed on the floor, while the poster is placed on the wall.

Inferring objects for a scene type Figures 5.11 and 5.12 show an example of inferring all the objects present in a scene from the input “bedroom” and “living room”. Some of the placements are good, while others can clearly be improved. Figure 5.11 has a bed and a reasonable placement for the desk but the position of the picture frame is unrealistic. In Figure 5.12 the TV, desk and couch are well chosen but the layout of the bookshelves and the distances between TV and couch could be improved. Figure 5.13 shows an example of a “living room with velvet curtains and a exercise bike”. Overall, we get the appropriate objects (including the exercise bike) and the parse looks correct. However, the selection and placement of the velvet curtain has failed.

5.4.2 Disambiguating “on”

As it will be shown in § 6.1.2, the English preposition “on”, when used as a spatial relation, corresponds strongly to the *supported_by* relation. In our trained model, the *supported_by* feature also has a high positive weight for “on”.

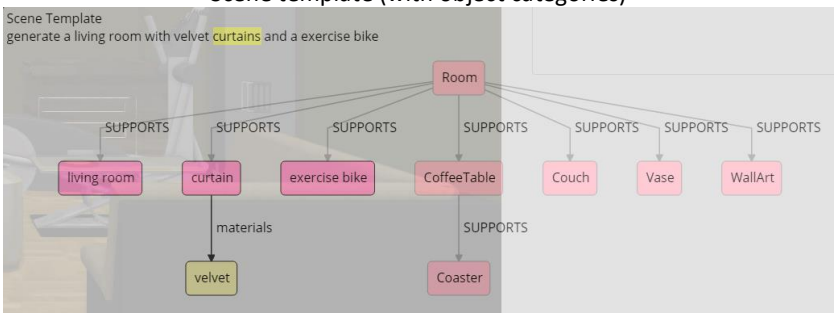


Figure 5.13: Generated scene for “living room with velvet curtains and a exercise bike”.

Figure 5.14: Different interpretations of “on” for support. From top left clockwise: “There is a cup on the table”, “There is a cup on the bookshelf”, “There is a poster on the wall”, “There is a hat on the chair”. Note the different geometric interpretations of “on”.

Our model for supporting surfaces and hierarchy allows interpreting the placement of "A on B" based on the categories of A and B. Figure 5.14 demonstrates four different interpretations for "on". Given the input "There is a cup on the table" the system correctly places the cup on the top surface of the table. In contrast, given "There is a cup on the bookshelf", the cup is placed on a supporting surface of the bookshelf, but not necessarily the top one which would be fairly high. Given the input "There is a poster on the wall", a poster is pasted on the wall, while with the input "There is a hat on the chair" the hat is placed on the seat of the chair.

5.4.3 Qualitative comparison

We qualitatively compare the scenes generated by our model with different variants. Figure 5.16 shows the impact of modeling the support hierarchy and the relative positions in the layout of the scene. Clearly a good support hierarchy is critical to the quality of the generated scenes, and there is much that could be improved upon in our simple approach.

We now compare the output of our text to scene system when different scene template parsers are used. We compare the rule based scene template parser presented in § 5.1 with a modification that converts from scene graphs produced by Schuster et al. (2015) instead of the Stanford dependencies. Given a scene graph the transformation to a scene template is a fairly straightforward process. Figure 5.15 shows the output of the three scene template parsing systems: **st-rule** (rule-based scene template parsing directly on Stanford dependencies from Stanford CoreNLP pipeline), **sg-rule** translates from scene graph rule-based parser presented in Schuster et al. (2015), **sg-bow** translates from scene graph trained bag-of-words parser from Schuster et al. (2015). From this example, we see that the trained bag of words parser actually captures some of the priors that are learned and inferred by our system.

For the input sentences "There is a desk with a notepad on it" and "There is a chair with a notepad on it", the original output from the Stanford CoreNLP system does not identify the "it" as being coreferent with "desk" and "chair". The scene graph rule-based parser (**sg-rule**) has improved rules for determining coreferent promoninal mentions and therefore was able to successfully identify that "it" maps to "desk" and "chair" in the two

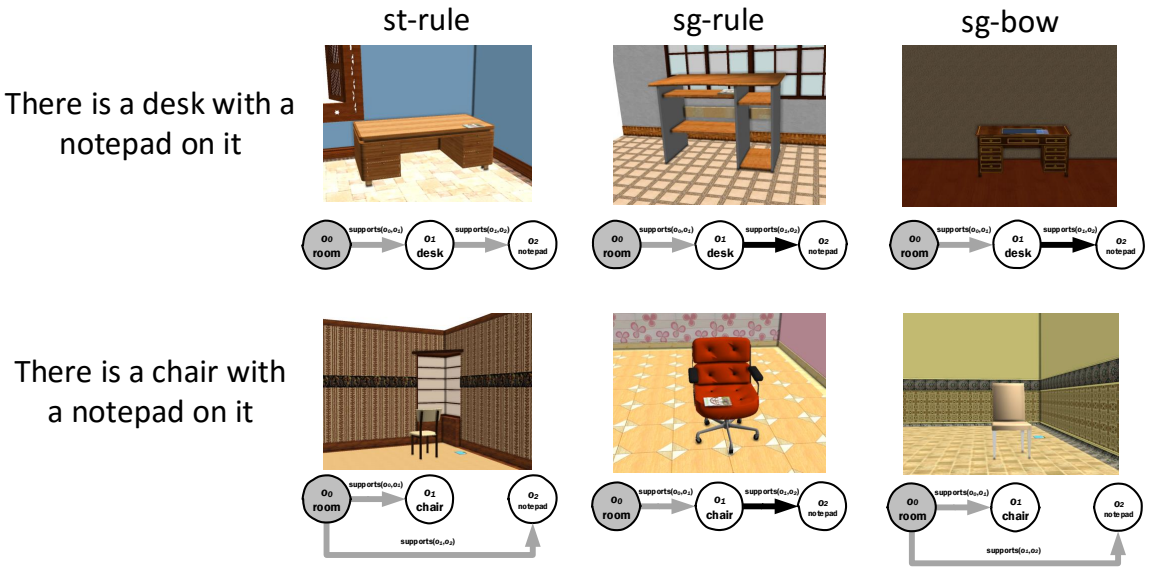


Figure 5.15: Example contrasting using the rule-based parser presented here versus the scene graph parser developed in Schuster et al. (2015). Gray nodes and arrows indicate inferred objects and relations correspondingly. The black arrows indicate parsed relations. Note that **st-rule** did not parse the "on" relation correctly for either sentence while **sg-rule** was able to parse both and **sg-bow** was able to parse the "notepad on desk" example but not the "notepad on chair" example.

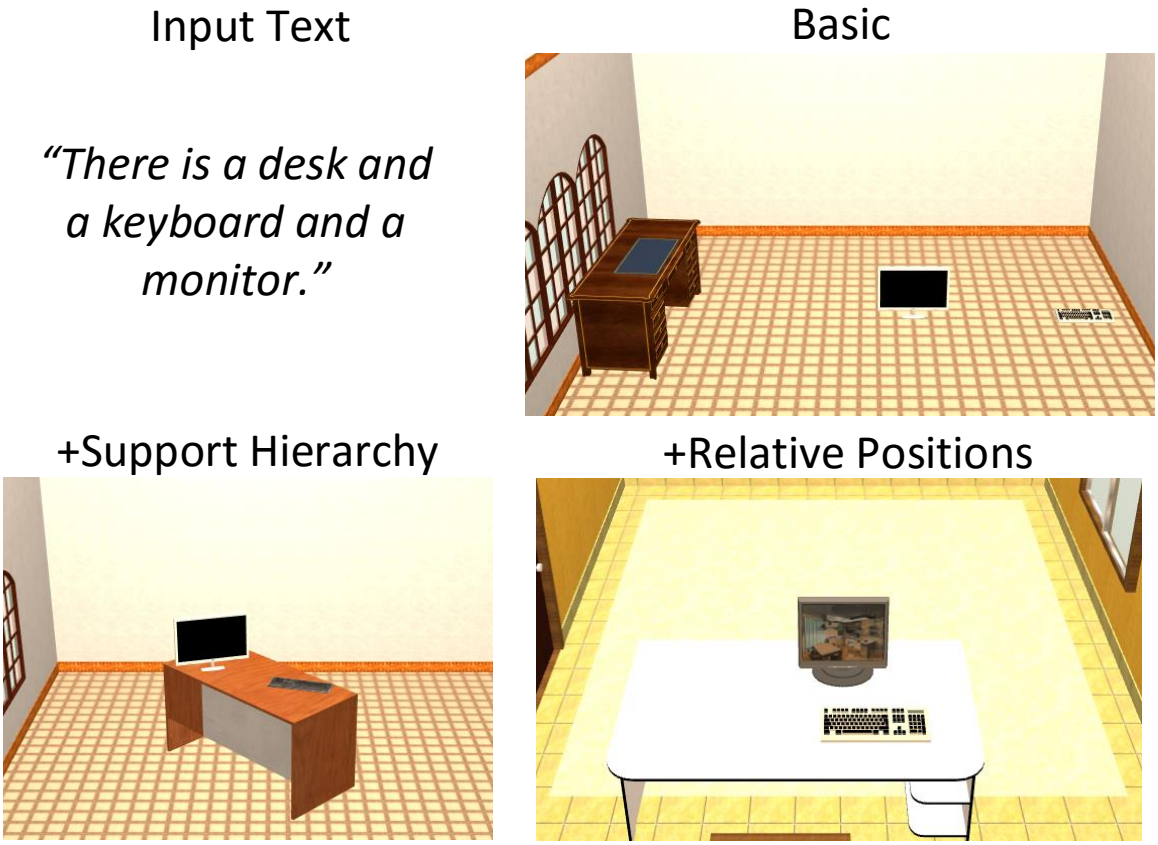


Figure 5.16: Qualitative comparison of influence of support priors on generated scenes. Generated scenes for randomly placing objects on the floor (*Basic*), with inferred *Support Hierarchy*, and with priors on *Relative Positions*.

examples. Figure 5.15 shows that the scene template has the correct form and the resulting scenes are both correct. In contrast **st-rule** was not able to correctly parse either sentence but due to the scene inference we were able to still deduce that notepads go on desks so it is successful for the first case. However, since notepads are not typically found on chairs, the notepad is placed on the ground in the second example. When we use the trained classifier (**sg-bow**) it is able to parse the first example correctly, but not the second one. This is because the training captures prior knowledge about the relation and the entities involved and thus the "notepad on chair" is not interpreted as corresponding to the spatial "on" relation.

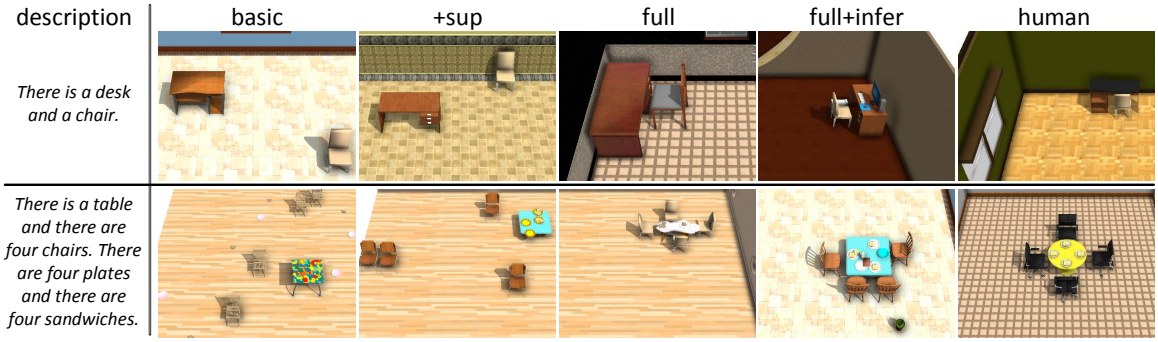


Figure 5.17: Some example textual descriptions and scenes generated by our system in different conditions, as well as scenes manually designed by people.

5.5 Empirical evaluation

We evaluate the output of our system by asking people to judge how well generated scenes match given input descriptions. This is an appropriate initial evaluation since in a practical usage scenario, a scene matching the input description well would provide a good starting point for further refinement. We compare versions of our system contrasting the benefit of different components. We also establish an upper-bound baseline by asking people to manually design scenes corresponding to the same descriptions, using a simple scene design interface used in prior work (Fisher et al., 2012).

Conditions We compare seven conditions: `basic`, `+sup`, `+sup+spat`, `+sup+prior`, `full`, `full+infer`, and `human`. The `basic` condition is a simple layout approach which does not use any learned support, spatial relation, or placement priors. The conditions `+sup`, `+sup+spat`, and `+sup+prior`, and `full` (which includes all three priors) test the benefit of adding these learned priors to the system. Finally, `full+infer` performs inference for selecting and laying out likely unmentioned objects, while `human` consists of the manually designed 3D scenes that people created given the descriptions. For each of these conditions we create a set of 50 scenes, one for each of the input textual descriptions. In total, we have 350 stimulus scene-description pairs to be rated by people during our experiment (see Figure 5.17 for example descriptions and scenes).

Participants Participants were recruited online on Amazon Mechanical Turk and were required to be fluent speakers of English. We recruited a total of 97 participants for evaluating the quality of generated scenes with respect to reference textual descriptions. For the `human` condition, the scenes were created by a different set of 50 participants who were given the textual descriptions and asked to create a corresponding scene (see last column of Figure 5.17).

Procedure During the experiment, a randomly sampled set of 30 pairs of generated scene and input textual descriptions were shown to each participant. The pairs were drawn from all conditions and present in randomized order. The participants were asked to rate each pair on a 7-point Likert scale to indicate "how well the scene matches the description", with a score of 1 indicating a very bad match, and 7 indicating a very good match. The participants were instructed to consider three points in particular: (1) Are the objects mentioned in the description included in the scene? (2) Are the described relationships between the objects correct in the scene? and (3) Overall, does the scene *fit the description*? Figure 5.18 shows the full instructions and examples of rated scenes shown to participants, while Figure 5.19 shows a screenshot of the rating UI that we used to carry out the experiment.

Design The experiment was a within-subjects factorial design with the condition {`basic`, `+sup`, `+sup+spat`, `+sup+prior`, `full`, `full+infer`, `human`}, description {1...50}, and participant {1...97} as factors. The Likert rating for description-to-scene match was the dependent measure.

5.6 Evaluation results

In total the 97 participants gave 2910 scene ratings for the 350 stimulus scene-description pairs with 8.39 ratings on average per pair (standard deviation was 3.56 ratings).

Generated scene ratings The mean ratings for each condition in our experiment are summarized in Table 5.2 and the rating distributions are plotted in Figure 5.20. We see that the `basic` condition receives the lowest average rating, while predictably the scenes

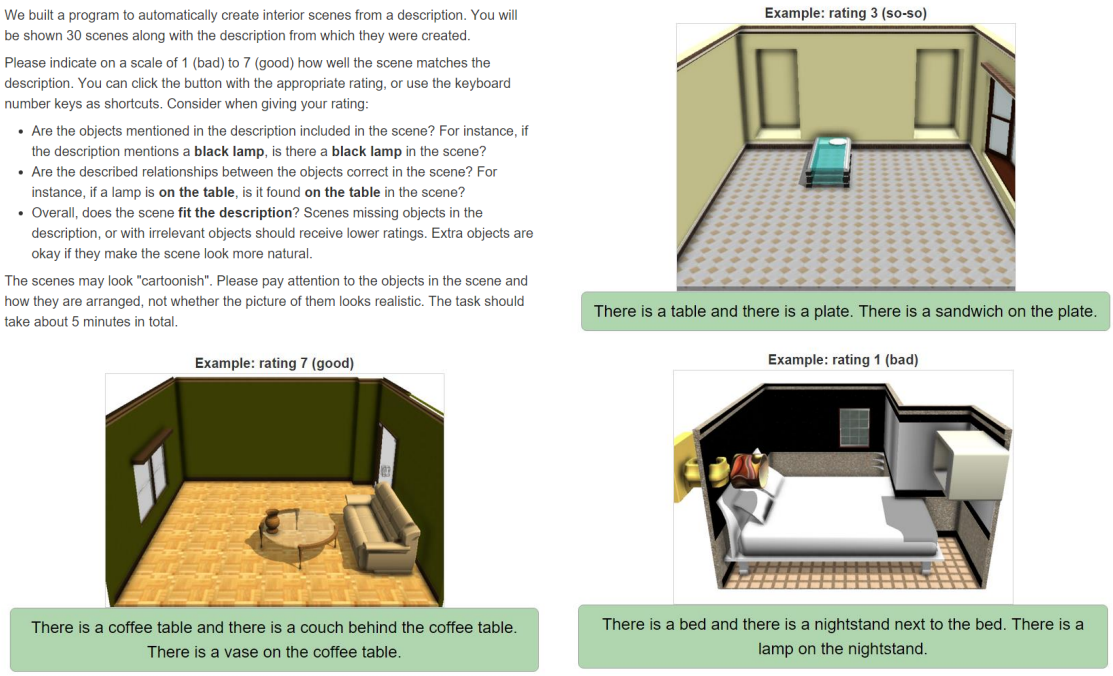


Figure 5.18: Instructions from our evaluation experiment. To help calibrate user's judgements, we provided examples of scenes to be rated as 7 (good), 3 (so-so), and 1 (bad).

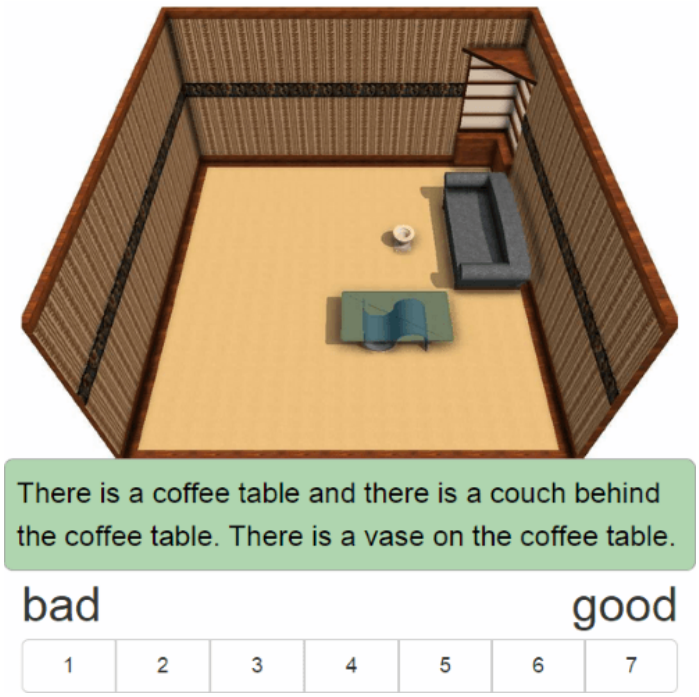


Figure 5.19: Screenshot from our evaluation experiment. Participants were asked to rate how well the displayed scenes match the description on a 7-point Likert scale.

condition	mean rating
basic	3.61
+sup	4.22
+sup+spat	4.72
+sup+prior	4.90
full	**5.03**
full+infer	4.65
human	5.89

Table 5.2: Average scene-description match ratings for each of the conditions in our experiment.

designed by people receive the highest rating. Adding learned support, spatial relation parsing, and priors for relative position improve the rating for the scenes generated by our system, and the full combined condition receives the highest average rating. We note that the scenes generated with additional inferred objects (full+infer) actually receive a lower rating. We hypothesize that two factors may contribute to the lower rating for full+infer. Firstly, adding extra objects makes the scene layout more complicated and prone to exhibiting object selection or spatial relation errors. Secondly, inferred objects are not explicitly mentioned in the description so participants may not have viewed them as significantly improving the quality of the scene (though they were instructed to consider additional objects positively if they add naturally to the appearance of the scene).

Statistical analysis The statistical significance of the effect of the condition factor on the ratings was determined using a mixed effects model (West et al., 2014), with the condition as a fixed effect and the participant and description as random effects, since the latter two are drawn randomly from a large pool of potential participants and descriptions.[2] Most pairwise differences between conditions for mean scene rating were significant under Wilcoxon rank-sum tests with the Bonferroni-Holm correction ($p < 0.05$). The exceptions are the comparisons between +sup+spat, +sup+prior and full which were not significant.

[2]We used the lme4 R package and optimized for maximum log-likelihood (Baayen et al., 2008).

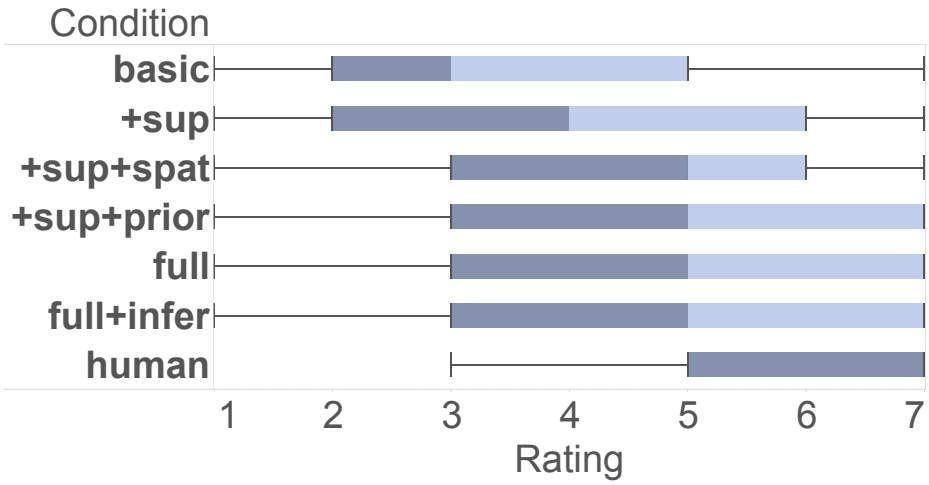


Figure 5.20: Distributions of scene-description match ratings for experiment conditions. The dark and light regions delineate the lower and upper quartile respectively, and the whiskers extend to 150% of the interquartile range.

Summary The experimental results show that our system can generate plausible 3D scenes given input descriptions. The different components of the system that leverage learned support relations, spatial relationship parsing, and relative position priors all contribute towards improving the quality of the generated scenes. Implicitly inferred additional objects do not improve the scene-to-description match ratings given by people. This could be an artifact of the experimental design as it may not be clear to the participants whether scenes with inferred objects should received a high rating or not. It is also possible that scenes with more objects are more likely to be arranged poorly. More experimentation is needed to examine this further.

5.7 Limitations and discussion

While the system shows promise, there are still many challenges in text-to-scene generation. For one, we did not address the difficulties of resolving objects. A failure case of our system stems from using a fixed set of categories to identify visualizable objects. For example, the sense of "top" referring to a spinning top, and other uncommon object types, are not handled by our system as concrete objects. Furthermore, complex phrases including

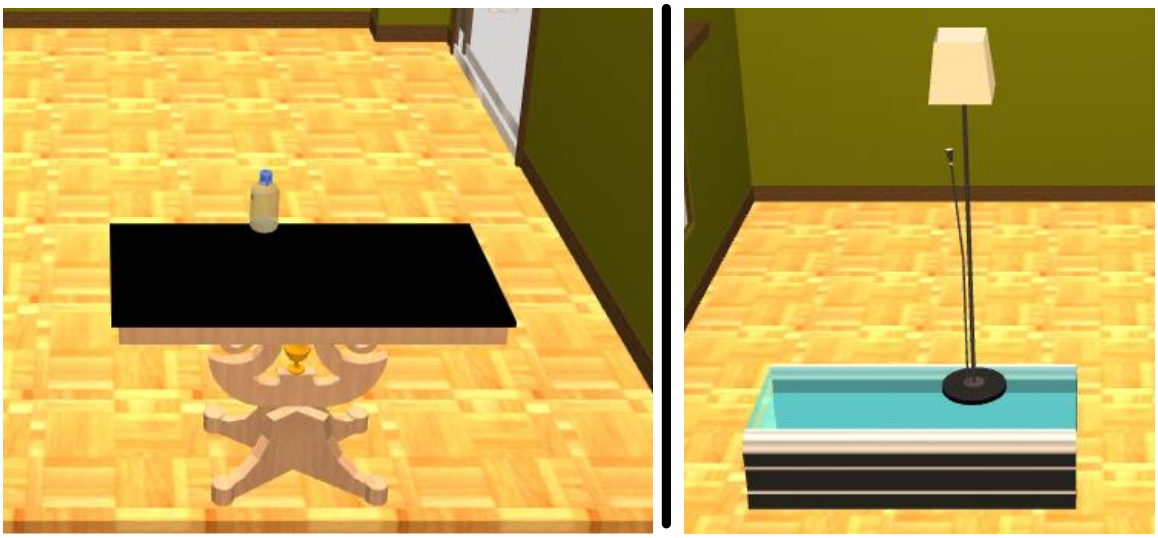


Figure 5.21: Inappropriate object selections for scene generation. **Left**: A water bottle instead of wine bottle is selected for "There is a bottle of wine on the table in the kitchen". In addition, the selected table is inappropriate for a kitchen. **Right**: A floor lamp is incorrectly selected for the input "There is a lamp on the table".

object parts such as "there's a coat on the seat of the chair" are not handled. Figure 5.21 shows some example cases where the context is important in selecting an appropriate object and the difficulties of interpreting noun phrases.

In addition, we rely on a few dependency patterns for extracting spatial relations so robustness to variations in spatial language is lacking. We only handle binary spatial relations (e.g., "left", "behind") ignoring more complex relations such as "around the table" or "in the middle of the room". Though simple binary relations are some of the most fundamental spatial expressions and a good first step, handling more complex expressions will do much to improve the system.

Another issue is that the interpretation of sentences such as "the desk is covered with paper", which entails many pieces of paper placed on the desk, is hard to resolve. With a more data-driven approach we can hope to link such expressions to concrete facts.

Finally, we use a traditional pipeline approach for text processing, so errors in initial stages can propagate downstream. Failures in dependency parsing, part of speech tagging, or coreference resolution can result in incorrect interpretations of the input language. For example, in the sentence "there is a desk with a chair in front of it", "it" is not identified as

coreferent with "desk" so we fail to extract the spatial relation `front_of`(`chair`, `desk`).

5.8 Conclusion

In this chapter I described the implementation of a text to scene system that can incorporate a set of learned priors to select 3D models for object references and lay them out into a concrete 3D scene such that the satisfy both explicit and inferred spatial constraints. In this implementation, the spatial relation and object grounding is done through a manually specified mapping. In the next chapter I will discuss provide more details on the set of predefined mapping supported by the system and how we can learn such a grounding automatically from data.

Chapter 6

Grounding

In the previous chapter I outlined the basic text to scene system pipeline incorporating prior knowledge. However, as we saw, the ability to ground object references is limited if we rely on available tags. In this chapter I will address the learning of lexical groundings of text to spatial relations, and of text referring to objects to concrete 3D model representations. We learn how to perform this lexical grounding using the 3D scene and textual description data that we have collected.

6.1 Grounding spatial relations

When considering spatial relations, there has been extensive literature that looks at the types of spatial relations, the use of prepositions, and how terms for spatial relations and their interpretation are influenced by the context and communicative goals of the speaker and listener (Levinson, 2003; Gärdenfors, 2014). In this thesis I focus on simple binary spatial relations from a viewer perspective (i.e., the position of the virtual camera). We do not consider relations such as "around", "piled up on", and other such multi-argument relations. We also do not consider when a relation is to be interpreted from the viewer's perspective or from an object-centric perspective (Levinson, 2003).

I define a set of formal spatial relations that we map to natural language terms (§ 6.1.1). In addition, we collect annotations of spatial relation descriptions from people, learn a mapping of spatial keywords to our formal spatial relations, and train a classifier that given

Relation	$P(relation)$
inside(A,B)	$\frac{Vol(A \cap B)}{Vol(A)}$
outside(A,B)	1 - $\frac{Vol(A \cap B)}{Vol(A)}$
left_of(A,B)	$\frac{Vol(A \cap \text{left\_of } (B))}{Vol(A)}$
right_of(A,B)	$\frac{Vol(A \cap \text{right\_of } (B))}{Vol(A)}$
near(A,B)	$\mathbb{1}(dist(A, B) < t_{near})$
faces(A,B)	$\cos(front(A), c(B) - c(A))$

Table 6.1: Definitions of spatial relation using bounding boxes. Note: $dist(A, B)$ is normalized against the maximum extent of the bounding box of B. $front(A)$ is the direction of the front vector of A and $c(A)$ is the centroid of A.

two objects can predict the likelihood of a spatial relation holding (§ 6.1.2).

6.1.1 Predefined spatial relations

For spatial relations we use a set of predefined relations: *left_of, right_of, above, below, front, back, supported_by, supports, next_to, near, inside, outside, faces, left_side, right_side*. We distinguish *left_of(A,B)* as A being left of the left edge of the bounding box of B vs *left_side(A,B)* as A being left of the centroid of B. We chose these as simple binary relations that are typically used in describing relative object arrangements in common scenes. These relations are measured using axis-aligned bounding boxes from the viewer's perspective. We take a 3D volumetric interpretation of the involved bounding boxes and score the predefined relations on a continuous range $[0, 1]$ so that there is a notion of goodness of each relation given a concrete scenario. The involved bounding boxes are compared to determine volume overlap or closest distance (for proximity relations; see Table 6.1). For simplicity, we take the axis aligned bounding box and rectilinear coordinate frame to measure the offsets and overlaps. It is possible to consider alternatives such as a spherical coordinate frame where the notions of left and right are mapped to sectors of the sphere, as done in Tappan (2008). Defining the best possible predefined set of spatial relations is however not the focus of the thesis. We are more interested in whether we can learn how to map these relations from data and how we can use them in the context of text to scene generation. For the full list of relations please refer to Appendix F.

Keyword	*Top Relations and Scores*
behind	$(back\_of, 0.46), (back\_side, 0.33)$
adjacent	$(front\_side, 0.27), (outside, 0.26)$
below	$(below, 0.59), (lower\_side, 0.38)$
front	$(front\_of, 0.41), (front\_side, 0.40)$
left	$(left\_side, 0.44), (left\_of, 0.43)$
above	$(above, 0.37), (near, 0.30)$
opposite	$(outside, 0.31), (next\_to, 0.30)$
on	$(supported\_by, 0.86), (on\_top\_of, 0.76)$
near	$(outside, 0.66), (near, 0.66)$
next	$(outside, 0.49), (near, 0.48)$
under	$(supports, 0.62), (below, 0.53)$
top	$(supported\_by, 0.65), (above, 0.61)$
inside	$(inside, 0.48), (supported\_by, 0.35)$
right	$(right\_of, 0.50), (lower\_side, 0.38)$
beside	$(outside, 0.45), (right\_of, 0.45)$

Table 6.2: Learned mapping of top keywords to spatial relations Appropriate mappings are marked in **bold**.

Feature	*#*	*Description*
$delta(A, B)$	3	Delta position (x, y, z) between the centroids of A and B
$dist(A, B)$	1	Normalized distance (wrt B) between the centroids of A and B
$overlap(A, f(B))$	6	Fraction of A inside left/right/front/back/top/bottom regions wrt B: $\frac{Vol(A \cap f(B))}{Vol(A)}$
$overlap(A, B)$	2	$\frac{Vol(A \cap B)}{Vol(A)}$ and $\frac{Vol(A \cap B)}{Vol(B)}$
$support(A, B)$	2	$supported\_by(A, B)$ and $supports(A, B)$

Table 6.3: Features for trained spatial relations predictor.

Figure 6.1: Our data collection task.

Since these spatial relations are resolved with respect to the view of the scene, they correspond to view-centric definitions of spatial concepts.

6.1.2 Learning spatial relations

To learn how to ground language to spatial relations, we first collect a set of text descriptions of spatial relations between two objects in 3D scenes by running an experiment on Amazon Mechanical Turk. We present a set of screenshots of scenes in our dataset that highlight particular pairs of objects and we ask people to fill in a spatial relationship of the form "The __ is __ the __" (see Fig 6.1). We collected a total of 609 annotations over 131 object pairs in 17 scenes. We use this data to learn priors on view-centric spatial relation terms and their concrete geometric interpretation.

For each response, we select one keyword from the text based on length. We learn a mapping of the top 15 keywords to our predefined set of spatial relations. We use our

predefined relations on annotated spatial pairs of objects to create a binary indicator vector that is set to 1 if the spatial relation holds, or zero otherwise. We then create a similar vector for whether the keyword appeared in the annotation for that spatial pair, and then compute the cosine similarity of the two vectors to obtain a score for mapping keywords to spatial relations. Table 6.2 shows the obtained mapping. Using just the top mapping, we are able to map 10 of the 15 keywords to an appropriate spatial relation. The 5 keywords that are not well mapped are proximity relations that are not well captured by our predefined spatial relations.

Using the 15 keywords as our spatial relations, we train a log linear binary classifier for each keyword over features of the objects involved in that spatial relation (see Table 6.3). We then use this model to predict the likelihood of that spatial relation in new scenes.

Figure 6.2 shows examples of predicted likelihoods for different spatial relations with respect to an anchor object in a scene. Note that the learned spatial relations are much stricter than our predefined relations. For instance, "above" is only used to referred to the area directly above the table, not to the region above and to the left or above and in front (which our predefined classifier will all consider to be above). I show in the results that we can obtain more accurate scenes using the trained spatial relations than the predefined ones.

6.1.3 Comparison

In Figure 6.3 I show a qualitative comparison of scene generation results when using predefined spatial relations and when using learned spatial relations. The baseline comparison against no spatial relation knowledge is also shown to make clear the importance of being able to ground spatial relations. The figure shows that the learned spatial relations can give a more accurate layout than the naive predefined spatial relations, since it captures pragmatic implicatures of language, e.g., left is only used for directly left and not top left or bottom left (Vogel et al., 2013; Dawson et al., 2013).

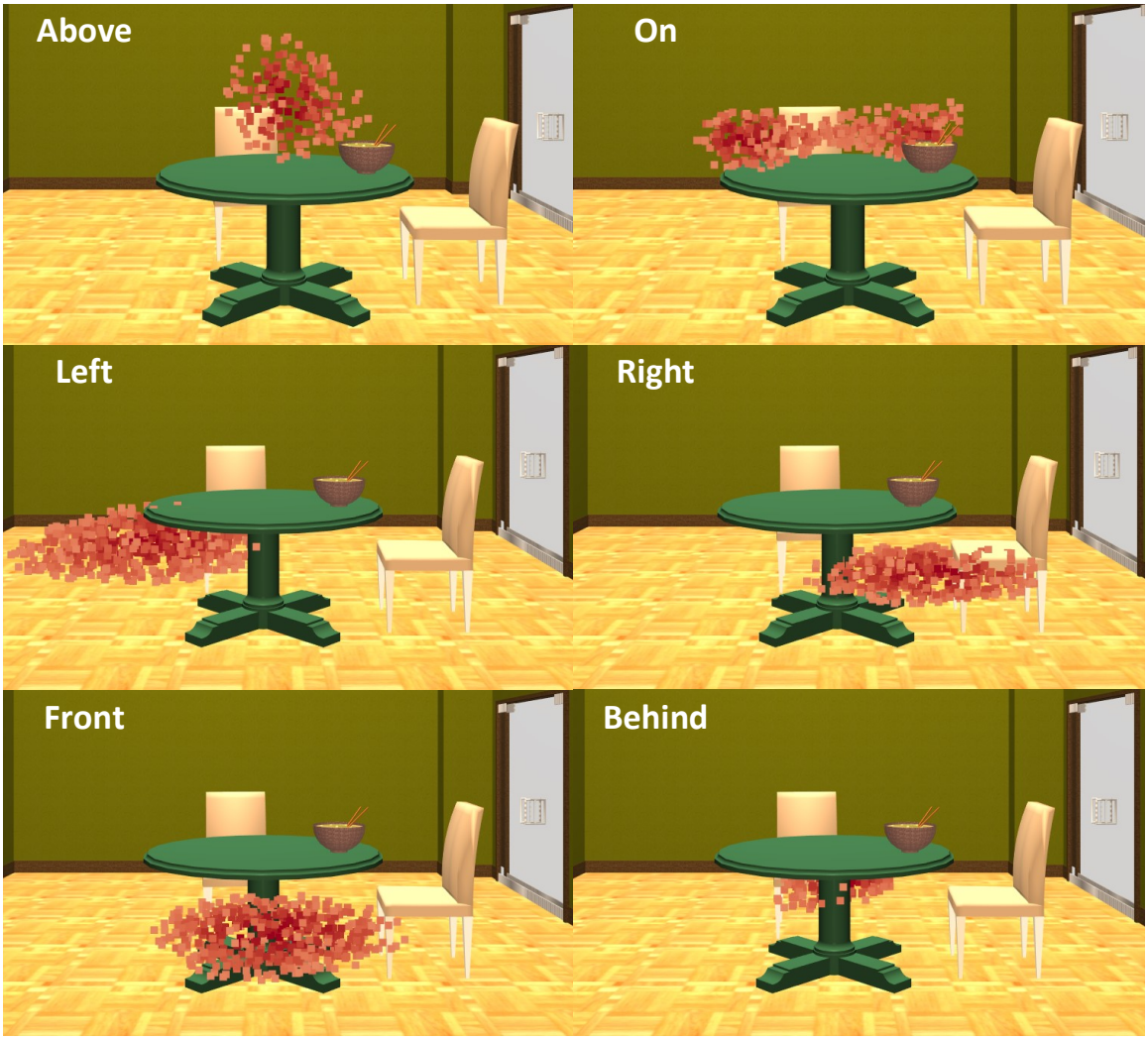


Figure 6.2: High probability regions for learned spatial relations Point clouds indicate where the center of another object would occur for some spatial relations with respect to a table: *above* (top left), *on* (top right), *left* (mid left), *right* (mid right), *in front* (bottom left), *behind* (bottom right).

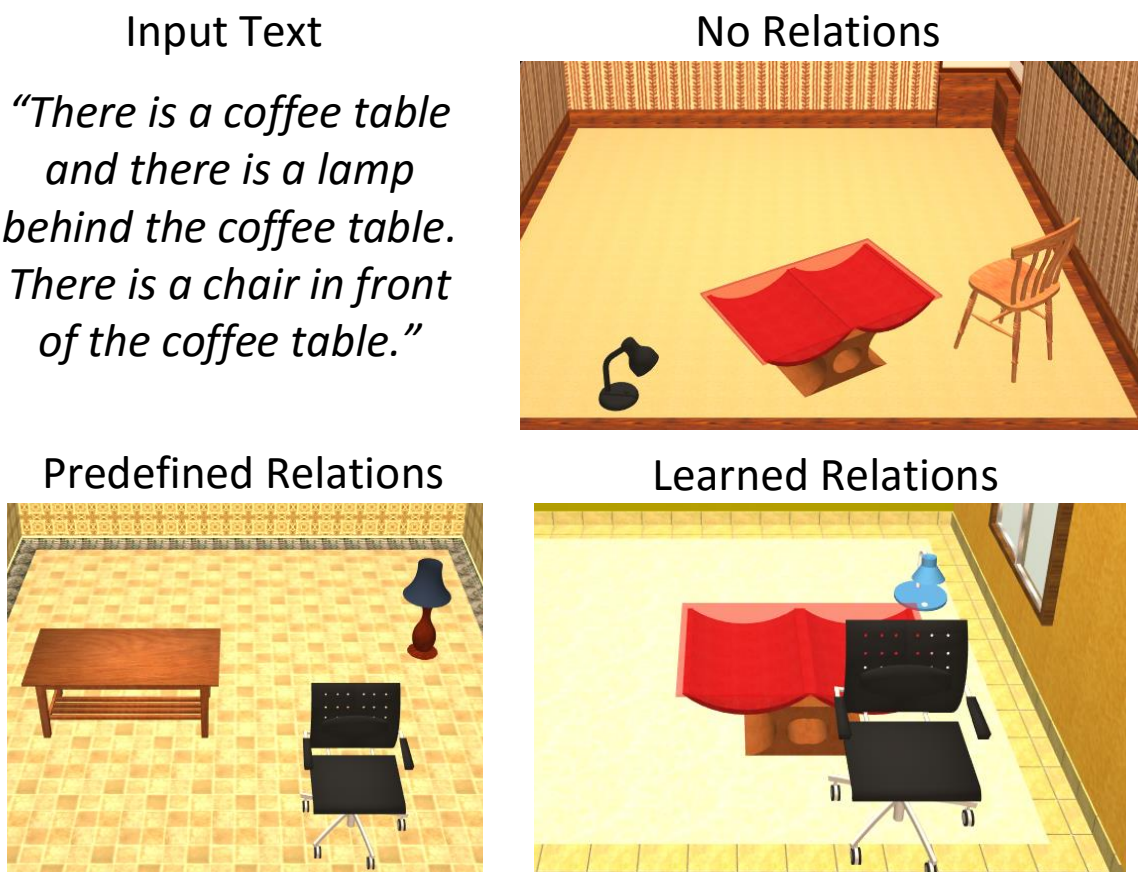


Figure 6.3: Qualitative comparison of learned vs predefined spatial relations. Generated scenes with no understanding of spatial relations (*No Relations*), scoring using *Predefined Relations* and *Learned Relations*.

6.2 Grounding objects

Once we determine from the input text what objects exist and their spatial relations, we select 3D models matching the objects and their associated properties. As described in § 5.3.1, each object in the scene template is grounded by querying a 3D models database with the appropriate category and keywords.

As I mentioned in § 3.5.1, we use a 3D model dataset collected from Google 3D Warehouse by prior work in scene synthesis and containing about 12490 mostly indoor objects (Fisher et al., 2012). These models have text associated with them in the form of names and tags. In addition, we semi-automatically annotated models with object category labels (roughly 270 classes). We used model tags to set these labels, and verified and augmented them manually. In addition, we automatically rescale models so that they have physically plausible sizes and orient them so that they have a consistent up and front direction (Savva et al., 2014a). We then indexed all models in a database that we query at run-time for retrieval based on category and tag labels (see § 4.1).

To create a model for generating scene templates from text, we train a classifier to learn lexical groundings. We then combine our learned lexical groundings with a rule-based scene generation model. The learned groundings allow us to select better models, while the rule-based model offers simple compositionality for handling coreference and relationships between objects. The work described in this section was published in Chang et al. (2015).

6.2.1 Learning lexical groundings

To learn lexical mappings from examples, we train a classifier on a related grounding task and extract the weights of lexical features for use in scene generation. This classifier learns from a "discrimination" version of our scene dataset, in which the scene in each scene–description pair is hidden among four other distractor scenes sampled uniformly at random. The training objective is to maximize the L_2-regularized log likelihood of this scene discrimination dataset under a one-vs.-all logistic regression model, using each true scene and each distractor scene as one example (with *true/distractor* as the output label).

Figure 6.4: Examples of learned lexical groundings These examples are extracted from the top 20 highest-weight features in our learned model: lexical terms from the descriptions in our scene corpus are grounded to 3D models within the scene corpus.

The learned model uses binary-valued features indicating the co-occurrence of a unigram or bigram and an object category or model ID. For example, features extracted from the scene-description pair shown in Figure 1.3 would include the tuples (*desk*, `modelId:132`) and (*the notepad*, `category:notepad`).

To evaluate our learned model's performance at discriminating scenes, independently of its use in scene generation, we split our scene and description corpus (augmented with distractor scenes) randomly into train, development, and test portions 70%-15%-15% by scene. Using only model ID features, the classifier achieves a discrimination accuracy of 0.715 on the test set; adding features that use object categories as well as model IDs improves accuracy to 0.833.

6.2.2 Rule-based model

We use the rule-based parsing component described in § 5.1. This system incorporates knowledge that is important for scene generation and not addressed by our learned model (e.g., spatial relationships and coreference). In § 6.2.3, I describe how we use our learned model to augment this model.

To summarize, this rule-based approach is a three-stage process using established NLP systems:

1. The input text is split into multiple sentences and parsed using the Stanford CoreNLP pipeline (Manning et al., 2014). Head words of noun phrases are identified as candidate object categories, filtered using WordNet (Miller, 1995) to only include physical objects.

2. References to the same object are collapsed using the Stanford coreference system.

3. Properties are attached to each object by extracting other adjectives and nouns in the noun phrase. These properties are later used to query the 3D model database.

6.2.3 Combined model

The rule-based parsing model is limited in its ability to choose appropriate 3D models. We integrate our learned lexical groundings with this model to build an improved scene generation system.

Identifying object categories Using the rule-based model, we extract all noun phrases as potential objects. For each noun phrase p, we extract features $\{\phi_i\}$ and compute the score of a category c being described by the noun phrase as the sum of the feature weights from the learned model in § 6.2.1:

$$\text{Score}(c \mid p) = \sum_{\phi_i \in \phi(p)} \theta_{(i,c)},$$

where $\theta_{(i,c)}$ is the weight for associating feature ϕ_i with category c. From categories with a score higher than $T_c = 0.5$, we select the best-scoring category as the representative for the noun phrase. If no category's score exceeds T_c, we use the head of the noun phrase for the object category.

3D model selection For each object mention detected in the description, we use the feature weights from the learned model to select a specific object to add to the scene. After using dependency rules to extract spatial relationships and descriptive terms associated with the object, we compute the score of a 3D model m given the category c and a set of descriptive terms d using a similar sum of feature weights. As the rule-based system may not accurately identify the correct set of terms d, we augment the score with a sum of feature weights over the entire input description x:

$$m = \arg\max_{m \in \{c\}} \lambda_d \sum_{\phi_i \in \phi(d)} \theta_{(i,m)} + \lambda_x \sum_{\phi_i \in \phi(x)} \theta_{(i,m)}$$

text	category	text	category
chair	Chair	round	RoundTable
lamp	Lamp	laptop	Laptop
couch	Couch	fruit	Bowl
vase	Vase	round table	RoundTable
sofa	Couch	laptop	Computer
bed	Bed	bookshelf	Bookcase

Table 6.4: Top groundings of lexical terms in our dataset to categories of 3D models in the scenes.

For the results shown here, $\lambda_d = 0.75$ and $\lambda_x = 0.25$. We select the best-scoring 3D model with positive score. If no model has a positive score, we assume the object mention was spurious and omit the object.

6.2.4 Learned lexical groundings

By extracting high-weight features from our learned model, we can visualize specific models to which lexical terms are grounded (see Figure 6.4). These features correspond to high frequency text–3D model pairs within the scene corpus. Table 6.4 shows some of the top learned lexical groundings to model database categories. We are able to recover many simple identity mappings without using lexical similarity features, and we capture several lexical variants (e.g., *sofa* for Couch). A few erroneous mappings reflect common co-occurrences; for example, *fruit* is mapped to Bowl due to fruit typically being observed in bowls in our dataset.

6.2.5 Evaluation

We conduct a human judgment experiment to compare the quality of generated scenes using the approaches we presented and baseline methods. To evaluate whether lexical grounding improves scene generation, we need a method to arrange the chosen models into 3D scenes. Since 3D scene layout is not a focus of our work, we use an approach based on prior work in 3D scene synthesis and text to scene generation (Fisher et al., 2012; Chang et al., 2014b), simplified by using sampling rather than a hill climbing strategy.

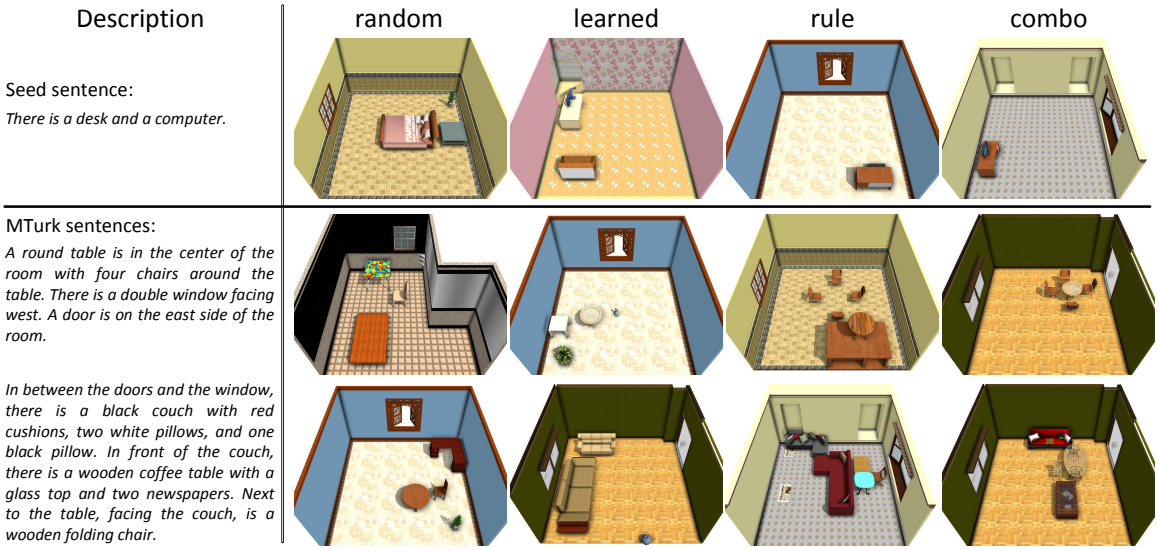


Figure 6.5: Qualitative comparison of generated scenes using lexical grounding Scenes are generated for three input descriptions (one *Seed* and two *MTurk*), using the four different methods: *random*, *learned*, *rule*, *combo*.

Conditions We compare five conditions: {*random, learned, rule, combo, human*}. The *random* condition represents a baseline which synthesizes a scene with randomly-selected models, while *human* represents scenes created by people. The *learned* condition takes our learned lexical groundings, picks the four[1] most likely objects, and generates a scene based on them. The *rule* and *combo* conditions use scenes generated by the rule-based approach and the combined model, respectively.

Descriptions We consider two sets of input descriptions: {*Seeds, MTurk*}. The *Seeds* descriptions are 50 of the initial seed sentences from which workers were asked to create scenes. These seed sentences were simple (e.g., *There is a desk and a chair*, *There is a plate on a table*) and did not have modifiers describing the objects. The *MTurk* descriptions are much more descriptive and exhibit a wider variety in language (including misspellings and ungrammatical constructs). Our hypothesis was that the rule-based system would perform well on the simple *Seeds* descriptions, but it would be insufficient for handling the complexities of the more varied *MTurk* descriptions. For these more natural descriptions, we

[1]The average number of objects in a scene in our human-built dataset was 3.9.

expected our combination model to perform better. Our experimental results confirm this hypothesis.

Qualitative evaluation

Figure 6.5 shows a qualitative comparison of 3D scenes generated from example input descriptions using each of the four methods. In the top row, the *rule-based* approach selects a CPU chassis for *computer*, while *combo* and *learned* select a more iconic monitor. In the bottom row, the rule-based approach selects two newspapers and places them on the floor, while the combined approach correctly selects a coffee table with two newspapers on it. The learned model is limited to four objects and does not have a notion of object identity, so it often duplicates objects.

Human evaluation

We performed an experiment in which people rated the degree to which scenes match the textual descriptions from which they were generated. Such ratings are a natural way to evaluate how well our approach can generate scenes from text: in practical use, a person would provide an input description and then judge the suitability of the resulting scenes. For the *MTurk* descriptions, we randomly sampled 100 descriptions from the development split of our dataset.

Procedure During the experiment, each participant was shown 30 pairs of scene descriptions and generated 3D scenes drawn randomly from all five conditions. All participants provided 30 responses each for a total of 5040 scene-description ratings. Participants were asked to rate how well the generated scene matched the input description on a 7-point Likert scale, with 1 indicating a poor match and 7 a very good one (see Figure 6.6). In a separate task with the same experimental procedure, we asked other participants to rate the overall plausibility of each generated scene without a reference description. This plausibility rating measures whether a method can generate plausible scenes irrespective of the degree to which the input description is matched. We used Amazon Mechanical Turk to recruit 168 participants for rating the match of scenes to descriptions and 63 participants

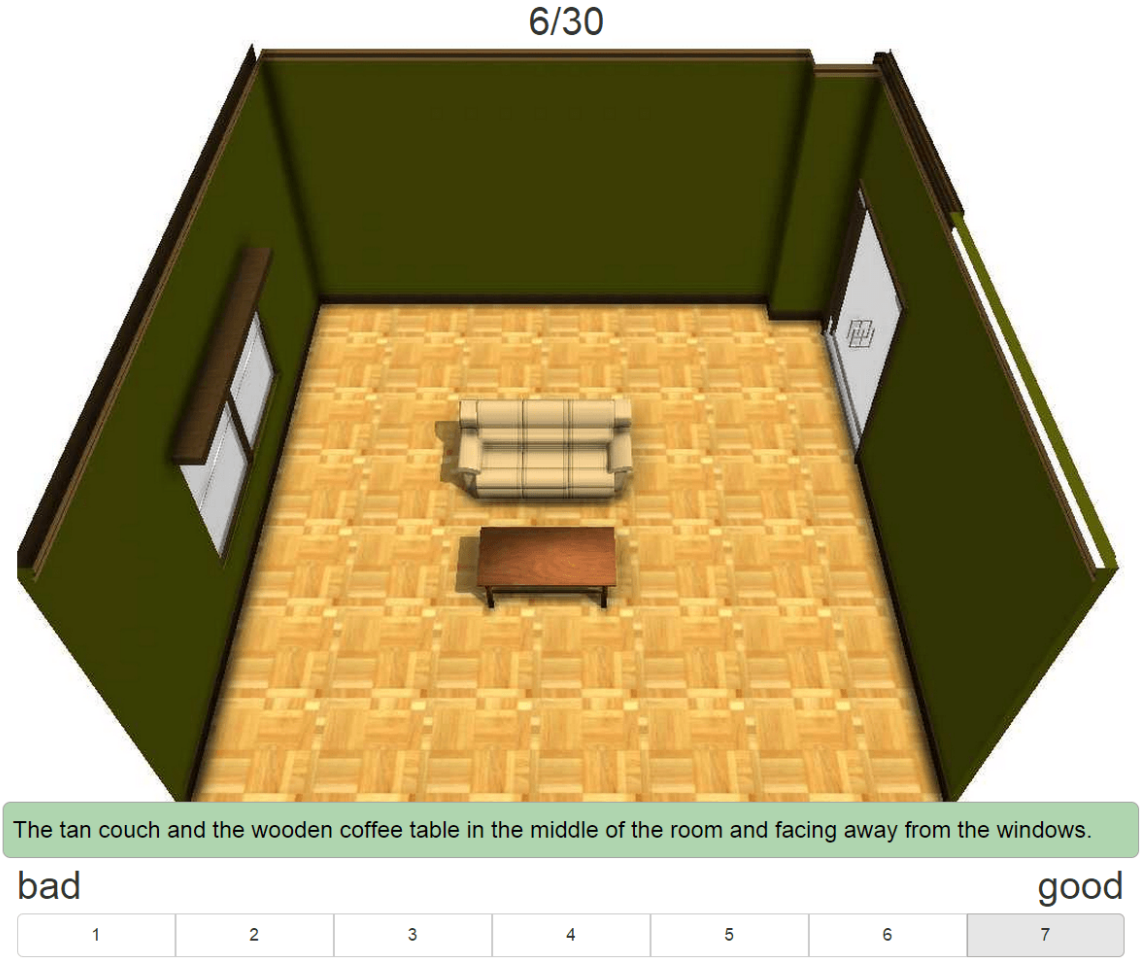


Figure 6.6: Screenshot of the UI for rating scene-description match.

for rating scene plausibility.

Design The experiment followed a within-subjects factorial design. The dependent measure was the Likert rating. Since per-participant and per-scene variance on the rating is not accounted for by a standard ANOVA, we use a mixed effects model which can account for both fixed effects and random effects to determine the statistical significance of our results[2]. We treat the participant and the specific scene as random effects of varying intercept, and the method condition as the fixed effect.

[2]We used the `lme4` R package and optimized fit with maximum log-likelihood (Baayen et al., 2008). We report significance results using the likelihood-ratio (LR) test.

method	Seeds	MTurk
random	2.03 (1.88 – 2.18)	1.68 (1.57 – 1.79)
learned	3.51 (3.23 – 3.77)	2.61 (2.40 – 2.84)
rule	**5.44** (5.26 – 5.61)	3.15 (2.91 – 3.40)
combo	5.23 (4.96 – 5.44)	**3.73** (3.48 – 3.95)
human	6.06 (5.90 – 6.19)	5.87 (5.74 – 6.00)

Table 6.5: Average scene-description match ratings across sentence types and methods (95% C.I.).

Results There was a significant effect of the method condition on the scene-description match rating: $\chi^2(4, N = 5040) = 1378.2, p < 0.001$. Table 6.5 summarizes the average scene-description match ratings and 95% confidence intervals for all sentence type–condition pairs. All pairwise differences between ratings were significant under Wilcoxon rank-sum tests with the Bonferroni-Holm correction ($p < 0.05$). The scene plausibility ratings, which were obtained independent of descriptions, indicated that the only significant difference in plausibility was between scenes created by people (*human*) and all the other conditions. We see that for the simple seed sentences both the rule-based and combined model approach the quality of human-created scenes. However, all methods have significantly lower ratings for the more complex *MTurk* sentences. In this more challenging scenario, the combined model is closest to the manually created scenes and significantly outperforms both rule-based and learned models in isolation.

Error Analysis

Figure 6.7 shows some common error cases in our system. The top left scene was generated with the rule-based method, the top right with the learned method, and the bottom two with the combined approach. At the top left, there is an erroneous selection of concrete object category (wood logs) for the *four wood chairs* reference in the input description, due to an incorrect head identification. At top right, the learned model identifies the presence of brown desk and lamp but erroneously picks two desks and two lamps (since we always pick the top four objects). The scene on the bottom right does not obey the expressed spatial constraints (*in the corner of the room*) since our system does not understand the grounding

Figure 6.7: Common scene generation errors. From top left clockwise: *Wood table and four wood chairs in the center of the room*; *There is a black and brown desk with a table lamp and flowers*; *There is a white desk, a black chair, and a lamp in the corner of the room*; *There in the middle is a table, on the table is a cup*.

of room corner and that the top right side is not a good fit due to the door. In the bottom left, incorrect coreference resolution results in two tables for *There in the middle is a table, on the table is a cup.*

6.3 Scene similarity metric

In addition to the subjective scene-description match judgments that we collected from people, we also introduce an automated metric for scoring scenes given a scene template representation: the *aligned scene template similarity* (ASTS). This metric provides a simple but useful measure of similarity between scene templates and description. Given a one-to-one alignment A between the nodes of a scene template and the objects in a scene, let the alignment penalty $J(A)$ be the sum of the number of unaligned nodes in the scene template and the number of unaligned objects in the scene. For the aligned nodes, we compute a similarity score S per node pair (n, n') where $S(n, n') = 1$ if the model ID matches, $S(n, n') = 0.5$ if only the category matches and 0 otherwise.

We define the ASTS of a scene with respect to a scene template to be the maximum alignment score over all such alignments:

$$\text{ASTS}(s, z) = \max_{A} \frac{\sum_{(n,n') \in A} S(n, n')}{J(A) + |A|}.$$

With this definition, we compare average ASTS scores for each method against average human ratings (Table 6.6). We test the correlation of the ASTS metric against human ratings using Pearson's r and Kendall's rank correlation coefficient r_τ. We find that ASTS and human ratings are strongly correlated ($r = 0.70$, $r_\tau = 0.49$, $p < 0.001$). This suggests ASTS scores could be used to train and algorithmically evaluate scene generation systems that map descriptions to scene templates.

6.4 Conclusion

In this chapter I described how we can learn grounding of object references and spatial relations by leveraging a parallel corpus of 3D scenes and natural language descriptions

method	Human	ASTS
random	1.68	0.08
learned	2.61	0.23
rule	3.15	0.32
combo	**3.73**	**0.44**

Table 6.6: Average human ratings (out of 7) and aligned scene template similarity scores.

of the scenes. Despite an improvement in the quality of the generated scenes, it is still unreasonable to expect that automatic output can fully satisfy a user's intent. In addition, 3D scene design itself is an iterative process, so in the following chapter, I will discuss how the text to scene system can be extended to handle interactive natural language commands for refining 3D scenes.

Chapter 7

Interactive scene design

In addition to allowing for end-to-end 3D scene generation, our framework can be leveraged to improve interactive 3D scene design systems. Designing 3D scenes is a challenging creative task. Expert users expend considerable effort in learning how to use complex 3D scene design tools. Still, immense manual effort is required, leading to high costs for producing 3D content in video games, films, interior design, and architectural visualization. How might we allow people to design 3D scenes using simple natural language?

A 3D scene interface driven by language can empower a broader demographic to create 3D scenes for games, interior design, and virtual storyboarding. Many subtasks in the scene design process can be facilitated through a system that can connect language to the structure of 3D scenes. For example, most basic scene editing operations involve low-level manipulation of object placements and have to be performed repeatedly throughout an interactive scene editing session. Furthermore, offering a set of scenes automatically generated from descriptions can be very useful as candidate starting points for interactive refinement.

In this light, we describe how we can augment our text to 3D scene system so that it can take as input high-level commands formulated with natural language, parse them in the context of a scene currently being viewed, and realize them as sets of low-level manipulations on the scene. In combination with traditional GUI-based manipulation of the scenes, we demonstrate how such a system can make interactive scene design a much more intuitive and rapid task.

Natural language commands Why should we use natural language as the input for scene editing commands? There is a long history of UI and HCI research aimed at making input devices and UI designs easier and more efficient to use. Though many low-level manipulations such as re-positioning of single objects, or selection of objects in a scene can be phrased in terms of direct keyboard and mouse inputs, many other seemingly simple operations that are easy to describe are tedious or complex to perform directly. For example, we can easily say "swap the side table with the flower pot in the back of the room with the couch by the right wall" but this statement would usually map to several translation and re-orientation operations.

The text to scene system presented in this thesis is of course limited in its ability to interpret and understand scene descriptions. However, it can provide good starting points for interactive editing particularly since scene design is an inherently interactive process and driving scene generation solely through language is likely to always be ambiguous and underspecified.

In this chapter, we will discuss how the current system can be extended for interpreting interactive natural language. We demonstrate that by a simple extension, we can handle simple textual commands and used them to interactively manipulate and refine generated 3D scenes such as the ones in Figure 7.1. Here we aim to demonstrate the potential of such a system through a set of high-level operations that we can parse from language. Using the spatial knowledge priors that we have learned, the system can update the arrangement of objects to reflect likely placements after each scene operation. For instance, new objects can be added in reasonable places without the user having to fully specify the position. A great avenue for future work would lie in enhancing the scope of the commands and in incorporating user feedback so that the system can learn and improve over time.

7.1 Scene operations

Once a scene is generated, the user can view the scene and manipulate it using both textual commands and mouse interaction. The system we have already described in this thesis supports traditional 3D scene interaction mechanisms such as navigating the viewpoint with mouse and keyboard, and selection and movement of object models by clicking. Now

There is a desk and a computer.

There is a living room with a red couch and a TV.

There is a desk with a chair and a poster. On the desk there is a red stapler.

Figure 7.1: Example generated scenes for three different input descriptions. These scenes can serve as good starting point candidates for further manipulation through natural language commands.

Verb	Operation	Example text	Example parse	Influenced parameters
select	SELECT	select the chair right of the table	$Select(\{\text{lamp}\},\{\text{right(lamp,table)}\})$	*Sel*
look,look at	LOOKAT	look at the lamp	$LookAt(\{\text{lamp}\})$	*Cam*
add,insert,place,put	INSERT	add a lamp to the table	$Insert(\{\text{lamp}\},\{\text{on(lamp,table)}\})$	$+object$
delete,remove	REMOVE	remove the lamp	$Remove(\{\text{lamp}\})$	$-object$
replace	REPLACE	replace the lamp with a vase	$Replace(\{\text{lamp}\},\{\text{vase}\})$	$+object, -object$
move, place, put	MOVE	move the chair to the left	$Move(\{\text{chair}\},\{\text{left(chair)}\})$	$pos, parent_{sup}, surf_{sup}$
enlarge, shrink	SCALE	enlarge the bowl	$Scale(\{\text{bowl}\})(1.5)$	σ

Table 7.1: Scene operations defined for our system. The natural language verbs are parsed to specific operations and subsets of the scene parameters are modified accordingly (last column).

we focus on enabling a user to give simple textual commands for selecting and modifying objects, or refining the scene.

To allow the user to interact with a generated scene via text, we define a set of high-level semantic operations for scene manipulation. These are higher level operations than typically found in traditional scene editing software. The text is treated as a set of constraints that we want the revised scene to satisfy, while trying to keep the revised scene as similar to the original scene as possible. More formally, given the original scene S and a scene operation O, we want to find the scene S' which is most similar to S, while still satisfying the constraints imposed by O.

To track the other elements of the scene, we maintain a scene state $Z = (S, \textit{Sel}, \textit{Cam})$ that consists of the scene S, the set of selected objects *Sel*, and the camera position *Cam*. Each operation is defined as a function $O : Z \rightarrow Z'$. The set of supported basic operations is summarized below (see also Table 7.1):

- SELECT changes the set of selected objects,
- INSERT adds objects into the scene,
- DELETE removes objects from the scene,
- REPLACE replaces objects in the scene with new objects,
- MOVE moves the set of selected objects,
- SCALE resizes the set of selected objects,
- LOOKAT repositions the camera to focus on the selected objects.

These basic operations demonstrate some simple scene manipulations through natural language. This set of operations can be extended, for example, to cover manipulation of parts of objects ("make the seat of the chair red").

To interpret a textual scene interaction command, the system first parses the input text u into sequence of scene operations $(O_1, \ldots, O_k)$, identifying the resulting set of constraints that should hold. For each parsed scene operation O_i, the system then executes the scene operation by resolving the set of objects on which the operation should be performed and then modifying the scene state accordingly.

Command parsing We deterministically map verbs to possible actions as shown in Table 7.1. Multiple actions are possible for some verbs (e.g., "place" and "put" can refer to either MOVE or INSERT). To differentiate, we assume new objects are introduced with the indefinite article "a" whereas old ones are modified with the definite article "the".

Object resolution To allow interaction with the scene, we must resolve references to objects within a scene. Objects are disambiguated by category and view-centric spatial relations. In addition to matching objects by their categories, we use the WordNet hierarchy to handle hyponym or hypernym referents. For example, if an object category is not found in the scene, then the WordNet hierarchy is traversed up the hypernym edges until a match is found (see Figure 7.2). This is a fairly simplistic approach that works well for many cases but it is prone to failure. We did not focus on this subtask of referring expression resolution since it is partially addressed in prior work such as Golland et al. (2010). A more advanced approach would be to explore leveraging similarities on word embeddings to identify the most likely referent for a given input term.

Another important part of the object resolution is the influence of the current view. Depending on how the user is viewing the scene, spatial relations such as "left" or "right" can refer to different objects (as illustrated in Figure 7.2). Here we can use the set of predefined spatial relations or the set of learned spatial relations presented in Section 6.1.2. For either the predefined or learned relations, we can just evaluate the probability that the relation holds given a candidate pair of target and reference objects.

In Figure 7.2 (left), the user can select a chair to the right of the table using the phrase "chair to the right of the table" or "object to the right of the table". The user can then change their viewpoint by rotating and moving around. Since spatial relations are resolved with respect to the current viewpoint, a different chair is selected for the same phrase from a different viewpoint in the right screenshot.

Camera positioning For the LOOKAT command, we first identify the set of objects being referred to. Given an utterance u of the form "look at X", we get the set of objects that are the most likely referent $Sel = \arg\max_{o \in S} P(o|X)$. The system then selects a optimal

Figure 7.2: Example of view-centric selection. **Left**: chair is selected using "the chair to the right of the table" or "the object to the right of the table". Chair is not selected for "the cup to the right of the table". **Right**: Different view results in different chair being selected for the input "the chair to the right of the table".

viewpoint for looking at Sel. Viewpoint selection is an active area of research in graphics (Feixas et al., 2009). In this work, we perform a simple viewpoint optimization where we sample camera positions c and find the position that maximizes a view function $f(c)$, giving us $Cam_{pos} = \arg\max_c f(c)$.

Camera positions are sampled from 12 equally spaced points around the up-axis of the selected objects and at a fixed height slightly above the bounding box of the objects. The camera is targeted at the centroid point of the selected objects. The view function we use is: $f(c) = vis_{Sel}(c) + b(scr_{Sel}(c)) + vis_{all}(c)$ where vis_{Sel} is the number of selected objects visible, vis_{all} is the percentage of all objects visible, and b is a function of scr_{Sel}[1], the percent of the screen that is taken up by the selected objects.

This is a simple approach for camera positioning which is challenging to perform manually for novice users of 3D design software and yet critical for navigating and interacting with the environment.

Scene modification Based on the operation, we modify the scene by maximizing the probability of a new scene template given the requested action and previous scene template. The set of objects in the scene is modified according to the operation:

- INSERT: Select a new object that satisfies the constraints from the model database

[1] linear ramp from $b(0.2) = 0$ to $b(0.4) = 1$

Figure 7.3: Example of LOOKAT operation. **Left**: initial scene. **Right**: after input "Look at vase", the camera zooms to the flower vase and the vase is selected (green highlight).

and place it in the scene.

- REPLACE: Select a new object that satisfies the constraints from the model database and replace the objects to be replaced with the new object.
- DELETE: Remove the old objects from the scene.

After the set of objects is determined, re-layout is performed on the scene by attempting to satisfy the constraints while minimizing the change in position for all unmodified objects. For operations such as MOVE and SCALE the set of objects remain the same but their position or size will need to change to satisfy the new constraints. When doing relayout, we do not vary all parameters, but just the set of influenced parameters for the operation (see Table 7.1).

Figure 7.4 shows a series of INSERT operations that each leverage the spatial priors of child attachment surface and support parent surface to determine where each newly inserted object should be placed within the scene. Without the learned spatial priors it is hard to resolve automatically from the simpe "add an X" commands where each of the inserted objects should go. As a consequence, in traditional scene design UIs these objects would each have to be manually oriented and positioned within the room.

In general, depending on the operation, operand objects, and their scene context, the resulting changes in the scene vary in complexity. The simplest operations only influence the target objects while more complex operations require adjustment of supported objects

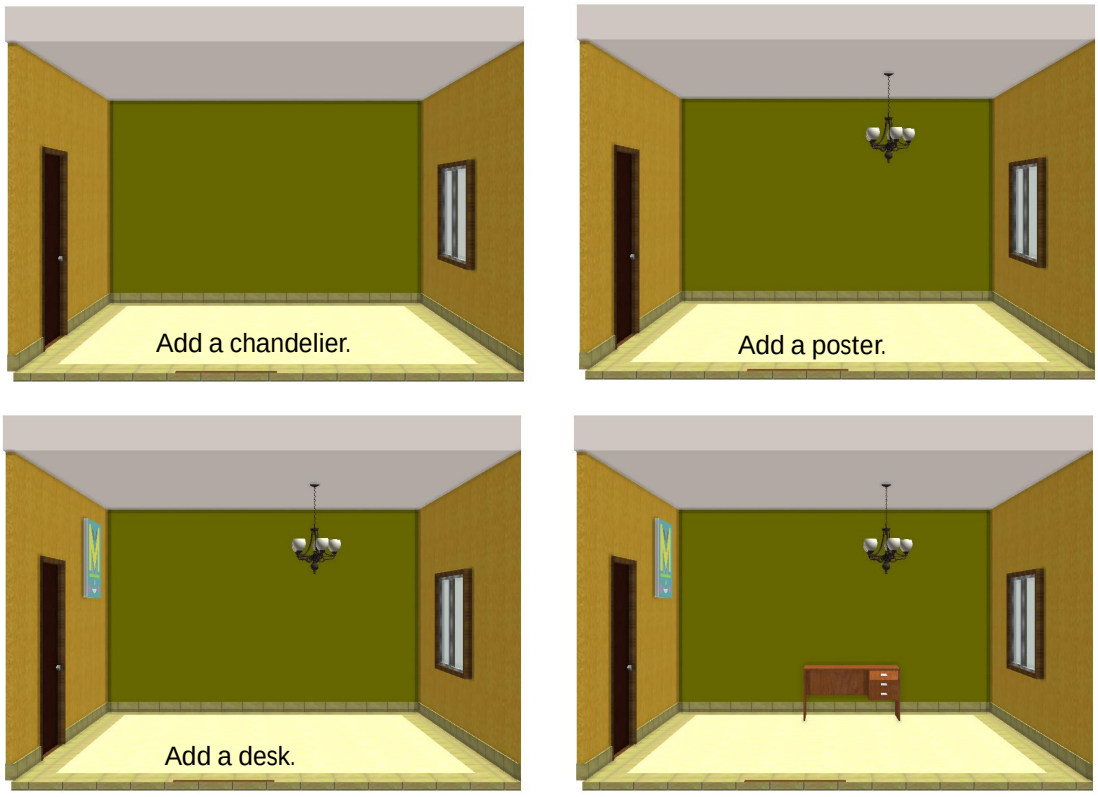


Figure 7.4: Examples of a sequence of INSERT operation illustrating the importance of attachment surface and support surface. **Top Left**: Start with empty room. **Top Right**: After "Add a chandelier". **Bottom Left**: After "Add a poster". **Bottom Right**: After "Add a desk".

Figure 7.5: Examples of REPLACE operation. **Top**: "Replace the bowl with a red lamp". **Middle**: "Replace the table with a coffee table". **Bottom**: "Replace the table with a round table".

or even surrounding objects. Figure 7.5 shows examples of the REPLACE operation with varying complexity. Replacing the bowl on the table with a table lamp is relatively easy since the system only needs to ensure the attachment point remains the same. However, when we replace the table with a coffee table in Figure 7.5 (middle), the bowl needs to be repositioned. In the bottom row of the figure we see a failure case due to the layout algorithm since the chairs are not repositioned to accommodate the bigger round table. Instead the table is pushed to the side to avoid collisions. Manually performing a replace operation can be extremely tedious due to such object position dependencies, thus making this a valuable high-level operation. This illustrates how our approach attempts to bridge the semantic gap between high-level intent and low-level geometric manipulation. In our implementation, the scene operations are initiated by the user using text commands. It is also possible to incorporate these operations into a graphical user interface. For instance, REPLACE can be implemented in a GUI as a sequence of clicks to select an object, search from a model database, select a desired model to use as a replacement.

7.2 Scene refinement examples

With the set of high-level operations that we defined, a user can progressively refine a generated scene. Figure 7.6 illustrates the design process where a user provides our text to scene system with a simple description of a scene. The system then generates several candidate scenes, from which the user selects one they like. Using the selected scene as the basis, the user can then use the GUI or issue a sequence of language based scene manipulate commands to further refine the scene. Figure 7.7 shows a sequence of operations for adding a rug to the living room we generated earlier. Combined with traditional mouse and keyboard navigation, these textual interactions allow high-level scene editing operations that are interpreted within the current viewing context (e.g., moving the rug to the back of the room from the viewer's perspective in the third panel). Without such high-level operations, the necessary interaction would include several steps to find a rug model, place it, orient it, and manually adjust its position.

This manipulation sequence clearly demonstrates the importance of prior knowledge. With the learned static support priors we were able to determine that the rug should be

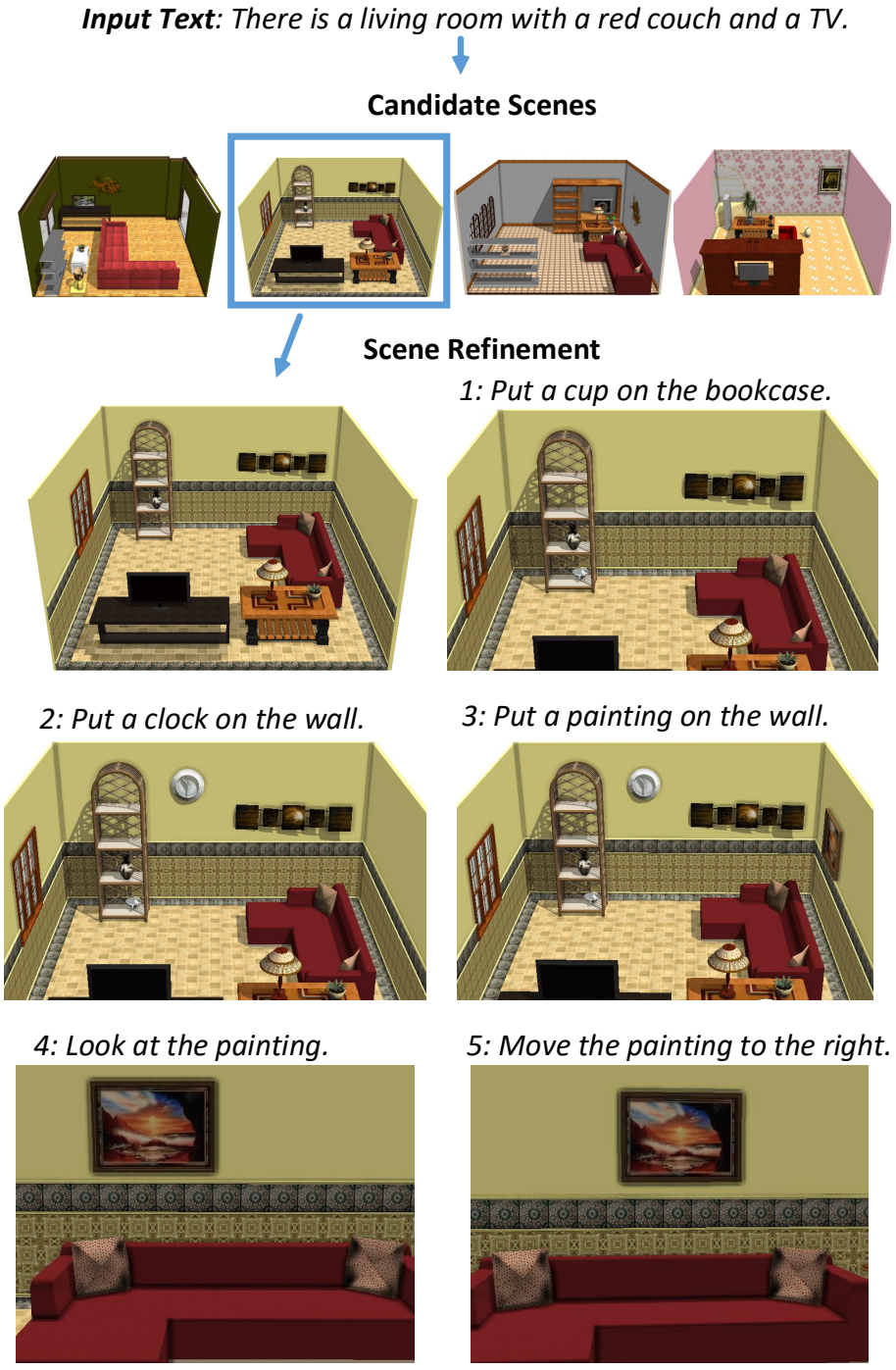


Figure 7.6: A sequence of language-based interactions to refine a living room scene

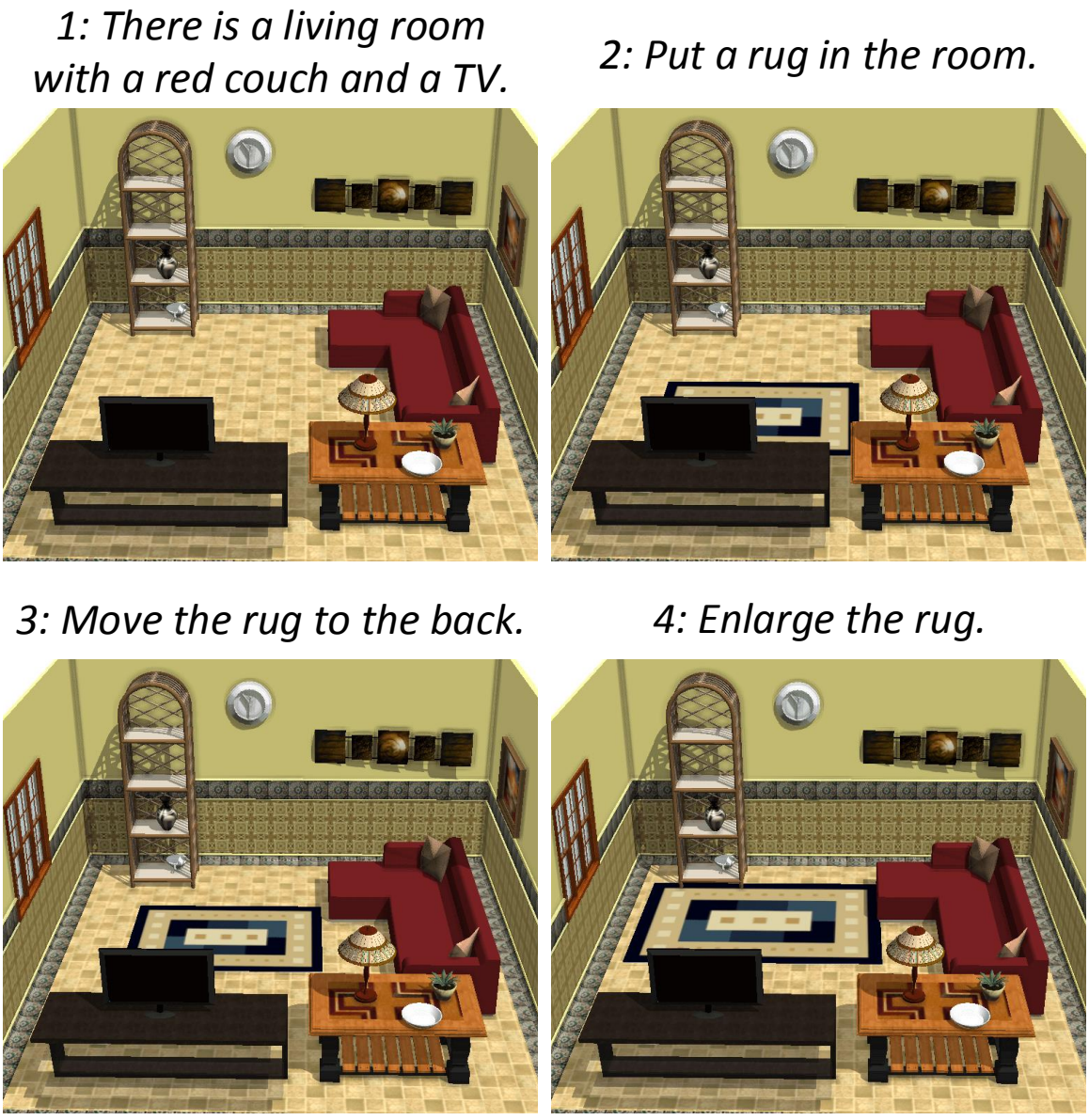


Figure 7.7: Further language-based interactions with a refined 3D scene

placed on the floor even though that was not explicitly stated. Furthermore, the rug was sized to have reasonable dimensions in relation to the room, and the command to make the rug "slightly larger" so it looks better can be resolved to a rescaling of the dimensions. When placing the clock and painting on the wall we were also able to orient the objects appropriately such that the front side faces out and their positions on the wall are reasonable. Note that this is done without having manually annotated any specialized spatial tags on the models.

7.3 Semantic scene querying

So far we have been using the learned spatial knowledge as a set of priors for guiding scene generation. However, there is an alternative view of this information that is closely tied to interactive scene manipulation. We can also view the spatial knowledge priors as information that allows us to answer questions such as "where does the cup go in the kitchen?", or "is the TV on top of the TV stand?", or "is the clock to the right of the bookshelf".

In Chapter 4, we showed how the priors allowed us to answer questions such as "On what surface is a floor desk likely to go? What about a desk lamp". Or given a scene, "Where should the chair go with respect to the desk? What about the mouse?". By combining the different learned spatial priors, we can also directly answer the question of where is an object X likely to go in a scene. In Figure 7.8, we show some results of querying the system for likely locations for a "poster", a "rug", a "floor lamp", and a "hat" in a scene.

This view presents a different problem statement that is close to a recent line of work in systems for visual question answering (Antol et al., 2015; Zhu et al., 2015b). Though this problem was not a focus of this thesis, it is likely that a question answering system for 3D scenes could be built with a very similar pipeline to the one presented here. This direction, as well as others we will discuss next constitute exciting future research opportunities.

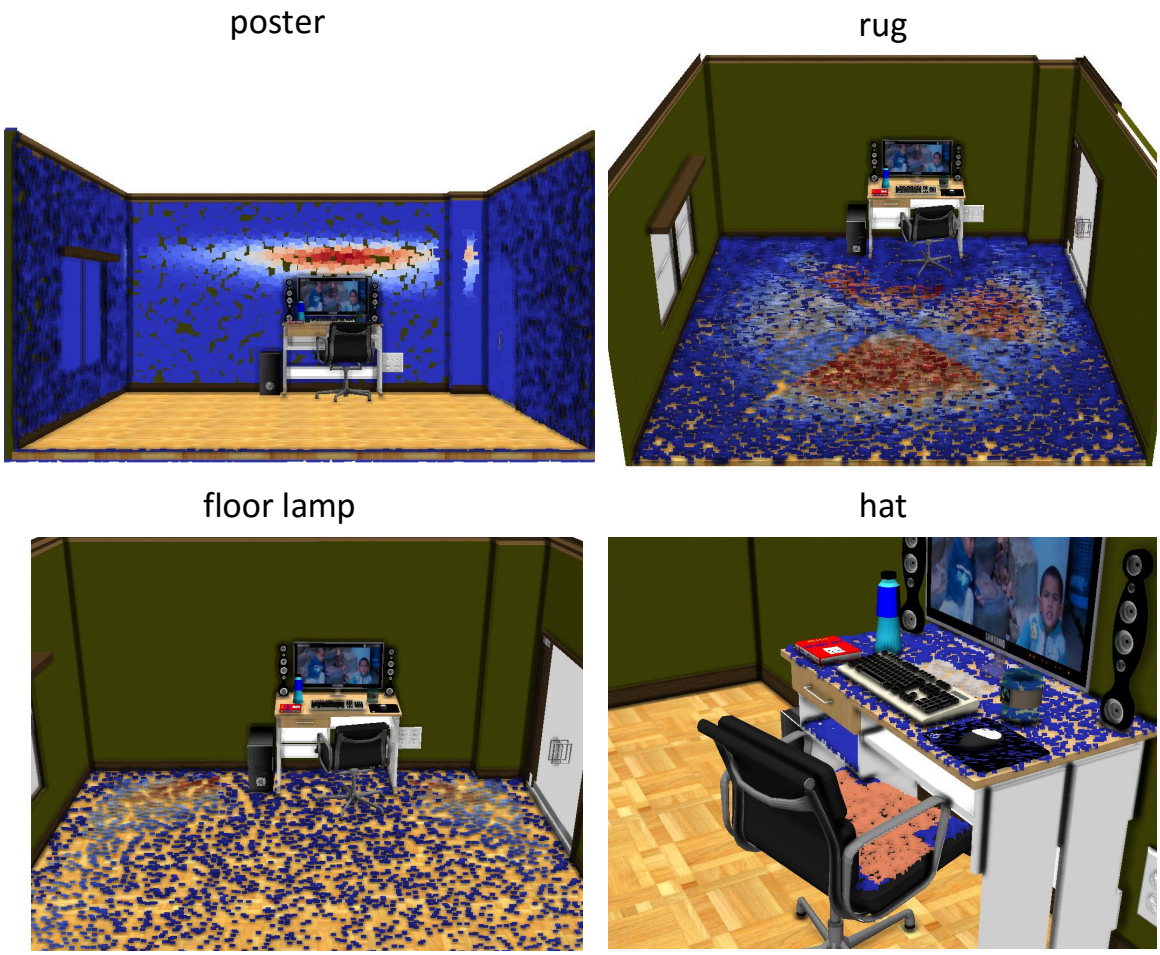


Figure 7.8: Use our learned priors to query likely positions for different types of objects

7.4 Conclusion

In the chapter, I have described how the text to scene system presented in this thesis can be augmented to interactively handle language-based commands that manipulate the state of given 3D scenes. This is preliminary work that suggests many avenues for future research. One exciting possibility is to integrate with a full dialog system that can carry out a conversation with the user asking for clarifications when a command is ambiguous, and cooperating in real-time to assist the user in designing the a scene. A related extension is to integrate with a speech to text systems that would allow for voice-driven, natural dialogue interactions. This could be particularly relevant in the context of increasingly popular virtual and augmented reality systems where text input is more cumbersome.

A dialogue-based system can also allow the user to correct the system by explicitly indicating failures and providing corrections, allowing the system to interactively learn and improve over time. Leveraging this user interaction information to improve spatial knowledge is another exciting avenue for future work. For instance, by observing where the user decides to manually place objects, we can improve our placement priors.

If such an interactive, self-improving system is implemented, crowdsourcing the accumulation of spatial knowledge by scaling to web-based platforms would be another exciting research opportunity. In addition to the obvious benefits for the size and quality of the spatial knowledge priors, this would also provide us an opportunity for broader user studies that can give insight into context-specific preferences for text versus direct manipulation interactions, and provide useful data for informing the design of future text-to-scene systems.

In the following chapter, I conclude this thesis with a summary of contributions, and a more detailed discussion of potential future work.

Chapter 8

Conclusions

In this thesis, I have presented a text to 3D scene generation framework. The system implementation was demonstrated in the domain of common indoor scenes which are predominantly composed of furniture and other manmade artifacts so that existing corpora of 3D models and 3D scenes could be leveraged. The general problem of text to 3D scene generation was described and decomposed into several well-defined subtasks.

8.1 Summary of contributions

This thesis presented the following contributions:

- Proposed a probabilistic framework for text to scene generation that incorporates spatial priors learned from 3D scene data.
- Defined representations for scene structure on which inference can be performed during text to scene generation to recover implicit facts.
- Defined a set of subtasks in the text to 3D scene generation problem that can be decoupled and addressed in isolation.
- Presented a model for encoding spatial knowledge as a set of priors extracted from 3D scenes and natural language descriptions of the scenes.
- Demonstrated how to ground language referring to objects within scenes to 3D models, and how to ground spatial relation terms to geometric relations.

- Shown how to handle parsing and grounding of natural language commands for interactive manipulation of 3D scenes.

This thesis is only a small step towards enabling general, robust text to 3D scene generation. There are many directions for future research which we discuss in the following section.

8.2 Future work

From the starting point of this thesis we can seek to improve each of the main components of the text to scene system. In particular, handling of the language input, better modeling of spatial knowledge priors, and better handling of more complex and diverse scenes are three obvious directions for improvement.

8.2.1 Handling language complexity

The space of human language is much more complex than the language we have been dealing with as input in this thesis. Not only is there an axis of linguistic complexity along which we can try to improve our system's capabilities, there is also the issue of handling different languages themselves. This space of language complexity is illustrated in Figure 8.1. In order to handle multiple languages we would certainly want to leverage data-driven, and corpus-based approaches which can generalize beyond rules that are specifically designed to work in the context of one language and may not easily generalize to others. In leveraging data-driven approaches we also need to tackle the issue of determining which parts of natural language scene descriptions are relevant to the text to scene task, and which can be disregarded. This issue is exemplified by the descriptive essay of "my bedroom" provided by a young student.

8.2.2 Better spatial knowledge priors

Incorporating a richer model of how the geometric and functional contexts of object arrangements influence interpretation of spatial terms (Carlson and Kenny, 2005) can help us to build better spatial knowledge priors that are closer to human intuition.

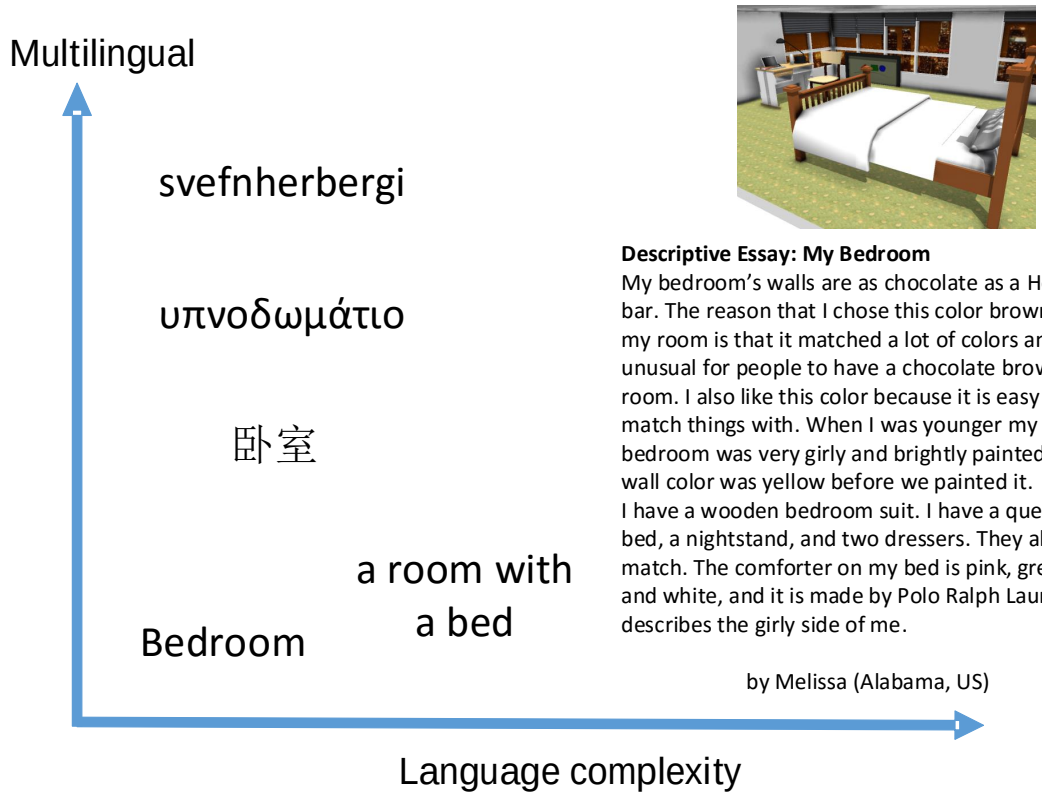


Figure 8.1: The challenging space of language In the horizontal, language is more and more complex and elaborate in its description of scenes. In the vertical we have many different languages which can also rely on different conceptualizations of spatial structure.

In addition to improving the model used for learning spatial priors from 3D scene data, we can also explore learning of scene priors from other visual modalities such as 2D images, or language not directly associated with describing scenes. There is much recent work in cross-modal learning between images and language (e.g., Fang et al. (2015)). Since there is currently much more image and text data available for analysis than 3D scene data, it would behoove us to try to extract more spatial knowledge from these more expansive modalities.

8.2.3 Handling scene complexity

This thesis has focused almost exclusively on common indoor scenes where the objects are mostly furniture and other manmade artifacts. This was prudent to do as a first step since most prior work in automatic scene layout was done in this domain so most 3D scene data available for research is again restricted to this domain. Future work could explore how interior scenes that exhibit larger scale structure than single small rooms (e.g., restaurants, lecture halls) can be handled (see Figure 8.2). In addition, there is a broad class of outdoor environments that may require significantly different principles for capturing spatial priors and common object co-occurrences. A notion of hierarchical organization at multiple scales may prove to be critical for expansive outdoor environments.

The approach we used for handling scene generation is based on a simple greedy sampling scheme which can be problematic especially in cases of complex scene layouts with long range relations and relations between many objects. An obvious way to improve the scene generation results is to use a more advanced sampling method to find out high likelihood scene layouts even with complicated constraints and many objects.

Another important limitation of this thesis is that it only considers static 3D scenes. Modeling dynamic 3D scenes with moving objects or agents will require a better understanding of physics and intuitive causality. Beyond the purely physical and geometric interpretation of how behaviors affect 3D scenes, we would also need to have an understanding of implied human emotions and their impact on visible behaviors. For example, there are important differences in trying to visualize the action of two people “chatting” vs. the action of two people “bickering”.

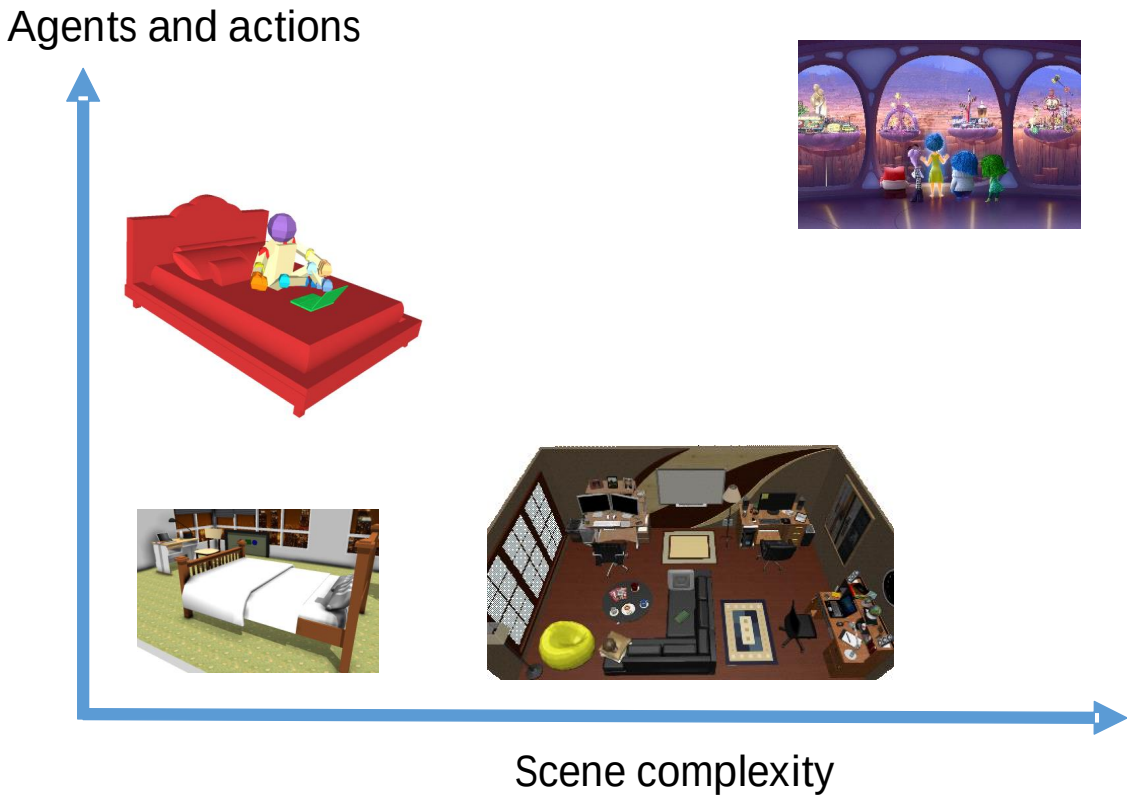


Figure 8.2: The space of scene and action complexity In the horizontal scenes are incresingly more complex in terms of their static structure, while in the vertical scenes are more complex in terms of their dynamics and behaviors of characters acting within them.

8.2.4 Improved integrative model

A very promising direction for improving how scene descriptions and commands should be interpreted is to combine the meaning of the utterance, the context, and the prior knowledge about the world in a joint model that explicitly model the influence of pragmatics. For instance, a more advanced speaker listener model than the "literal listener" that this thesis implicitly assumed can allow us to better extract information from implicatures that are dictated by both the linguistic and concrete 3D scene context. Models that take into account the communicative goals of the speaker such as the rational speech acts model (Goodman and Stuhlmüller, 2013) are good candidates for exploring this direction.

8.2.5 Other directions

An alternative research avenue to the one presented in this thesis would try to bypass the need for representing scenes in 3D, instead focusing on directly translating text to 2D images. This has recently been demonstrated to be possible in some scenarios through generative neural network models that can produce images (Gregor et al., 2015; Lazaridou et al., 2015; Mansimov et al., 2015). However, these approaches are severely restricted in the resolution and complexity of the images that they can generate.

Another exciting direction for future research is to consider learning of spatial knowledge priors directly from observations of how people interact with real environments. Recently, there has been some research into learning how to evaluate the degree to which particular locations in a scene allow people to perform common actions (Savva et al., 2014b). We could try to leverage similar approaches to automatically extract semantic attributes for objects that people are observed to interact with, or to obtain priors on the saliency of different object relations for allowing particular actions to take place.

Farther into the future, we could expand on the ideas presented in this thesis to address the text to animation task. Though prior work has attempted to address this problem in a limited fashion, there is much more research to be done in order to understand and generate the patterns of dynamic human behaviors and the state changes that result in the 3D environments within which actions take place. Our focus on static 3D scenes has already shown that there is much world knowledge that we need to address in just one slice in time and

the issue is of course more complex as soon as the temporal dimension has to be modeled as well.

Appendix A

Seed sentences

List of the 60 seed sentences that we used to create the scene and description corpus.

There is a desk and a chair.

There is a desk and a computer.

There is a desk and a laptop.

There is a desk and a lamp.

There is a desk and a poster.

There is a desk and a notepad.

There is a desk and a keyboard and a monitor.

There is a desk and a monitor and a mouse.

There is a desk and a monitor and a keyboard and a mouse.

There is a desk and there is a notepad on the desk. There is a pen next to the notepad.

There is a desk and there is a notepad on the desk. There is a pen on the notepad.

There is a desk and there is a stapler on the desk. There is a notepad to the left of the stapler.

There is a desk and there is a stapler on the desk. There is a keyboard to the right of the stapler.

There is a table and a chair.

There is a table and there is a plate on the table.

There is a table and there are two plates on the table.

There is a table and there is a plate. There is a sandwich on the plate.

There is a table and there is a sandwich.

There is a table and there is a lamp.

There is a table and there is a chair to the left of the table.

There is a table and there is a chair to the right of the table.

There is a table and there is a lamp on the table.

There is a table and there are two chairs.

There is a table and there are four chairs.

There is a table and there are four chairs. There are four plates and there are four sandwiches.

There is a table and there is a chair to the left of the table. There is a chair to the right of the table.

There is a table and there is a chair in front of the table. There is a chair behind the table.

There is a table and there is a chair to the left of the table. There is a chair to the right of the table. There is a chair in front of the table. There is a chair behind the table.

There is a bed next to the wall.
There is a bed and there is a laptop on the bed.
There is a bed and there is a pillow on the bed.
There is a bed and there is a nightstand next to the bed.
There is a bed and there is a nightstand next to the bed. There is a floor lamp next to the nightstand.
There is a bed and there is a nightstand next to the bed. There is a floor lamp next to the bed.
There is a bed and there is a nightstand next to the bed. There is a lamp on the nightstand.
There is a bed and there is a chair next to the bed.
There is a coffee table.
There is a coffee table and there is a vase on the coffee table.
There is a coffee table and there is a couch behind the coffee table.
There is a coffee table and there is a couch behind the coffee table. There is a vase.
There is a coffee table and there is a couch behind the coffee table. There is a vase on the coffee table.
There is a coffee table and there is a couch. There is a chair next to the coffee table.
There is a couch. There is a vase.
There is a couch. There is a coffee table and there is a plant.
There is a couch. There is a coffee table and there is a plant on the coffee table.
There is a couch and there is a pillow on the couch. There is a chair next to the couch.
There is a lamp and there is a chair next to the lamp. There is a coffee table next to the chair.
There is a coffee table and there is a lamp behind the coffee table. There is a chair in front of the coffee table.
There is a desk and there is a lamp next to the desk. There is a chair in front of the desk.
There is a desk and there is a lamp next to the desk. There is a chair in front of the desk. There is a trash bin next to the desk.
There is a bookcase and a couch.
There is a bookcase and a desk.
There is a bookcase and a vase on the bookcase. There is also a desk next to the bookcase.
There is a bookcase and a couch. There is a plant by the couch.
There is a couch and a coffee table. There is a cup on the coffee table.
There is a bookcase and a couch and a coffee table. There is a cup on the bookcase.
There is a table with candles and a bowl of fruit.
There is a couch and a coffee table. There is a bowl of fruit on the coffee table.
There is a couch and a coffee table. There is a trash bin next to the couch.
There is a desk with a lamp and a bowl of fruit.

Appendix B

Scene taxonomy

Using the scene names from Fisher et al. (2012)'s dataset of 133 indoor scenes, I assign scene types and create the basic scene taxonomy below, along with the number of scenes for each scene type. The dataset consists of simple scenes with localized structure (Desk, Table, Dresser, Bookshelf, etc) as well as some more complex scenes that shows a entire room arrangement (LivingRoom, Study, Bedroom, etc).

- Laboratory: 5.0
- Studio: 1.0
- Nightstand: 4.0
- EntertainmentCenter: 9.0
- Counter: 5.0
  - KitchenCounter: 5.0
- Desk: 36.0
  - ComputerDesk: 16.0
  - StudyDesk: 6.0
  - OfficeDesk: 9.0
    - BarrenOfficeDesk: 1.0
  - GamerDesk: 5.0
    - BarrenGamerDesk: 1.0
- Room: 26.0
  - LivingRoom: 4.0
    - EuropeanLivingRoom: 1.0
  - Kitchen: 4.0
    - AfricanKitchen: 1.0
    - EuropeanKitchen: 1.0
  - LaundryRoom: 3.0
  - Study: 6.0
  - Bathroom: 1.0
  - GameRoom: 1.0
  - CleanRoom: 1.0
  - Bedroom: 6.0
    - BarbieBedroom: 1.0
    - KidBedroom: 1.0
    - ArtBedroom: 1.0
- Hood: 3.0
  - FumeHood: 3.0
- Bed: 3.0
- Table: 25.0
  - DiningTable: 12.0
    - BarrenDiningTable: 1.0
  - LivingRoomTable: 3.0
  - LabTable: 7.0
  - CoffeeTable: 3.0
- EntertainmentCenterWithSofa: 4.0
  - EuropeanEntertainmentCenter: 1.0
- Dresser: 2.0
- Bookshelf: 10.0

Appendix C

Surface segmentation and feature extraction

Here I describe some of the simple geometric processing I used to automatically extract surfaces and surface features. In this thesis, I work mainly with models that are represented as triangle based mesh surfaces. The representation necessitates some preprocessing to identify candidate support surfaces.

Segmentation Models are segmented using the SuperFace algorithm (Kalvin and Taylor, 1996). In this algorithm, we take the a set of triangles in a mesh and cluster them into surfaces by considering the similarity of the normals. The largest unclustered triangle is selected, and the surface cluster grows by considering adjacent triangles that have a normal that is sufficiently similar to its neighbor (i.e. the cosine similarity is greater than some threshold). This process is repeated iteratively until all triangles have been clustered.

When building these surface clusters, we make the following assumptions:

- Each model is represented as a set of triangle meshes. We assume that each triangular mesh is semantically meaningful, and thus only run the segmenting algorithm on each mesh (i.e. a surface will not cross meshes).
- Triangles are adjacent if they have a overlapping edges with a shared vertex. This allows for efficient processing of the triangles. Note that this misses edges cases with

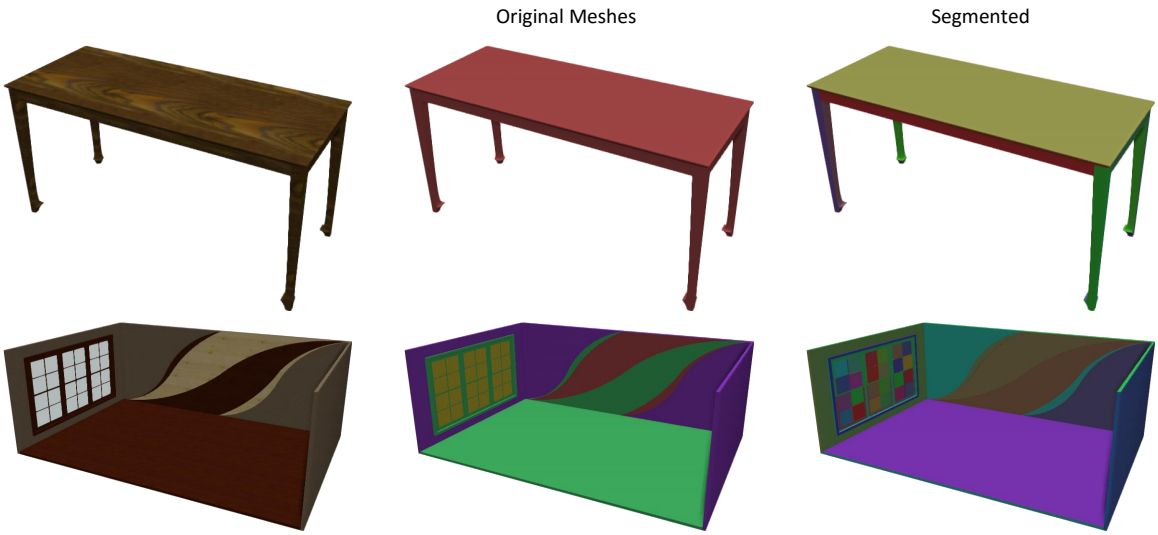


Figure C.1: Comparison of over segmented model vs original mesh Colors indicate different meshes or segments.

adjacent triangles that do not share any vertices. We believe this to be a rare case in geometric meshes used in modeling objects.

- Support surfaces are assumed to be planar, so when identifying support surfaces, only adjacent triangles whose normal is sufficiently close to the overall normal of the plane is considered.

Figure C.1 shows some example of how models are segmented using this algorithm. Notice that the table model is represented as one mesh and that the segmentation allows us to differentiate the top surface from the rest of the model. In contrast, the room model was already represented by several meshes. Our segmentation allows us to different between the different walls. However, because the back wall is represented as several meshes, we fail to identify it as one contiguous surface.

Surface direction Each surface is associated with a surface normal. I use the surface normal to determine if the surface is upward, downward, or lateral facing.

Inside and outside surfaces Consider a model of a room. Example: a single wall maybe modeled using four surfaces. Two surfaces facing in and two surfaces facing out. The surfaces facing out are considered external surfaces while the surfaces facing in are considered internal surfaces. In this thesis, I only allow positioning of objects on external surfaces. As the wall is part of a larger object, the room, I consider the external surface that is facing into the room a interior surface while the external surface facing outside of the room is considered a exterior surface. Figure C.2 shows the difference between internal and external surfaces for a wall, and the difference between an interior and exterior wall.

To summarize, in this work, I differentiate between the following types of surfaces:

- Internal vs External: Internal surface is a surface that is facing inside the model and will never be visible to the user (like the inside of a wall).
- Interior vs Exterior: Interior surface is a external surface that is facing toward interior of the model.

Whether a surface is internal vs external is identified by sampling the surface and raytracing outwards and colliding with other parts of the object. A surface is an internal surface if more than half the rays intersect the object at a distance that is less than some threshold θ_a. In this thesis, I used $\theta_a = 0.1|n \bullet d|$ where n is the normal of the surface, and d are the extents of the axis aligned bounding box of the object.

Whether a surface is interior vs exterior is identified in a similar fashion. In this case, by sampling the surface raytracing outwards and colliding with the axis aligned bounding box of the object. A surface is an interior surface if more than half the rays intersect the object at a distance that is greater than some threshold θ_b. In this thesis, I used $\theta_b = 0.05x$ where x is the maximum extent of the axis aligned bounding box of the object.

These are very simplistic methods for identifying whether a surface is internal/external, and interior/exterior. More sophisticated methods can be used to improve the identification of candidate support surfaces.

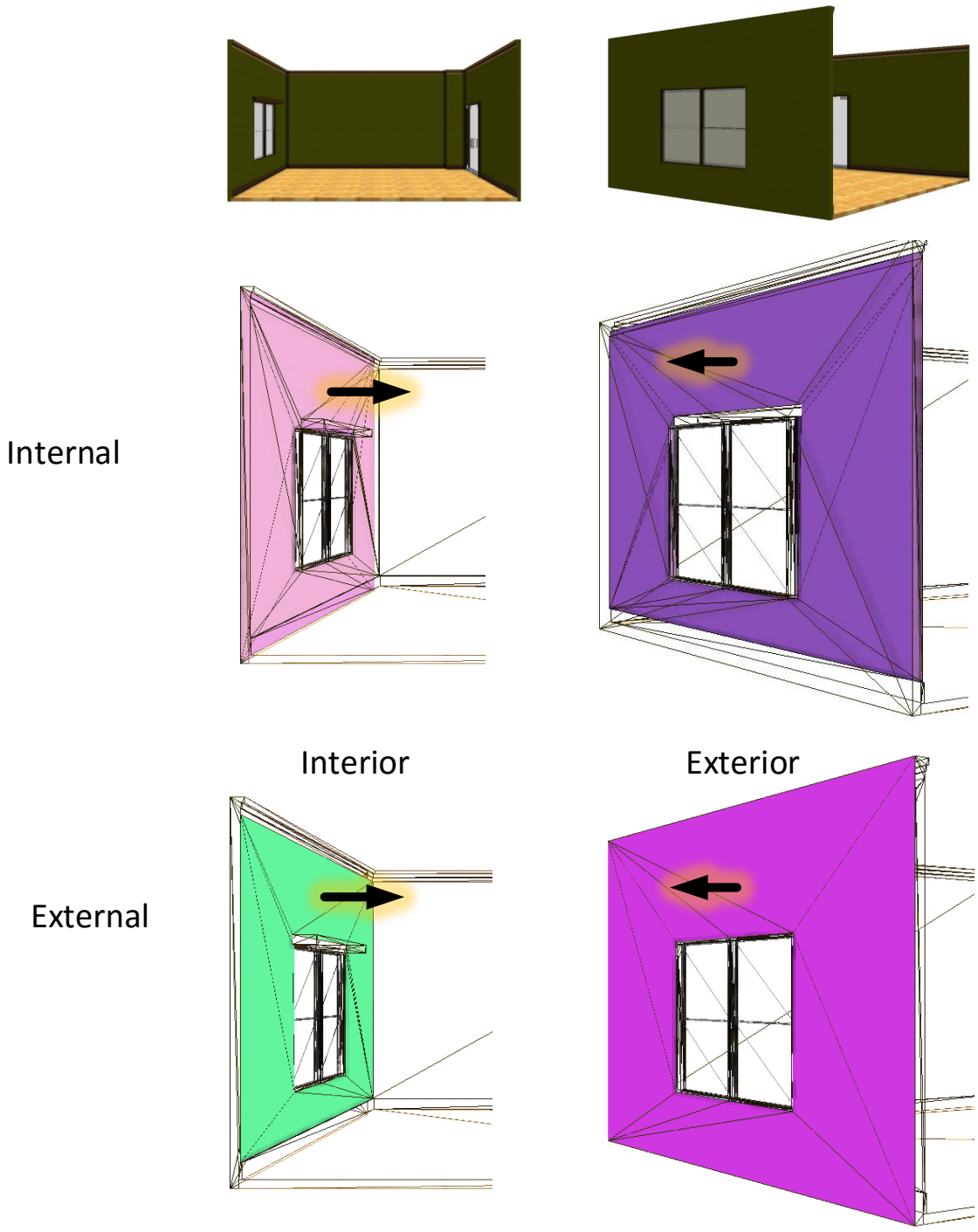


Figure C.2: Different surfaces of a wall. Here we illustrate the difference between internal and external surfaces, and between interior and exterior walls.

Appendix D

Word lists

Word lists extracted from WordNet 3.1

Colors Color terms were extracted by taking all hyponyms under the nouns "chromatic color" and "achromatic color". The full list of colors is below:

chromatic color, chromatic colour, spectral color, spectral colour, red, redness, sanguine, chrome red, Turkey red, alizarine red, cardinal, carmine, crimson, ruby, deep red, dark red, burgundy, claret, oxblood red, wine, wine-colored, wine-coloured, purplish red, purplish-red, magenta, fuschia, maroon, cerise, cherry, cherry red, scarlet, vermilion, orange red, orange, orangeness, reddish orange, tangerine, salmon, yellow, yellowness, canary yellow, canary, amber, gold, brownish yellow, gamboge, lemon, lemon yellow, maize, old gold, orange yellow, saffron, ocher, ochre, pale yellow, straw, wheat, greenish yellow, blond, blonde, green, greenness, viridity, greenishness, sea green, sage green, bottle green, chrome green, emerald, olive green, olive-green, yellow green, yellowish green, chartreuse, Paris green, pea green, bluish green, blue green, teal, cyan, jade green, jade, blue, blueness, azure, cerulean, sapphire, lazuline, sky-blue, powder blue, steel blue, Prussian blue, dark blue, navy, navy blue, greenish blue, aqua, aquamarine, turquoise, cobalt blue, peacock blue, purplish blue, royal blue, ultramarine, purple, purpleness, lavender, mauve, reddish purple, royal purple, Tyrian purple, violet, reddish blue, indigo, pink, pinkness, carnation, rose, rosiness, old rose, solferino, purplish pink, yellowish pink, apricot, peach, salmon pink, coral, brown, brownness, Vandyke brown, chestnut, chocolate, coffee, deep brown, umber, burnt umber, hazel, light brown, tan, topaz, dun, greyish brown, grayish brown, fawn, beige, ecru, mocha, reddish brown, sepia, burnt sienna, Venetian red, mahogany, brick red, copper, copper color, Indian red, yellowish brown, raw sienna, buff, caramel, caramel brown, snuff-color, snuff-colour, puce, olive brown, taupe, olive, olive drab, drab, pastel, complementary color, complementary, achromatic color, achromatic colour, black, blackness, inkiness, coal black, ebony, jet black, pitch black, sable, soot black, white, whiteness, alabaster, bleach, bone, ivory, pearl, off-white, chalk, frostiness, hoariness, gray, grayness, grey, greyness, ash grey, ash gray, silver, silver grey, silver gray,

charcoal, charcoal grey, charcoal gray, oxford grey, oxford gray, dapple-grey, dapple-gray, dappled-grey, dappled-gray, iron-grey, iron-gray, tattletale grey, tattletale gray, iron blue, steel grey, steel gray, Davy's grey, Davy's gray,

Patterns Pattern terms were extracted by taking all hyponyms under the adjective "patterned". The full list of patterns is below:

patterned, banded, black-and-tan, black-barred, black-marked, blotched, blotchy, splotched, brindled, brindle, brinded, tabby, brown-speckled, brownish-speckled, brown-striped, brownish-striped, burled, checked, checkered, chequered, cross-banded, dappled, mottled, dark-spotted, dotted, flecked, specked, speckled, stippled, figured, floral, flowered, freckled, lentiginous, lentiginose, laced, marbled, marbleized, marbleised, maroon-spotted, moire, watered, patched, spotty, spotted, pointillist, pointillistic, pinstriped, purple-veined, purple-spotted, red-striped, reddish-striped, red-streaked, ringed, slashed, sprigged, streaked, streaky, striped, stripy, tessellated, tiger-striped, veined, venose, veinlike, violet-streaked, white-blotched, white-ribbed, white-streaked, yellow-banded, yellow-marked, yellow-spotted, yellow-striped,

Materials Material terms were extracted by taking all hyponyms under the nouns "fabric" (excluding those under "piece" or "meshwork"), "rock", "animal material" (excluding those under "piece" or "organism"), "paper" (excluding those under "card" or "piece of paper"), "wood", "glass", "metal" (excluding atomic numbers), "alloy", "ceramic", and "plastic". The full list of materials is below:

silk, cord, Canton crepe, quilting, chintz, cotton flannel, shantung, metallic, whipcord, batiste, serge, tarpaulin, camelhair, velveteen, khaddar, Velcro, swan's down, vulcanized fiber, pique, pina cloth, screening, macrame, seersucker, camo, damask, satinette, buckram, nankeen, homespun, satinet, Bedford cord, denim, vicuna, herringbone, bombazine, Valenciennes, diaper, etamine, needlepoint, velour, frieze, satin, voile, cloth, nainsook, viscose, pepper-and-salt, gauze bandage, trousering, gunny, khadi, dungaree, hair, shirting, tricot, terry, fleece, crinoline, oilcloth, felt, suede, haircloth, burlap, print, waterproof, pongee, sponge cloth, terrycloth, balbriggan, shirttail, monk's cloth, macintosh, fiber, twill, terry cloth, tartan, sackcloth, ticking, horsehair, crepe de Chine, grogram, jean, cobweb, wincey, wide wale, leatherette, baize, Orlon, panting, sateen, chenille, rayon, viscose rayon, Terylene, mousseline de sole, camel's hair, marseille, russet, duffel, double knit, fustian, flannel, silesia, faille, Valenciennes lace, moquette, acetate rayon, camouflage, polyester, crape, flannelette, chino, camlet, brocade, duck, cerecloth, plush, towelling, stockinette, tapis, moleskin, aba, fibre, chambray, upholstery material, sarsenet, cambric, olive drab, motley, scrim, velours, elastic, sailcloth, web, fabric, gabardine, nylon, organdy, petrolatum gauze, woolen, linen, mackinaw, tapestry, grosgrain, spandex, knit, pillow lace, Viyella, repp, lame, lisle, stockinet, permanent press, huck, batik, stammel, wool, drapery, tarp, sacking, tweed, georgette, huckaback, taffeta, Aertex, foulard, diamante, tucker, tapa, Harris Tweed, madras, coating, jacquard, alpaca, watered-silk, Canton flannel, linsey-woolsey, doeskin, Dacron,

velvet, crepe, sheeting, challis, poplin, suede cloth, corduroy, gingham, percale, shag, pilot cloth, basket weave, wash-and-wear, Courtelle, duffle, yoke, filet, textile, organdie, woollen, pinstripe, hopsacking, material, sharkskin, moire, winceyette, mackintosh, marocain, jaconet, tappa, boucle, moreen, dimity, imitation leather, chiffon, khaki, hopsack, mohair, paisley, bagging, bobbin lace, rep, suiting, acrylic, samite, lace, organza, etamin, belting, tammy, worsted, muslin, chinchilla, cotton, durable press, Ultrasuede, jersey, lint, toweling, plaid, cretonne, wire cloth, acetate, narrow wale, sarcenet, webbing, broadcloth, canopy, crepe marocain, point lace, ninon, bunting, calico, cashmere, Brussels lace, rock, intrusion, rock outcrop, enterolith, sill, gallstone, bedrock, stone, cystolith, Plymouth Rock, crystallite, pebble, tor, concretion, whin, renal calculus, wall rock, chondrite, petrifaction, shore boulder, crystal, bowlder, achondrite, clastic rock, stepping stone, kidney stone, outcrop, bilestone, calculus, xenolith, boulder, urolith, salivary calculus, urinary calculus, outcropping, bladder stone, sialolith, belay, outthrust, glacial boulder, river boulder, whinstone, ptyalith, nephrolith, crystallization, broadtail, bearskin, silk, down, primary feather, mutton tallow, goose down, sable, bone oil, train oil, swan's down, sperm oil, shammy leather, wash leather, dubbin, acylglycerol, quill feather, beaver, stick lac, plume, vicuna, shell, ambergris, shellac, parchment, aftershaft, saddle feather, quill, Ghedda wax, animal material, garnet lac, tortoiseshell, kidskin, Persian lamb, oleo oil, fleece, beeswax, fell, horsehide, suede, animal fibre, cod-liver oil, seal oil, plumule, drying oil, white leather, morocco, shoe leather, leopard, porpoise oil, vellum, beef tallow, goose grease, fish oil, deerskin, horsehair, Levant morocco, fish-liver oil, Levant, baleen, bone fat, tail feather, nacre, triglyceride, alligator, scapular, shark oil, neat's-foot oil, dolphin oil, glyceride, lanolin, mother-of-pearl, plumage, butterfat, blubber, tuna oil, seal, osseous tissue, cowhide, pelt, bone, saddle hackle, Russia leather, ooze leather, animal fiber, dentin, dentine, calfskin, virgin wool, primary quill, otter, wool fat, cordovan, astrakhan, whale oil, contour feather, Sonora lac, pinion, halibut-liver oil, spurious wing, down feather, leather, mocha, raccoon, animal product, wool, alula, sickle feather, hudson seal, suede leather, hide, wool oil, tallow oil, Dippel's oil, duck down, muskrat fur, salmon oil, alpaca, fish glue, eiderdown, tusk, hackle, chamois, cod oil, doeskin, pigskin, fur, menhaden oil, crush, cod liver oil, whit leather, tallow, box calf, beaver fur, grain, lard oil, Shetland wool, shoddy, sardine oil, raw wool, Golden Fleece, lambskin, animal oil, goatskin, sealskin, animal fat, glove leather, chamois leather, whalebone, roan, ivory, bastard wing, rawhide, patent leather, horn, animal skin, feather, muskrat, shammy, chammy, crushed leather, ermine, mink, lac, wool grease, chinchilla, seed lac, chammy leather, primary, sheepskin, cowskin, shark-liver oil, animal glue, gum-lac, lapin, flight feather, cashmere, blotting paper, manilla paper, wove paper, manila, newspaper, strawboard, wrapping paper, decal, greaseproof paper, flimsy, oilpaper, manifold, toilet roll, blueprint paper, ticker tape, dimple, blotter, wallpaper, decalcomania, bond paper, paperboard, parchment, filter paper, manila paper, timecard, pad of paper, art paper, butcher paper, pasteboard, ledger paper, crepe paper, dimpled chad, computer paper, notepad, vellum, typing paper, tablet, cigarette paper, flypaper, paper-mache, typewriter paper, message pad, tar paper, carbon, scratch pad, olla, letter paper, letterhead, paper tape, india paper, construction paper, transfer paper, graph paper, pad, ola, roofing paper, posterboard, notepaper, drawing paper, rag paper, linen, confetti, trichad, rice paper, corrugated cardboard, kraft, tissue, score paper, Kleenex, tissue paper, composition board,

pregnant chad, poster board, kraft paper, cardboard, toilet tissue, corrugated board, paper toweling, binder board, paper, crepe, scratch paper, chad, carbon paper, millboard, bond, tracing paper, toilet paper, rolling paper, laid paper, manifold paper, music paper, wax paper, papyrus, binder's board, papier-mache, bathroom tissue, facial tissue, newsprint, scribbling block, hanging chad, cartridge paper, litmus paper, manilla, linen paper, gift wrap, onionskin, stationery, writing paper, Post-It, swinging chad, writing pad, waste paper, Port Jackson pine, ruby wood, sandarac, basswood, Juniperus horizontalis, California redwood, guaiacum, zebrawood, red cypress pine, sweet gum, bamboo, whitewood, hazelwood, locust, briarwood, Ahuehuete, Sequoia gigantea, linden, elm, larch, dwarf juniper, fir, Montezuma cypress, east African cedar, red cedar, logwood, knotty pine, cabinet wood, swamp cypress, birch, Bermuda cedar, Sequoiadendron giganteum, wicker, raw wood, driftwood, granadilla wood, poplar, cocoswood, teak, Juniperus communis depressa, Sierra redwood, amboyna, hardwood, boxwood, sapwood, mahogany, saw log, juniper, heartwood, spruce, pencil cedar tree, lignum vitae, silver quandong, ironwood, Juniperus silicicola, satinwood, Callitris glauca, Callitris endlicheri, poon, Taxodium distichum, lemonwood, Mexican swamp cypress, yew, chestnut, eastern red cedar, red lauan, elmwood, splinters, brazilwood, gumwood, obeche, Callitris parlatorei, Turkish boxwood, hazel, Callitris glaucophylla, Taxodium mucronatum, Callitris quadrivalvis, walnut, satin walnut, matchwood, savin, bald cypress, red sandalwood, pine, brushwood, redwood, white pine, eucalyptus, douglas fir, black cypress pine, drooping juniper, black locust, ebony, burl, Juniperus flaccida, ash, Sequoia Wellingtonia, cedarwood, big tree, cigar-box cedar, orangewood, red gum, citronwood, nurse log, Philippine mahogany, dyewood, pyinma, yellow pine, sandarac tree, cypress pine, lacewood, cherry, balsa wood, beechwood, knot, Panama redwood, cypress, alder, oak, sandalwood, pulpwood, sumac, guaiacum wood, sequoia, sabicu wood, hemlock, bird's-eye maple, Juniperus sabina, Tule tree, creeping juniper, Tetraclinis articulata, Callitris calcarata, Juniperus procera, dogwood, wood, true tulipwood, sabicu, fruitwood, pond bald cypress, hickory, guaiac wood, brierwood, Juniperus bermudiana, white cypress pine, tulipwood, Andaman redwood, sawdust, teakwood, southern cypress, blackwood, yellowwood, yellow poplar, Juniperus virginiana, tupelo, pencil cedar, white poplar, kingwood, cocuswood, incense wood, beefwood, ground cedar, quira, Taxodium ascendens, rosewood, Port Orford cedar, coast redwood, common juniper, gum, duramen, Mexican juniper, kauri, olive, holm oak, pond cypress, bentwood, maple, sycamore, deal, stringybark pine, balsa, beech, guaiac, applewood, log, cedar, fumed oak, red juniper, southern red cedar, lancewood, pecan, giant sequoia, Callitris cupressiformis, shittimwood, Sequoia sempervirens, softwood, Juniperus communis, brier-wood, optical crown glass, flint glass, opal glass, ground glass, lechatelierite, paste, quartz, optical glass, lead glass, wire glass, optical flint, water glass, crystal, shatterproof glass, quartz glass, soft glass, stained glass, milk glass, Tiffany glass, natural glass, Pyrex, soluble glass, sodium silicate, vitreous silica, tektite, glass, safety glass, crown glass, laminated glass, optical crown, yttrium, beta iron, cerium, cinder pig, rubidium, titanium, rhenium, alloy cast iron, Ni-resist iron, osmium, palladium, barium, bismuth, berkelium, lead, delta iron, radium, thorium, gold, wrought iron, spiegel iron, lutetium, Ni-hard, samarium, spelter, barium hydroxide, tantalum, antimony, thorium-228, glucinium, cast iron, potash alum, promethium, radiothorium, copper, barium oxide, antimonial lead, scrap iron, nickel, tungsten, californium, alpha iron, alum, niobium, molybdenum, europium, coin silver, 24-karat gold,

uranium 235, gadolinium, gold dust, barium protoxide, basic iron, zirconium, hafnium, einsteinium, pig lead, silver, americium, tin, Ni-resist, columbium, protactinium, sodium, zinc, technetium, lithium, lanthanum, spiegeleisen, wolfram, Ni-hard iron, ruthenium, alkaline-earth metal, calcium, beryllium, ingot iron, thallium, pure gold, holmium, cesium, chrome, neodymium, francium, aluminium, rhodium, pig iron, platinum, ammonia alum, neptunium, curium, fermium, praseodymium, barium peroxide, alkali metal, magnesium, potassium alum, barium dioxide, iron, mine pig, scandium, heavy metal, dysprosium, gamma iron, thulium, uranium 238, mercury, spiegel, alkaline earth, vanadium, alloy iron, hydrargyrum, hard lead, potassium, guinea gold, baryta, strontium 90, structural iron, Swedish iron, polonium, aluminum, cadmium, galvanized iron, erbium, noble metal, grid metal, alkaline metal, factor IV, pot metal, lutecium, caesium, calcium ion, cobalt, terbium, protoactinium, green gold, ytterbium, metal, barium monoxide, blister copper, gallium, indium, metallic element, uranium, quicksilver, strontium, cesium 137, cobalt 60, base metal, chromium, ammonium alum, iridium, manganese, nickel bronze, cheoplastic metal, alpha-beta brass, cartridge brass, chisel steel, aluminium bronze, tungsten steel, Muntz metal, gunmetal, brass, misch metal, dental gold, latten, eutectoid steel, austenitic manganese steel, alloy, bearing brass, bearing metal, wolfram steel, low-carbon steel, Admiralty brass, Wood's metal, Wood's alloy, Admiralty Metal, phosphor bronze, soft solder, 22-karat gold, manganese bronze, hot-work steel, austenitic steel, hyper-eutectoid steel, cupronickel, structural steel, solder, electrum, ormolu, steel, 18-karat gold, type metal, Duralumin, vanadium steel, Permalloy, Stellite, hard solder, high-strength brass, Carboloy, type slug, German silver, naval brass, Tobin bronze, nickel-base alloy, chromium steel, mild steel, low brass, Eureka, beryllium bronze, slug, alpha bronze, molybdenum steel, Damask steel, nickel alloy, Inconel, drill steel, soft-cast steel, case-hardened steel, high-speed steel, leaded bronze, tombac, white gold, pewter, silicon bronze, pinchbeck, bell metal, oil-hardened steel, Elinvar, white metal, tambac, alloy steel, copper-base alloy, quenched steel, alpha brass, bronze, heavy metal, aluminum bronze, tombak, drill rod, hypo-eutectoid steel, oroide, dental amalgam, silver solder, Britannia metal, hard steel, medium steel, pyrophoric alloy, guinea gold, stainless, red brass, oreide, tool steel, Monel metal, chrome-nickel steel, pot metal, babbitt, Alnico, Nichrome, manganese steel, ferrocerium, gilding metal, stainless steel, Invar, high brass, carbon steel, metal, chrome-tungsten steel, sterling silver, Monell metal, Babbitt metal, constantan, yellow metal, crucible steel, nickel steel, amalgam, fusible metal, shot metal, Damascus steel, nickel silver, clinker brick, adobe brick, coping, brick, adobe, clinker, ceramic, header, mud brick, firebrick, cope, Vinylite, thermoplastic, coumarone resin, Teflon, thermoplastic resin, aminoplast, thermosetting compositions, amino plastic, coumarone-indene resin, polypropene, polyester, resinoid, thermosetting resin, phenolic plastic, Bakelite, plastic, vinyl, acrylonitrile-butadiene-styrene, silicone resin, Mylar, cellulosic, amino resin, saran, polypropylene, fluorocarbon plastic, celluloid, ABS, phenolic urea, polytetrafluoroethylene, polyvinyl-formaldehyde,

Appendix E

Dependency patterns

The full list of dependency patterns used for extracting attributes and spatial relations is presented here. As in Table 5.1, each rule is show with an example of parsed text and semantic form. Rules are specified using Semgrex patterns over Stanford dependencies.

In addition to the dependency patterns presented in Table 5.1, the patterns in Table E.1 are used to extract spatial relationships from noun phrases. These patterns are combined with the rules in Table E.2, the attribute word lists (see Appendix D) and a dictionary of spatial relation terms to spatial relations (see Appendix F) to determine the best attachment and the most appropriate semantic form.

rule	`{}=nsubj >prep ({}=prep >pobj =pobj)`
text	left`[nsubj]` of`[prep]` the chair`[pobj]`.
	attribute`(verb,pobj)(nsubj,pobj)`
	`material(chair,wood)`
rule	`{}=nn >amod {}=amod`
text	the left`[amod]` chair`[nn]`
	attribute`(dobj)(nsubj,dobj)`
	`color(chair,red)`

Table E.1: Dependency patterns for extracting attributes and spatial relations from sentence fragments.

Table E.2 contains all the patterns over sentences for scenes from text.

rule	`{tag:VBN}=verb >nsubjpass {}=nsubj >prep ({}=prep >pobj {}=pobj)`
text	The chair`[nsubj]` is made`[verb]` of`[prep]` wood`[pobj]`.
	attribute`(verb,pobj)(nsubj,pobj)`
	`material(chair,wood)`
rule	`{}=dobj >cop {} >nsubj {}=nsubj`
text	The chair`[nsubj]` is red`[dobj]`.
	attribute`(dobj)(nsubj,dobj)`
	`color(chair,red)`
rule	`{}=dobj >cop {} >nsubj {}=nsubj >prep ({}=prep >pobj {}=pobj)`
text	The table`[nsubj]` is next`[dobj]` to`[prep]` the chair`[pobj]`.
	spatial`(dobj)(nsubj, pobj)`
	`next_to(table,chair)`
rule	`{}=nsubj >advmod ({}=advmod >prep ({}=prep >pobj {}=pobj))`
text	There is a table`[nsubj]` next`[advmod]` to`[prep]` a chair`[pobj]`.
	spatial`(advmod)(nsubj, pobj)`
	`next_to(table,chair)`
rule	`{word:is} >nsubj {}=nsubj >prep ({}=prep >pobj {}=pobj)`
text	The lamp`[nsubj]` is on`[prep]` the table`[pobj]`.
	spatial`(prep)(nsubj, pobj)`
	`supported_by(lamp,table)`
rule	`{word:is} >nsubj {}=nsubj >prep ({}=prep >pobj {}=pobj)`
text	The table`[nsubj]` is to`[prep]` the left`[pobj]` of the chair.
	spatial`(pobj)(nsubj, pobj2)`
	`next_to(table,chair)`

Table E.2: The full list of dependency patterns used for extracting attributes and spatial relations. These patterns are combined with those in Table E.1.

Table E.3 contains all the patterns for interpreting the manipulation commands described in Chapter 7. See Table 7.1 for a mapping of verbs to scene operations.

rule	{tag:VB}=verb >dobj {}=dobj
text	Add[verb] a chair[nsubj].
	op(verb)(dobj)
	Insert(chair)

Table E.3: Dependency patterns used for interpreting commands. These patterns are combined with those in Table E.1 to given additional constraints.

Appendix F

Spatial relations

In this section, I describe the full list of predefined spatial relations considered in this dissertation. I focus on binary relations involving two objects that take up volume (i.e., objects are not simplified to points). For simplicity, spatial relations are computed using the axis-aligned bounding box of the two objects with respect to the viewer (i.e., the camera position). In the tables below, the following notation is used: $Vol(A)$ is the volume A approximated using the volume of the axis-aligned bounding box of A, $front(A)$ is the direction of the front vector of A, and $c(A)$ is the centroid of A.

Directional relations For all relative directional relations, the system uses a right handed coordinate frame, with Y up and $-Z$ being the front direction (thus left is mapped to $-X$ and right to $+X$).

The system differentiates between the concept of *left_of* (as in "the chair is to the left of the table") and *left_side* (as in "the chair is on the left side of the room"). The notation left_of (B) (used for *left_of*) indicates the region of space that is completely to the left of object B. In contrast, left_of $(c(B))$ (used for *left_side*) is considered from the centroid of the bounding box of B. The two sets of directional relations are listed below:

Table F.1: **Directional**

Relation	$P(relation)$
left_of(A, B)	$\frac{Vol(A \cap \text{left\_of } (B))}{Vol(A)}$
right_of(A, B)	$\frac{Vol(A \cap \text{right\_of } (B))}{Vol(A)}$
above(A, B)	$\frac{Vol(A \cap \text{above } (B))}{Vol(A)}$
below(A, B)	$\frac{Vol(A \cap \text{below } (B))}{Vol(A)}$
front_of(A, B)	$\frac{Vol(A \cap \text{front\_of } (B))}{Vol(A)}$
back_of(A, B)	$\frac{Vol(A \cap \text{back\_of } (B))}{Vol(A)}$
left_side(A, B)	$\frac{Vol(A \cap \text{left\_of } (c(B)))}{Vol(A)}$
right_side(A, B)	$\frac{Vol(A \cap \text{right\_of } (c(B)))}{Vol(A)}$
upper_side(A, B)	$\frac{Vol(A \cap \text{above } (c(B)))}{Vol(A)}$
lower_side(A, B)	$\frac{Vol(A \cap \text{below } (c(B)))}{Vol(A)}$
front_side(A, B)	$\frac{Vol(A \cap \text{front\_of } (c(B)))}{Vol(A)}$
back_side(A, B)	$\frac{Vol(A \cap \text{back\_of } (c(B)))}{Vol(A)}$

Table F.2: **Support**

Relation	$P(relation)$
supports(A, B)	1 if A is support parent of B, 0 otherwise
supported_by(A, B)	1 if B is support parent of A, 0 otherwise
on_top_of(A, B)	$P(above(A, B))$ if B is support parent of A, 0 otherwise

Table F.3: **Containment**

Relation	$P(relation)$
inside(A, B)	$\frac{Vol(A \cap B)}{Vol(A)}$
contains(A, B)	$\frac{Vol(A \cap B)}{Vol(B)}$
outside(A, B)	1 - $\frac{Vol(A \cap B)}{Vol(A)}$

Table F.4: **Orientation**

Relation	$P(relation)$
faces(A, B)	$\cos(front(A), c(B) - c(A))$

Table F.5: **Proximity**

Proximity relations are defined using the distance between objects A and B. The distance $dist(A, B)$ is normalized against the maximum extent of the bounding box of B. In this thesis, we used f to be the symmetric triangular distribution with the given endpoints.

Relation	$P(relation)$
next_to(A, B)	$1.0 - f(dist(A, B), 0.25, 1.0))$
near(A, B)	$1.0 - f(dist(A, B), 1.0, 3.0))$
across_from(A, B)	$f(dist(A, B), 3.0, 5.0))$

Bibliography

Aditya, Somak, Yezhou Yang, Chitta Baral, Cornelia Fermuller, and Yiannis Aloimonos (2015). "From image to sentences through scene description graphs using commonsense reasoning and knowledge". In: *arXiv preprint arXiv:1511.03292*.

Adorni, Giovanni, Mauro Di Manzo, and Fausto Giunchiglia (1984). "Natural language driven image generation". In: *Proceedings of the 10th International Conference on Computational Linguistics and 22nd annual meeting of the Association for Computational Linguistics*.

Åkerberg, Ola, Hans Svensson, Bastian Schulz, and Pierre Nugues (2003). "CarSim: an automatic 3D text-to-scene conversion system applied to road accident reports". In: *European Association for Computational Linguistics (EACL)*.

Alcantara, Kim D, Jomar P Calandria, Junika S Calupas, Jean Paula R Echas, and Ria A Sagum (2014). "StorVi (Story Visualization): A Text-to-Image Conversion". In: *International Journal of Future Computer and Communication* 3.5, p. 363.

Antol, Stanislaw, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh (2015). "VQA: Visual Question Answering". In: *arXiv preprint arXiv:1505.00468*.

Artzi, Yoav and Luke Zettlemoyer (2013). "Weakly supervised learning of semantic parsers for mapping instructions to actions". In: *Transactions of the Association for Computational Linguistics (TACL)*.

Baayen, R.H., D.J. Davidson, and D.M. Bates (2008). "Mixed-effects modeling with crossed random effects for subjects and items". In: *Journal of Memory and Language* 59.4, pp. 390–412.

Badler, Norman I., Rama Bindiganavale, Jan Allbeck, William Schuler, Liwei Zhao, and Martha Palmer (2000). "Parameterized action representation for virtual human agents". In: *Embodied Conversational Agents*. MIT Press, pp. 256–284.

Baker, Collin F, Charles J Fillmore, and John B Lowe (1998). "The Berkeley Framenet project". In: *Proceedings of the 17th international conference on Computational linguistics*.

Barbu, Andrei, Alexander Bridge, Zachary Burchill, Dan Coroian, Sven Dickinson, Sanja Fidler, Aaron Michaux, Sam Mussman, Siddharth Narayanaswamy, Dhaval Salvi, et al. (2012). "Video in sentences out". In: *arXiv preprint arXiv:1204.2742*.

Berger, John (1972). *Ways of seeing*. Vol. 1. Penguin UK.

Besl, Paul J and Ramesh C Jain (1985). "Three-dimensional object recognition". In: *ACM Computing Surveys (CSUR)* 17.1, pp. 75–145.

Boberg, Richard Wayne (1973). "Generating line drawings from abstract scene descriptions". MA thesis. Massachusetts Institute of Technology.

Boroditsky, Lera (2011). "How languages construct time". In: *Space, time and number in the brain: Searching for the foundations of mathematical thought*. Ed. by Stanislas Dehaene and Elizabeth Brannon. Oxford University Press, pp. 333–341.

Bowerman, Melissa and Erik Pederson (1992). "Topological relations picture series". In: *Space stimuli kit 1.2*. Ed. by Stephen C. Levinson. Max Planck Institute for Psycholinguistics, p. 51.

Bukowski, Richard W and Carlo H Séquin (1995). "Object associations: a simple and practical approach to virtual 3D manipulation". In: *Proceedings of the Symposium on Interactive 3D Graphics*.

Carlson, Laura A and Ryan Kenny (2005). "Constraints on Spatial Language Comprehension: Function and geometry". In: *Grounding Cognition: The Role of Perception and Action in Memory, Language, and Thinking*. Ed. by Diane Pecher and Rolf A Zwaan, pp. 35–64.

Cassell, Justine, Hannes Högni Vilhjálmsson, and Timothy Bickmore (2001). "BEAT: the behavior expression animation toolkit". In: *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*.

Chambers, Nathanael, Daniel Cer, Trond Grenager, David Hall, Chloe Kiddon, Bill MacCartney, Marie-Catherine de Marneffe, Daniel Ramage, Eric Yeh, and Christopher D. Manning (2007). "Learning alignments and leveraging natural logic". In: *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*.

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

Chang, Angel X., Manolis Savva, and Christopher D. Manning (2014a). "Interactive Learning of Spatial Knowledge for Text to 3D Scene Generation". In: *Proceedings of the ACL 2014 Workshop on Interactive Language Learning, Visualization, and Interfaces (ACL-ILLVI)*.

— (2014b). "Learning Spatial Knowledge for Text to 3D Scene Generation". In: *Empirical Methods in Natural Language Processing (EMNLP)*.

Chen, Xinlei and C Lawrence Zitnick (2015). "Minds eye: A recurrent visual representation for image caption generation". In: *Computer Vision and Pattern Recognition (CVPR)*.

Clark, Herbert H (1996). *Using language*. Cambridge university press.

Clay, Sharon Rose and Jane Wilhelms (1996). "Put: Language-based interactive manipulation of objects". In: *Computer Graphics and Applications, IEEE*.

Colin, Christian, Emmanuel Desmontils, Jean-Yves Martin, and Jean-Philippe Mounier (1998). "Working modes with a declarative modeler". In: *Computer Networks and ISDN Systems* 30.20, pp. 1875–1886.

Coyne, Bob and Richard Sproat (2001). "WordsEye: an automatic text-to-scene conversion system". In: *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*.

Coyne, Bob, Richard Sproat, and Julia Hirschberg (2010). "Spatial relations in text-to-scene conversion". In: *Computational Models of Spatial Language Interpretation, Workshop at Spatial Cognition*.

Coyne, Bob, Daniel Bauer, and Owen Rambow (2011). "Vignet: Grounding language in graphics using frame semantics". In: *Proceedings of the ACL 2011 Workshop on Relational Models of Semantics*.

Coyne, Bob, Alexander Klapheke, Masoud Rouhizadeh, Richard Sproat, and Daniel Bauer (2012). "Annotation Tools and Knowledge Representation for a Text-To-Scene System". In: *International Conference on Computational Linguistics (COLING)*.

Davidov, Dmitry and Aro Rappoport (2010). "Extraction and approximation of numerical attributes from the Web". In: *Association of Computational Linguistics (ACL)*.

Dawson, Colin Reimer, John Wright, Antons Rebguns, Marco Valenzuela Escárcega, Daniel Fried, and Paul R Cohen (2013). "A generative probabilistic framework for learning spatial language". In: *Development and Learning and Epigenetic Robotics (ICDL), 2013 IEEE Third Joint International Conference on*.

Dema, Mesfin and Hamed Sari-Sarraf (2014). "A Relevancy, Hierarchical and Contextual Maximum Entropy Framework for a Data-Driven 3D Scene Generation". In: *Entropy* 16.5, pp. 2568–2591.

Desmontils, Emmanuel (2000). "Expressing constraint satisfaction problems in declarative modeling using natural language and fuzzy sets". In: *Computers & Graphics* 24.4, pp. 555–568.

Devlin, Jacob, Saurabh Gupta, Ross Girshick, Margaret Mitchell, and C Lawrence Zitnick (2015). "Exploring Nearest Neighbor Approaches for Image Captioning". In: *arXiv preprint arXiv:1505.04467*.

Donahue, Jeff, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell (2015). "Long-term recurrent convolutional networks for visual recognition and description". In: *Computer Vision and Pattern Recognition (CVPR)*.

Elliott, Desmond and Frank Keller (2013). "Image Description using Visual Dependency Representations". In: *Empirical Methods in Natural Language Processing (EMNLP)*.

Elliott, Desmond and Arjen P de Vries (2015). "Describing Images using Inferred Visual Dependency Representations". In: *Association for Computational Linguistics and International Joint Conference on Natural Language Processing (ACL-IJCNLP)*.

Elliott, Desmond, Victor Lavrenko, and Frank Keller (2014). "Query-by-Example Image Retrieval using Visual Dependency Representations". In: *International Conference on Computational Linguistics (COLING)*.

Fang, Hao, Saurabh Gupta, Forrest Iandola, Rupesh Srivastava, Li Deng, Piotr Dollar, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John Platt, Lawrence Zitnick, and Geoffrey Zweig (2015). "From Captions to Visual Concepts and Back". In: *Computer Vision and Pattern Recognition (CVPR)*.

Farhadi, Ali, Mohsen Hejrati, Mohammad Amin Sadeghi, Peter Young, Cyrus Rashtchian, Julia Hockenmaier, and David Forsyth (2010). "Every picture tells a story: Generating sentences from images". In: *European Conference on Computer Vision (ECCV)*.

Feixas, Miquel, Mateu Sbert, and Francisco González (2009). "A unified information-theoretic framework for viewpoint selection and mesh saliency". In: *ACM Transactions on Applied Perception (TAP)* 6.1, p. 1.

Fisher, Matthew, Daniel Ritchie, Manolis Savva, Thomas Funkhouser, and Pat Hanrahan (2012). "Example-based synthesis of 3D object arrangements". In: *ACM Transactions on Graphics (TOG)*.

FitzGerald, Nicholas, Yoav Artzi, and Luke Zettlemoyer (2013). "Learning Distributions over Logical Forms for Referring Expression Generation". In: *Empirical Methods in Natural Language Processing (EMNLP)*.

Freeman, John (1975). "The modelling of spatial relations". In: *Computer graphics and image processing*.

Fu, Hongbo, Daniel Cohen-Or, Gideon Dror, and Alla Sheffer (2008). "Upright orientation of man-made objects". In: *ACM Transactions on Graphics (TOG)*.

Gaildrat, Véronique (2007). "Declarative modelling of virtual environments, overview of issues and applications". In: *International Conference on Computer Graphics and Artificial Intelligence (3IA)*.

Gapp, Klaus-Peter (1994). "Basic Meanings of Spatial Computation and Evaluation in 3D Space". In: *Association for the Advancement of Artificial Intelligence (AAAI)*.

Gärdenfors, Peter (2014). *The geometry of meaning: Semantics based on conceptual spaces*. MIT Press.

Gildea, Daniel and Daniel Jurafsky (2002). "Automatic labeling of semantic roles". In: *Computational linguistics* 28.3, pp. 245–288.

Glass, Kevin (2008). "Automating the conversion of natural language fiction to multi-modal 3D animated virtual environments". PhD thesis. Rhodes University.

Golland, Dave, Percy Liang, and Dan Klein (2010). "A game-theoretic approach to generating spatial descriptions". In: *Empirical Methods in Natural Language Processing (EMNLP)*.

Goodman, Noah D and Andreas Stuhlmüller (2013). "Knowledge and implicature: Modeling language understanding as social cognition". In: *Topics in cognitive science* 5.1, pp. 173–184.

Gorniak, Peter and Deb Roy (2004). "Grounded semantic composition for visual scenes". In: *Journal of Artificial Intelligence Research (JAIR)* 21.1, pp. 429–470.

— (2005). "Probabilistic grounding of situated speech using plan recognition and reference resolution". In: *Proceedings of the 7th International Conference on Multimodal Interfaces*.

Gregor, Karol, Ivo Danihelka, Alex Graves, and Daan Wierstra (2015). "DRAW: A recurrent neural network for image generation". In: *International Conference on Machine Learning (ICML)*.

Guadarrama, Sergio, Lorenzo Riano, Dave Golland, Daniel Gohring, Yangqing Jia, Dan Klein, Pieter Abbeel, and Trevor Darrell (2013). "Grounding Spatial Relations for Human-Robot Interaction". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Gupta, Rakesh and Mykel J Kochenderfer (2004). "Common sense data acquisition for indoor mobile robots". In: *Association for the Advancement of Artificial Intelligence (AAAI)*.

Hanser, Eva, Paul Mc Kevitt, Tom Lunney, Joan Condell, and Minhua Ma (2010). "SceneMaker: multimodal visualisation of natural language film scripts". In: *Knowledge-Based and Intelligent Information and Engineering Systems*. Springer, pp. 430–439.

Herzog, Gerd and Peter Wazinski (1994). "Visual translator: Linking perceptions and natural language descriptions". In: *Artificial Intelligence Review* 8.2-3, pp. 175–187.

Hodhod, Rania, Marc Huet, and Mark Riedl (2014). "Toward Generating 3D Games with the Help of Commonsense Knowledge and the Crowd". In: *Tenth Artificial Intelligence and Interactive Digital Entertainment Conference*.

Jackendoff, Ray (1983). *Semantics and cognition*. MIT press.

— (1992). *Semantic structures*. MIT press.

Jiang, Yun and Ashutosh Saxena (2013). "Infinite latent conditional random fields for modeling environments through humans". In: *Robotics: Science and Systems*.

Jiang, Yun, Marcus Lim, and Ashutosh Saxena (2012). "Learning Object Arrangements in 3D Scenes using Human Context". In: *International Conference on Machine Learning (ICML)*.

Johnson, Justin, Ranjay Krishna, Michael Stark, Li-Jia Li, David Shamma, Michael Bernstein, and Li Fei-Fei (2015). "Image Retrieval using Scene Graphs". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Joshi, Dhiraj, James Z Wang, and Jia Li (2006). "The Story Picturing Engine—a system for automatic text illustration". In: *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 2.1, pp. 68–89.

Kahn, Kenneth Michael (1979). "Creation of computer animation from story descriptions". PhD thesis. Massachusetts Institute of Technology.

Kallmann, Marcelo and Daniel Thalmann (1999). "Direct 3D Interaction with Smart Objects". In: *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*.

Kalvin, Alan D. and Russell H. Taylor (1996). "Superfaces: Polygonal mesh simplification with bounded error". In: *Computer Graphics and Applications, IEEE*.

Karpathy, Andrej and Li Fei-Fei (2015). "Deep visual-semantic alignments for generating image descriptions". In: *Computer Vision and Pattern Recognition (CVPR)*.

Karpathy, Andrej, Armand Joulin, and Li Fei-Fei (2014). "Deep fragment embeddings for bidirectional image sentence mapping". In: *Advances in Neural Information Processing Systems*.

Kazemzadeh, Sahar, Vicente Ordonez, Mark Matten, and Tamara L. Berg (2014). "ReferItGame: Referring to Objects in Photographs of Natural Scenes". In: *Empirical Methods in Natural Language Processing (EMNLP)*.

Kelleher, John and Josef van Genabith (2004). "Visual salience and reference resolution in simulated 3-D environments". In: *Artificial Intelligence Review* 21.3-4, pp. 253–267.

Kelleher, John D and Fintan J Costello (2009). "Applying computational models of spatial prepositions to visually situated dialog". In: *Computational Linguistics* 35.2, pp. 271–306.

Kiros, Ryan, Ruslan Salakhutdinov, and Richard S Zemel (2014). "Unifying visual-semantic embeddings with multimodal neural language models". In: *arXiv preprint arXiv:1411.2539*.

Kong, Chen, Dahua Lin, Mohit Bansal, Raquel Urtasun, and Sanja Fidler (2014). "What are you talking about? Text-to-image coreference". In: *Computer Vision and Pattern Recognition (CVPR)*.

Krahmer, Emiel and Kees Van Deemter (2012). "Computational generation of referring expressions: A survey". In: *Computational Linguistics* 38.1, pp. 173–218.

Krishnamoorthy, Niveda, Girish Malkarnenkar, Raymond Mooney, Kate Saenko, and Sergio Guadarrama (2013). "Generating natural-language video descriptions using text-mined knowledge". In: *North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL-HLT)*.

Krishnamurthy, Jayant and Thomas Kollar (2013). "Jointly learning to parse and perceive: Connecting natural language to the physical world". In: *Transactions of the Association for Computational Linguistics (TACL)*.

Kulkarni, Girish, Visruth Premraj, Sagnik Dhar, Siming Li, Yejin Choi, Alexander C. Berg, and Tamara L. Berg (2011). "Baby talk: Understanding and generating simple image descriptions". In: *Computer Vision and Pattern Recognition (CVPR)*.

Lakoff, George and Mark Johnson (2008). *Metaphors we live by*. University of Chicago press.

Lazaridou, Angeliki, Dat Tien Nguyen, and Marco Baroni (2015). "Do Distributed Semantic Models Dream of Electric Sheep? Visualizing Word Representations through Image Synthesis". In: *Proceedings of the Fourth Workshop on Vision and Language (VL15)*.

Le Roux, O, V Gaildrat, and R Caube (2004). "Constraint satisfaction techniques for the generation phase in declarative modeling". In: *Geometric modeling: techniques, applications, systems and tools*. Springer, pp. 193–215.

LeCun, Yann and Yoshua Bengio (1995). "Convolutional networks for images, speech, and time series". In: *The handbook of brain theory and neural networks*. Ed. by Michael A. Arbib. MIT Press, pp. 255–257.

Lee, Heeyoung, Yves Peirsman, Angel Chang, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky (2011). "Stanford's Multi-Pass Sieve Coreference Resolution System at the CoNLL-2011 Shared Task". In: *Proceedings of the CoNLL-2011 Shared Task*.

Lenat, Douglas B. (1995). “CYC: A large-scale investment in knowledge infrastructure”. In: *Communications of the ACM* 38.11.

Levinson, Stephen C. (2003). *Space in language and cognition: Explorations in cognitive diversity*. Cambridge University Press.

Li, Bo, Yijuan Lu, Chunyuan Li, Afzal Godil, Tobias Schreck, Masaki Aono, Qiang Chen, Nihad Karim Chowdhury, Bin Fang, Takahiko Furuya, et al. (2014). “Shrec14 track: Large scale comprehensive 3D shape retrieval”. In: *Eurographics Workshop on 3D Object Retrieval*.

Lin, Dahua, Sanja Fidler, Chen Kong, and Raquel Urtasun (2014). “Visual semantic search: Retrieving videos via complex textual queries”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Liu, Hugo and Push Singh (2004). “ConceptNet - a practical commonsense reasoning toolkit”. In: *BT technology journal* 22.4.

Liu, Tianqiang, Siddhartha Chaudhuri, Vladimir G Kim, Qixing Huang, Niloy J Mitra, and Thomas Funkhouser (2014). “Creating consistent scene graphs using a probabilistic grammar”. In: *ACM Transactions on Graphics (TOG)* 33.6, p. 211.

Liu, Ying, Dengsheng Zhang, Guojun Lu, and Wei-Ying Ma (2007). “A survey of content-based image retrieval with high-level semantics”. In: *Pattern Recognition* 40.1, pp. 262–282.

Liu, Zhi-Qiang and Ka-Ming Leung (2006). “Script visualization (ScriptViz): a smart system that makes writing fun”. In: *Soft Computing* 10.1, pp. 34–40.

Lu, Jiajie, Canlin Li, Chao Yin, and Lizhuang Ma (2010). “A new framework for automatic 3D scene construction from text description”. In: *Progress in Informatics and Computing (PIC), 2010 IEEE International Conference on*.

Lu, Ruqian and Songmao Zhang (2002). *Automatic generation of computer animation: using AI for movie animation*. Springer-Verlag.

Ma, Minhua (2006). “Automatic conversion of natural language to 3D animation”. PhD thesis. University of Ulster.

Manning, Christopher D., Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky (2014). “The Stanford CoreNLP Natural Language Processing Toolkit”. In: *Association for Computational Linguistics (ACL): System Demonstrations*.

Mansimov, Elman, Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov (2015). "Generating Images from Captions with Attention". In: *arXiv preprint arXiv:1511.02793*.

Mao, Junhua, Wei Xu, Yi Yang, Jiang Wang, and Alan Yuille (2014). "Deep captioning with multimodal recurrent neural networks (m-RNN)". In: *arXiv preprint arXiv:1412.6632*.

Matuszek, Cynthia, Nicholas Fitzgerald, Luke Zettlemoyer, Liefeng Bo, and Dieter Fox (2012). "A Joint Model of Language and Perception for Grounded Attribute Learning". In: *International Conference on Machine Learning (ICML)*.

Matuszek, Cynthia, Evan Herbst, Luke Zettlemoyer, and Dieter Fox (2013). "Learning to parse natural language commands to a robot control system". In: *Experimental Robotics*.

McMahan, Brian and Matthew Stone (2015). "A Bayesian Model of Grounded Color Semantics". In: *Transactions of the Association for Computational Linguistics (TACL)* 3, pp. 103–115.

Merrell, Paul, Eric Schkufza, Zeyang Li, Maneesh Agrawala, and Vladlen Koltun (2011). "Interactive furniture layout using interior design guidelines". In: *ACM Transactions on Graphics (TOG)*.

Miller, George A. (1995). "WordNet: a lexical database for English". In: *Communications of the ACM*.

Misra, Dipendra K., Jaeyong Sung, Kevin Lee, and Ashutosh Saxena (2014). "Tell me Dave: Context-sensitive grounding of natural language to mobile manipulation instructions". In: *Robotics: Science and Systems (RSS)*.

Mitchell, Margaret, Xufeng Han, Jesse Dodge, Alyssa Mensch, Amit Goyal, Alex Berg, Kota Yamaguchi, Tamara Berg, Karl Stratos, and Hal Daumé III (2012). "Midge: Generating image descriptions from computer vision detections". In: *European Association for Computational Linguistics (EACL)*.

Olivier, Patrick, Toshiyuki Maeda, and J Ichi Tsujii (1994). "Automatic depiction of spatial descriptions". In: *Association for the Advancement of Artificial Intelligence (AAAI)*.

Oshita, Masaki (2010). "Generating animation from natural language texts and semantic analysis for motion search and scheduling". In: *The Visual Computer* 26.5, pp. 339–352.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu (2002). "BLEU: A method for automatic evaluation of machine translation". In: *Association for Computational Linguistics (ACL)*.

Pick, Herbert L. and Linda P. Acredolo (1983). *Spatial orientation: Theory, research, and application*. Plenum Press.

Plemenos, Dimitri and Georgios Miaoulis (2009). "Intelligent scene modeling". In: *Visual Complexity and Intelligent Computer Graphics Techniques Enhancements*. Springer, pp. 27–64.

Roberts, Lawrence Gilman (1965). "Machine perception of three-dimensional solids". PhD thesis. Massachusetts Institute of Technology.

Rohrbach, Marcus, Wei Qiu, Igor Titov, Stefan Thater, Manfred Pinkal, and Bernt Schiele (2013). "Translating video content to natural language descriptions". In: *International Conference on Computer Vision (ICCV)*.

Rosch, Eleanor and Barbara B Lloyd (1978). *Cognition and categorization*. Hillsdale, New Jersey: Lawrence Erbaum Associates.

Rosch, Eleanor and Carolyn B Mervis (1975). "Family resemblances: Studies in the internal structure of categories". In: *Cognitive psychology* 7.4, pp. 573–605.

Rosman, Benjamin and Subramanian Ramamoorthy (2011). "Learning spatial relationships between objects". In: *The International Journal of Robotics Research*.

Rouhizadeh, Masoud, Margit Bowler, Richard Sproat, and Bob Coyne (2011). "Collecting semantic data by Mechanical Turk for the lexical knowledge resource of a text-to-picture generating system". In: *Proceedings of the Ninth International Conference on Computational Semantics*.

Roy, Deb K (2002). "Learning visually grounded words and syntax for a scene description task". In: *Computer Speech & Language* 16.3, pp. 353–385.

Ruchaud, William and Dimitri Plemenos (2002). "Multiformes: a declarative modeller as a 3D scene sketching tool". In: *International Conference on Computer Vision and Graphics (ICCVG)*.

Savva, Manolis, Angel X. Chang, Gilbert Bernstein, Christopher D. Manning, and Pat Hanrahan (2014a). "On Being the Right Scale: Sizing Large Collections of 3D Models".

In: *SIGGRAPH Asia 2014 Workshop on Indoor Scene Understanding: Where Graphics meets Vision*.

Savva, Manolis, Angel X Chang, Pat Hanrahan, Matthew Fisher, and Matthias Nießner (2014b). "SceneGrok: Inferring Action Maps in 3D Environments". In: *ACM Transactions on Graphics (TOG)* 33.6, p. 212.

Savva, Manolis, Angel X. Chang, and Pat Hanrahan (2015). "Semantically-Enriched 3D Models for Common-sense Knowledge". In: *Computer Vision and Pattern Recognition (CVPR) Workshop on Vision meets Cognition: Functionality, Physics, Intentionality and Causality*.

Saxena, Ashutosh, Ashesh Jain, Ozan Sener, Aditya Jami, Dipendra K Misra, and Hema S Koppula (2015). "Robo Brain: Large-Scale Knowledge Engine for Robots". In: *International Symposium on Robotics Research (ISRR)*.

Schuster, Sebastian, Ranjay Krishna, Angel Chang, Li Fei-Fei, and Christopher D. Manning (2015). "Generating Semantically Precise Scene Graphs from Textual Descriptions for Improved Image Retrieval". In: *Proceedings of the Fourth Workshop on Vision and Language (VL15)*.

Schwarz, Katharina, Pavel Rojtberg, Joachim Caspar, Iryna Gurevych, Michael Goesele, and Hendrik PA Lensch (2010). "Text-to-video: story illustration from online photo collections". In: *Knowledge-Based and Intelligent Information and Engineering Systems*. Springer, pp. 402–409.

Seversky, Lee M. and Lijun Yin (2006). "Real-time automatic 3D scene generation from natural language voice and text descriptions". In: *Proceedings of the 14th annual ACM international conference on Multimedia*.

Simmons, Robert F (1975). "The clowns microworld". In: *Proceedings of the 1975 workshop on Theoretical issues in natural language processing*.

Singh, Push, Thomas Lin, Erik T Mueller, Grace Lim, Travell Perkins, and Wan Li Zhu (2002). "Open Mind Common Sense: Knowledge acquisition from the general public". In: *Proceedings of the First International Conference on Ontologies, Databases, and Applications of Semantics for Large Scale Information Systems*.

Snoek, Cees GM and Marcel Worring (2008). "Concept-based video retrieval". In: *Foundations and Trends in Information Retrieval* 2.4, pp. 215–322.

Speer, Robert and Catherine Havasi (2013). "ConceptNet 5: A large semantic network for relational knowledge". In: *The Peoples Web Meets NLP*. Springer, pp. 161–176.

Sproat, Richard (2001). "Inferring the environment in a text-to-scene conversion system". In: *Proceedings of the 1st international conference on Knowledge capture*.

Srihari, Rohini K (1994). "Computational models for integrating linguistic and visual information: A survey". In: *Artificial Intelligence Review* 8.5-6, pp. 349–369.

Tabia, Hedi and Ngoc-Son Vu (2014). "3D Shape classification using information fusion". In: *International Conference on Pattern Recognition (ICPR)*.

Takashima, Yosuke, Hideo Shimazu, and Masahiro Tomono (1987). "Story driven animation". In: *ACM SIGCHI Bulletin*.

Tapaswi, Makarand, Martin Bäuml, and Rainer Stiefelhagen (2015). "Book2movie: Aligning video scenes with book chapters". In: *Computer Vision and Pattern Recognition (CVPR)*.

Tappan, Dan (2008). "Monte Carlo Simulation for Plausible Interpretation of Natural-Language Spatial Descriptions". In: *International Conference on Artificial Intelligence (ICAI)*.

Tappan, Daniel Allen (2004). "Knowledge-based spatial reasoning for automated scene generation from text descriptions". PhD thesis. New Mexico State University.

Tellex, Stefanie, Pratiksha Thaker, Joshua Joseph, and Nicholas Roy (2014). "Learning perceptually grounded word meanings from unaligned parallel data". In: *Machine Learning*.

Tobler, Robert F (2011). "Separating semantics from rendering: a scene graph based architecture for graphics applications". In: *The Visual Computer* 27.6-8, pp. 687–695.

Tutenel, Tim, Rafael Bidarra, Ruben M Smelik, and Klaas Jan De Kraker (2008). "The role of semantics in games and simulations". In: *Computers in Entertainment (CIE)* 6.4, p. 57.

Unal, Emre (2014). "A Language Visualization System". MA thesis. Koç University.

Van Kaick, Oliver, Hao Zhang, Ghassan Hamarneh, and Daniel Cohen-Or (2011). "A survey on shape correspondence". In: *Computer Graphics Forum*. Vol. 30. 6. Wiley Online Library, pp. 1681–1707.

Vedantam, Ramakrishna, Xiao Lin, Tanmay Batra, C Lawrence Zitnick, and Devi Parikh (2015). "Learning Common Sense Through Visual Abstraction". In: *International Conference on Computer Vision (ICCV)*.

Venugopalan, Subhashini, Marcus Rohrbach, Jeff Donahue, Raymond Mooney, Trevor Darrell, and Kate Saenko (2015a). "Sequence to Sequence - Video to Text". In: *International Conference on Computer Vision (ICCV)*.

Venugopalan, Subhashini, Huijuan Xu, Jeff Donahue, Marcus Rohrbach, Raymond Mooney, and Kate Saenko (2015b). "Translating videos to natural language using deep recurrent neural networks". In: *North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL-HLT)*.

Vinyals, Oriol, Alexander Toshev, Samy Bengio, and Dumitru Erhan (2014). "Show and Tell: A Neural Image Caption Generator". In: *arXiv preprint arXiv:1411.4555*.

Vogel, Adam and Dan Jurafsky (2010). "Learning to Follow Navigational Directions". In: *Association for Computational Linguistics (ACL)*.

Vogel, Adam, Christopher Potts, and Dan Jurafsky (2013). "Implicatures and Nested Beliefs in Approximate Decentralized-POMDPs". In: *Association for Computational Linguistics (ACL)*.

Waltz, David L (1980). *Generating and Understanding Scene Descriptions*. Tech. rep. DTIC Document.

West, Brady T, Kathleen B Welch, and Andrzej T Galecki (2014). *Linear mixed models: a practical guide using statistical software*. CRC Press.

Winograd, Terry (1972). "Understanding natural language". In: *Cognitive psychology*.

Xu, Kelvin, Jimmy Ba, Ryan Kiros, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio (2015). "Show, attend and tell: Neural image caption generation with visual attention". In: *arXiv preprint arXiv:1502.03044*.

Xu, Ken, James Stewart, and Eugene Fiume (2002). "Constraint-based automatic placement for scene composition". In: *Graphics Interface*. Vol. 2.

Yao, Li, Atousa Torabi, Kyunghyun Cho, Nicolas Ballas, Christopher Pal, Hugo Larochelle, and Aaron Courville (2015). "Video description generation incorporating spatio-temporal features and a soft-attention mechanism". In: *arXiv preprint arXiv:1502.08029*.

Ye, Patrick and Timothy Baldwin (2008). "Towards Automatic Animated Storyboarding." In: *Association for the Advancement of Artificial Intelligence (AAAI)*.

Yu, Lap-Fai, Sai-Kit Yeung, and Demetri Terzopoulos (2015). "The Clutterpalette: An Interactive Tool for Detailing Indoor Scenes". In: *IEEE Transactions on Visualization and Computer Graphics* 21.

Zaragoza Rios, Jaime Alberto (2009). "Declarative modeling based on knowledge". PhD thesis. Université de Toulouse, Université Toulouse III-Paul Sabatier.

Zhang, Jiawan, Yukun Hao, Liang Li, Di Sun, and Li Yuan (2012). "StoryWizard: a framework for fast stylized story illustration". In: *The Visual Computer* 28.6-8, pp. 877–887.

Zhu, Yuke, Alireza Fathi, and Li Fei-Fei (2014). "Reasoning about Object Affordances in a Knowledge Base Representation". In: *European Conference on Computer Vision (ECCV)*.

Zhu, Yuke, Ce Zhang, Christopher Ré, and Li Fei-Fei (2015a). "Building a Large-scale Multimodal Knowledge Base for Visual Question Answering". In: *arXiv:1507.05670*.

Zhu, Yuke, Oliver Groth, Michael Bernstein, and Li Fei-Fei (2015b). "Visual7W: Grounded Question Answering in Images". In: *arXiv preprint arXiv:1511.03416*.

Zhu, Yukun, Ryan Kiros, Richard Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler (2015c). "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books". In: *arXiv preprint arXiv:1506.06724*.

Zitnick, C. Lawrence, Devi Parikh, and Lucy Vanderwende (2013). "Learning the Visual Interpretation of Sentences". In: *International Conference on Computer Vision (ICCV)*.