Action-Driven 3D Indoor Scene Evolution

Rui Ma[1*] Honghua Li[2*] Changqing Zou[1] Zicheng Liao[3] Xin Tong[4] Hao Zhang[1]
[1]Simon Fraser University [2]Shandong University [3]Zhejiang University [4]Microsoft Research

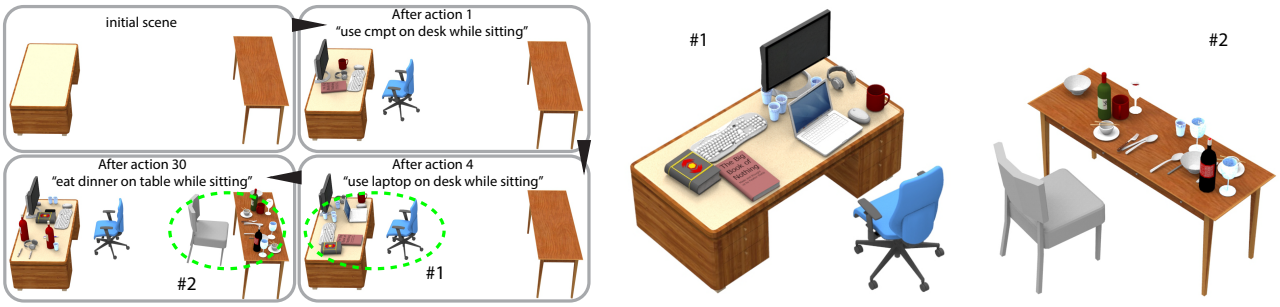

Figure 1: *Action-driven scene evolution alters an initial scene consisting of a desk and a dining table. The initial scene and three intermediate evolution snapshots (after 1, 4, and 30 actions, respectively) are shown (left) with zoom-ins for better viewing on the right. Applied actions trigger both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

Abstract

We introduce a framework for *action-driven evolution* of 3D indoor scenes, where the goal is to simulate how scenes are altered by human actions, and specifically, by object placements necessitated by the actions. To this end, we develop an *action model* with each type of action combining information about one or more human poses, one or more object categories, and spatial configurations of objects belonging to these categories which summarize the object-object and object-human relations for the action. Importantly, all these pieces of information are learned from annotated *photos*. Correlations between the learned actions are analyzed to guide the construction of an *action graph*. Starting with an initial 3D scene, we probabilistically sample a sequence of actions from the action graph to drive progressive scene evolution. Each action triggers appropriate object placements, based on object co-occurrences and spatial configurations learned for the action model. We show results of our scene evolution that lead to realistic and messy 3D scenes, as well as quantitative evaluations by user studies which compare our method to manual scene creation and state-of-the-art, data-driven methods, in terms of scene plausibility and naturalness.

Keywords: 3D indoor scene, action-driven scene evolution

Concepts: •**Computing methodologies** → **Shape analysis;**

*Rui Ma and Honghua Li are co-first authors.

SA '16 Technical Papers,, December 05 - 08, 2016, Macao
ISBN: 978-1-4503-4514-9/16/12
DOI: http://dx.doi.org/10.1145/2980179.2980223

1 Introduction

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

Most indoor scenes available from public data repositories, e.g., the 3D warehouse, possess the organization and cleanness of a showroom; these scenes were mostly designed. In real life, our offices, labs, and bedrooms are often messier. So far, aside from scene construction from images [Liu et al. 2015; Fisher et al. 2015] and sketches [Xu et al. 2013], the predominant approach to realistic scene synthesis has been based on exemplar-based learning [Fisher et al. 2012], where a scene is produced by sampling from a probabilistic distribution learned from 3D scene examples. In real life however, a scene is not a random whole "event", but a snapshot from a continuous scene evolution.

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

We develop an *action model* which supports action-driven 3D scene evolution. The model is *data-driven* and learned from annotated scene data. Each type of action combines information about one

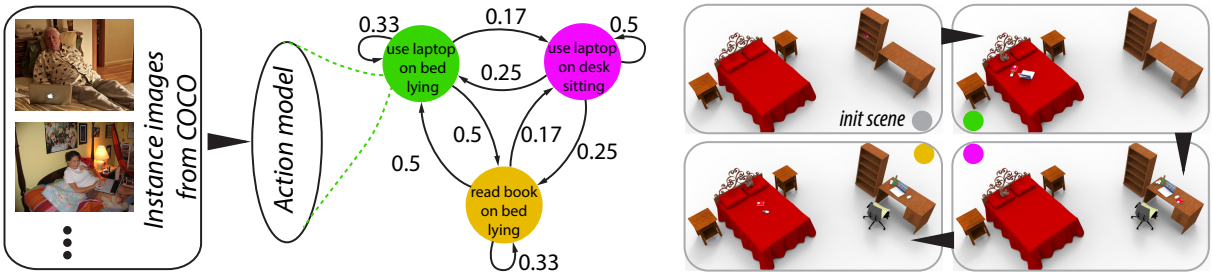

Figure 2: *An overview of our action-driven 3D scene evolution. Action models are learned from annotated photos (left). An action graph (middle) is built, whose nodes are the learned actions, and edges represent transition probabilities between the actions. To drive the evolution of an initial 3D scene, we apply a sequence of actions sampled from the action graph onto the scene. Each sampled action triggers appropriate object placements, i.e., relocation or insertion, producing a continuous scene evolution (right).*

or more human poses, one or more object categories, and spatial configurations of objects belonging to these categories which summarize the object-to-object and object-to-human (the pose in the action model) relations for the action. *Correlation* between the learned actions are analyzed to guide the construction of an *action graph*, whose nodes correspond to actions and edges encode correlations between actions in the form of transitional probabilities. Scene evolution starts with an initial 3D indoor scene. We probabilistically sample an action sequence from the action graph, where each action triggers appropriate object placements, which continuously evolve the scene; see Figures 1 and 2.

A key question facing any data-driven approach is choice of the data. To drive 3D scene synthesis, 3D data of human actions and human-object interactions are most directly applicable. However, acquiring such data in large volume is highly costly with challenges from reconstruction, tracking, and annotation. Annotating existing 3D scenes is an option, but such scenes are limited in number and variety, and they were mostly designed without human presence or intimate connections to human actions. With these in mind, we turn to the vast source of *photographs* of indoor scenes with daily human activities. Specifically, we utilize the Microsoft COCO (Common Objects in Context) database [Lin et al. 2014], which offers a solid baseline for our data requirement: a large number of photos with object segmentations, labels, and text captions describing the contents, including human actions, in each photo. Yet, to learn our action model, much information about human poses and inter-object relations is still missing. Recovering necessary 3D action data to drive 3D scene synthesis is a challenging problem in general.

To learn action models from COCO, we first analyze the photo captions therein to collect photos related to certain actions. Then, photos of the same action are clustered based on object categories and human poses within the photos. Each cluster defines an action node in our action graph. Object co-occurrences, object-human, and object-object spatial relations are learned within each action node. For the action graph, edges are added to all pairs of action nodes, and also a self loop to the node itself. We compute the transitional probabilities by examining the overlap of associated objects. From the action graph, plausible action sequences are generated automatically and stochastically. Each action applied triggers appropriate object relocations or insertions, based on object co-occurrences and spatial configurations learned for the action model.

Example-based synthesis [Fisher et al. 2012] is also data-driven, but it takes a holistic view of scene generation: the produced scene must be similar to the exemplars overall and likely belonging to the same scene category. In contrast, action-driven evolution is a *procedural* and more atomic form of scene generation that is not tied to scene categories; the key criterion is what actions are applicable in a given scene context. For example, the action "read book on desk while sitting" is applicable in any scene with a desk. No less important is the fact that action data is more compact and more atomic than whole scene exemplars. Applying actions one at a time allows more local control and generates scenes with higher granularity.

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

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

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

2 Related work

Aside from serving VR/AR applications, large collections of 3D scenes are valuable both as training data to support machine learning for scene understanding and as model repositories for model-driven 3D scene modeling [Kim et al. 2012; Shao et al. 2012; Xu et al. 2013; Fisher et al. 2015]. Recently, there have been a great deal of work in computer graphics and computer vision on the processing and analysis of indoor scenes, e.g., reconstruction, understanding, and editing. In this section, we only focus on works most closely related to ours, i.e., those on 3D scene generation as well as human- or action-oriented scene processing.

Scene modeling. Interactive, user-centric tools for 3D scene modeling exist commercially, e.g., *Autodesk Homestyler*[1], *Sweet Home 3D*[2], and in the research literature. The use of such tools requires advanced modeling skills and the modeling time is often quite long. Data-driven scene modeling or reconstruction from X has gained much interest lately where X could be a sketch [Xu et al. 2013], a photograph [Liu et al. 2015], or a depth scan [Kim et al. 2012; Shao et al. 2012; Chen et al. 2014]. In these cases, the object

[1]http://www.homestyler.com/
[2]http://www.sweethome3d.com/

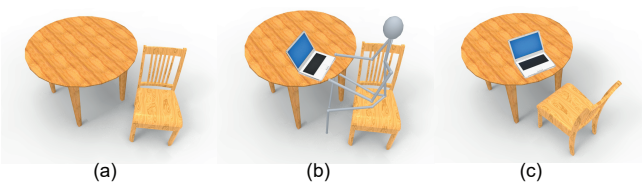

Figure 3: *Given an initial scene (a), placing a laptop based on pairwise human-object relations would rely on a human pose predicted for the chair (b). In contrast, applying the action "use laptop on table while sitting" by our method simultaneously places the laptop and rotates the chair (c), owing to the joint positions learned of all three objects in the scene: table, chair, and laptop.*

Figure 4: *With only a desk in an initial scene (a), the action "use laptop on table while sitting" can insert a chair and a laptop along with other objects; see (b) and (c) for two possible results from our action-driven scene synthesis. In contrast, without a chair in (a), a sitting pose is unlikely to be predicted near the desk solely by pose estimation, hence a chair is unlikely to be placed or retrieved.*

arrangements inferrable from X are fixed and guide the retrieval and placement of suitable 3D objects from a model repository. In our work, we are interested in a more open-ended and less constrained synthesis, not modeling from X.

Furniture layout optimization. Several approaches have been proposed for the furniture layout optimization problem. Germer and Schwarz [2009] arrange a room by letting each piece of furniture act as an agent in a multi-agent system, following manually specified room semantics and furniture layout rules. Merrell et al. [2011] turn furniture layout guidelines into a probabilistic model and suggest sensible room layouts by sampling from the density function, as a user interactively moves furniture in a room. In contrast, Yu et al. [2011] learn the layout rules from 3D scene exemplars. All of these solutions optimize the layout of a given room with a given set of furniture. In our work, we evolve a 3D scene by moving and inserting objects progressively.

Learning from 3D data. Existing 3D scene synthesis methods predominantly resort to probabilistic reasoning from 3D exemplars and 3D scene databases to drive the synthesis [Fisher et al. 2012; Jiang et al. 2012; Fisher et al. 2015; Savva et al. 2016; Sadeghipour et al. 2016]. Some of these methods, e.g., Fisher et al. [2012], take a holistic approach to scene generation, while others progressively alter an initial scene. These approaches could all be limited by the availability of well constructed and annotated 3D scenes. For reference, the state-of-the-art work by Fisher et al. [2012] worked with 130 user-constructed 3D scenes. The lack of data limits both the variability and the scale of the generated scenes. In our work, we utilize thousands of photos with action-related text annotations from the COCO database. The challenge is to recover the 3D scene layout and properly embed human poses into the photos.

Action-driven scene understanding. There has been a great deal of work in robotics and computer vision, and more recently in computer graphics, on utilizing human actions for various analysis tasks. After all, humans understand the world and function in it through their actions. Works that have been applied to 3D scenes include geometry estimation [Fouhey et al. 2012], object labeling [Jiang et al. 2013], and affordance learning [Savva et al. 2014], to name just a select few. The key problem is to fit static human poses or pose sequences into various scene contexts to understand the structure and functionality of a scene and the objects therein. In our work, such a fitting task is necessary. But our focus is not on automated analysis, but on how to organize the fitted human poses and their surroundings, learn a suitable action model, and then apply the model for 3D scene synthesis.

Human- or activity-centric scene modeling. Jiang et al. [2012] propose an interesting, *human-centric* approach to arrange objects in a room, focusing more on human-object relations rather than object-object relations. Specifically, they learn, from 3D scene exemplars, density functions which characterize how each type of object is placed relative to a human pose. The role of the density functions is similar to that of our action model for scene generation. However, one distinction is that these density functions encode *pairwise* human-object relations while an action in our model can encode *joint* relations among multiple humans and objects; see Figure 3 for a comparison. When arranging a room, they first infer possible human poses and then place one object at a time, based on the predicted poses and the learned human-object or object-object relations. In contrast, our actions can trigger the placement of one or more objects at a time and the placement is not predicated only by pose estimation — it accounts for both pose fitting and object co-occurrence; see Figure 4 for a visual example.

Sharf et al. [2013] study object mobilities in 3D scenes and edit scenes by altering object arrangements and configurations (e.g., drawers opening or closing) based on their mobilities. Mobilities of objects arise from their movements due to human actions. However, they learn mobilities, again from 3D scene exemplars, by analyzing only object-object relations between reoccurring objects.

Fisher et al. [2015] propose an *activity-centric* approach to functional scene modeling, which generates 3D scenes that allow the same human activities as real environments captured through noisy and incomplete 3D scans. Given an input scan, affordance analysis [Savva et al. 2014] is first performed to detect potential activity regions and activity types. Then objects relevant to the activities are retrieved and fitted to the scan over the activity regions under human-object interaction priors learned from 3D scene databases. While their work focuses on modeling functionally similar scenes conditioned on a coarse geometric input scan, the input scan is not a must. The synthesis problem we address in this paper has a different input and different goal while learning priors from different data sources. Our action model is learned from annotated photos with only spatial constraints, while their activity model is learned from 3D scene exemplars with semantic annotations. More importantly, their synthesis is designed to serve functional scene modeling where object placements are mainly constrained by human activities inferred from a given scene; see Figure 4. In contrast, our action model evolves a scene with object placements conditioned on both human-object relations and object co-occurrences.

Most recently, Savva et al. [2016] synthesize *interaction snapshots* by sampling prototypical interaction graphs learned from real-world observations of human-object interactions captured with commodity RGB-D sensors. In contrast, we learn atomic actions from annotated photos for progressive scene synthesis. If we were to place a rigged human character in a single scene synthesized in our work, the result would be an interaction snapshot. However, our goal is not to sample a single scene instance, but to produce a continuously evolving sequence of snapshots. Furthermore, we

aim to produce realistic and messy scenes populated with many objects, while [Savva et al. 2016] focuses on accurately depicting the interaction between a human and few key objects.

COCO+action database. A highly related recent development is the COCO-a database established by [Ronchi and Perona 2015]. COCO-a enriches the Microsoft COCO database with comprehensive annotations of visual actions, designed to facilitate action discrimination and scene understanding. However, more than half of the annotated actions in COCO-a happen between people, e.g., talking and playing games, and occur during sports play or outdoors. As well, human representations remain as whole segments without pose embedding or joint labeling, which are necessary for action-driven object placement. For these reasons, we produced our own action-oriented annotations over photos from COCO and elsewhere that are designed to serve action-driven 3D scene evolution.

3 Overview

We first introduce the notations of our action model and action graph, with which we present an overview of our two-stage learning and synthesis framework; see Figure 2 for an illustration.

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

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

System overview. As shown in Figure 2, our system consists of two stages: an offline action learning stage and an online scene synthesis stage. In the learning stage, we construct an action graph from action nodes learned from the Microsoft COCO database. First, we retrieve a set of instance images from the COCO database for each action type based on keyword searching, and infer 3D human pose and object layout information from each image (Section 4.1). We then cluster the instance images according to their associated actions – $(\mathcal{T}, \mathcal{K}, \mathcal{H})$. After that, we construct the action model for each cluster by analyzing the occurrence and spatial layouts of objects (Section 4.2). Lastly, we construct an action graph over the action nodes and compute edge transitional probability based on the *correlation* between action nodes (Section 4.3).

After the action graph is constructed, we use it to drive the evolution of a scene by applying a sequence of actions sampled from the graph – this is the online scene synthesis stage. Given an initial scene, we first adapt the action graph by disabling action nodes that cannot be applied to the scene (Section 5.1). Then we generate an action sequence by sampling the adapted action graph. Note that the action sequences are not learned in the learning stage; they are instances of Markov chains sampled from the action graph. The realization of an action involves placement of 3D human pose and synthesis (insertion and relocation) of corresponding objects (Section 5.2). In the end, we obtain a sequence of evolved scenes after applying a series of actions to a target scene.

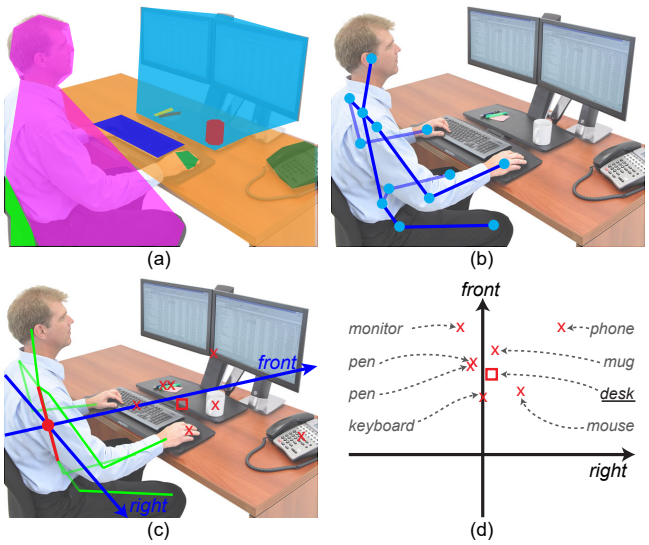

Figure 5: *(a) A typical image from COCO with block annotation for each object. (b) The annotated 2D skeleton for 3D human pose recovery. (c) The recovered 3D pose (green) is projected onto the image, as well as its right and frontal directions (blue). The center of torso (red bar) is defined as body center. Placing the right and frontal vectors there forms a Cartesian coordinate system. Each object is reduced to a 2D point (red square for the key object and red crosses for others) located at its polygonal center. (d) All object locations are mapped into a standard Cartesian system and their polar coordinates are recorded for learning their spatial layout.*

4 Data-driven model construction

In this section, we describe the procedure for data-driven action learning and action graph construction. Given a set of COCO images with manually labeled human joints, our method automatically infer various action models and construct an action graph over all action nodes. As far as we know, this represents the first attempt at using 2D images to construct action models for 3D objects and scenes. Without loss of generality, we first introduce the procedure for learning actions performed by a single person (Sections 4.1-4.3). We then describe how to extend the procedure to learn group actions that involve multiple persons (Section 4.4).

4.1 Preparing action instances

The first step of our learning procedure is to collect a large number of action instances (exemplars), each of which describes the human pose, the key object, and the object-object and object-human relationships involved in the action. To this end, we take as input the Microsoft COCO database, which consists of a large set of pre-segmented, annotated photos providing the exact labels we need for extracting action instances: human-object segmentation, object category labeling, and five captions per photo that linguistically describes the scene; see Figure 5(a). Starting from the COCO database, our method first finds a set of instance images for each action type. Then, the 3D human pose is recovered for each instance with the help of manually labeled joints. After that, we infer the

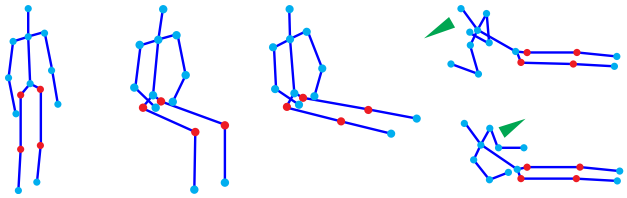

Figure 6: *Five representative 3D human poses used for our action models. From left: standing, sitting, sitting with straight legs, lying with the face down, and lying with the face up. Angles at red skeletal joints are used for matching the recovered skeletons.*

key object and 3D object layout in each instance with the help of recovered 3D human pose.

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

Given an action type, we retrieve relevant instance photos from the COCO database using their associated photo captions. To maximize recall, we collect synonyms and different tenses of the action type word and use them for keyword-based searching. For example, for action *use laptop*, the set of keywords we used include *use laptop, using laptop, work laptop, working laptop, operate laptop, operating laptop, utilize laptop, utilizing laptop*. Some of the returned photos may contain multiple irrelevant persons in the background, or too few object categories to describe a meaningful action instance. We manually filter out these photos by examining the object category labels in the photo. We further remove photos that do not contain a visible human pose. This gives us a clean set of photo instances, of size 60 - 150, for each action type.

3D human pose recovery. The COCO database only provides a 2D image region of the human body (Figure 5(a)), while the human pose definition in our action model is 3D. Hence we need to recover the most plausible 3D human pose from the 2D image. To this end, we first manually annotate the human skeleton joints in each action instance image. Then we apply the method in [Zhou et al. 2015] to find a 3D pose configuration $\mathcal{H}^p$ whose projection on the image plane matches the 2D joint annotations. To obtain a stable 3D pose estimation, we also need to manually provide the plausible locations of as many missing joints as possible, as in the case of partial occlusion; see Figure 5(b). The output is a 3D human skeleton and a weak camera projection matrix of the input image.

The resulting 3D human skeleton defines a 3D local frame in the scene, with origin at the center of the torso skeleton, the right direction determined by the vector between two shoulder joints, and the up direction defined by the torso. The frontal direction can be computed by the cross product of the up and right directions; see Figure 5(c). We use this coordinate frame for inferring and encoding all spatial layout information in the next step; see Figure 5(d).

However, the recovered 3D skeletons may have large variance and may be partial due to occlusions and thus cannot be directly used as the representative human pose for our action model. We thus follow the idea from [Jiang et al. 2012] and introduce five representative human poses (see Figure 6) for our action model. For each action instance, we find one representative pose $\mathcal{H}^*$ that best matches the recovered partial skeleton $\mathcal{H}^p$ by minimizing

$$D(\mathcal{H}^p, \mathcal{H}^*) = \sum_i \omega_i \|\theta_i(\mathcal{H}^p) - \theta_i(\mathcal{H}^*)\|, \quad (1)$$

where $\theta_i \in \{\theta_{\text{hip}}^{\text{left}}, \theta_{\text{hip}}^{\text{right}}, \theta_{\text{knee}}^{\text{left}}, \theta_{\text{knee}}^{\text{right}}\}$ are angles at the hip (between torso and thighs) and knee joints (marked in red in Figure 6), and $\omega_i = 1$ if $\mathcal{H}^p$ contains the corresponding joint, with $\omega_i = 0$ otherwise. We assign the resulting representative 3D human pose to $\mathcal{H}$ of the current action instance. This representative pose $\mathcal{H}$ is only used to identify the action of the current instance, while the original recovered pose $\mathcal{H}^p$ is used to infer object layout.

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

Object layout. Given an action instance image with recovered 3D human pose and labeled object segmentations, we learn the object-human and object-object relationships in this step. Without a full reconstruction of the 3D scene, we encode an object's relative position to the human in a 2D polar coordinate system represented by the projected local human frame. Specifically, we project the local human frame onto the instance image and assume that the projected torso skeleton has unit length. The right axis is defined as the polar axis, as shown in Figure 5(c).

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

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

4.2 Generating action nodes

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

Occurrence analysis. An action node usually involves many objects, which however might vary in their dependencies on the action

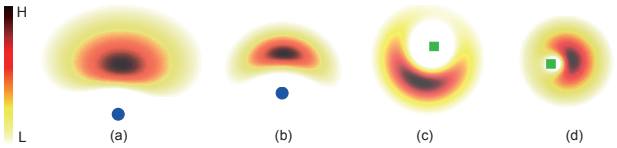

Figure 7: *The learned spatial distributions of a few constituent objects in the action "use computer on desk while sitting": the object-human distribution for monitor (a) and keyboard (b) with the human marked as blue dot, and the keyboard-monitor (c) and mouse-keyboard (d) distributions with the reference objects (the second object in each pair) marked as green squares.*

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

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

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

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

$$\mathcal{P}(o, \mathcal{H}; \Theta) = \mathcal{P}(r, \mathcal{H}; \Theta_r) * \mathcal{P}(\phi, \mathcal{H}; \Theta_\phi). \tag{2}$$

We apply MLE to estimate the parameters of the log-normal distribution $\mathcal{P}(r, \mathcal{H}; \Theta_r)$, which has closed form solution. The number of von Mises distributions in the mixture $\mathcal{P}(\phi, \mathcal{H}; \Theta_\phi)$ is a latent variable; we test values ranging from 1 to 4 and choose the one that maximizes the Akaike information criterion [Akaike 1973]. We then fix the number of von Mises distributions and utilize the numerical solution provided by [Fisher 1993] for estimating the parameters in the model. Figure 7(a-b) illustrate the learned distributions of monitor and keyboard with respect to human pose in the action node of "use computer on desk while sitting".

Finally, we formulate the object-object relationship $\mathcal{P}(o, o')$ in the same way as in Equation 2, with a similar learning procedure. The interdependencies between objects should not be treated equally; we would like to extract "reliable" object-object relationships that occur more frequently in the input instances. We compute the co-occurrence frequency $f(o, o')$ of two objects $o, o' \in O$, and object pairs with $f(o, o') < 0.5$ will be rejected from establishing the object-object relationships. Figure 7(c-d) illustrate the learned distributions for two pairs of frequent relationships in the action node of "use computer on desk while sitting".

4.3 Creating action graph

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

We construct an action graph for modeling the transitional probability between actions. The action graph $G = (V, E)$ is a weighted directed graph, where V is a set of action nodes, and E are edges connecting all pairs of action nodes, including action nodes to themselves. An edge $e_{i\to j} \in E$ directs from node a_i to a_j and carries the transition probability $w_{i\to j}$, i.e., how likely action a_j is to occur after action a_i. The transition probability is defined by

$$w_{i\to j} = \frac{\hat{s}_{ij}}{\sum_j \hat{s}_{ij}}, \tag{3}$$

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

$$\hat{s}_{ij} = \begin{cases} 1, & \text{if } i = j; \\ \max\{s_{ij}, 0.5\}, & \text{otherwise.} \end{cases} \tag{4}$$

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

$$s_{ij} = \delta(\mathcal{K}_i - \mathcal{K}_j) + \frac{\sum_o \min\{\mathcal{C}_i(o), \mathcal{C}_j(o)\}}{\sum_o \max\{\mathcal{C}_i(o), \mathcal{C}_j(o)\}}. \tag{5}$$

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

With this formulation, the two nodes with no object category intersection have the lowest transitional probability, whereas those with many shared object categories have higher transitional probability. The resulting graph is a directed state diagram, and the bi-directional transition probabilities are asymmetric (due to per node normalization in Equation 3). Sampling over the graph produces a Markov chain of human actions. The Markov assumption ("memoryless" transition probabilities) generates a new action based on the previous action instead of long-time causality relations – this simplifies the sampling process.

4.4 Group actions

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

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

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

$$\begin{cases} d_o^{\max}/d_h < 2/3, \\ d_o^{\max}/d_o^{\min} < 2. \end{cases} \tag{6}$$

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

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

$$P(d_h^{i,j}) = \begin{cases} \frac{1}{b-a}, & a \leq d_h^{i,j} \leq b, \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

Given the configuration of multiple human poses, a shared object o must locate in the region defined by Equation 6, and its distribution over that region follows the following uniform potential:

$$\mathcal{P}(o, \{\mathcal{H}, n\}) \propto \frac{1}{d_o^{\max}}. \tag{8}$$

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

5 Action-driven scene synthesis

Given the action graph learned from the annotated COCO images, we are now ready to synthesize human actions to drive the evolution of a scene. Our solution follows recent work [Fisher et al. 2012; Fisher et al. 2015] and models human actions including group activities to synthesize lively messy 3D indoor scenes.

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

5.1 Graph adaptation

Our action graph is constructed from all types of actions learned from the COCO database and covers actions for various types of scenes: bedroom, kitchen, office room, etc. Applying it to a specific scene requires a preprocess of graph adaptation. That is, given an input scene with key objects, we need to prune certain action nodes if their key objects are missing in the scene. For example, if silverware is not present in the scene, the action node of dining could be on – the realization of this action will cause the insertion of silverware; but if the dining table (key object) is missing, dining should not be allowed to happen, because our action model relies solely on the key object to place the human pose and therefore all other constituent objects. After the graph adaptation, we also remove dangling edges connected to pruned nodes and update edge weights in the adapted graph according to Equation 3.

5.2 Action realization

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

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

Placing active objects (O). Given the human pose and key object of the action, we need to figure out the set of constituent objects O to be inserted, as well as their locations and orientations in the scene. First, we generate the set of constituent objects according to the learned occurrence model $\mathcal{C}$. For the active objects that have been in the scene, we have two options – either reuse it or insert a new one. We predefine an upper bound for the occurrence times of each object category in the whole scene. For example, a scene can

contain at most one monitor, one keyboard, but two coffee mugs, ten books. New objects are inserted into the scene before their time of occurrence reaches the upper bound; otherwise, we only allow reuse of objects for realizing new actions.

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

$$f^1(o) = \mathcal{L}(o) * \mathcal{S}(o) * \mathcal{P}(o). \quad (9)$$

The *collision penalty* term $\mathcal{L}(\cdot)$ enforces no physical collision between objects; $\mathcal{L}(o) = 0$ if o collides with any other objects in the scene, and is of value 1 otherwise. The *overhang penalty* term $\mathcal{S}(\cdot)$, similar to that in [Fisher et al. 2012], prevents o from hanging off the edge of a supporting surface. We project the bounding box of o onto the supporting surface and compute the intersection area $A(o)$ between the projection and the supporting region. Precisely, $\mathcal{S}(o) = 1$ if $A(o) \geq 0.5$, otherwise $\mathcal{S}(o) = 0$. That says the placement of o is not plausible if more than half of its volume hangs off the supporting surface. The last term $\mathcal{P}(o)$ combines the spatial layouts of o w.r.t. both human pose and other objects:

$$\mathcal{P}(o) = \mathcal{P}(o, \mathcal{H}) + \sum_{o'} f(o, o') * \mathcal{P}(o, o'), \quad (10)$$

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

Note that the positioning of o so far is still in the human centric system, which might generate a 3D location floating in the air. To make the synthesized scene physically plausible, we move o vertically until it reaches the closest supporting surface of either a key object or the floor. A sampling strategy is utilized in the placement procedure to ensure a balance between the diversity and the plausibility of object configuration. In our implement, we uniformly sample $k = 2,000$ locations and select the one that maximizes the score defined in Equation 9 as the final position for an object. To determine the orientation of an object o, we manually specify a *facing direction* for o w.r.t. human. Each placed object is rotated horizontally so that its facing direction pointing to the human center.

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

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

$$f^2(\tilde{o}) = \mathcal{L}(\tilde{o}) * \mathcal{S}(\tilde{o}) * \mathcal{W}(\tilde{o}) * \exp(-\Delta(\tilde{o})). \quad (11)$$

The first two terms, $\mathcal{L}(\cdot)$ and $\mathcal{S}(\cdot)$, are the same as that in Equation 9. The third term $\mathcal{W}(\cdot) = 1$ if $\tilde{o}$ falls outside of the working zone; otherwise, $\mathcal{W}(\cdot) = 0$. The $\Delta(\tilde{o})$ in the last term measures the distance of shift of $\tilde{o}$ from its original position.

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

5.3 Realizing group action

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

We place human poses into the scene one at a time. The first human pose is randomly placed around the key object. Suppose k human poses have been placed, the location of the $(k+1)$-th human pose must also be around the key object, as well as obey the distance constraint in Equation 7 to all other k human poses. This procedure stops until the group size reaches four or no further human pose can be placed, which leads to groups with size of 2 and 3.

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

$$f^3(o) = \mathcal{L}(o) * \mathcal{S}(o) * \mathcal{R}(o) * \mathcal{P}(o, \{\mathcal{H}, n\}), \quad (12)$$

where $\mathcal{L}(\cdot)$ is the collision penalty, $\mathcal{S}(\cdot)$ is the overhang penalty, $\mathcal{R}(o)$ is a binary indicator of whether o falls in the sharing region specified by Equation 6, and $\mathcal{P}(o, \{\mathcal{H}, n\})$ is the spatial distribution potential of o; see Equation 8.

6 Results and evaluation

In this section, we present results of action-driven scene evolution and compare the results, through user studies, to those created by an artist, and those by the most closely related methods for human- or activity-centric scene modeling [Jiang et al. 2012; Fisher et al. 2012; Fisher et al. 2015]. We also demonstrate the capability of our method to synthesize messy scenes at larger scales.

Action data and graph. Action learning is conducted exclusively over 1,216 annotated photographs, 936 of which are from the Microsoft COCO database [Lin et al. 2014]. The remaining photos were collected on-line to enrich or complement action data extracted from COCO. The photos were all annotated with embedded human poses as well as action and object labels. It takes less than 30 seconds to mark all the joints for one human in a photo. An unlabeled on-line photo typically takes less than three minutes to annotate using LabelMe for a scene with 5-10 objects. For our experiments, the learned action graph is a complete graph (with self-loops) composed of 20 nodes and covering 8 types of actions. After annotating all the photos, action learning and graph construction take about 8 minutes in total to complete.

3D scene evolution. Figure 14 (also see Figures 1 and 8) shows a gallery of 3D scene evolution results that highlights the various

features offered by our method. Timing-wise, sampling an action from our action graph takes on average 0.1 second and object placements take on average 1 second.

In each row of Figure 14, the action sequence is probabilistically sampled from the learned action graph and applied to the initial scene in order. Applied actions are indicated in the figure, with zoom-in views to better visualize the insertion and relocation of objects. The two final rows in the figure show how actions involving working and dining learned from bedrooms and living rooms can be transferred to never-seen scene categories such as dining halls and computer labs to create quite a mess. It should be reiterated that the mess is purely the result of action-driven scene evolution, starting from a clean initial scene. The work of Fisher et al. [2015] was able to show the modeling of messy 3D scenes when depth scans of cluttered scenes are given as input.

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

Group actions. Figure 8 contrasts scenes synthesized via group actions to those generated by applying a "baseline" method mainly involving single-person actions. In both cases, we first sample from a distribution of persons based on the same initial scene. For group actions, we sample and place a set of shared objects learned from photos of group actions. Then for each person in the group, we apply a single-person action model to place additional objects into the scene. For the baseline method, the shared objects are randomly assigned to persons in the group, then we apply a series of single-person actions to place all the objects including the shared ones.

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

To assess the plausibility of our learned group action model, we prepared 6 pairs of scenes like the ones shown in Figure 8, using our group action model and the baseline method described above. We then asked 24 human participants to select, one out of each pair, the scene they believe to "appear more like a scene during or after a group of people having a meal together". The participants selected the scenes generated by our group action model 75% of the times.

6.1 Plausibility tests against artist

A key evaluation for our method is whether the results from action learning and scene generation are plausible. Similar to previous works [Fisher et al. 2015], we leave such judgments to human subjects. We ask users to give a score from 1 (least plausible) to 5 (most plausible) to a generated scene based on two criteria: *plausibility*, the scene is a plausible result after a given action is performed on an initial scene; and *naturalness*, the scene looks natural.

Object placement test (OPA). To compare our results to human-generated scenes, we hired a professional artist to manually create scenes based on given actions. When asking users to rate scene plausibility, we found that providing them with more contexts with which to make judgements is more likely to gather more reliable feedback. Therefore, for each initial scene and a given action, instead of providing only a pair of scenes to rate, we provide five scenes: three synthesized by our method with random initialization, and two scenes created by the artist. In each case, the same

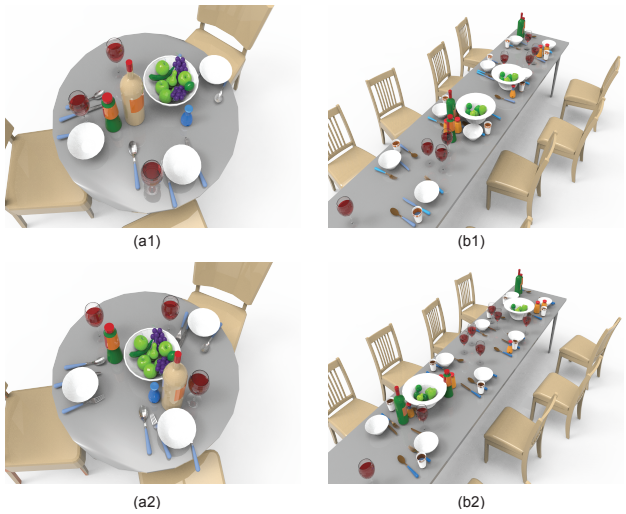

Figure 8: *Two scenes synthesized from a group action "group dining" (bottom) vs. those by a baseline method (see text) mainly involving single-human actions for dining (top). Both syntheses start with the same initial scene and place the same set of objects. Notable distinctions can be observed for the placements of the large fruit bowl, which is a shared object.*

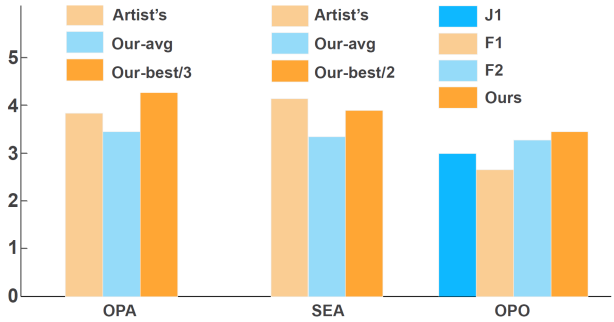

Figure 9: *Overall average user ratings for scene plausibility test, when comparing our results to scenes created by an artist and to scenes created by closely related works.*

set of objects were inserted or relocated. The five scenes, randomly ordered, make up one query for the Object Placement test against Artist or OPA, for short. Our user study consists of 20 OPAs covering 10 actions. With 28 human participants working on the OPA test, we gathered a total of 140 user ratings.

Figure 9 (left) plots the overall average user rating for the artist's results and our results from the OPA test. Per-action average ratings are plotted in Figure 10. Our results received an average rating of 3.46, which approaches closely to that of the artist (3.83). If we take the best out of our three results from each OPA query, then the (best-of-3) average rating goes up to 4.29 and is higher than artist's average rating, indicating that our method is able to produce results no worse than an average artist in the best scenario.

Scene evolution test (SEA). We scale up the OPA test to scene evolution involving multiple actions. Each Scene Evolution test against Artist, or SEA, consists of two sequences of scenes evolved using our method via probabilistic sampling of the action graph and one sequence of scenes created by the artist using the same action sequence, all with the same initial scene and same set of objects. Three evolution steps are applied for each initial scene and the SEA

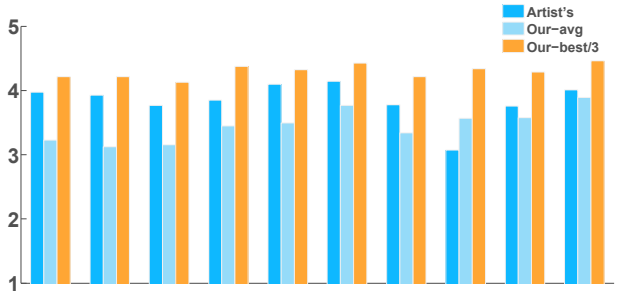

Figure 10: *Per-action OPA test results: comparison to scenes created by an artist in terms of average user ratings. Along the x-axis, the 10 actions are: "use computer on desk while sitting", "use laptop on desk while sitting", "use laptop on coffee table while sitting", "use laptop on bed while lying",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

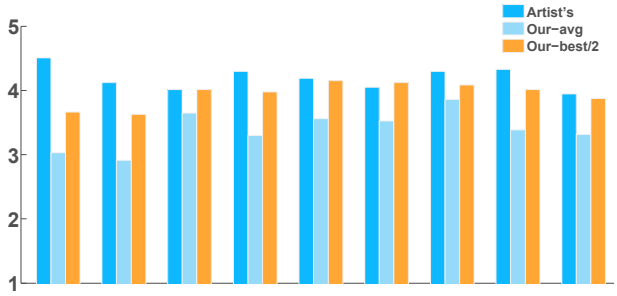

Figure 11: *Per-action-sequence SEA test results: comparison to scene evolutions created by an artist, over 9 initial scenes and action sequences.*

test covers 9 initial scenes. A total of 29 human participants rated for the SEA test. For each query, the participants were asked to rate the overall plausibility of each of three scene sequences. The overall average and per-action-sequence average ratings are shown in Figures 9 (middle) and 11, respectively.

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

6.2 Comparisons to closely related works

Among all the previous works on 3D scene modeling, the example-based synthesis method (F1) of Fisher et al. [2012] is the closest to our work in terms of end goal: both aim to develop an open-ended tool to synthesize 3D scenes, but take different routes to achieve the goal. Judging by modeling methodologies, the works by Jiang et al. [2012] (J1) and Fisher et al. [2015] (F2) come the closest as they both take a human- or activity-centric approach. All of these works, including ours, are data-driven. However, F1, J1, and F2 all learn from 3D scene exemplars, we learn from annotated photos.

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

Table 1: *Numerical average user ratings for the OPO test, for three actions and two initial scenes and object replacements per action. The three actions are: CK = "use computer on desk while sitting"; LS = "use laptop on coffee table while sitting"; D = "eat dinner on table while sitting".*

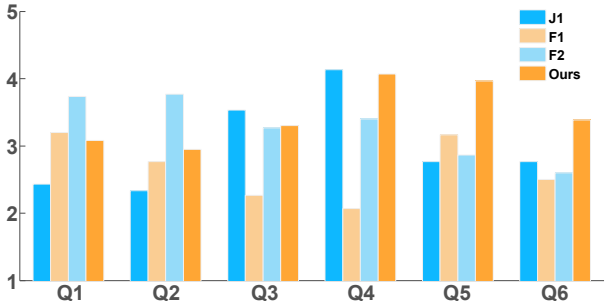

Figure 12: *OPO test results on 6 queries: comparison to three related works in terms of scene plausibility. Along the x-axis, the actions are "use computer on desk while sitting"(Q1, Q2), "use laptop on coffee table while sitting"(Q3, Q4), and "eat dinner on table while sitting"(Q5, Q6).*

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

Since the compared methods perform learning from different data sources, which would influence object category occurrence, we only compare the plausibility of object placement where the set of objects placed are the same, as in the OPA test. Also, only single-frame object placements are compared since the other three methods were not all designed for scene evolution. Given an action applied to an initial scene, the four scenes resulting from J1, F1, F2, and our method make up one query in the Object Placement test against Other methods or OPO, for short. Human participants are asked to provide plausibility scores for these queries.

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

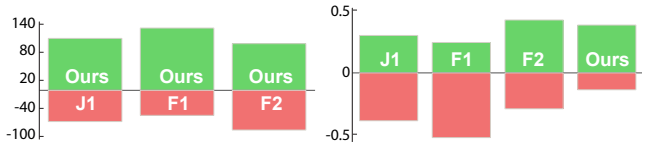

Figure 13: *Additional comparisons among the four methods in terms of scene plausibility. Left: the number of times our method was rated higher (green) or lower (red) than other methods in the OPO test, respectively. Right: the percentage of each method being rated as the best (green) or the worst (red) among all.*

put scene examples and scenes in the 3D scene database from the original paper.

For each of the three actions selected for the OPO test, two object placements are performed by all four methods. Thus, there are six OPO queries in total. The user study involved 30 human subjects to provide a total of 180 ratings. Table 1 summarizes the OPO test results, including average user ratings for each scene-action combination. The last column shows the overall averages and the standard deviations of the ratings. The numbers indicate that in the OPO test, our action-driven method outperforms all of the three methods compared. Significance tests also show that the performance gains are statistically significant. Figures 9 (right) and 12 visualize the numbers given in Table 1.

Test results in Figure 12 show that our method performed best for the action "eat dinner on table while sitting", which involves multiple small items. The likely reason is that our action model considers human-object relations as well as relations among multiple objects, e.g., the bowls and utensils, while both J1 and F2 focus on pairwise human-object relations. For the action "use computer on desk while sitting", F2 received the highest scores. This may be attributed to the *semantic* human-object relations that can be learned from their 3D scene data possessing semantic annotations, e.g., monitor is not blocked for visibility, keyboard and mouse need to be touchable. On the other hand, our action model learns the human-object distributions only with *spatial constraints* from photos.

In Figure 13, we show additional comparison results, including head-to-head ratings. These results consistently demonstrate the advantage of our method from different perspectives. Note that all the human subjects in our user studies are graduate students in the fields of computer science and engineering.

7 Discussion, limitation, and future work

Our ultimate goal is to automatically synthesize truly messy yet realistic 3D indoor scenes, a task we believe no current work has been able to come close to achieving. In reality, scenes are messed up by our daily actions, not a probabilistic scene/sub-scene mixing. In addition, a mess is hardly something that can be properly learned from limited exemplars as the messiness is expected to be highly varied and unpredictable. In our work, we propose an action-driven approach with a progressive layout scheme, which we call a scene evolution. Furthermore, we believe that the data source ought to come from images since so far, only images possess the richness and variety to allow the synthesis of truly messy scenes.

The method we have presented only represents a first attempt at executing the above ideas. Yet, the action model we develop and apply, as well as action extraction and learning from annotated photographs, offers unique features which set our approach apart from previous works on human- or activity-centric 3D scene modeling [Fisher et al. 2012; Jiang et al. 2012; Savva et al. 2014; Fisher et al. 2015; Savva et al. 2016]. Even with the various simplifying design choices in our work, comprehensive user study results positively support the action-driven approach in terms of its generative capabilities and plausibility of the generated scenes, as compared to human effort and state-of-the-art data-driven methods. That said, the action-driven approach should only *complement* and not replace other scene modeling paradigms. We also reiterate that the success of any data-driven approach relies heavily on the quantity and quality of its data. What our approach can accomplish is still limited by having only a handful of action types to work with.

Scene synthesis at two scales. An important lesson from our current pursuit is that indoor scene generation is inherently a process that ought to operate at *two* scales. At the coarse scale, one is concerned with how to layout large furniture pieces such as beds and sofas. How a human sits or lies on a sofa does not play a major role in its placement in a room. The layout problem is more suited for a rule-based approach [Merrell et al. 2011], since in reality, it is mainly guided by design guidelines and functionality considerations. The movements or arrangements of small, mobile items are naturally tied to human actions. There are also *passive* actions such as watching TV, which do not involve moving an object but may trigger an object insertion. The action-driven approach is best suited for scene synthesis at such finer granularity level.

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

Pattern-driven vs. action-driven. Arranging small objects over a relatively small workspace such as a shelf or desk *is* action-driven, but the relevant actions are difficult to capture and annotate from depth scans [Savva et al. 2014] or photographs. Since these kinds of arrangements typically exhibit predicable patterns, *pattern-driven* syntheses have been successful, where the patterns are rule- or style-based [Majerowicz et al. 2014] and can be learned from examples [Fisher et al. 2012]. On the other hand, messed-up rooms after a party or kids play would hardly share common statistical properties or reveal predictable patterns. An action-driven, progressive approach is more befitting to the modeling of such scenes.

Static vs. dynamic action data. Our action model is learned from static photos showing snapshots of human-object and object-object relations. Without capturing dynamic transitions between actions, we currently assume that the transition probability from one action to others can be estimated by the correlation between actions; see Equation 3. Such a substitution of transition probabilities by correlations between action models is a limitation which could be lifted if dynamic action data, e.g., video or RGBD motion data [Savva et al. 2014; Savva et al. 2016], can be utilized to study the transition probabilities. Then the learned model can be adapted to our action graph to drive the scene evolution.

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

tary to this, efforts can be made to acquire more meaningful photos which enrich the action data while facilitating action learning.

Future work. The set of action types we learn and apply in this paper were hand-picked from the COCO database; they are clearly limited. To fully realize the potential of action-driven scene analysis and synthesis, a lot more action data should be acquired and prepared. To extend the effort to a much larger scale, action data should ideally be mined first from large text and image sources in a future pursuit. If we are able to obtain much more action data with text annotations, then we would not be far from achieving 3D scene scene evolution from textual instructions [Savva et al. 2016]. Also, our current object placement scheme can be enhanced with more advanced geometric and even functional analysis of 3D objects [Hu et al. 2016]. A final interesting thought would be to *reverse* the scene evolution: instead of going forward in time from an input scene, reconstruct a plausible scene sequence backward in time so that the sequence would lead to the input scene.

Acknowledgements

We would like to thank all the reviewers for their comments and suggestions. We are grateful to Matthew Fisher for his help in providing data and results during our evaluation. Thanks also go to Ashutosh Saxena for off-line discussions on related works. The first author carried out the earlier phase of the research work at Microsoft Research Asia with their support and a MITACS Globalink Research Award. This work was also supported in part by NSERC (611370), NSFC (61502153,61602406), and NSF of Hunan Province of China (2016JJ3031).

References

AKAIKE, H. 1973. Information theory and an extension of the maximum likelihood principle. In *Second International Symposium on Information Theory*, vol. 1, 267–281.

CHEN, K., LAI, Y.-K., WU, Y.-X., MARTIN, R., AND HU, S.-M. 2014. Automatic semantic modeling of indoor scenes from low-quality RGB-D data using contextual information. *ACM Trans. on Graph. 33*, 6, 208:1–12.

FISHER, M., RITCHIE, D., SAVVA, M., FUNKHOUSER, T., AND HANRAHAN, P. 2012. Example-based synthesis of 3D object arrangements. *ACM Trans. on Graph. 31*, 6, 135:1–11.

FISHER, M., LI, Y., SAVVA, M., HANRAHAN, P., AND NIESSNER, M. 2015. Activity-centric scene synthesis for functional 3D scene modeling. *ACM Trans. on Graph. 34*, 6, 212:1–10.

FISHER, N. I. 1993. *Statistical Analysis of Circular Data*. Cambridge University Press, Cambridge.

FOUHEY, D. F., DELAITRE, V., GUPTA, A., EFROS, A. A., LAPTEV, I., AND SIVIC, J. 2012. People watching: Human actions as a cue for single-view geometry. In *ECCV*, 732–745.

GERMER, T., AND SCHWARZ, M. 2009. Procedural arrangement of furniture for real-time walkthroughs. *Computer Graphics Forum 28*, 8, 2068–2078.

HU, R., VAN KAICK, O., WU, B., HUANG, H., SHAMIR, A., AND ZHANG, H. 2016. Learning how objects function via co-analysis of interactions. *ACM Trans. on Graph. 35*, 4, 47:1–12.

JIANG, Y., LIM, M., AND SAXENA, A. 2012. Learning object arrangements in 3d scenes using human context. In *Proc. Int. Conf. on Machine Learning (ICML)*.

JIANG, Y., KOPPULA, H., AND SAXENA, A. 2013. Hallucinated humans as the hidden context for labeling 3d scenes. In *IEEE CVPR*, 2993–3000.

KIM, Y. M., MITRA, N. J., YAN, D.-M., AND GUIBAS, L. 2012. Acquiring 3D indoor environments with variability and repetition. *ACM Trans. on Graph. 31*, 6, 138:1–138:11.

LIN, T.-Y., MAIRE, M., BELONGIE, S., HAYS, J., PERONA, P., RAMANAN, D., DOLLR, P., AND ZITNICK, C. L. 2014. Microsoft COCO: Common objects in context. In *ECCV*, 740–755.

LIU, Z., ZHANG, Y., WU, W., LIU, K., AND SUN, Z. 2015. Model-driven indoor scenes modeling from a single image. In *Proc. of Graphics Interface*, 25–32.

MAJEROWICZ, L., SHAMIR, A., SHEFFER, A., AND HOOS, H. H. 2014. Filling your shelves: Synthesizing diverse style-preserving artifact arrangements. *IEEE Trans. Visualization & Computer Graphics 20*, 11, 1507–1518.

MERRELL, P., SCHKUFZA, E., LI, Z., AGRAWALA, M., AND KOLTUN, V. 2011. Interactive furniture layout using interior design guidelines. *ACM Trans. on Graph. 30*, 4, 87:1–10.

RONCHI, M. R., AND PERONA, P. 2015. Describing common human visual actions in images. In *Proc. of the British Machine Vision Conference (BMVC)*, 52:1–12.

SADEGHIPOUR, Z., LIAO, Z., TAN, P., AND ZHANG, H. 2016. Learning 3D scene synthesis from annotated RGB-D images. *Computer Graphics Forum (SGP) 35*, 5.

SAVVA, M., CHANG, A. X., HANRAHAN, P., FISHER, M., AND NIESSNER, M. 2014. SceneGrok: Inferring action maps in 3D environments. *ACM Trans. on Graph. 33*, 6, 212:1–10.

SAVVA, M., CHANG, A. X., HANRAHAN, P., FISHER, M., AND NIESSNER, M. 2016. PiGraphs: Learning interaction snapshots from observations. *ACM Trans. on Graph. 35*, 4.

SHAO, T., XU, W., ZHOU, K., WANG, J., LI, D., AND GUO, B. 2012. An interactive approach to semantic modeling of indoor scenes with an RGBD camera. *ACM Trans. on Graph. 31*, 6, 136:1–11.

SHARF, A., HUANG, H., LIANG, C., ZHANG, J., CHEN, B., AND GONG, M. 2013. Mobility-trees for indoor scenes manipulation. *Computer Graphics Forum 32*, 1–13.

XU, K., CHEN, K., FU, H., SUN, W.-L., AND HU, S.-M. 2013. Sketch2Scene: Sketch-based co-retrieval and co-placement of 3D models. *ACM Trans. on Graph. 32*, 4, 123:1–10.

YU, L.-F., YEUNG, S. K., TANG, C.-K., TERZOPOULOS, D., CHAN, T. F., AND OSHER, S. 2011. Make it home: automatic optimization of furniture arrangement. *ACM Trans. on Graph. 30*, 4, 86:1–12.

ZHOU, X., LEONARDOS, S., HU, X., AND DANIILIDIS, K. 2015. 3D shape estimation from 2D landmarks: A convex relaxation approach. In *IEEE CVPR*, 4447–4455.

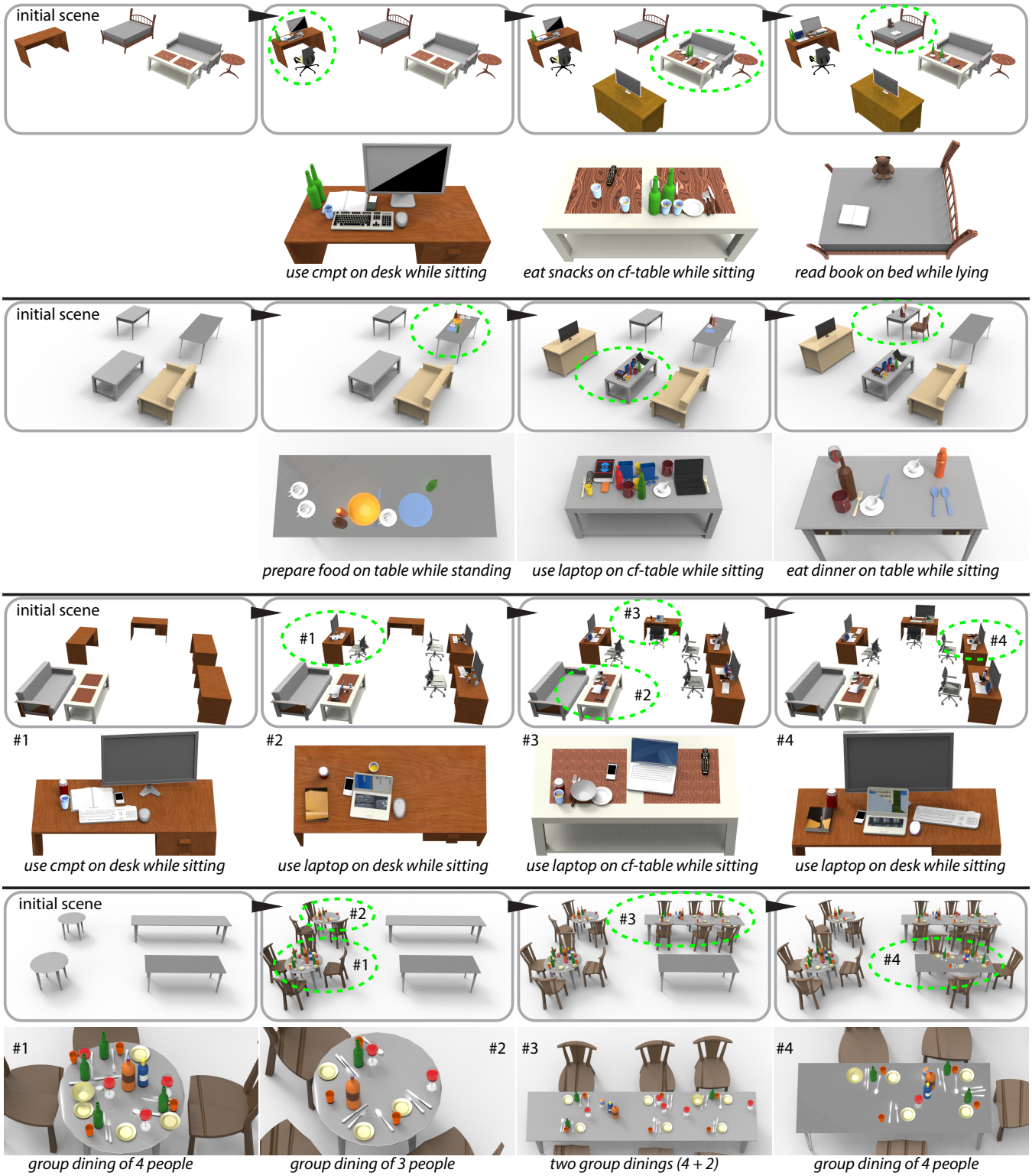

Figure 14: *A gallery of our action-driven 3D scene evolution results. The figures and annotations should be self-explanatory. Sub-scenes enclosed in green ellipses are zoomed in below the scene sequence for better visualization. Last two rows show how actions involving laptop or computer use and dining, which were learned from small-scale bedrooms and living rooms from Microsoft COCO photos, can be transferred to never-seen scene categories such as a computer lab and a dining hall to create quite a mess.*