Language-Driven Synthesis of 3D Scenes from Scene Databases

RUI MA*, Simon Fraser University and AltumView Systems Inc.
AKSHAY GADI PATIL*, Simon Fraser University
MATTHEW FISHER, Adobe Research
MANYI LI, Shandong University and Simon Fraser University
SÖREN PIRK, Stanford University
BINH-SON HUA, University of Tokyo
SAI-KIT YEUNG, Hong Kong University of Science and Technology
XIN TONG, Microsoft Research Asia
LEONIDAS GUIBAS, Stanford University
HAO ZHANG, Simon Fraser University

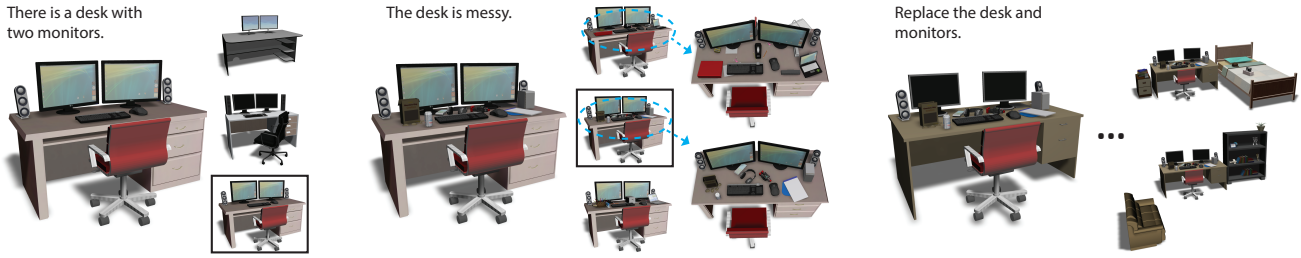


Fig. 1. Our modeling tool uses compact and natural language to generate and edit 3D indoor scenes. Each language command triggers a scene evolution where the user can choose a scene (marked by a box) from a list of suggested outcomes to alleviate ambiguities in natural language. One or groups of objects can be inserted, replaced, or re-arranged based on attributes or relations (e.g., "desk is messy") specified in the command.

We introduce a novel framework for using natural language to generate and edit 3D indoor scenes, harnessing scene semantics and text-scene grounding knowledge learned from large annotated 3D scene databases. The advantage of natural language editing interfaces is strongest when performing semantic operations at the *sub-scene* level, acting on *groups* of objects. We learn how to manipulate these sub-scenes by analyzing existing 3D scenes. We perform edits by first parsing a natural language command from the user and transforming it into a *semantic scene graph* that is used to retrieve corresponding sub-scenes from the databases that match the command. We then augment this retrieved sub-scene by incorporating other objects that may be implied by the scene context. Finally, a new 3D scene is synthesized by aligning the augmented sub-scene with the user's current scene, where new objects are spliced into the environment, possibly triggering appropriate adjustments to the existing scene arrangement. A suggestive modeling interface with multiple interpretations of user commands is used to alleviate ambiguities in natural language. We conduct studies comparing our approach against both prior text-to-scene work and artist-made scenes and find that our method significantly outperforms prior work and is comparable to handmade scenes even when complex and varied natural sentences are used.

CCS Concepts: • **Computing methodologies** → **Computer graphics**; **Shape modeling**; **Shape analysis**;

Additional Key Words and Phrases: Data-driven 3D scene generation and editing, natural language interface, semantic scene graph, relational model

ACM Reference Format:
Rui Ma, Akshay Gadi Patil, Matthew Fisher, Manyi Li, Sören Pirk, Binh-Son Hua, Sai-Kit Yeung, Xin Tong, Leonidas Guibas, and Hao Zhang. 2018. Language-Driven Synthesis of 3D Scenes from Scene Databases. *ACM Trans. Graph.* 37, 6, Article 212 (November 2018), 16 pages. https://doi.org/10.1145/3272127.3275035

1 INTRODUCTION

Recent advances in computational design, VR/AR, and robotics are placing an increasing demand for rich 3D content, especially those in indoor environments [Hua et al. 2016; Silberman et al. 2012; Song et al. 2015, 2017]. While most efforts on 3D indoor scene modeling have been devoted to high-quality and interactive scene *reconstruction* from photographs or depth scans, there has also been a push for more open-ended, and often user-centric, approaches to synthesize and edit 3D scenes [Chang et al. 2014b; Fisher et al.

*Co-First Authors

Authors' addresses: Rui Ma, Simon Fraser University and AltumView Systems Inc. Akshay Gadi Patil, Simon Fraser University; Matthew Fisher, Adobe Research; Manyi Li, Shandong University and Simon Fraser University; Sören Pirk, Stanford University; Binh-Son Hua, University of Tokyo; Sai-Kit Yeung, Hong Kong University of Science and Technology; Xin Tong, Microsoft Research Asia; Leonidas Guibas, Stanford University; Hao Zhang, Simon Fraser University.

0730-0301/2018/11-ART212 $15.00
https://doi.org/10.1145/3272127.3275035

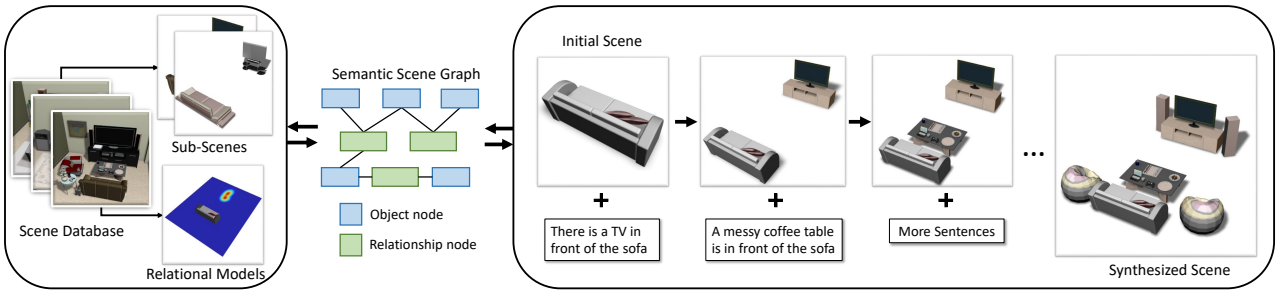


Fig. 2. An overview of our language-driven scene synthesis.

2012; Ma et al. 2016; Savva et al. 2016; Wang et al. 2018], aiming for improved richness and creativity of the generated content.

Natural language is arguably the most accessible input for content creation. With no modeling skills or training required, language- or text-driven modeling is fun, open-ended, and stimulates the imagination. Directly converting languages to 3D scenes has been a long and on-going pursuit since the pioneering work on WordsEye [Coyne and Sproat 2001]. To date, most research has emerged from the natural language processing (NLP) community, e.g., [Chang et al. 2014b; Coyne et al. 2012; Seversky and Yin 2006], where the focus has been on mapping explicit scene arrangement languages, e.g., "a bed with three pillows and a bedside table next to it", to object arrangements. Clearly, such explicit commands can be tedious and unnatural. Moreover, one may argue that precise, object-level controls are more effectively achieved by direct object manipulation using a mouse, e.g., to rotate a cup for its logo to face front. On the other hand, scene modifications at a semi-global or *sub-scene* scale or those involving changes in *group relations* (e.g., "make the tabletop arrangement messy or formal") represent situations where a language-based modeling interface can excel.

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

The second objective is to improve both the complexity and realism of the generated 3D scenes. As a scene increases in complexity, it provides increasingly richer spatial and semantic contexts beyond pairwise object relations [Chang et al. 2014b]. To account for these, we need to encode and learn more complex object relations, particularly those involving *groups* of objects. With this in mind, we enhance existing 3D scene datasets such as SUNCG [Song et al. 2017] and SceneNN [Hua et al. 2016] with both finer-scale objects and annotations of group relations, e.g., "around", "aligned", "messy", "work", etc., to support our modeling task.

To accomplish the goals set above, our key idea is to treat language-driven 3D scene modeling primarily as a combination of two tasks operating at the sub-scene level: language-driven sub-scene *retrieval* from a 3D scene database, and scene *accommodation* which merges an appropriately retrieved sub-scene with the current 3D environment to *synthesize* a new 3D scene. Playing a central role in our modeling framework is a *semantic scene graph* representation, which encodes geometric and semantic scene information and serves as the bridge between user language commands and scene modeling operations that directly modify a 3D indoor scene. Specifically, a semantic scene graph is defined by object instances along with object-level attributes and pairwise, as well as group-wise, object relations. Both annotated 3D scene data and natural language commands are converted into semantic scene graphs.

We develop an interactive scene modeling tool which allows the creation of an initial 3D indoor scene followed by progressive editing to evolve the scene, where all the scene generation and editing commands are given in natural languages; see Figure 1 and Figure 2. Specifically, at each editing iteration, an input language statement is turned into a semantic scene graph to retrieve a suitable sub-scene via graph alignment from the 3D scene database. Once a sub-scene is retrieved, it is augmented with additional objects based on the input text and scene context. Next, this augmented sub-scene is semantically aligned with the current scene. Finally, a new scene is synthesized by splicing the augmented sub-scene into the current scene, possibly triggering appropriate adjustments to the existing scene arrangement. In addition, our tool supports scene edits with *verb* commands, which may trigger, e.g., an object replacement, as shown in Figure 1. Overall, the accommodation of the sub-scene into the current scene is guided by object co-occurrences and relations learned from 3D scene databases. Since natural language is inherently imprecise, we develop a suggestive interface to provide multiple interpretations of user commands for the user to select one or more scene options during modeling.

Compared to the holistic example-based scene synthesis via probabilistic sampling [Fisher et al. 2012], the fine-grained progressive synthesis by sampling human actions [Jiang et al. 2012; Ma et al. 2016], and executing texts on precise object-level controls [Chang et al. 2014b; Coyne and Sproat 2001; Seversky and Yin 2006], our method makes the following contributions:

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

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

Our text-to-scene modeling framework is realized on a dataset consisting of thousands of annotated 3D scenes from SUNCG [Song et al. 2017], among others. We show results of our language-driven scene synthesis and evolution, leading to the generation of plausible 3D indoor scenes with much improved versatility and complexity than previous methods. We conduct studies evaluating our method's ability to generate plausible 3D scenes and robustness against variations in scene description languages. We compare our approach to both prior text-to-scene work and artist-made scenes in terms of scene plausibility. The results show that our method significantly outperforms prior work and is comparable to handmade scenes even when complex and varied natural sentences are used.

2 RELATED WORK

There has been a great deal of work in computer graphics and vision on 3D indoor scene modeling, and so far, most efforts have been invested into *reconstruction*. 3D scene reconstruction takes one or more images or depth scans and aims to reproduce the captured scene geometry and even semantics. There are surveys on the topic, e.g., [Seitz et al. 2006; Slabaugh et al. 2001; Xiao 2012], to name just a few, and most recent works are taking a data-driven approach, e.g., [Kim et al. 2012; Shao et al. 2012; Xu et al. 2016]. In this section, we mainly focus on works most closely related to ours, i.e., those on 3D scene synthesis where textual inputs are used.

3D indoor scene synthesis. One line of work for 3D scene synthesis focuses on furniture layout optimization [Merrell et al. 2011; Yu et al. 2011] for a given room with a given set of furniture. Example-based approaches rely on probabilistic reasoning from 3D exemplars and scene databases to drive the synthesis [Fisher et al. 2015, 2012; Jiang et al. 2012; Sadeghipour et al. 2016; Savva et al. 2016]. A few recent works aim to populate a 3D scene with small objects to increase its realism. Majerowicz et al. [2014] develop a method to fill a shelf-like environment based on styles learned from a photograph or a 3D exemplar. The ClutterPalette by Yu et al. [2016] offers an interactive tool to allow a user to insert and position one object at a time to mess up an otherwise clean scene. As the user clicks on a region of the scene, the tool suggests a set of suitable objects to insert, which are based on support and co-occurrence relations learned from data. The first deep convolutional neural network for scene synthesis has been introduced by Wang et al. [2018], whereby top-down views of scene layouts can be automatically and progressively synthesized from a room architecture based on learned convolutional object placement priors. In contrast to these works, our scene generation framework is language-driven and it follows a pipeline of interpret (from texts), retrieve and synthesize (from scene databases), and disambiguate (via a suggestive interface).

Recently, Ma et al. [2016] introduce a framework for action-driven 3D scene evolution which progressively alters a scene by object placements necessitated by human actions. They learn action models from annotated photographs and probabilistically sample a sequence of actions conditioned on scene contexts to evolve a scene. While the actions applied in their work are labeled textually, e.g., "group dining" or "use laptop", and the texts may not explicitly describe object presence or placement, the action texts themselves are pre-determined (only 8 action types in the paper) with each manually bound to an appropriate class of scene contexts retrieved from the photos. In contrast, our work must learn a mapping between an immensely richer text corpus to object perturbation. Furthermore, none of existing works have considered learning, applying, or adjusting object group relations, e.g., "messy", "aligned", etc.

There is a slight resemblance between the group actions supported by [Ma et al. 2016] and our group relation model. For example, applying the action "group dining on table while sitting" would trigger the insertion of a group of objects into a scene, while in our framework, the object relations and placements would have to be more explicitly specified by a language command. On the other hand, all scene affections in their work must be executed through human actions. For example, "making the table clean" and "moving the chairs apart from table" are well-supported by our synthesis tool through group relations, but would be quite involved to realize by the action-driven approach; it would have to capture and learn actions involving humans tidying up a table or moving chairs.

Text2Scene. One of the earliest systems for text-driven 3D scene synthesis is WordsEye [Coyne and Sproat 2001]. This work, along with other follow-ups [Coyne et al. 2012; Seversky and Yin 2006], is capable of generating 3D scenes directly from natural language, but relies on manual mapping between language and object placements in a scene. The rigidity of the mapping forces the users to issue unnatural commands, e.g., "the chair is three feet to the north of the window". More recently, Zitnick et al. [2013] learn visual interpretation of sentences by focusing on mapping visual features to semantic phrases extracted from the sentences, where the phrases describe binary relations that are either spatial (e.g., "Mike is after Jen") or semantic (e.g., "Alice wants the ball"). However, the generated scenes are 2D, composed of clip art elements. Also relevant is the recent work by Savva et al. [2016], which is able to convert single-sentence descriptions (e.g., "He is lying on the couch" or "He is sitting in bed and using a laptop") into scene prototypes which depict simple human-object interactions.

A series of papers by Chang et al. have provided improvements over the early systems. The key improvement in [Chang et al.

2014a,b] is the utilization of spatial knowledge, learned from 3D scene data, to better constrain scene generations by unstated facts or common sense. The first work [Chang et al. 2014a] also allows interactivity, but with object movements directly controlled by the user. The more recent improvement [Chang et al. 2015b] is on lexical grouding of textual terms to 3D model references, i.e., on selecting more appropriate objects, by combining a rule-based model [Chang et al. 2014b] and lexical grounding learned from user annotation of 3D scenes. In their latest paper [Chang et al. 2017], they extend the pipeline in [Chang et al. 2014b] with interactive text-based scene editing operations and improve the UI. The languages supported by all of these systems are explicit about object presence, but possibly implicit about object spatial relations. In our work, we allow both to be implicit and enable scene annotation, retrieval, and synthesis at the sub-scene level, with the additional consideration of group relations. Moreover, in previous works, the learned spatial knowledge is quite basic, e.g., only binary object relations [Chang et al. 2014a,b; Zitnick et al. 2013]. In our work, we learn and employ richer scene contexts for scene synthesis by utilizing a richer data source.

Robotics. Controlling a robot with textual commands is one of the central problems in robotics, e.g., [Guadarrama et al. 2013; Misra et al. 2014; Tenorth and Beetz 2013]. Rather than learning to map language to object/scene arrangements, as for Text2Scene, the key problem in robotics is to map texts to robot motions, as well as robot-object interactions. However, both problems and their solutions share commonalities, including the need to utilize both rule-based models and data-driven learning of spatial and semantic relations, as well as the exploitation of common sense knowledge [Tenorth and Beetz 2013] to incorporate unstated facts.

Scene graphs. Fisher et al. [2011] represent 3D scenes as object graphs with edges denoting pairwise semantic relationships, then use graph kernels to estimate scene similarity. Xu et al. [2014] represent a 3D indoor scene by a graph of its constituent objects and detect focal points over a heterogeneous collection of 3D indoor scenes as representative sub-scenes. These focal points enable part-in-whole sub-scene retrievals and scene exploration, which could be adapted for our task. However, the object relations encoded in their scene graphs only include binary support, proximity, and symmetry groups. We convert natural language into a semantic scene graph representation that uses both pairwise relationships and multi-object relation annotations.

3 OVERVIEW

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

Semantic scene graph. The semantic scene graph (SSG) encodes the objects, as well as their attributes and relationships in a graph (Section 4). By modeling both 3D scenes and text commands as semantic scene graphs, this uniform representation enables us to apply the edit specified by the command to the input 3D scene and update the scene with the help of the database.

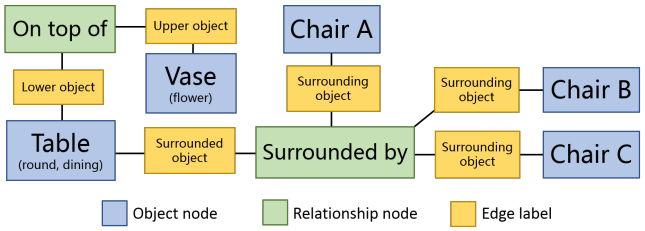


Fig. 3. The semantic scene graph for the sentence "The round dining table is surrounded by three chairs and there is a flower vase on top of the table."

3D scene dataset processing. We collect thousands of annotated 3D scenes as the exemplars for scene editing (Section 5). We construct the semantic scene graph for each 3D scene in the dataset. For later scene editing, we also learn relational models from all 3D scenes in the dataset by modeling the object co-occurrence and relative distribution. This database preprocessing step is executed only once.

Natural language processing. Our system evolves the scene with a sequence of natural language commands. Each command sentence can either specify a scene edit operation (e.g., "put plates on the table.") or passively describe the evolution of the scene (e.g. "there are two plates on the table."). The language processing module transforms each sentence into a canonical semantic scene graph representation (Section 6). For this purpose, we first extract the command and associated scene entities (i.e., objects, their attributes, and their relationships) from the sentence and then convert the command and entities into a semantic scene graph.

Language-driven scene synthesis. We edit the input scene according to the SSG parsed from natural language (Section 7). If the language refers to a scene edit, we directly manipulate the target objects with the specified edit operations. Otherwise, we use the SSG of the input text to search from the scene database, a set of 3D sub-scene exemplars that best match the scene described in the text. We then augment the SSG of the retrieved scene with the SSG of the input text as well as the scene context, and align the result to the SSG of the current scene. We then insert new objects from the augmented SSG into the current scene. The location of the inserted objects is determined by the surrounding objects in the scene and the relational model learned from the scene database.

4 SEMANTIC SCENE REPRESENTATION

To enable matching between a text description and a 3D indoor scene, we employ a *Semantic Scene Graph* (SSG), a semantic scene representation which contains rich information about objects, as well as their attributes and relationships. Both 3D scenes and natural language descriptions of scenes are converted into this uniform scene representation through scene processing (Section 5) and natural language processing (Section 6).

SSG's are udirected graphs with labeled edges and two node types: object nodes and relationship nodes. Object nodes represent objects in the scene and may be annotated with a list of per-object attributes (e.g. "antique", "wooden"). Relationship nodes represent a specific instance of a relationship between two or more objects (e.g., "to the left of", "on each side of", "around"). Each edge connects an

object node to a relationship node and is labeled with the type of connection between the object and relationship. Figure 3 shows an example of a SSG. Each relationship type has a pre-defined set of possible edge labels, e.g., surrounded and surrounding. Each instance of a relationship has its own relationship node. In the rest of the paper, edge labels are not shown for simplicity.

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

5 3D SCENE PROCESSING

Our scene database includes 2984 3D scenes from SUNCG dataset [Song et al. 2017], 133 scenes from SceneSynth [Fisher et al. 2012] and 80 scenes modeled from the real-world SceneNN dataset [Hua et al. 2016]. For each scene, we build a semantic scene graph to encode the object relationships. For each type of pairwise or group relationship, a relational model encoding object co-occurrences and relative distributions is learned from the database.

5.1 Database preparation

To enable semantic matching of objects between natural language and 3D models, consistent and meaningful semantic labels need to be associated with each model from the database. We use annotations from ShapeNetSem [Chang et al. 2015a; Savva et al. 2015] for object labels such as category, attributes, and front orientation. To facilitate learning, each real-world scan from the SceneNN dataset is also converted into a synthetic scene, by retrieving the best match 3D model from SceneSynth using the object tag in SceneNN.

Unlike previous methods which focus on pairwise relationships, we also consider group relationships, which enable more complex text to scene processing. We augment the scenes with high-level group relationships by annotating sub-scenes with semantic labels on corresponding objects, for scenes in all three datasets. In our current work, a total of eleven relationships are handled - nine high-level group relationship types including *messy, clean, organized, disorganized, formal, casual, study, work, dining,* and two spatial group relationships, *around, aligned* are considered (see Figure 4). These group relationships are extracted from a set of sentences that people use to describe the indoor environments (Section 6). As some relationships such as *messy* and *disorganized* have similar arrangements in a scene, we annotate such object groups with multiple labels to provide more instances for each group relation type. As our group relational models characterize the spatial arrangements relating to high-level semantic relations and that they are mostly subjective, we have tried to be as consistent as possible when annotating the scenes.

5.2 Semantic graph from a scene

To build a semantic graph, we first build an object node for each object in the scene and save its category as a node label. We also encode enriched object annotations from ShapeNetSem as node attributes to describe the refined properties of an object. Attributes may refer to many different properties of an object, including shape ("round", "square"), color, material, or usage ("dining", "study"), which could increase the accuracy and control for retrieval of specific object instances as studied in Chang et al. [2015b].

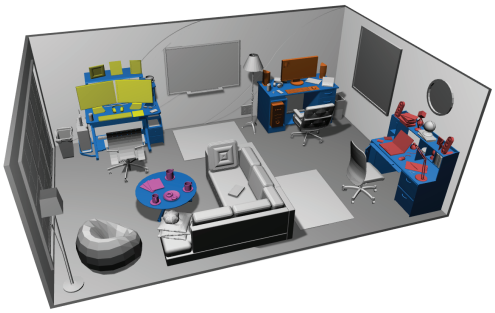


Fig. 4. A database scene annotated with group relationships. Each object in blue corresponds to the anchor object in one relation group; other objects in other color are the active objects annotated for the corresponding group: yellow, orange - *organized*; red - *messy, disorganized*; magenta - *casual*. Note that yellow and orange objects belong to two independent *organized* groups; the group of red objects are annotated with two group relationship types.

Next, we extract the relationships between objects and encode them into the graph. We use a similar approach as in [Chang et al. 2014b] to extract a set of pre-defined spatial relations by examining arrangements of the bounding boxes of two objects. The list of spatial relations include: *on (vertical support), on left, on right, on center, under, left side, right side, front, back* and *near.*

As spatial relations (such as "left side" and "front") are ambiguous when using different reference frames, previous works [Chang et al. 2014a,b] use a view-centric coordinate frame to extract and apply such relationships. Unlike them, we employ an object-centric reference frame to extract spatial relationships between objects. This allows us to directly extract and retrieve view-oblivious object relationships from database scenes for future language-driven object layout. For each object type, we first define a coordinate frame based on its front facing direction. Then, we adjust this object-centric frame based on how humans would perceive and interact with the object (Figure 5(a)). Given a pair of objects, we define one object as the *anchor object* and the other as the *active object*, and record the relative position and orientation of the active object in anchor's frame for future learning (Figure 5(b)).

For each relation (pairwise or group), we create a relation node and connect it with the related objects. Labels are added to the edges based on whether the connected object is an anchor or active object. Unlike previous approaches that encode relations as edge labels between the object nodes, we add relation nodes to the semantic graph, which facilitates representing group relationships more clearly, as all involved objects in a group relation will be linked to just one relation node (e.g., see the "Surrounded by" node in Figure 3).

5.3 Relational model

Given the scene database, we learn a *relational model* to encode object relationships (pairwise or group) for a given text description. The relational model contains an *arrangement model* $\mathcal{A}$, which records the spatial distribution of objects w.r.t the anchor object,

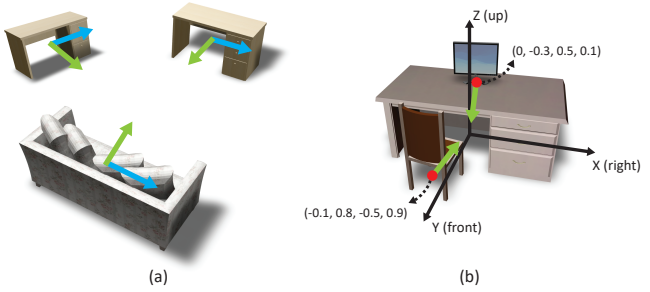


Fig. 5. (a) A scene with two desks and a sofa annotated with adjusted object-centric frame: front (green), right (cyan; note that the sofa and desk have different handed frames); up is the same as the scene's up and is not shown here; (b) Position (red dot) and Orientation (green arrow) of active objects - chair and monitor in the frame originated at the center of the desk (anchor object); relative position is normalized by desk's bounding box size and orientation is normalized by pi, and together they are represented as (x, y, z, θ).

and an *object occurrence model* O, which describes the occurrence of individual objects and the co-occurrence of object pairs in a group relationship.

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

$$\mathcal{A}(o_{\mathbf{act}}, o_{\mathbf{anchor}}, r) = \mathcal{P}(x, y, z, \theta), \tag{1}$$

where $o_{\mathbf{act}}$ is the active object, $o_{\mathbf{anchor}}$ the anchor object, and r the relationship between them. $\mathcal{P}(x, y, z, \theta)$ is the probability distribution of relative position $[x, y, z]$ and orientation θ between the active object and the anchor object. We first assume each arrangement model to be a Gaussian Mixture Model (GMM) and learn the parameters from observations in the database scenes, similar to Fisher et al. [2012].

As for some relations, observations of certain object categories may be limited and cannot be fit with reliable Gaussian distributions (e.g., a stapler on the desk does not have many instances in the entire database, including SUNCG dataset). Based on our experiments, we set a threshold of 15 on the observation count, for fitting a GMM . If the observations are less than the threshold, we directly record all observed (x, y, z, θ) tuples for the corresponding arrangement model and fit a discrete probability distribution. This discrete arrangement probability is defined as: $\mathcal{P} = 1$ if a new object placement is close to any saved observation within a certain spatial threshold (5 cm for the position and 15° for the orientation), otherwise $\mathcal{P} = 0$. Such an arrangement model definition will allow objects to be placed based on all observed locations and orientations, even if the observations of particular object relationships are limited.

Group relationships. For a group relationship, we first define the occurrence model characterizing the occurrence probability of an object in a group as:

$$O(o_i^m, r) = C(o_i^m, r)/C(s, r), \tag{2}$$

where $C(o_i^m, r)$ is the number of scenes annotated with relationship r that are observed with m instances of o_i. $C(s, r)$ is the total number

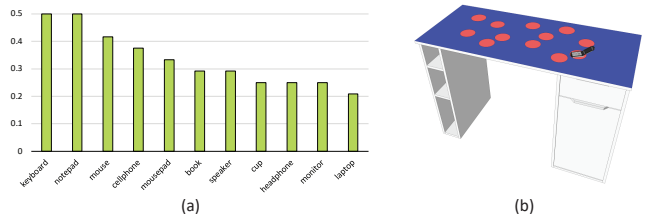


Fig. 6. An example of learned group relational model *(desk, messy)*: (a) Occurrence probability of active objects; (b) Arrangement of cellphone on a desk; as the observation number is not enough to fit a good distribution, we store all observations and use the discrete arrangement model for placing the cellphone.

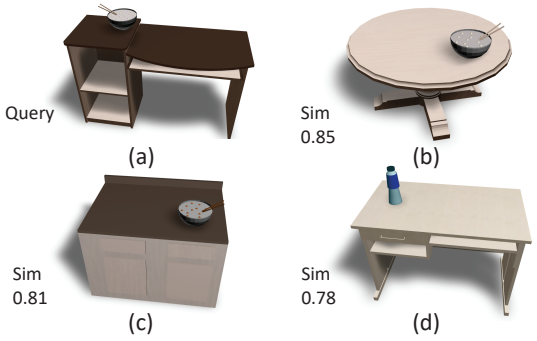


Fig. 7. A query pairwise relational model (a) $\mathcal{A}(bowl, desk, on)$ and three retrieved models (b) $\mathcal{A}(bowl, table, on)$ (c) $\mathcal{A}(bowl, counter, on)$ (d) $\mathcal{A}(bottle, desk, on)$ ranked by the relation similarity.

of scenes s annotated with r. We also compute the co-occurrence probability of an object pair in a group using

$$O(o_i^m, o_j^n, r) = \frac{C(o_i^m, o_j^n, r)}{max\{C(o_i^m, r), C(o_i^n, r)\}}, \tag{3}$$

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

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

$$\mathcal{A}(\mathbf{O}, r) = \sum_{o_i, o_j \in \mathbf{O}} \omega \mathcal{A}(o_i, o_j, r), \tag{4}$$

where $\mathbf{O}$ is the set of objects in the group; ω is the weight for the corresponding pairwise model $\mathcal{A}(o_i, o_j, r)$, which equals to 1, if one object in the pair is the anchor; otherwise ω is set to $O(o_i, o_j, r)$, i.e., the co-occurrence probability of object instance o_i and o_j in the group. Figure 6 shows a learned model for a group relationship.

Relational model similarity. To further harness the knowledge from the scene databases, we define the similarity between relational models and allow to retrieve and sample from similar relational models to find additional placements for an object, when needed. For any two pairwise relational models with the same relation r, their similarity is defined analogously to that of *graph kernel* in

Fisher et al. [2011]. Specifically, we define

$$Sim(\mathcal{A}_o, \mathcal{A}_{o'}) = k(o_{\textbf{anchor}}, o'_{\textbf{anchor}}, r)k(o_{\textbf{act}}, o'_{\textbf{act}}, r), \quad (5)$$

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

$$k(o, o', r) = 0.5\delta_{\textbf{cat}}(o, o') + 0.5k_{\textbf{geo}}(o, o', r), \quad (6)$$

here $\delta_{\textbf{cat}}(o, o')$ is a Dirac delta function which returns 1 if object o and o' are in the same category and 0 otherwise; $k_{\textbf{geo}}(o, o', r)$ encodes the similarity of their geometry features including elevation from the floor, bounding box height and volume. Figure 7 shows an example of relational model retrieval.

Relative relation priors. When editing scenes based on language, object placements must satisfy the *explicit* relations that are specified by the input sentence. At the same time, *implicit* relations imposed by existing objects must also be satisfied when placing a new object, as there might be objects already in the scene. Therefore, in addition to the specific pairwise or group relations, we also learn the *relative* relations, which encode arrangements between all pairs of objects in the database to provide prior knowledge of implicit constraints for object placements. The relative relation model is represented as $\mathcal{A}_I$ and defined similar to Equation 1, but without the specific relationship constraint r. The relative relation priors and the specified relational models are jointly used to provide plausible constraints when placing new objects (Section 7.2).

6 NATURAL LANGUAGE PROCESSING

People use many types of language to interact with scenes. Sentences such as "there are three plates on the table" are descriptive, indicating the way the user desires the scene to be. On the other hand, a sentence such as "put three plates on the table" are commands, indicating changes the program should make to the scene. In both cases, we attempt to transform the natural language input into an equivalent semantic scene graph as detailed in Section 4. Sometimes it may not be possible to represent a command as a scene graph; e.g., "remove the plates from the table". We represent such sentences as scene graphs with verb annotations that connect to entities in the graph.

6.1 Text parsing

We use the Stanford CoreNLP framework [Manning et al. 2014] to perform part-of-speech tagging and convert input statements into a dependency tree. This dependency tree assigns a parent token and annotation label to each token in the sentence; an example is shown in Figure 8. We use the Universal Dependencies representation[1] for our dependency relationships.

6.2 Entity-command representation

We seek to convert the low-level dependency representation shown in Figure 8 into a list of entities annotated with attributes and relationships, as well as a list of command verbs which operate over these entities. We call this the *entity-command representation* (ECR) of the sentence; see Figure 10 for an example.

A scene entity consists of the following:

[1]http://universaldependencies.org/

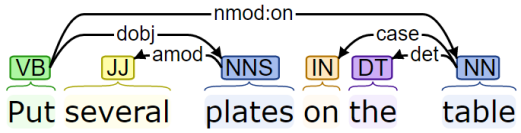


Fig. 8. An example dependency parse using "Enhanced++ Dependencies" from Stanford CoreNLP. VB=verb, JJ=adjective, NN(S)=(plural) noun, IN=preposition, DT=determiner. Edges represent directed dependencies between tokens, such as dobj=direct object and nmod=nominal modifier.

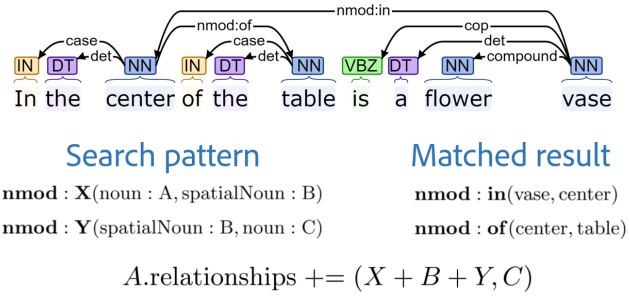


Fig. 9. A pattern matching example using spatial nouns. The "in center of:table" relationship will be recorded for the vase entity.

- **Category** — the base noun used to describe the object(s). Ex. "table", "plate", "arrangement".
- **Attributes** — a list of attributes, each of which may have a set of modifier words. Ex. "modern", "blue (very, dark)".
- **Count** — either an integer representing the number of entities in a group, or a qualitative descriptor. Ex. "many", "some".
- **Relationships** — a set of (string, entity) pairs that describe a connection to another specific entity in the sentence. Ex. "on:desk", "left-of:keyboard".
- **Determiners** — a list of determiners such as "a", "another", "the", "each". These are useful for estimating object binding (Section 7.2).

A command verb is defined by the following:

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

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

Coreference. To dereference pronouns, we use the coreference information obtained from the CoreNLP framework. This assists with sentences such as "Add a dining room table and put plates on top of it", where we will not create a new entity called "it". However,

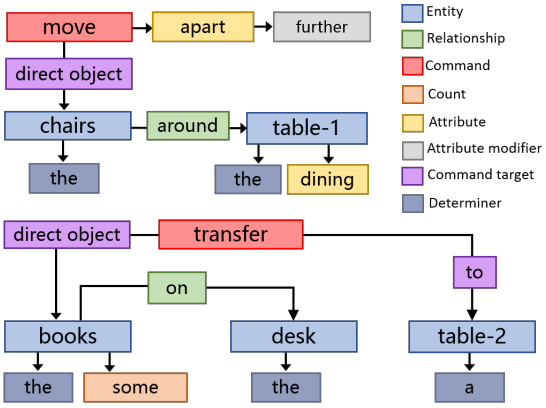


Fig. 10. The sentence "Move the chairs around the dining table further apart and transfer some of the books on the desk to a table" in our entity-command representation.

we do not use coreference for non-pronoun scenarios such as the two table entities in Figure 10. In this case, it is possible that the two tables refer to different tables in the scene, and we will resolve this ambiguity when we align entities to objects in the user's scene (Section 7.2).

6.3 Pattern matching

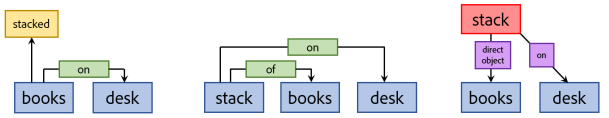
After determining the seed tokens for all scene entities and commands, we use pattern matching over the dependency parse to assign all the other properties enumerated in Section 6.2. For some dependencies, this pattern matching is very simple; for example, **amod**(noun : **A**, adjective : **B**) assigns token **B** as either an attribute or a count of the entity seeded at **A**, if one exists (see Figure 8 for an example). When pattern matching, we augment the standard parts of the input text used by our dependency parser with four special classes that are important for scene understanding:

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

An example of a more complicated pattern involving spatial nouns is shown in Figure 9.

6.4 Canonical entity-command representation

The same scene editing concept can be expressed in many different ways, resulting in different representations; see Figure 11 for an example. We define the descriptive form, where base nouns are objects that have sets of attributes that describe them, to be canonical and when possible transform our input entity-command representation into the descriptive form. These transforms are also executed via a set of pattern matching rules, which transform forms such as Figure 11(b) and (c) into the descriptive form. We detail the patterns used



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

Fig. 11. Entity-command representation of three sentences representing the same underlying concept. We define the descriptive form, (a), to be canonical and transform (b) and (c) into (a) by group noun transformation and verb application, respectively.

in our system for both dependency tree parsing and conversion to a canonical form as supplemental materials.

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

6.5 Conversion to a semantic scene graph

Our entity-command representation transforms easily into a semantic scene graph as defined in Section 4.

Base noun to object category. Our model database uses a fixed set of object categories, and we start by mapping the base noun for each entity into a corresponding object category. We use equivalence sets derived from WordNet[2]. We discard entities that do not map to a category in our model database. Attributes and determiners are added as annotations to the corresponding object nodes.

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

Qualifiers. Some qualifiers such as "each" and "all" imply the presence of multiple object nodes, but are left as qualifiers over a single object node until the SSG is grounded (Section 7.2).

Relationships. Relationships in the ECR transfer directly into relationship nodes in the SSG. The edge label is determined by the directionality of the relationship in the ECR. Relationships that support multiple objects are grouped together into a single relationship node in the SSG; for example "the table is surrounded by two benches and some chairs" will create one "surrounded by" relationship node with appropriate edge labels shown in Figure 3.

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

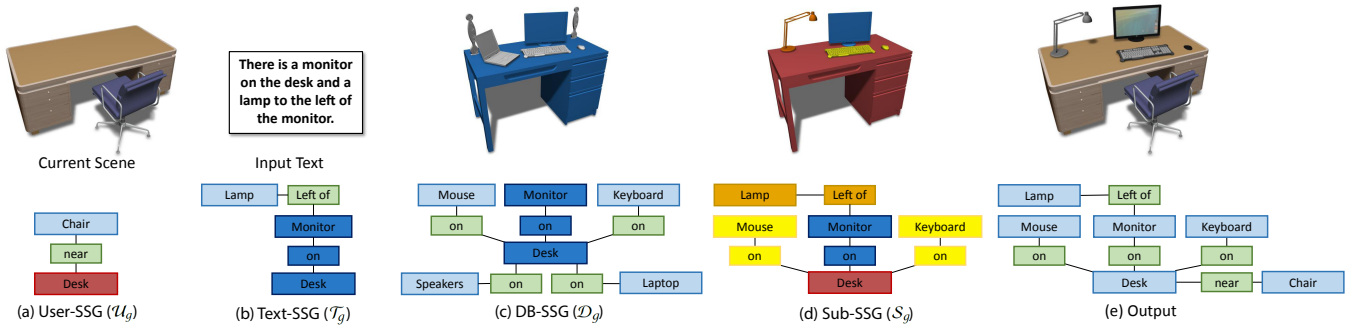


Fig. 12. (a) An input scene and its graph $\mathcal{U}_g$; (b) an input text and its graph $\mathcal{T}_g$; (c) a retrieved database scene and $\mathcal{D}_g$ using the text with aligned nodes shown in blue; (d) a sub-scene $\mathcal{S}_g$ extracted from $\mathcal{D}_g$ with synthesized nodes (in orange) and nodes enriched by context (in yellow); desk is in red since it is aligned to the desk in (a); (d) updated scene by transforming objects from the sub-scene to current scene using desk as the anchor object.

7 LANGUAGE-DRIVEN SCENE EDITING

Given a semantic scene graph constructed from an input sentence (Text-SSG, noted as $\mathcal{T}_g$), our system has two modes for evolving the user-engaged scene (User-SSG, noted as $\mathcal{U}_g$). For graphs *without* a verb node (Figure 12(b)), our system starts by aligning $\mathcal{T}_g$ with semantic scene graphs of the database scenes (DB-SSG, noted as $\mathcal{D}_g$) and finds a subgraph (Sub-SSG, noted as $\mathcal{S}_g$) that best matches the given sentence (Figure 12(c)). Unaligned nodes from the sentence are added to $\mathcal{S}_g$ as synthesized nodes. Furthermore, we enrich this $\mathcal{S}_g$ with scene context. For each updated $\mathcal{S}_g$, we align and merge it to $\mathcal{U}_g$ and insert unaligned objects to current scene, based on their relationship to the aligned objects (Figure 12(d, e)). Whereas for graphs *with* verb nodes, we directly align their object nodes to objects in the current scene, and execute the scene editing functions specified by the verb. By repeating this process, complex and realistic scenes can be generated by using a sequence of sentences.

7.1 Sub-scene retrieval and augmentation

Our goal is to extend the current scene by the obtained set of sub-scenes as well as the specified relationships in the text. One way to generate a candidate sub-scene is to synthesize it with object distributions learned from a 3D scene database (e.g., as proposed in Text2Scene [Chang et al. 2014b]). However, synthesizing complex scenes by explicitly specifying each object is tedious due to the number of objects and the inherent ambiguity of language. To overcome these limitations, we employ the scene context from already existing scenes. We retrieve related sub-scenes by aligning $\mathcal{T}_g$ to each $\mathcal{D}_g$ from our scene database. The output is a list of $\mathcal{S}_g$ ordered by the matching score to $\mathcal{T}_g$.

Text-scene graph alignment. A node in $\mathcal{T}_g$ represents an object or a relationship that the user explicitly wants to be present in the intended scene. We align these nodes with the nodes of every $\mathcal{D}_g$ to find the matches that best correspond to the candidate sub-scenes (blue nodes in Figure 12(c)). A node in $\mathcal{D}_g$ can only be aligned with one node in $\mathcal{T}_g$. An object node is aligned when its category and associated attribute labels are all matched, and a pairwise relation node is aligned when its type and two connected object nodes as well as the edge labels are matched. For a group relation node, since the text may only mention the anchor object to represent a group, e.g., a messy table, we set the node to be aligned when its type and connected anchor object are matched. To find the best matched sub-scene from $\mathcal{D}_g$ for a given $\mathcal{T}_g$, we rank the alignment based on following metric:

$$\mathcal{M}(\mathcal{T}_g, \mathcal{D}_g) = \sum_{N_i \in \mathcal{T}_g, N_j \in \mathcal{D}_g} M(N_i, N_j). \tag{7}$$

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

Language-driven scene graph augmentation. As the database is unlikely to store a $\mathcal{S}_g$ exactly as specified by the given text, some nodes in $\mathcal{T}_g$ may be unaligned because an object instance is missing, or some relationships are not satisfied by the database scenes. Since every node from the input text is crucial in producing the desired scene, we synthesize missing nodes in the subgraph to ensure exact alignment with the given text. New object or relation nodes are added to the retrieved $\mathcal{S}_g$ so that each of them is matched with an unaligned node in $\mathcal{T}_g$. Next, we link the synthesized nodes to existing ones according to the edge connections specified in $\mathcal{T}_g$. By now, we get a $\mathcal{S}_g$ (the graph without the yellow nodes and related edges in Figure 12(d)) with all nodes and edges exactly aligned to $\mathcal{T}_g$. Moreover, as the group relation node is only aligned based on the anchor object, we further synthesize nodes for active objects in the group based on the occurrence (Equation 2) and co-occurrence model (Equation 3), and add their connections to the anchor object. Since a synthesized object node does not correspond to a fixed object instance, we use the same object model if an object with same category appears in current scene; otherwise we randomly sample an object model from the database according to its category.

Enriching $\mathcal{S}_g$ with scene context. Natural language does not conclusively describe objects and their relationships. Moreover, it is challenging to produce complex scenes and evolved object relationships by only specifying them with text. To enable the generation of scenes with a high level of detail and complexity, we exploit the stored scene context from the scene database to enrich the sub-scenes obtained from the previous step. Similar to Chang et al. [2014b], we use the scenes in the database to learn the support

hierarchy of objects. If the most likely support parent of an object node is not present in a retrieved $\mathcal{S}_g$, we find its parent node in the original $\mathcal{D}_g$ and add related nodes to the subgraph. Moreover, we incorporate more scene context to enrich sub-scenes by introducing relevant objects based on their co-occurrence probability w.r.t the initial objects encoded in $\mathcal{S}_g$. The co-occurrence probability of two object categories is computed using Equation 3, while the relationship r is restricted to sibling, i.e., we only consider objects supported by the same parent object when introducing the scene context by object co-occurrence. Objects with co-occurrence probability larger than the context control parameter α will be added to the subgraph.

7.2 Sub-scene accommodation

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

Scene graph alignment and merging. During scene evolution, sentences specified by the user commonly contain objects that already exist in the scene. For example, given the input "There is a desk to the right of the bed", a bed might already be in the scene. Such objects could be used as anchors for graph merging. Specifically, we first align $\mathcal{S}_g$ and $\mathcal{U}_g$ using the similar way of aligning $\mathcal{T}_g$ and $\mathcal{D}_g$ (Section 7.1), but with consideration of determiners extracted from $\mathcal{T}_g$ of an object. Besides the category, two object nodes are aligned when the corresponding determiner of the object is *"the"*, with 50% likelihood if the determiner is absent or *"a"*, and are never aligned if the determiner is *"another"*. Moreover, if the aligned object nodes have qualifiers such as *"each"* or there are multiple aligned objects in $\mathcal{U}_g$, we also duplicate their connected edges and nodes in $\mathcal{S}_g$ based on the number of aligned objects in $\mathcal{U}_g$. For example, "put a lamp on each nightstand" will bring two lamps into the scene if there are two nightstands in $\mathcal{U}_g$. Then, we merge $\mathcal{S}_g$ to $\mathcal{U}_g$ by using the aligned object node in $\mathcal{U}_g$ as anchor and add all non-aligned nodes and related edges from $\mathcal{S}_g$ to $\mathcal{U}_g$. In the 3D scene, objects corresponding to newly added nodes are inserted and their placements are resolved by layout adjustment in next stage.

Layout adjustment. Newly inserted objects come with positions as they are stored in the original database scenes. Therefore, we need to adjust their location to satisfy all relationships encoded in the updated $\mathcal{U}_g$. We compute a transformation matrix to align position and orientation of the anchor object in $\mathcal{S}_g$ and its correspondent in $\mathcal{U}_g$, and set it as the initial transformation matrix for objects from retrieved sub-scene. Since the aligned anchor objects may have different geometric features, applying the initial transformation to these objects is likely to cause artifacts, e.g., intersections, floating objects and wrong orientations. Also, the objects introduced by graph augmentation from $\mathcal{T}_g$ do not have an initial location w.r.t the retrieved sub-scene. Therefore, we refine the placement of each new inserted object based on its specified relationship to the anchor object, as well as its implicit relations to other existing objects in the scene.

For an object o involved in a pairwise relationship, we define the layout score as follows:

$$Score(o) = \mathcal{L}(o) \cdot \mathcal{H}(o) \cdot \mathcal{R}(o). \quad (8)$$

Here $\mathcal{L}(o)$ is the *collision penalty* term which returns 0 if o intersects any object in the scene and 1 otherwise; $\mathcal{H}(o)$ is the *overhang penalty* term defined analogous to that in [Fisher et al. 2012] to prevent o from hanging off of the edge of a supporting surface. $\mathcal{R}(o)$ includes all explicit and implicit relation constraints for o:

$$\mathcal{R}(o) = \omega \sum_{o_i \in \mathbf{O_E}, r \in \mathbf{E}} \mathcal{A}(o, o_i, r) + (1-\omega) \sum_{o_i \in \mathbf{O_U}} \mathcal{A}_{\mathcal{I}}(o, o_i), \quad (9)$$

where $\mathbf{E}$ contains the explicit relationships from the text and $\mathbf{O_E}$ is the set of objects involved in $\mathbf{E}$, and $\mathcal{A}(o, o_i, r)$ represents the pairwise arrangement score for o; $\mathbf{O_U}$ is the set of objects in current scene and $\mathcal{A}_{\mathcal{I}}(o, o_i)$ is relative relation prior between o and o_i; ω is set to 0.7 in our current implementation to weight more on the explicit constraints. Layout scores for group relationships are extended by considering the collision and overhang penalty for each object in the group, and replacing pairwise arrangement model for explicit relations in Equation 9 with the group arrangement model $\mathcal{A}(\mathbf{O}, r)$ (Equation 4).

Ideally, the placement of an object retrieved from a sub-scene would immediately produce the maximum layout score. In practice, the object arrangement may already violate the relation constraints after being aligned to the scene. Therefore, we define a layout quality threshold for an object based on the observed distribution for the explicit relation and its relative relations to existing objects. When the initial alignment causes an intersection or fails to pass the threshold, we optimize the above layout score by hill climbing, i.e., adjusting the layout of the object using the placement that produces the maximum score. New candidate locations are sampled depending on the distribution learned by the relational model. A location is blocked from future sampling if the layout score returns 0. In the case that there is no distribution learned and points near all observations in the current relational model have been tested, we find more candidate positions by sampling from the most similar relational models which are retrieved using Equation 5.

For the placement of a group, object positions are adjusted in an order based on their level of support hierarchy and bounding box sizes. Thus, larger objects which support other objects will be placed first. As the object number increases during the scene evolution, there might be no position to place a new object. Similar to [Ma et al. 2016], when the placement of an object fails to be within a prescribed threshold, we allow the layout algorithm to roll back to the previously placed object, modifying its placement in seeking a relaxed solution. If an object still cannot be placed after one roll-back step, we will skip placing this object and return a failure message to the user.

Scene editing by verb commands. If $\mathcal{T}_g$ contains a verb node, we directly align its related nodes with the current $\mathcal{U}_g$ to find anchor and target objects. We define a set of functions based on commonly used verbs for scene modification: *Replace(A), Move_to(A,B), Move_on(A, B), Move_closer(A, B), Move_apart(A, B), Delete(A), Rotate(A, degree), Scale(A, value)*, where A is the target object or objects for the verb command, and B is the anchor object. The effect of these functions

is the same as indicated by their names. Although some of these editing functions can also be performed through a click-and-drag UI, defining these operations through verb commands allows a more efficient way of editing groups of objects. For example, replacing the chairs around a table using a normal 3D UI may involve inserting new instance of chairs, aligning them with each existing chair and deleting the original chairs, which would need a larger number of operations than using text commands for editing.

7.3 Suggestive interface

Our system provides a suggestive interface to support two-way communication with the users (as shown in Figure 1). For each input sentence, our system produces a set of suggestions (set to 5 when producing all results in the paper). Users can interactively explore all suggested scenes and choose the one they favor the most for the next iteration of text to scene generation. When ambiguities exist in the input text, the system returns possible scene arrangements as suggestions. For example, when there are two desks in the current scene, and the textual command is "put a monitor on the desk", our system will return two possible results with the monitor on either one of the desks.

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

8 RESULTS AND EVALUATIONS

We present the results of our language-driven scene synthesis method, evaluate its performance, and compare the results to those created by artists and those obtained by the state-of-the-art text-to-scene generation method of Chang et al. [2014b]. We demonstrate various aspects of our modeling tool in the supplementary video. Furthermore, the code and data are available on our project page.

Scene synthesis results. Figure 21 shows a gallery of 3D scenes generated by our method. Taking advantage of sub-scene level scene synthesis, group relational models and adding contexts from scene databases, we are able to achieve a higher level of language efficiency for scene modeling than previous attempts at text to scene generation. For example, a scene with 20 to 30 objects can be synthesized with only three sentences (Figure 21, last column). More synthesis results can be found in Figure 1 and supplementary materials.

Parameter and timing. The main tunable parameter in our method is α, which controls the level of introducing context objects into the scene. The conversion of input language to semantic scene graphs is instantaneous. For scenes of 15-20 objects, sub-scene retrieval and synthesis take 1-5 seconds, excluding the 3D model loading time, on a Windows PC with a 2.7 GHz i7 CPU, 16 GB RAM and a NVIDIA GTX 750 Ti.

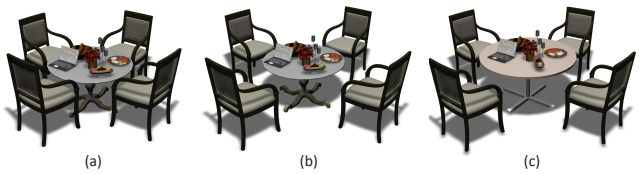


Fig. 13. Verb commands applied to refine the current scene (a). (b) scene after "move the chairs apart from the table". (c) scene after applying the command "replace the table"; note that the elevation of all objects on the table are increased since a taller table is inserted.

Verb commands. Verb commands serve as secondary editing options in our modeling tool to complement synthesis commands. They are realized via pre-defined scene editing operations and potential layout adjustment. Verb commands are useful when a scene needs to be refined, but no object or sub-scene retrieval is required (Figure 13(b)). Another use case is object replacement, as shown in Figure 1 and Figure 13(c), allowing the user to refine scenes more quickly.

Comparison to [Chang et al. 2014b]. The key evaluation for our method is whether the generated scenes are plausible and natural. Similar to previous works [Fisher et al. 2015; Ma et al. 2016], we leave such judgments to human participants. Additionally, we compare our results to those generated by Chang et al.[2014b]. Unlike their work, our system supports group relations. For the comparison, we used the implementation provided by the authors and ran our method using relational models trained from SceneSynth [Fisher et al. 2012], on which Chang's is trained.

We set up 10 editing scenarios, each described by three input sentences, with α set to 0 (i.e., no scene contexts). The editing commands cover bedroom, dining room, living room, and office scenes, with both pairwise and group relations were accounted for. However, since Chang et al. [2014b] only model pairwise relations, we split specifications of group relations into sets of pairwise constraints. Further, we chose the best result from Chang et al. out of five instances run through their implementation.

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

Our first user study is the Plausibility Test against Chang et al. (PTC). We split the total of 60 scenes generated for this study into two sets. Each subject is shown three scenes from the three methods/options at a time. The subjects were asked to give a score from 1 (least favoured) to 5 (most favoured) to each scene based on two criteria: plausibility – whether the scene is plausible with respect to the given editing command, and naturalness – whether

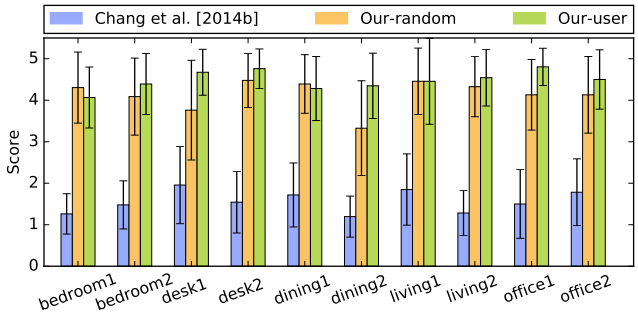


Fig. 14. PTC study results: average plausibility-naturalness scores by subjects for all 10 scene editing scenarios.

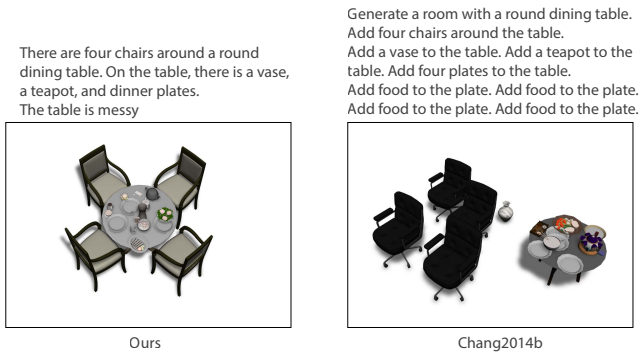
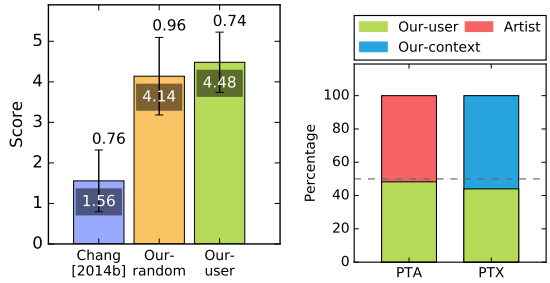


Fig. 15. Qualitative Comparison: The figure shows the generated scenes using our method and that of Chang et al. [2014b]. Since Chang et al. [2014b] do not use group relations like *messy*, group relations are fragmented into pairwise relations and fed to their system.

the scene and its object arrangement appear natural. We collected feedback from 23 participants, resulting in 46 scores per test scenario per method/option and 460 scores per method/option. All the user study materials are provided as supplementary materials.

Figure 14 plots the average subject scores and variances for each test scenario. Figure 16(a) shows the outcome of the user study. Our method outperforms Chang et al. [2014b] with average scores of 4.14 (Our-random) and 4.48 (Our-user), compared to 1.56 for Chang. We attribute this to a number of improvements, including the ability to handle group relations (Section 5.3), the transformation into a canonical graph representation (Section 6.4), and the sub-scene retrieval and accommodation pipeline (Figure 12). Interestingly, Our-random is rated only slightly lower than Our-user, suggesting that our method has a good average performance. In Figure 15, we present one case where Chang et al. [2014b] fails whereas our system succeeds.

Comparison to artist creations. Our second user study is the Plausibility Test against Artist (PTA), where we replaced the method of Chang et al. [2014b] by a professional artist and repeated the PTC study. When the artist created the scenes, she was given the natural language prompt, the same object database used by our tool, and was not limited by modeling time. The artist spent at least five minutes for modeling each scene. Importantly, the artist had total



(a) PTC average (b) PTA and PTX average

Fig. 16. Average PTC scores and PTA and PTX percentages.

freedom in choosing which objects to add to the scene, based on the language input, and how to place them using a modeling software. Therefore, the comparison covers the full spectrum of our scene synthesis method.

Instead of asking the subjects to score scenes, we asked them to *vote* between a pair of scenes, one created by the artist and the other an Our-user scene from PTC. The subjects were asked to vote based on their judgment of plausibility and naturalness of the scenes. We received feedback from 20 subjects, resulting in 400 subject votes over the 10 editing scenarios since the artist also generated two scene variants per scenario. Figure 16(b) (first column) plots the average percentages of the artist and Our-user results being voted on by the subjects, respectively. Per test scenario results from the study can be found in the supplementary material. Overall, our method received 48.25% of the votes. Based on Wilson score confidence interval for Bernoulli trials [Agresti and Coull 1998], there is a 95% chance that human users would prefer our method at least 43.3% of the time.

Effect of adding context. Our third user study is the Plausibility Test with Context (PTX), where we aim to evaluate the effect of adding scene contexts to the generated scenes. In this study, we set the context control parameter α to 0.5. For each query, two scenes were presented to the subjects, one generated with context added (Our-context) and the other is an Our-user result from the previous studies, without adding context. Feedback was received from 25 subjects, who selected for each editing scenario (from the previous two studies), which of the two scenes was preferred based on plausibility and naturalness, as before. As shown in Figure 16(b) (second column), Our-context scenes received 54% of the total votes over all 10 editing scenarios, suggesting that there is a 95% chance that our method is preferred by the users at least 47.8% of the time and adding context does improve favourability of our results.

Reproducing a photographed scene. In our fourth user study, we essentially repeat the PTA but under a realistic modeling scenario with available ground truth. Specifically, we test the efficacy of our text parsing and scene generation modules in *reproducing* a 3D scene when shown a photograph of it, where the photograph serves as the ground truth. Given a photograph of a 3D scene, users are asked to describe the scene in natural language and then apply the language description to interactively synthesize a 3D scene using our tool under the "Our-user" setting. That is, at each step, users

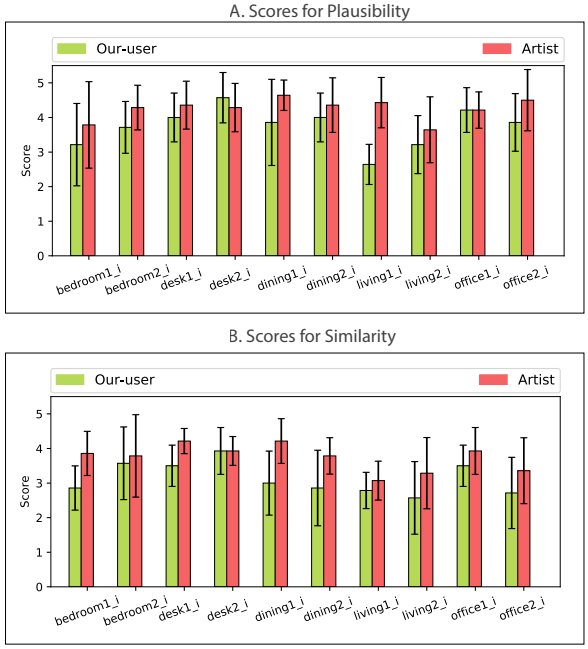


Fig. 17. Results from the user study on reproducing photographed scenes: user plausibility ratings of generated scenes (top) and user similarity ratings (on a scale of 1 to 5) of the scenes to reference photographs (bottom).

select the best suggestion from among the five options our tool offers. On average, a user spent 4-5 minutes to generate one scene. The photographed scenes cover multiple categories as shown in Figure 17. To distinguish between 3D scenes in PTC and this study, the scene names here have been appended with 'i'(interactive scene synthesis from photographs). The study employed 10 participants, all graduate students in visual computing, each describing 10 scenes, resulting in a total of $10 \times 10 = 100$ distinct text descriptions.

A *separate* group of participants were asked to rate the plausibility of the generated scenes (with respect to the given descriptions) and their similarity (in terms of object layout) to the original photographs. We then pass the same set of scene descriptions to an artist to create 3D scenes accordingly, where the available objects are the same as those from the corresponding user-generated scenes. Note that the artist was not shown the reference photograph. On average, the artist spent 10-12 minutes to create one scene. Then the same group of raters were asked to score the artist creations. All data and results for this study are provided in the supplementary material.

Figure 17 shows the plausibility and similarity ratings for this study. We can see that the average ratings from Our-user scenes are comparable to those of artist's creations. Some factors that could have affected the scores are: (1) There was no time constraint on the artist. (2) Sometimes the users did not notice an object while describing a photographed scene. When it comes to similarity w.r.t the reference photograph, the overall scores go down for both Our-user and artist created scenes. This is likely due to natural languauge being ambiguous and the measure of similarity being subjective.

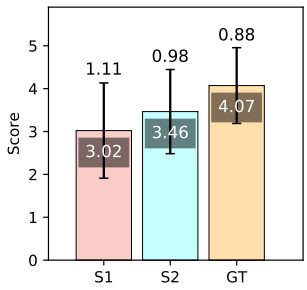


Fig. 18. User ratings (on a scale of 1 to 5) for two user-generated scenes (S_1, S_2) and the reference scene (GT) based on how well they match a description of GT that is *different* from those that generated S_1 and S_2. A Score of 3 is for "Overall matches the description", 4 for "Matches the description closely" and 5 is a "Perfect Match".

Robustness against language variations. In our final user study, we test the robustness of our scene modeling tool against language variations. This study reuses the user descriptions of the photographed scenes from the previous (fourth) study; these descriptions are expected to contain varying degrees of variations. We selected two best user-generated scenes (S_1, S_2) resulting from the previous user study and grouped them with the corresponding reference photograph (designated as GT). Both S_1 and S_2 were rendered to images from the same view point as for the reference (GT) photograph. Then, among the remaining 8 user descriptions for GT (i.e., those not corresponding to S_1 or S_2), we selected the best description D*. A survey records scores from 15 users for the 10 groups of scene triplets, presented in random order, on the basis of how closely each rendered scene (S_1, S_2, or GT) matches the description D*.

Figure 18 shows the average scores for the three scenes. As expected, GT holds the highest score since the description D* is meant for it. Scenes S_1 and S_2, which were generated by our tool based on descriptions that are *different* from D*, also obtained reasonable, above-average scores, reflecting a certain degree of robustness of our tool amid language variations. Figure 19 shows two examples of S_1 and S_2 where the corresponding scene descriptions vary in sentence order and details. Note that the raters saw all three scenes including the GT, whose presence can serve as a good reference but may also compromise the scores received by S_1 and S_2.

Failure cases. Two representative failure cases from our scene modeling method are illustrated in Figure 20. First, our method cannot guarantee global plausibility of the generated scenes, since it processes the input sentences sequentially and can only produce results based on the relations encoded in each individual sentence. As shown in Figure 20(a), while correct scene accommodation is executed for each sentence, the final generated scene has a globally implausible layout. Second, our scene synthesis is shape-unaware, as shown in Figure 20(b). Currently, when learning our relational model, we extract the relative positions with respect to the object bounding boxes (see Section 5.3) and do not account for the geometric shapes of the 3D objects. Thus, when the learned model is applied to the objects that are not tightly enclosed in their bounding boxes, the generated scene may be implausible as shown.

Fig. 19. Two user-generated scenes for the same reference photograph, but based on different scene descriptions. As shown, the descriptions vary in sentence order and details, while the corresponding scenes are both plausible with similar layout, except for a symmetric flip and some object details (e.g., the speaker and the lamps were missed in the description on the left).

Fig. 20. Two representative failure cases of our method.

9 DISCUSSION, LIMITATION, AND FUTURE WORK

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

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

Limitations. We still regard our method only as a preliminary prototype for data-driven 3D scene modeling with natural language inputs. In addition to the failure cases shown, there are several other limitations arising out of our current execution:

- *Generality of the learned model.* Our current implementation only supports a limited set of group relations and limited classes of commands for language-scene grounding. Enriching both would require expanded data annotation and natural language processing capabilities beyond what our current implementation can support. Also, with a strict reliance on object co-occurrences and arrangements in the scene dataset for scene accommodation, our current relation model does not generalize well to learning of concepts or abstractions. For example, it cannot learn to generate messy rooms based only on messy tabletops from available data. Learning the *intrinsic* meaning of "messiness" and other group relations would be an interesting direction for future work.
- *Object geometry, symmetry, style, and functionality.* Our current scene synthesis algorithm is not symmetry-aware — there is no guarantee of symmetry between the lamps around a bed, even though the scene data would hint so. Also, our current descriptors for object-object relations are still quite rudimentary. As a result, the system does not take into account finer-scale object geometries, style, functionality, or human-object interactions. Incorporating style compatibility metrics [Liu et al. 2015] and functionality encodings, e.g., via the ICON descriptor [Hu et al. 2015], into the mix are both worth exploring.
- *Creative modeling.* For text-to-scene applications, creative modeling depends on both the user input and the richness of the database. User input can facilitate creativity while knowledge from the database helps in plausibility. Unrealistic or unobserved object arrangements and scene layout could be explicitly specified by the user, like placing a vase on a plate. Our method can generate such object arrangements, to a reasonable extent, depending on the input. However, *truly* creative modeling may be explored with the use of deep generative models in the future.

Future work. Aside from learning and retrieving everything from available 3D scene databases, we could also extend knowledge acquisition to other sources such as ImageNet or KnowledgeNet. We are also interested in applying techniques developed in our work to other contexts. For example, our scene alignment algorithm may hold the potential to enrich a set of synthetic scenes by aligning them with real scenes as a way to produce variations. Ultimately, an intelligent language-to-scene modeling tool should be able to learn and adapt *on-the-fly*. Examples of such intelligence include automatic requests for new or additional scene exemplars, annotations for unknown attributes, or ungrounded textual commands. With the additional data, the semantic scene graphs and underlying learning mechanism can be adjusted and enhanced on the fly.

Using natural language to assist creative tasks is a growing field. Many techniques explored in the context of scene editing using languages have applications to other creative media such as images and video. These systems must all answer challenging questions. How should voice or text input interleave with other input modalities? How do we deal with uncertainty in understanding the user's intent or the system's inability to fully execute the request? Our

work has explored some of these issues in the context of 3D scenes and believe that language-powered interactive systems will soon be able to significantly lower the entry barrier for creative tools.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their valuable comments. This work was supported, in parts, by an NSERC grant (611370), an NSF grant IIS-1528025, the Stanford AI Lab-Toyota Center for Artificial Intelligence Research, the Singapore MOE Academic Research Fund MOE2016-T2-2-154, an internal grant from HKUST (R9429), and gift funds from Adobe and Amazon AWS. We also thank Phuchong Yamchomsuan for creating the artist scenes, as well as Quang-Hieu Pham and Chenyang Zhu for helping with pre-processing the scene databases.

REFERENCES

Alan Agresti and Brent A Coull. 1998. Approximate is better than exact for interval estimation of binomial proportions. *The American Statistician* 52, 2 (1998), 119–126.

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

Angel X. Chang, Mihail Eric, Manolis Savva, and Christopher D. Manning. 2017. SceneSeer: 3D Scene Design with Natural Language. *CoRR* abs/1703.00050 (2017). http://arxiv.org/abs/1703.00050

Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. 2015a. ShapeNet: An Information-Rich 3D Model Repository. (2015).

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

Angel X. Chang, Manolis Savva, and Christopher D. Manning. 2014b. Learning Spatial Knowledge for Text to 3D Scene Generation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Bob Coyne, Alex Klapheke, Masoud Rouhizadeh, Richard Sproat, and Daniel Bauer. 2012. Annotation Tools and Knowledge Representation for a Text-To-Scene System. In *COLING*. 679–694.

Bob Coyne and Richard Sproat. 2001. WordsEye: An Automatic Text-to-scene Conversion System. In *Proc. of SIGGRAPH*. 487–496.

Matthew Fisher, Yangyan Li, Manolis Savva, Pat Hanrahan, and Matthias Nießner. 2015. Activity-centric Scene Synthesis for Functional 3D Scene Modeling. *ACM Trans. on Graph.* 34, 6 (2015), 212:1–10.

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

Matthew Fisher, Manolis Savva, and Pat Hanrahan. 2011. Characterizing structural relationships in scenes using graph kernels. 30, 4 (2011), 34.

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

Ruizhen Hu, Chenyang Zhu, Oliver van Kaick, Ligang Liu, Ariel Shamir, and Hao Zhang. 2015. Interaction Context (ICON): Towards a Geometric Functionality Descriptor. *ACM Trans. on Graph.* 34, 4 (2015), Article 83.

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

Yun Jiang, Marcus Lim, and Ashutosh Saxena. 2012. Learning Object Arrangements in 3D Scenes using Human Context. In *Proc. Int. Conf. on Machine Learning (ICML)*.

Young Min Kim, Niloy J. Mitra, Dong-Ming Yan, and Leonidas Guibas. 2012. Acquiring 3D Indoor Environments with Variability and Repetition. *ACM Trans. on Graph.* 31, 6 (2012), 138:1–138:11.

Tianqiang Liu, Aaron Hertzmann, Wilmot Li, and Thomas Funkhouser. 2015. Style Compatibility for 3D Furniture Models. *ACM Trans. on Graph.* 34, 4, Article 85 (2015), 85:1–85:9 pages.

Rui Ma, Honghua Li, Changqing Zou, Zicheng Liao, Xin Tong, and Hao Zhang. 2016. Action-Driven 3D Indoor Scene Evolution. *ACM Trans. on Graph.* 35, 6 (2016).

Lucas Majerowicz, Ariel Shamir, Alla Sheffer, and Holger H. Hoos. 2014. Filling Your Shelves: Synthesizing Diverse Style-Preserving Artifact Arrangements. *IEEE Trans. Visualization & Computer Graphics* 20, 11 (2014), 1507–1518.

Christopher D. Manning, 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*. 55–60. http://www.aclweb.org/anthology/P/P14/P14-5010

Paul Merrell, Eric Schkufza, Zeyang Li, Maneesh Agrawala, and Vladlen Koltun. 2011. Interactive Furniture Layout Using Interior Design Guidelines. *ACM Trans. on Graph.* 30, 4 (2011), 87:1–10.

Dipendra Kumar Misra, Jaeyong Sung, Kevin Lee, and Ashutosh Saxena. 2014. Tell Me Dave: Context-Sensitive Grounding of Natural Language to Manipulation Instructions. In *Proc. of Robotics: Science and Systems*.

Zeinab Sadeghipour, Zicheng Liao, Ping Tan, and Hao Zhang. 2016. Learning 3D Scene Synthesis from Annotated RGB-D Images. *Computer Graphics Forum (SGP)* 35, 5 (2016).

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

Manolis Savva, Angel X. Chang, Pat Hanrahan, Matthew Fisher, and Matthias Nießner. 2016. PiGraphs: Learning Interaction Snapshots from Observations. *ACM Trans. on Graph.* 35, 4 (2016).

Steven M. Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. 2006. A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms. In *IEEE CVPR*. 519–528.

Lee M. Seversky and Lijun Yin. 2006. Real-time Automatic 3D Scene Generation from Natural Language Voice and Text Descriptions. In *Proc. of ACM International Conference on Multimedia*. 61–64.

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

N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. 2012. Indoor segmentation and support inference from RGBD images. In *ECCV*.

Greg Slabaugh, Bruce Culbertson, Tom Malzbender, and Ron Schafer. 2001. A Survey of Methods for Volumetric Scene Reconstruction from Photographs. In *Proc. of Eurographics Conference on Volume Graphics*. 81–101.

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

Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. 2017. Semantic scene completion from a single depth image. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*. IEEE, 190–198.

Moritz Tenorth and Michael Beetz. 2013. KnowRob: A Knowledge Processing Infrastructure for Cognition-enabled Robots. *Int. J. Rob. Res.* 32, 5 (2013), 566–590.

Kai Wang, Manolis Savva, Angel X. Chang, and Daniel Ritchie. 2018. Deep Convolutional Priors for Indoor Scene Synthesis. *ACM Trans. on Graphics (Proc. of SIGGRAPH)* 37, 4 (2018).

Jianxiong Xiao. 2012. *3D Reconstruction is Not Just a Low-level Task: Retrospect and Survey*. Technical Report. MIT9.S912: What is Intelligence?

Kai Xu, Rui Ma, Hao Zhang, Chenyang Zhu, Ariel Shamir, Daniel Cohen-Or, and Hui Huang. 2014. Organizing Heterogeneous Scene Collection through Contextual Focal Points. *ACM Trans. on Graph.* 33, 4 (2014), Article 35.

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

Lap-Fai Yu, Sai Kit Yeung, Chi-Keung Tang, Demetri Terzopoulos, Tony F. Chan, and Stanley Osher. 2011. Make it home: automatic optimization of furniture arrangement. *ACM Trans. on Graph.* 30, 4 (2011), 86:1–12.

Lap-Fai Yu, Sai Kit Yeung, and Demetri Terzopoulos. 2016. The Clutterpalette: An Interactive Tool for Detailing Indoor Scenes. *IEEE Trans. Visualization & Computer Graphics* 22, 2 (2016), 1138–1148.

C. Lawrence Zitnick, Devi Parikh, and Lucy Vanderwende. 2013. Learning the Visual Interpretation of Sentences. In *Proc. ICCV*. 1681–1688.

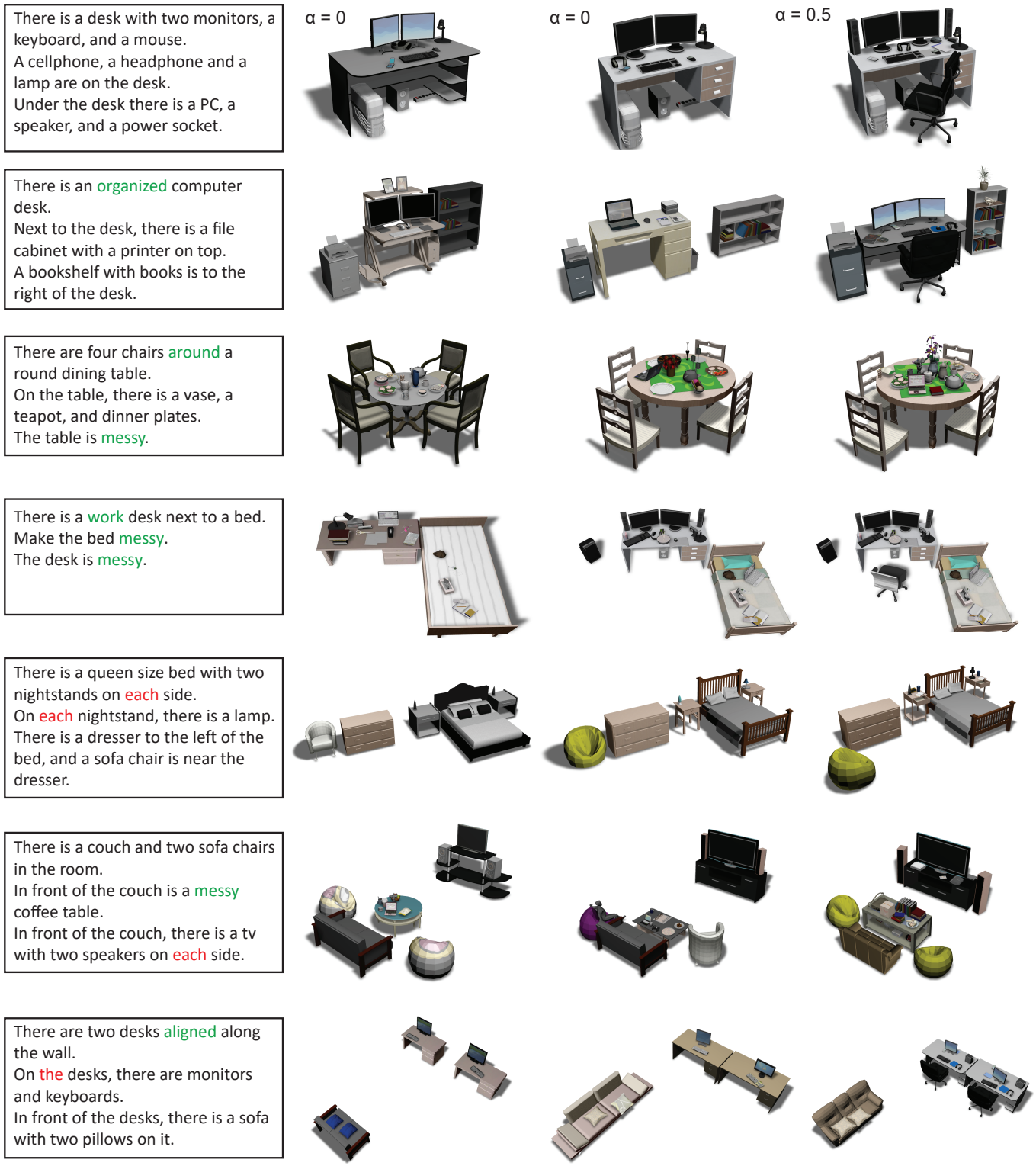


Fig. 21. A gallery of our language-driven scene synthesis results. In each row, we show from left to right: input sentences; an output scene selected from 5 results ($\alpha = 0$); another output scene selected from 5 results ($\alpha = 0$); an output scene with context added ($\alpha = 0.5$). In the sentences, highlighted are words that map to group relations (green) and some key determiners (red) that affect scene binding.