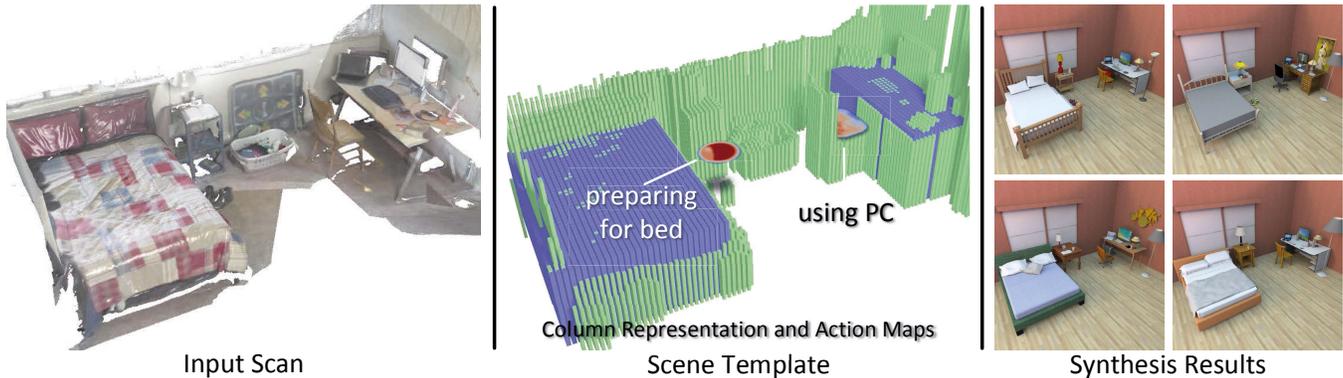


# Activity-centric Scene Synthesis for Functional 3D Scene Modeling

Matthew Fisher   Manolis Savva   Yangyan Li   Pat Hanrahan   Matthias Nießner  
Stanford University



**Figure 1:** Given a 3D scan of an environment (left), we produce a scene template that encodes where activities can occur and contains a coarse geometric representation describing the general layout of objects (middle). By instantiating this template, we synthesize many plausible scenes containing appropriate objects in arrangements that allow the encoded activities to be performed (right). The key idea of our method is to learn the distribution and arrangements of objects involved in each activity from an annotated 3D scene and model corpus.

## Abstract

We present a novel method to generate 3D scenes that allow the same activities as real environments captured through noisy and incomplete 3D scans. As robust object detection and instance retrieval from low-quality depth data is challenging, our algorithm aims to model semantically-correct rather than geometrically-accurate object arrangements. Our core contribution is a new scene synthesis technique which, conditioned on a coarse geometric scene representation, models functionally similar scenes using prior knowledge learned from a scene database. The key insight underlying our scene synthesis approach is that many real-world environments are structured to facilitate specific human activities, such as sleeping or eating. We represent scene functionalities through virtual agents that associate object arrangements with the activities for which they are typically used. When modeling a scene, we first identify the activities supported by a scanned environment. We then determine semantically-plausible arrangements of virtual objects – retrieved from a shape database – constrained by the observed scene geometry. For a given 3D scan, our algorithm produces a variety of synthesized scenes which support the activities of the captured real environments. In a perceptual evaluation study, we demonstrate that our results are judged to be visually appealing and functionally comparable to manually designed scenes.

**CR Categories:** I.2.10 [Artificial Intelligence]: Vision and Scene Understanding—3D/stereo scene analysis

**Keywords:** scene synthesis, activities, scene understanding

## 1 Introduction

Understanding the identity, arrangement, and functionality of objects in real-world environments is a difficult but essential task for many applications. This problem is especially hard since a real environment can only be captured by observations from an incomplete set of viewpoints, and each observation is limited by occlusions and sensor quality. Nevertheless, people are able to robustly understand their surroundings by mapping them to environments they have previously experienced. Even when we observe only a small part of an environment, we are still able to effectively interact with since we rely upon our prior understanding of what objects to expect and how they are arranged. One significant component of the human understanding process is to imagine someone performing specific activities and identify nearby objects in the context of this activity [Tversky and Hard 2009]. This approach is successful because many environments we encounter have been designed for human-centric needs; once we successfully identify that a certain type of activity can be performed, we can identify and interact with nearby objects by aligning them relative to the activity.

In contrast to people, computers struggle to understand objects and their arrangement in 3D environments. One significant problem is identifying object instances when only part of the object is visible or when an exact geometric match does not exist in the training database. A promising research direction to overcome these limitations is to augment low-level geometric features with higher-level contextual information [Galleguillos and Belongie 2010]. Unfortunately, modeling semantic context in real settings is challenging because it is difficult to know which contextual relationships between objects are important. We would like to enable functional parsing of environments through an activity-centric perspective. Our insight is to use human activities as a hidden variable to represent contextual relationships between objects. This admits a simple and sparse scene representation. Our approach is motivated by the fact that scenes are designed with specific functions in mind. Bedrooms facilitate sleep, while office environments enable easy access to desks and computers. Although we evaluate our method on a 3D scene generation task, we believe an understanding of activities is fundamental to many applications beyond scene reconstruction, including image understanding and robot-human interaction.

In this paper, we anchor an observed scene in a set of activities, and use associated activity regions as focal points to guide scene modeling. We take as input an RGB-D scan of an environment and generate an arrangement of 3D models that allows the same types of interactions as the scan (see Figure 1). These activity regions guide scene modeling towards plausible results, especially in the presence of significant occlusion or noisy data.

In summary, we present a novel activity-centric method for synthesizing scenes that respect the functional and geometric properties of an input RGB-D scan:

- We propose a *scene template* as an intermediate representation encoding functional and geometric properties of a scene.
- We utilize existing methods for action recognition in 3D environments to build a scene template from a 3D scan.
- We estimate the functional plausibility of an arrangement of objects using an activity model learned from an annotated 3D scene and model database.
- We present a method that uses this activity model to synthesize plausible scenes respecting the geometric and functional properties specified in the scene template.
- We evaluate our method against scenes made by people and prior scene synthesis methods to demonstrate that our approach produces plausible and functionally similar scenes, even from noisy and incomplete range data.

## 2 Related Work

### 2.1 Action Understanding

As our goal is to reconstruct scenes based on a human-centric view, it is important to understand how actions influence scene structure. The human-centric view of environments has a long history, with a strong connection to cognitive psychology [Gibson 1977; Tversky and Hard 2009].

In computer graphics and vision, understanding actions has become a fundamental basis for many research directions. Grabner et al. [2011] evaluate “chair-ness” by probing environments with a mesh in a sitting pose. Geometry supporting the posed mesh is assumed to be chair-like. Fitting skeletal poses, instead of posed meshes, to 3D geometry has been explored by Kim et al. [2014]. This idea has gained relevance with the feasibility of tracking skeletal poses with commodity RGB-D sensors [Shotton et al. 2013]. Such pose data enabled research on the classification of objects and actions in the context of human activities [Koppula et al. 2013; Wei et al. 2013a; Wei et al. 2013b]. The key idea of these approaches is the utilization of temporal features obtained by observing an environment over a specific period of time.

A particularly relevant line of work hypothesizes the arrangement of objects based on the plausibility of “hallucinated” skeletal poses [Jiang et al. 2012; Jiang and Saxena 2013]. Recently, Savva et al. [2014] generated probability distributions corresponding to likelihoods of activities in real and virtual environments. We build on these techniques, as we predict likely activities for environments prior to reconstruction. While our method is orthogonal to the specific activity prediction approach, our implementation uses SceneGrok [Savva et al. 2014] as a prior for our functional reconstruction.

### 2.2 Model Retrieval and Semantic 3D Reconstruction

One way to semantically reconstruct a scanned 3D scene is by retrieving models from a shape database. Once objects are retrieved, higher-level semantic information is drawn from the database and utilized to interpret the 3D scene. Unfortunately, object recognition

and retrieval based on low-level geometric features – e.g., [Johnson and Hebert 1999; Lowe 2004; Drost and Ilic 2012] – is challenging in the context of noisy and unsegmented range data. Recent methods combine local and global feature descriptors to jointly segment, retrieve, and align objects; e.g., [Matuschek et al. 2014; Li et al. 2015]. Another direction is to recognize models through machine learning, where classifiers are used to detect objects based on geometric features. For instance, Kim et al. [2012] learn a deformable part model from multiple range images, while Nan et al. [2012] use a random decision forest to classify objects from over-segmented high-quality range data. Shao et al. [2014], abstract environments as collections of cuboids, and analyze the scene structure via cuboid arrangements.

To improve reconstructions, prior work has augmented object instance retrieval with spatial context information. This is particularly relevant if the sensor data is incomplete and noisy. Shao et al. [2012] semi-automatically obtain semantic regions to drive their object retrieval algorithm. Chen et al. [2014] learn object relationships from a scene database (cf. Section 2.3) and use Markov random fields to perform model retrieval with respect to the learned semantic context.

As our main goal is to model semantically plausible scenes, our method is less constrained by the scanned input geometry. Instead of focusing on object instance recognition, we capture the arrangement and relationship of objects while relaxing the requirement for exact instance matching.

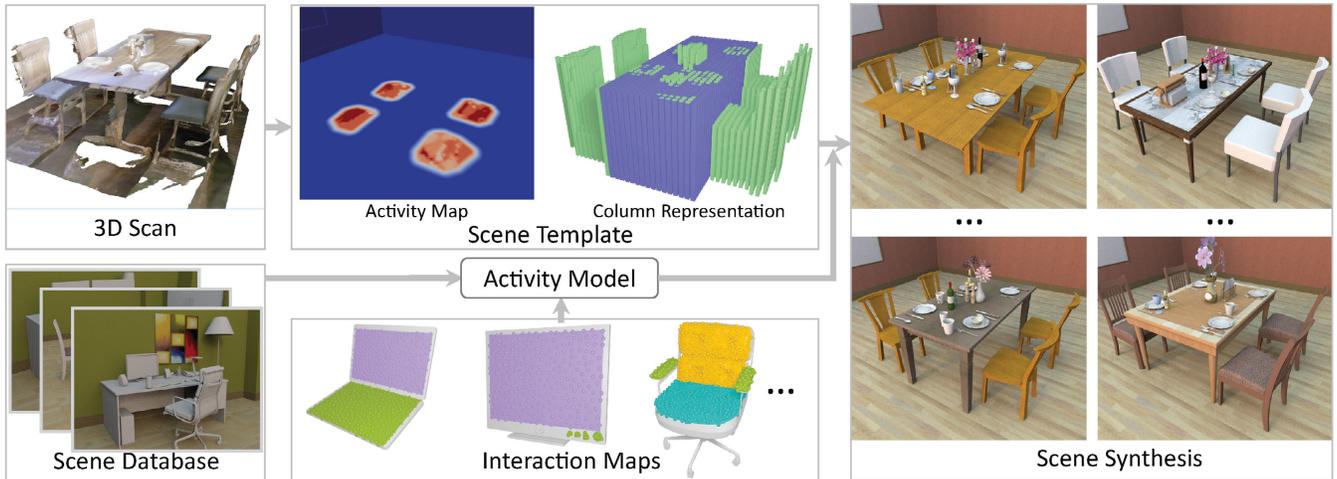
### 2.3 Contextual Scene Understanding

Contextual scene understanding has become important for automated content creation, as well as for the organization and maintenance of scene databases. A common way to model the context between objects in a scene is a relationship graph. The core idea is to encode semantic scene structure in a graph representation in order to enable data-mining operations, such as comparisons and database queries [Fisher et al. 2011]. Xu et al. [2014] cluster scene relationship graphs into contextual focal groups, allowing them to simplify graph connectivity and organize large scene collections much more efficiently. Liu et al. [2014] share the same goal; they use a probabilistic grammar to obtain consistent graph hierarchies over large scene collections. Our method extends these ideas; specifically, we express relationships in the presence of activities, and model object relationships indirectly via hidden agent variables. This allows us to minimize the number of relationships, as we only consider high-level semantic meaning via the agents, and avoid the need to recognize object instances to infer scene relationships.

Fisher et al. [2012] generate synthetic scenes by modeling these object relationships with a Bayesian network learned from a scene database and user-provided input examples. Sketch2Scene [Xu et al. 2013] is another scene synthesis system using simple image sketches to drive the synthesis algorithm. Though we also use a scene database for synthesis, we model our scene prior using activities and condition on real-world RGB-D scans.

## 3 Algorithm Overview

An overview of our approach is given in Figure 2. The core of our method is the use of a scene template to serve as an intermediate representation capturing both the prominent geometric properties of a scan and the activities that are likely to be performed in the environment. The scene template represents activities using a continuous activity map that can be sampled to produce a set of *proxy agents*. These are oriented and localized virtual agents who per-



**Figure 2:** Starting from a 3D scan, we learn a scene template that describes different activities that can take place in the environment and the rough geometric layout of the scene. Using an activity model trained from an annotated database of scenes and 3D objects, we can synthesize many scenes that respect the activity and geometric constraints of the scene template.

form a specific labeled action. We formalize our scene template representation in Section 4. Our goal is to arrange objects from a categorized 3D model database to produce scenes that respect the activity and geometric properties of the scene template.

Given a 3D scan, we first build a corresponding scene template for the captured environment (Section 5). We use an activity classifier trained from observations of people in real environments to compute a continuous-valued activity map over the scan for many possible activities [Savva et al. 2014]. We also capture stable geometric features of the scan, such as dominant supporting planes and the rough layout of the room. Section 6 describes how we quantitatively evaluate how similar a synthesized scene matches the geometric features in the scene template.

Our synthesis algorithm relies upon learning an activity model for each type of activity in the scene template (Section 7). This model is trained from a database of virtual scenes, each annotated with agent proxies that represent the presence of an agent performing a specific activity in the scene along with a set of objects the agent interacts with while performing that activity; an example is shown in Figure 3. The activity model also makes use of a 3D model database annotated with *interaction maps*. An interaction map for a model shows the regions on the model surface where specific interactions are required for an activity. For example, Figure 3 shows the regions on a monitor that need to be visible to an agent for the monitor to be useful while an agent is using a desktop PC. Our activity model is independent of any specific input scan and is trained as a pre-process over the entire training database.

Given a scene template and a model for each activity, we show how to synthesize scenes that respect both the geometric and activity properties specified in the template (see Section 8). We define an activity scoring function that uses the activity model to estimate how well each proxy agent in the scene template can perform the given activity. Activities might be difficult to perform because certain objects important to the activity are missing (e.g., no monitor when using a computer), or because they are not positioned or oriented well (e.g., a monitor rotated away from the agent). For synthesis, we start from a blank scene and iteratively insert new objects, biasing our selection towards objects that increase the combined geometric and activity scoring functions. We repeat this synthesis process many times, and choose final results where the geometric properties of the scan are sufficiently respected and the proxy agents

can effectively perform the target activities.



**Figure 3:** Example training scene for the activity model. A “prepare for bed” agent (tan) and a “use PC” agent (red) are labeled in the scene. All models in the database are labeled with interaction maps indicating different regions on the model an agent would interact with while performing each activity; here, we visualize the per-model interaction map annotations on the training scene.

## 4 Scene Templates

A scene template is a high-level representation of a scene that encodes both geometric and activity properties of an environment. We use a scene template as an intermediate representation between an input scan and our output object arrangements — once we convert a scan to a scene template, no additional information about the scan is needed. Scene templates can also be modeled directly or constructed from other data sources such as labeled 2D images or textual descriptions. Our scene template is defined over a fixed rectangle of space with known spatial extents.

### 4.1 Geometry Representation

We would like our scene template to capture a coarse representation of the geometric layout of an environment. Our focus is on directing the synthesized results towards scenes with a realistic layout similar to an observed input while not otherwise restricting the space of scenes that can be generated. We want to capture broad

Activities	Database Instances
Use a desktop computer	42
Eat at a dining table	52
Prepare for bed	16
Sit on a couch	20

**Table 1:** Summary of the dataset. For each activity, we show the number of example instances of this activity in the scene database to learn our activity model (see Section 7).

scene properties such as the presence of furniture or the density and height of objects on a surface, but not specific properties such as the brand of a monitor or the title of a book.

Our representation divides the environment into a uniform grid of vertical columns, each covering 5 cm by 5 cm. Within each column  $c$ , we store two distances:  $c_s$ , the height of the highest supporting plane above the floor, and  $c_r$ , the height of the highest observed geometry above the highest supporting plane. A supporting plane is a surface that supports other objects, such as the surface of a desk, a bed, or a coffee table. The division into supporting planes and residual geometry that sits above the supporting plane is useful for capturing the approximate placement of short objects such as plates that might otherwise be lost in the relative variation of table heights. We use  $T_G$  to refer to the desired grid of  $(c_s, c_r)$  height values for a scene template  $T$ .

Figure 2 (top middle) visualizes this representation. The blue columns represent the expected height of a supporting plane in that region ( $c_s$ ) and the green columns represent the height of residual geometry ( $c_r$ ) which sits on top of a supporting plane.

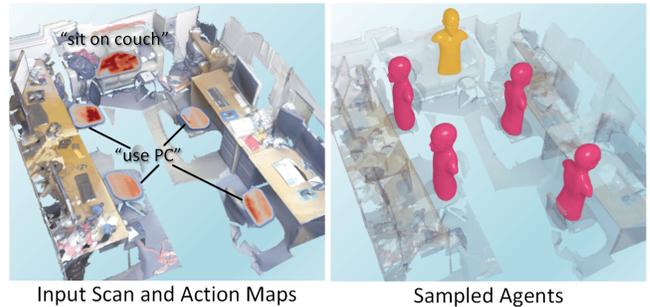
Environments generated from the scene template will attempt to match both the support plane and residual geometry heights for all vertical columns in the domain. We quantify how well an object arrangement  $O$  agrees with  $T_G$  with a function  $P_{\text{geo}}(O_G, T_G)$ , as described in Section 6.

## 4.2 Activity Representation

We represent activities in a scene template as a set of activity maps which are continuous distributions defined over the entire 2D floor rectangle. We use the set of activities given in Table 1. For each such activity, a scene template defines a dense set of  $(x, y, \theta)$  activation samples ranging from 0% (not likely an agent could perform this activity here) to 100% (extremely likely an agent could perform the activity). The  $(x, y, \theta)$  components encode the expected position and orientation of the agent’s head while the agent is performing the activity. This representation may encode several distinct regions corresponding to different agents, such as a computer lab with many distinct computer setups. Environments generated from the scene template will try to position objects such that an agent can perform the activities specified in the activity maps.

## 4.3 Sampling Proxy Agents

When working with activity maps, we start by sampling the continuous activity distribution with a discrete set of *proxy agents*  $T_A$ . A proxy agent is identified by an activity label and an  $(x, y, z, \theta)$  coordinate in the scene representing the expected position and orientation of the agent’s head. The agent’s head position implicitly constrains expected positions of body parts (such as the shoulders) as fixed offsets relative to the agent’s head and face direction. In Section 7, we quantify how well a proxy agent can act in a given object arrangement. Importantly, we do not pick a single set of



**Figure 4:** On the left, we show an input scan along with two continuous activity maps. On the right, we show one sample of discrete and oriented agents drawn from this distribution.

proxy agents, but instead sample a new agent set each time we synthesize a new scene, as described in Section 8.

To sample proxy agents from our continuous activation samples, we start by choosing a random sample in the top 10% of sample activations to be a proxy agent at this location. The height of the proxy agent is taken as the average height of the activity observations in the scene database, as described in Section 7.1. To prevent multiple agents from being sampled too close together, we then exclude all samples within 50 cm of the randomly sampled location. We repeatedly sample agents from the top 10% of activations until there are no valid samples left. We determine the orientation of a proxy agent by integrating the directions of nearby activation samples weighted by the amount of activation and the distance to the agent location (20 cm with a linear falloff). We then snap the agent orientation towards the nearest object support plane, which is determined by the spatial distance between the plane and the agent location and the deviation to the original agent orientation. Our complete set of sampled proxy agents is the union of all agents produced by the activity activation maps for each activity type in the scene template. Figure 4 shows an example of agent sampling from a set of activity maps.

## 5 Analyzing 3D Scans

By analyzing an input RGB-D scan, we can generate a scene template that will produce scenes similar in both geometry and functionality to the environment captured by the scan.

### 5.1 Plane Extraction

We start by extracting a set of dominant planes from the scan’s 3D point cloud using an existing approach designed for indoor scan classification [Mattausch et al. 2014]. Figure 5 shows an input scan and the extracted planes. We then classify each plane into one of three categories: *agent support* planes of a particular category, such as the seat of a chair or sofa where an agent might stand or sit while interacting with objects, *object support* planes, such as the surface of a desk or table which are large surfaces that support many other objects, and other planes not falling into either category. We classify each type of plane by simple normal, surface area, and height heuristics. An object support plane must have a normal within 5 degrees of the gravity vector, a surface area greater than  $0.5 \text{ m}^2$ , and its centroid must be more than 0.5 m above the floor. For each category of agent support objects (ex. bed, sofa, and chair), we compute the observed surface area of these support planes in the activity database described in Section 7.1 to get a minimum and maximum observed surface area. We classify a plane as an agent support plane for the category if it is within 15% of these bounds.

Figure 5 middle-right shows the planes classified as object support and agent support (chair).

## 5.2 Geometry from Scans

We use a simple ray-tracing approach to convert our RGB-D scan into the column-based representation for scene template geometry described in Section 4.1. For each column in  $T_G$ , we shoot a ray from the ceiling downwards into the scan, recording the first geometry in the 3D mesh that is intersected as well as the height of the first support plane. Figure 5 (right) visualizes this representation for the scan shown on the left.

## 5.3 Activity Retrieval from Scans

As introduced in Section 4.3, we sample proxy agents from continuous activity maps. We obtain these maps directly from the scanned input geometry using an action-retrieval approach [Savva et al. 2014]. To run this method, we associate actions with agent types and train the algorithm accordingly. For a given RGB-D scan, we then obtain a probability distribution for each activity type. Each distribution is represented as a densely-sampled  $uv$  map where every sample contains an activation probability, and a dominant orientation given by a scene-pose classifier (for details, we refer to Savva et al. [2014]). In addition, we filter each activity distribution with the extracted agent support planes (see Section 5.1); i.e., we set the activation probabilities to zero if the sample is not above an agent support plane associated with the agent type).

Note that our method is orthogonal to the underlying action retrieval approach, and could be combined with a different scene categorization technique (e.g., Jiang and Saxena [2013]).

## 6 Geometry Model

To quantify how well a synthesized object arrangement  $O$  agrees with  $T_G$ , we start by computing  $O_G$ , an analogous grid-of-columns representation for the synthesized scene. Each object in  $O$  comes from a categorized 3D model database. We identify some of these categories as being support categories by looking at an existing scene database (see Section 7.1) and looking for all categories that have been observed supporting at least two other objects. To compute  $O_G$ , we shoot a ray for each column from the ceiling downward into the scene, recording the height of the first object of any type hit as well as the height of the first supporting object hit. This mimics the method used for 3D scans in Section 5.2, except we identify supporting objects by category rather than by detecting dominant horizontal planes.

To compare  $T_G$  and  $O_G$  we average a comparison function for all columns  $c$ :

$$P_{\text{geo}}(T_G, O_G) = \sum_{c \in T_G} \frac{e^{-\frac{(\Delta c_s)^2}{\sigma_s^2}} + e^{-\frac{(\Delta c_r)^2}{\sigma_r^2}}}{2|T_G|} \quad (1)$$

$\Delta c_s$  and  $\Delta c_r$  are the differences between the supporting plane height and residual geometry height for column  $c$  in  $T_G$  versus  $O_G$ .  $\sigma_s$  and  $\sigma_r$  control how strongly the geometry of the synthesized scene is penalized for deviating from the scene template; we use  $\sigma_s = 15$  cm and  $\sigma_r = 15$  cm.  $P_{\text{geo}}(T_G, O_G)$  ranges from 0 (no geometric similarity) to 1 (exact geometric match).

## 7 Activity Model

We want to estimate how well a given object arrangement supports each activity map in a scene template. After sampling the activity

map to produce a set of proxy agents (see Section 4.3), we need to determine how well each proxy agent can execute its labeled activity. To accomplish this, we use a corpus of annotated scenes to train a model for each activity. We use these examples to represent the expected distribution of objects involved and their expected position and orientation relative to the agent. Our activity model also incorporates knowledge of the ways that an agent interacts with each object while performing the activity. For example, some objects might have regions that the agent needs to be able to see or touch.

### 7.1 Scene and Model Database

For learning the distributions of objects associated with an activity, we use 100 indoor scenes from Fisher et al. [2012] augmented with 25 scenes to support the activities listed in Table 1. The objects are arranged in a static support hierarchy, and are grouped into one of 125 categories. We manually augment the database with activity knowledge by inserting proxy agents into each scene. Each activity is also annotated with a set of objects in the scene that participate in that activity. The same object may be used by different agents — for example, a group of agents that are eating dinner may have their own chair and plate objects but collectively share the dining table. Our activity model is broken into two parts: an object occurrence model and an object interaction model.

### 7.2 Object Occurrence Model

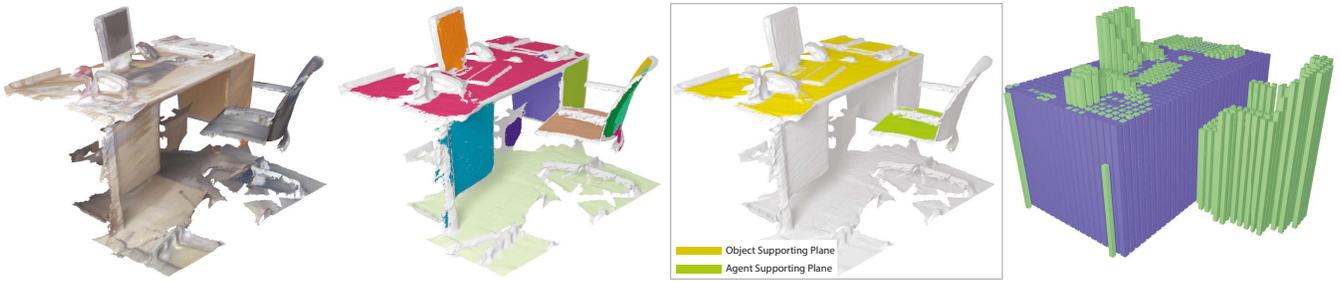
We use a Bayesian network to model the distribution of objects that participate in each activity. We follow the learning approach used for modeling object occurrence in example-based synthesis and semantic modeling [Fisher et al. 2012; Chen et al. 2014]. We assume each activity instance contains objects drawn independently from some unknown distribution. For each activity observation in the database, we count the number of instances of each object category involved and represent this as an *occurrence vector*  $V$  in  $\mathbb{Z}^{|C|}$  where  $|C|$  is the number of categories.

The set of all occurrence vectors is used to learn the structure and parameters of a Bayesian network using structure learning with the Bayesian information criterion and dense conditional probability tables [Margaritis 2003]. We use the booleanization transform from prior work on modeling object occurrence in scenes to help reduce the number of parameters in the conditional probability tables [Fisher et al. 2012]. Occurrence vectors which contain a never-before-observed count in a category will receive zero probability — if an agent has never been observed to use either zero or more than three monitors at once, such setups will be deemed impossible by our occurrence model.

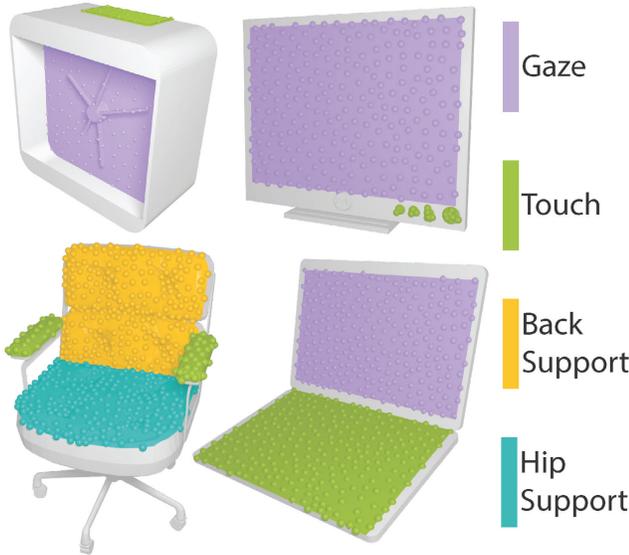
There is one such Bayesian network for each activity, which can assign a probability  $P_{\text{occur}}(a, O_a)$  to any candidate object collection  $O_a$  bound to agent  $a$  by first counting the number of instances of each category to produce an occurrence vector  $V$ . This Bayesian network structure can capture many logical dependencies observed in the training data, such as a requirement to have either a laptop or a monitor and keyboard to use a computer. For simple activities such as an agent preparing for bed, we have often observed it sufficient to simply assume all object categories occur independently, which is equivalent to a Bayesian network with only the edges due to booleanization and no additional learned edges.

### 7.3 Object Interaction Model

For an agent to execute an activity, each object needs to be positioned such that the agent can easily interact with it. We explicitly



**Figure 5:** Scan plane extraction and geometric representation. From left to right: the input scan, the extracted planes, the extracted object and agent support planes, and the final geometric representation inferred from the scan. Blue columns represent the expected height of an object support plane and green columns represent the expected height of residual geometry that sits above the supporting plane.



**Figure 6:** Object interaction annotations for an alarm clock, a monitor, a chair, and a laptop.

model the way agents interact with objects by defining a set of *interaction maps* for each model that represent polygons on the mesh surface that an agent might interact with in different ways (see Figure 6). We consider the following types of interaction maps:

- **Gaze:** Regions that need to be visible to the agent.
- **Touch:** Regions that the agent needs to touch with its hands.
- **Back support:** Regions that support the agent’s back.
- **Hip support:** Regions that support the agent’s hips.

To evaluate how well a candidate object placement supports each interaction, we start by sampling each interaction map surface into a discrete set of surface points with associated normals, using a density of approximately 1 point for every  $20 \text{ cm}^2$  and guaranteeing at least one point per connected component of the interaction map surface. Given a candidate placement, we compute the interaction score for a single point of our four interaction map types as follows:

- **Gaze:** If the ray from the agent’s head position to the surface point is obstructed, the score is zero. Otherwise, the score is  $\max(\cos(\pi - \theta), 0)$ , where  $\theta$  is the angle between the surface normal and the gaze vector from surface point to agent head.
- **Touch:** The score is  $1 - \epsilon d$  if a ray originating from either the agent’s left or right shoulder to the point is unobstructed; otherwise the score is 0.  $d$  is the distance in meters between the point and the agent’s closest shoulder, and  $\epsilon = 0.01$ . This term means that all else being equal, agents will prefer objects

they touch to be closer to them.

- **Back support:** The score is  $\max(\cos(\pi - \theta), 0)$ , where  $\theta$  is the angle between the surface normal and the support vector from the surface point to the middle of the agent’s spine.
- **Hip support:** Instead of using a point discretization, an object has a score of 1 for the hip support interaction if a cylinder with radius 10 cm centered at the agent’s head pointing down hits only polygons marked as “hip support” and 0 otherwise.

For gaze, touch, and back support, we further set a point’s interaction score to zero if the spatial offset from the point to the corresponding body part is more than 5% from any observed offset in one of the training scenes. This captures important properties of the interaction distribution, such as the expected distance between a viewer’s eyes and the television screen or between a typist and the keys on a keyboard. The total interaction score is the average score over all interaction points. Using this approach, we assign an interaction plausibility score between 0 and 1 to any agent  $a$  and object  $o$  by taking the product of all interaction scores:

$$P_{\text{interact}}(a, o) = \prod_{x \in \{\text{gaze, touch, back, hip}\}} P_x(a, o) \quad (2)$$

When an interaction type is not supported by an object, it is removed from this equation. The interaction score for a collection of objects  $O_a$  bound to an agent  $a$  is simply the product of the score for each object:

$$P_{\text{interact}}(a, O_a) = \prod_{o \in O_a} P_{\text{interact}}(a, o) \quad (3)$$

**Types of interactions.** We do not consider many relevant types of interactions such as “listen” for speakers or “feet” for piano pedals. These could be modeled independently, but for simplicity, we map them onto another interaction type. In our results, the only instances of this remapping are to represent “listen” and “provide light” with the “gaze” interaction.

**Object orientation.** The orientation of most objects, such as monitors and desks, does not need to be modeled explicitly; incorrect orientations can be detected from interaction constraints such as the inability of the agent to see one side of a monitor or to touch the drawers of a desk. However, a small number of object categories such as keyboards have a strong expected orientation relative to the agent that cannot be inferred from a simple “touch” map over the object surface. If a keyboard is rotated 180 degrees relative to its expected orientation, a person can still touch every key of the keyboard but typing is nevertheless much harder for most people to perform. For objects in these challenging categories, we explicitly record the orientation of each occurrence of each model relative to

the proxy agent in the scene database. We set  $P_{\text{touch}}(a, o)$  to zero if an object deviates by more than ten degrees from an observed orientation. For our results, the only categories oriented by observation are keyboards, computer mice, and silverware.

## 7.4 Activity Free Space

When acting, an agent needs to have space to comfortably move around. For example, large obstructions under a desk or table can make it difficult to maneuver our feet to easily get in and out of a chair. To represent this expectation of free space, each activity has an associated free-space region we model as a collection of bounding boxes relative to the agent’s coordinate frame. When synthesizing scenes, objects are not allowed to have any geometry in the free-space region of any agent. The exception is an object with a positive hip-support interaction for the agent (ex. a chair or sofa). We determine the free-space region for each activity by scanning a real person in a resting configuration during the activity. We then voxelize the torso and lower body of the scanned skeleton and take the set of voxels to be the free-space region.

## 8 Scene Synthesis

Our goal is to generate scenes that exemplify a scene template, which encodes both the geometry of an input scan (Section 6) and activities detected in the scan (Section 7). We start by defining a scoring function for a given arrangement of objects by combining the terms from sections 6 and 7. Our final pipeline works by generating many plausible candidate scenes using a simple greedy approach, choosing high-scoring results from this collection of candidate scenes.

### 8.1 Scoring Function

Our scene template scoring function takes as input a scene template  $T$  and object arrangement  $O$  and returns a value between 0 (totally dissimilar) and 1 (complete agreement):

$$\text{Score}(T, O) = P_{\text{geo}}(T_G, O_G) \prod_{a \in T_A} P_{\text{interact}}(a, O_a) \quad (4)$$

Here,  $T_A$  is the set of proxy agents in the scene template and  $O_a$  is the set of objects in  $O$  bound to the proxy agent  $a$ .  $P_{\text{geo}}$  ensures that the generated scene agrees with the high-level geometric properties encoded in the scene template.  $P_{\text{interact}}$  asserts that objects needed for an activity are at locations and orientations where the agent will be able to interact with them effectively.  $P_{\text{occur}}$ , which returns a high score only when each proxy agent in the template has a usable set of objects bound, does not need to be explicitly modeled in our scoring function because we will directly sample from the object distribution for each activity. This ensures that each proxy agent in the template will have a usable set of objects bound.

### 8.2 Synthesis Pipeline

We generate scenes that match a scene template by repeatedly generating candidate scenes using a simple greedy approach, choosing scenes with the highest score according to Equation 4. To generate a candidate, we first sample a set of proxy agents from the continuous action distribution encoded in the scene template using the method described in Section 4.3. Next, we use the per-activity Bayesian network trained in Section 7.2 to sample a set of objects to insert for each agent. This set is represented as a collection of category-integer pairs indicating the number of each category that should be present and bound to the agent in the final scene. We then start from an empty scene and iteratively build up a list of candidate models

and densely-sampled locations to insert them, greedily adding the best candidate object into the scene. More specifically, a candidate object is defined by an agent the object belongs to, a scaled model from the backing model database, and a location and orientation in the scene where the model will be inserted.

```

procedure GENERATE_SCENE(SceneTemplate T)
  Scene  $S_0 \leftarrow \{\}$ 
  AgentList  $A \leftarrow \text{SampleAgents}(T)$ 
  for  $a \in A$  do
    SampledObjects( $a$ )  $\leftarrow \text{SampleActivityModel}(a)$ 
  while RemainingSampledObjects( $a \in A$ )  $\neq \{\}$  do
    CandidateObjectList  $N \leftarrow \{\}$ 
    for  $a \in A$  do
      for  $c \in \text{RemainingSampledObjects}(a)$  do
        for  $m \in \text{SampleModels}(c, k_m)$  do
          for  $l \in \text{SampleLocations}(S_i, m, k_l)$  do
             $N \leftarrow N \cup \{a, m, l\}$ 
     $S_{i+1} \leftarrow S_i \cup \arg \max_{o \in N} \text{Score}(T, S_i \cup o)$ 

```

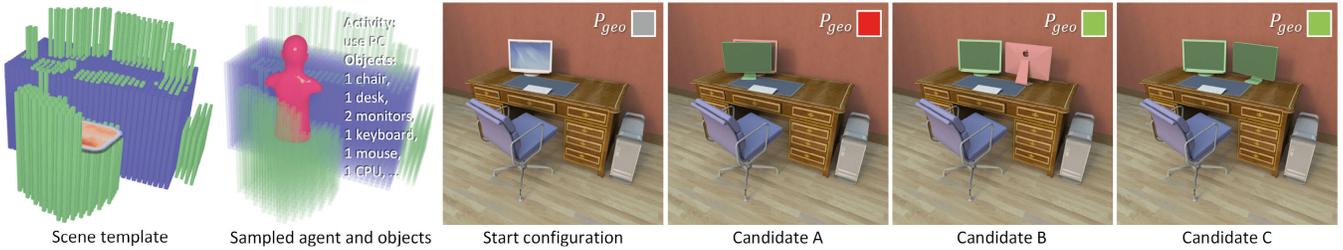
**Figure 7:** Pseudocode for generating a candidate scene that conforms to a given scene template.

The pseudocode for our synthesis method is given in Figure 7 and steps of the algorithm execution are visualized in Figure 8. At each step, we iterate over all possible agents in the template, and for each agent, we iterate over categories that should be bound to that agent in the final scene but have not yet been inserted. Within each category, we randomly sample  $k_m$  different models from the backing model database along with a uniform scaling term that converts the model’s original coordinate axes into physical units. For each model, we sample the surface of existing objects in the scene for  $k_l$  random locations and rotations where this object could plausibly be placed. A location is plausible if three conditions are met:

- The object must be placed on the surface of a category that it has been observed on at least once in the training scene database (e.g., a rug cannot be placed on a table).
- The contact normal between an object and its parent surface must not deviate more than 5 degrees from an observed contact normal (e.g., mugs cannot be placed on a vertical wall).
- When placed at this location, the object must not collide with any other object.

We evaluate the change in the scene scoring function for the  $k_l$  locations for each of the  $k_m$  models for each agent and category. We then insert the object that yields the greatest increase in score, stopping when all objects have been added. If no valid object placement can be found or the resulting scene has zero score, the candidate scene is rejected. All results in this paper use  $k_m = 5$  and  $k_l = 10000$ . Note that inserting an object may affect the interaction score of a previously-placed object, such as a picture frame blocking the visibility of a monitor.

Our final synthesis pipeline simply repeats the per-scene synthesis process many times, choosing the top-ranked results. This produces a diverse set of plausible scenes while avoiding local minima which can occur due to poor activity model sampling (e.g., sampling three monitors on a desk that can only comfortably fit one monitor) or a bad model choice (e.g., choosing a dining table model that is not large enough to comfortably support the given number of diners). We explored other scene sampling approaches, such as using a Markov Chain Monte Carlo method; however, we found the high proposal cost to be prohibitive and the chain was extremely slow to mix [Yeh et al. 2012].



**Figure 8:** Visualization of our scene synthesis algorithm. Given a scene template, we first sample a random set of proxy agents and the objects needed by each agent. We then iteratively insert objects into the scene. The start of an iteration is shown along with three candidate placements proposing to add a second monitor to the scene. In (A),  $P_{geo}$  decreases and the interaction score of the first monitor decreases because its screen is no longer visible by the agent. In (B),  $P_{geo}$  improves; however, the monitor screen is facing away from the agent and its  $P_{interact}$  term is 0. In (C), both  $P_{geo}$  and  $P_{interact}$  improve.

**Scene Diversity.** For content creation applications, the goal is to produce a diverse set of results that span the design space of the problem [Xu et al. 2012]. For our dataset, the key source of diversity comes from sampling a different sets of objects for each agent with the activity occurrence model in Section 7.2. Additional diversity comes from choosing  $k_m$  to be smaller than the number of models in the database with a given category. By restricting the set of models available, the algorithm will be forced to take different choices that often have cascading effects on later stages as the subsequently placed objects are forced to redistribute themselves to enable agent interaction. If desired, additional control over the diversity can be implemented by choosing at each iteration any object whose score improvement is within  $D\%$  of the maximum improvement. Our method yields the least diversity with  $D = 0\%$ , and as  $D$  increases so does the scene diversity at the potential cost of placing objects at locations that are harder for the agent to interact with.

**Multi-agent objects.** Some objects in a scene need to be shared by many agents, such as diners at a dining table. To accommodate this, when inserting an object  $o$  to a scene we bind it to any agent  $a$  who has a non-zero interaction score for this object according to  $P_{interact}(a, o)$ . This method works well for many environments, but can be confused by tightly packed scenes with many similar and nearby agents, such as restaurants or classrooms. For such environments, a more effective approach could be to use a hierarchical scene decomposition to model the way agents cluster into subgroups [Liu et al. 2014].

**Model consistency.** When multiple instances of the same category occur in a scene, it is sometimes the case that the instances are consistent (all instances use the same geometric model) or diverse (different instances use different models). This is typically a stylistic choice and does not affect the functionality of the scene. To model this in our synthesized scenes, for each category we compute the percentage of training scenes that use a consistent model for this category. We use the same frequency when determining whether a category in a modeled scene should use a consistent model.

## 9 Results and Evaluation

To evaluate our work, we use a Microsoft Kinect sensor to scan a set of seven cluttered indoor environments that support the activities listed in Table 1. We use a volumetric fusion approach [Nießner et al. 2013] to align scanned RGB-D frames and reconstruct 3D meshes, which are used as the input of our method (see Figure 9, left). We then extract scene templates for each environment using

the method described in Section 5 and synthesize 500 scenes from each scene template.

The time required to synthesize each scene is approximately linear in the number of objects present in the environment; the student office depicted in Figure 4 took 41.0 seconds per synthesized scene on a 4-core Intel i7 CPU. Smaller scenes such as the dining table in Figure 2 take an average of 15.7 seconds per scene. Over 50% of the compute time is spent evaluating bounding-volume hierarchy intersections at each candidate location to verify that it does not collide with an existing object.

Our scene and model database, as well as the per-scene activity annotations and the per-model interaction annotations, can be found on the project website<sup>1</sup>.

### 9.1 Synthesis Results

Figure 9 shows five scans along with three of the scenes generated by our method. Our synthesis algorithm produces a diverse and plausible set of scenes that emulate the functionality of the scanned environment. For each agent implied by the scan, we directly synthesize a viable set of objects up to our understanding of each activity as encoded in the activity model from Section 7.

Our approach deals well with scenes that mix different types of activities. For example, in the student office scene in Figure 9, a couch is present amongst a large number of computer desks. The algorithm correctly separates the computer agents from the agent sitting on the couch. Furthermore, although most database scenes with a couch occur in a living room which also contains a coffee table, the algorithm does not synthesize a coffee table in front of the couch. While this relationship is strongly reinforced by the training examples, the presence of a coffee table is not supported by the geometric matching term  $P_{geo}$  for this scan.

### 9.2 Baseline Synthesis Comparison

To demonstrate the advantage of our activity-centric approach, we compare our synthesis method against a baseline synthesis algorithm based on previous work that we condition on the geometric layout of a 3D scan [Fisher et al. 2012]. This method requires a set of training scenes which contain the desired object distribution. For scenes containing a single, discrete activity such as the computer desks shown in Figure 10, our scene database already contains many similar scenes. For the larger scenes shown in Figure 9, to enable a comparison we manually created four additional scenes with similar object distributions but different object arrangements.

<sup>1</sup><http://graphics.stanford.edu/projects/actsynth/>

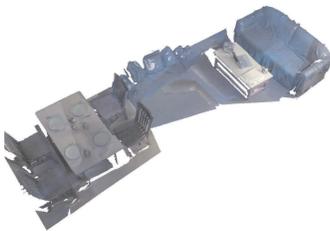
Double dining table



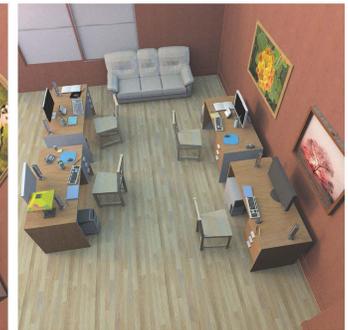
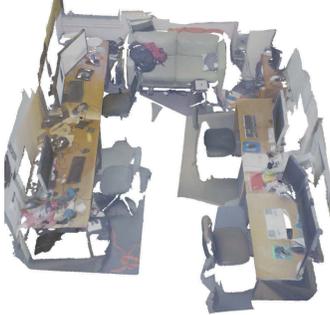
Living room



Studio apartment



Student office



Dining hall

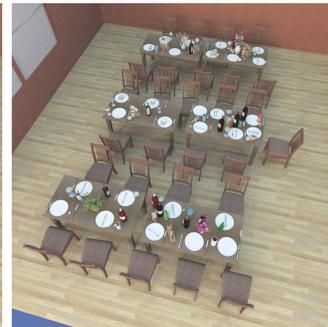
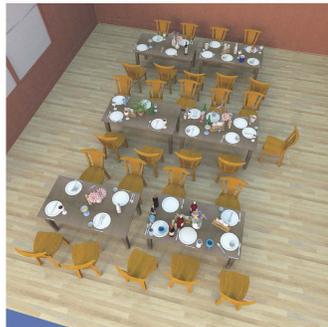


Figure 9: Left: input 3D scan. Right: modeling results for different environments using our method.



**Figure 10:** Baseline synthesis results. For the two dining scenes shown, the input scan and results from our method are depicted in Figure 2.

These scenes were constructed to be as favorable as possible to the prior method; for example, each training scene created for the “student office” scene contains exactly four desks and one couch.

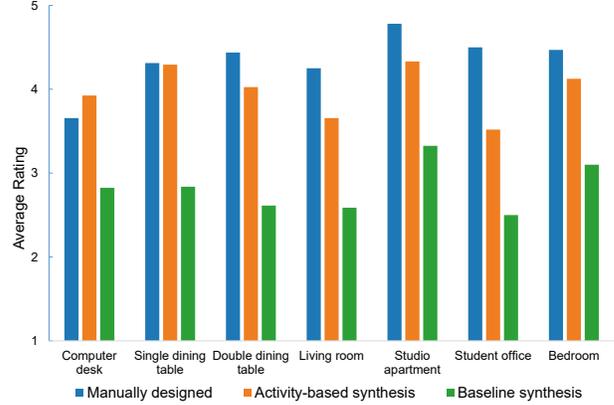
Our comparison implements the example-based arrangement approach in Fisher et al. [2012]. This method starts by sampling the total set of objects (the “occurrence” model), then arranging the objects according to a per-category multivariate-Gaussian score function  $S(O)$  (the “arrangement” model). To support loosely conditioning the generated scene on an input scan, we instead consider the final arrangement score to be a linear combination of the arrangement score  $S(O)$  and the scan-based  $P_{\text{geo}}(O)$  term described in Section 6. Because it is required by the baseline method, for each category we also manually oriented all objects in our database into a canonical coordinate frame; this information is not needed or used by our synthesis method.

Figure 10 shows example scenes generated using this baseline synthesis method. The baseline method generates a valid set of objects and correctly recovers the orientation of some large objects such as the chairs from the scan geometry, but it still produces many invalid arrangements and object orientations. Because it fails to understand how people interact with dining tables it does not position or orient objects such as silverware and plates well. For the isolated computer desk scenes, the objects are mostly arranged and oriented reasonably because the scene is governed by a single, dominant orientation. However, because this method does not understand how people interact with computers, the resulting scenes still contain significant errors such as monitors and paintings that cannot be viewed easily.

### 9.3 Perceptual Evaluation Study

We use a judgment study to quantitatively evaluate the functionality and plausibility of our synthesized scenes compared against human-made scenes and our baseline synthesis approach. We use seven indoor scenes including the top four shown in Figure 9. For each input scan, we then obtained a set of 3D scenes for each experimental condition:

1. *Manually designed*: Four students not associated with the project were presented with a picture of each environment and the 3D scan and asked to model a functionally similar scene. Scenes were modeled with the same model database described



**Figure 11:** Perceptual evaluation results. We ask a group of users to evaluate the plausibility and functional similarity to an RGB-D scan of scenes designed by humans and scenes synthesized by our method and a baseline method. Here, we show the average rating the users provided to each experimental condition.

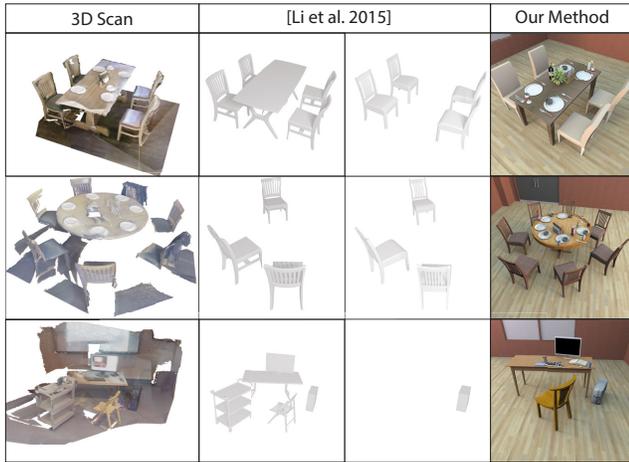
in Section 4.2 and modeling took approximately fifteen minutes per scan.

2. *Activity-centric synthesis*: We sort our 500 synthesized scenes according to  $P_{\text{geo}}(T_G, O_G)$ , choosing 20 scenes at random from the top 100 scenes. This selects synthesized scenes that respect the geometry of the scan while also choosing scenes with a diverse set of objects.
3. *Baseline synthesis*: We synthesize ten scenes use the baseline approach described in Section 9.2, using the same technique to generate diverse scenes that also respect the scan geometry.

Twenty students not involved in the study and also separate from those who modeled scenes were recruited to evaluate each scene in each condition. They were presented with multiple pictures of the scan from different angles and asked to evaluate a virtual scene on its plausibility and functional similarity to the scanned environment on a Likert scale ranging from 1 (implausible and functionally dissimilar) to 5 (plausible and functionally similar). We provide the complete experimental setup for both the scene modeling and scene evaluation tasks as supplemental materials along with the modeled and synthesized scenes used for each condition.

The results of this evaluation are shown in Figure 11. As expected, on average evaluators found the scenes generated by people to be both plausible and functionally similar. Our algorithm performed best on single and double dining table scenes, where agents are well-defined and the expected distribution over the arrangement of objects is well-captured by our model. In comparison, by not incorporating an understanding of activities, the baseline synthesis algorithm received the lowest rating on these dining scenes. The baseline method consistently received the lowest scores, failing to capture the arrangement and orientation of objects in large scenes.

Our synthesized scenes received a functionality and plausibility score of either 4 or 5 for 76% of user ratings, indicating that users found them to be of overall high quality. This is valuable for content-generation tasks, where we would like to be able to generate large quantities of viable game and film content with little to no supervision. In comparison, the baseline method received a score of 4 or 5 only 22% of the time.



**Figure 12:** Retrieval results using [Li et al. 2015]. In the second column, we show the results using a 6,000 model database specifically tailored to contain the models in the scan. In the third column, we show the results using our 12,000 model database containing more general indoor objects. Geometric retrieval methods often fail when insufficient key points can be extracted (e.g., the round table and the small items on the table) or the extracted key points cannot form consistent constellations due to partial scan or occlusions (e.g., the chairs with only the upper parts scanned).

#### 9.4 Geometric Retrieval vs Scene Modeling

Our method uses a simple geometric matching term to capture the scan geometry. When scan quality is high, it is possible to instead use a finer-scale geometric matching term to produce stronger alignment between the generated scene and the provided scan [Satkin and Hebert 2013; Li et al. 2015]. Similarly, it is possible for the activity model presented in Section 7 to inform a geometric retrieval method. To better understand the behavior of geometric retrieval methods, in Figure 12, we show a geometric retrieval result on three different input scans. Although it is able to retrieve some large furniture objects, it is not always successful and is especially prone to error when the scanned model is not present in the database or contains smooth, indistinct features such as a round table. Our method, which focuses on scene modeling instead of geometric retrieval, is still able to produce a viable scene that roughly aligns with the scanned geometry. Note that our method is unable to model parts of the scene corresponding to activities it cannot detect, such as the cart in the bottom scan.

#### 9.5 Interaction Terms

In Figure 13 we show the effects of removing different interaction terms from Equation 2. Removing the “gaze” interaction causes objects such as monitors or speakers to sometimes rotate in ways that are geometrically consistent but not visible to an agent; similarly, objects such as posters may be occluded by other objects in the environment. Removing the “touch” interaction causes objects such as desktop computers to be oriented in unusual ways, or objects such as mugs to be placed in difficult to reach and occluded areas. Removing the “hip support” interaction can cause too many objects to be placed on sittable areas such as couches producing a scene whose functionalities are inconsistent with those of the input scan. Removing the “back support” interaction can cause sitting objects to be oriented in unusable ways.



**Figure 13:** Interaction map comparison. We show an input scan and synthesis results with different interaction terms removed. Removing interaction terms causes the resulting scenes to be functionally implausible in different ways.



**Figure 14:** Synthesis results for a scene which contains a laptop, mouse, and various food items.

#### 9.6 Limitations

The key assumption of our approach is that an environment can be decomposed into a set of regions where the layout of objects in each region are strongly influenced by their interactions with a hallucinated proxy agent. We can produce good results for scenes where this assumption holds and where we have good models for the activities that take place in each region. For environments where we do not have a model for all activities, our method simply ignores these objects. For example, in Figure 1, a clothes hamper and associated clothing on the floor of the bedroom is not captured by our method because this is not one of the activities represented in our database. The synthesized scenes still support the detected bed and computer activities.

Sometimes a scene contains a region that does not quantize well into one of our activities. Consider the environment shown in Figure 14. This scene contains a mix of objects belonging to both “dining table” and “use desktop PC” activities, but neither categorization is entirely appropriate. Our activity detector described in Section 5.3 successfully detects the agent support plane but our action classifier returns positive activation for both activity types.

Whether synthesized as a dining scene or as a computer desk scene, neither result is wholly satisfactory. In practice, many regions are a composite of multiple types of activities, and the ability to incorporate this “weighted activity vector” directly into the synthesis approach is an interesting avenue of future research.

Our method does not directly model large-scale object layout constraints which can cause the synthesized results to be implausible. Although we use the geometry of a 3D scan to capture coarse global layout, our approach can still produce scenes that violate many common design patterns in indoor scenes. Consider the large dining scene shown at the bottom of Figure 9. The synthesized scenes often separate the tables by a significant gap.

## 10 Conclusion

We have explored ways to use activities encoded in a scene template as a focal point to represent the semantics of 3D scenes. We believe that a deep understanding of activities is crucial for scene understanding and reconstruction tasks. Complex activities such as “cooking fried chicken” cover a wide range of sub-activities and are done over a broad area of space interacting with potentially hundreds of objects. A kitchen is structured so that these activities can be easily and efficiently performed. This is consistent across human-made environments, which are built precisely so that we can act in them. The activities we have explored are dramatically simplified compared to their real-life counterparts—the agents are well-localized and object interactions can be modeled without complex multi-stage dynamics such as opening cabinets or rearranging furniture. Nevertheless, we feel that our approach serves as a sound basis upon which to explore extensions that support a more general class of activities.

We evaluate how well an agent can perform an activity by its ability to interact with nearby objects in specific ways. Our activity model is learned from an augmented database of virtual scenes and each object is manually tagged with the regions that admit different types of interactions. The challenge with learning from these virtual databases is that they are static — the dynamic motion executed by agents performing actions in these environments must be manually approximated and annotated. A crucial long-term goal is to learn our activity model directly from observation of humans or other agents. This is challenging because object segmentation, recognition, and tracking are hard in natural environments. Nevertheless, learning from real observations allows for a more faithful activity model with a deeper understanding of how activities are decomposed into actions and the different ways agents interact with objects. We would like to move beyond the simple set of interaction types described in Section 7.3 to a richer model of the interaction between agents and objects.

The desire to generate novel, diverse content occurs in many applications. One direction that has recently seen growing success is generating content to train machine learning systems. For example, commercial skeleton tracking software works by training on a large set of synthetic skeletons derived from an underlying model of human motion [Shotton et al. 2013]. Moving forward, we believe that many scene understanding systems will rely upon training on a large, synthetic training set of annotated scenes. To be effective, these training scenes need to be arranged in plausible configurations similar to those observed in the real world. Collecting this training data without scene synthesis is prohibitive, as the cost of arranging and acquiring real-world data that is well-segmented and annotated is immense. We believe scene modeling methods such as the one we present will one day be an effective method to generate training data for recognition tasks in robotics and vision that deal with the relationships and arrangements of objects in the real world.

## Acknowledgements

We thank Angela Dai for helping with scanning and the video voice over. We also gratefully acknowledge the support from NVIDIA Corporation for hardware donations. This research is co-funded by the Max Planck Center for Visual Computing and Communications and NSFC grant 61202221.

## References

- 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 Transactions on Graphics (TOG)* 33, 6, 208.
- DROST, B., AND ILIC, S. 2012. 3d object detection and localization using multimodal point pair features. In *Second International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT)*, IEEE, 9–16.
- FISHER, M., SAVVA, M., AND HANRAHAN, P. 2011. Characterizing structural relationships in scenes using graph kernels. In *ACM Transactions on Graphics (TOG)*, vol. 30, ACM, 34.
- FISHER, M., RITCHIE, D., SAVVA, M., FUNKHOUSER, T., AND HANRAHAN, P. 2012. Example-based synthesis of 3D object arrangements. *ACM Transactions on Graphics (TOG)* 31, 6, 135.
- GALLEGUILLOS, C., AND BELONGIE, S. 2010. Context based object categorization: A critical survey. *Computer Vision and Image Understanding* 114, 6, 712–722.
- GIBSON, J. 1977. The concept of affordances. *Perceiving, acting, and knowing*.
- GRABNER, H., GALL, J., AND VAN GOOL, L. 2011. What makes a chair a chair? In *CVPR*.
- JIANG, Y., AND SAXENA, A. 2013. Hallucinating humans for learning robotic placement of objects. In *Experimental Robotics*, Springer, 921–937.
- JIANG, Y., LIM, M., AND SAXENA, A. 2012. Learning object arrangements in 3d scenes using human context. *arXiv preprint arXiv:1206.6462*.
- JOHNSON, A. E., AND HEBERT, M. 1999. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21, 5, 433–449.
- KIM, Y. M., MITRA, N. J., YAN, D.-M., AND GUIBAS, L. 2012. Acquiring 3d indoor environments with variability and repetition. *ACM Transactions on Graphics (TOG)* 31, 6, 138.
- KIM, V. G., CHAUDHURI, S., GUIBAS, L., AND FUNKHOUSER, T. 2014. Shape2pose: Human-centric shape analysis. *ACM Transactions on Graphics (TOG)* 33, 4, 120.
- KOPPULA, H., GUPTA, R., AND SAXENA, A. 2013. Learning human activities and object affordances from RGB-D videos. *IJRR*.
- LI, Y., DAI, A., GUIBAS, L., AND NIESSNER, M. 2015. Database-assisted object retrieval for real-time 3d reconstruction. *Computer Graphics Forum* 34, 2.
- LIU, T., CHAUDHURI, S., KIM, V. G., HUANG, Q.-X., MITRA, N. J., AND FUNKHOUSER, T. 2014. Creating Consistent Scene Graphs Using a Probabilistic Grammar. *ACM Transactions on Graphics (TOG)* 33, 6.

- LOWE, D. G. 2004. Distinctive image features from scale-invariant keypoints. *International journal of computer vision* 60, 2, 91–110.
- MARGARITIS, D. 2003. *Learning Bayesian network model structure from data*. PhD thesis, University of Pittsburgh.
- MATTAUSCH, O., PANOZZO, D., MURA, C., SORKINE-HORNUNG, O., AND PAJAROLA, R. 2014. Object detection and classification from large-scale cluttered indoor scans. *Computer Graphics Forum* 33, 2, 11–21.
- NAN, L., XIE, K., AND SHARF, A. 2012. A search-classify approach for cluttered indoor scene understanding. *ACM Transactions on Graphics (TOG)* 31, 6, 137.
- NIESSNER, M., ZOLLHÖFER, M., IZADI, S., AND STAMMINGER, M. 2013. Real-time 3d reconstruction at scale using voxel hashing. *ACM Transactions on Graphics (TOG)* 32, 6, 169.
- SATKIN, S., AND HEBERT, M. 2013. 3dnn: Viewpoint invariant 3d geometry matching for scene understanding. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, IEEE, 1873–1880.
- SAVVA, M., CHANG, A. X., HANRAHAN, P., FISHER, M., AND NIESSNER, M. 2014. Scenegrok: Inferring action maps in 3D environments. *ACM Transactions on Graphics (TOG)* 33, 6.
- 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 Transactions on Graphics (TOG)* 31, 6, 136.
- SHAO, T., MONSZPART, A., ZHENG, Y., KOO, B., XU, W., ZHOU, K., AND MITRA, N. J. 2014. Imagining the unseen: Stability-based cuboid arrangements for scene understanding. *ACM Transactions on Graphics (TOG)* 33, 6, 209.
- SHOTTON, J., SHARP, T., KIPMAN, A., FITZGIBBON, A., FINOCCHIO, M., BLAKE, A., COOK, M., AND MOORE, R. 2013. Real-time human pose recognition in parts from single depth images. *Communications of the ACM* 56, 1, 116–124.
- TVERSKY, B., AND HARD, B. M. 2009. Embodied and disembodied cognition: Spatial perspective-taking. *Cognition* 110, 1, 124–129.
- WEI, P., ZHAO, Y., ZHENG, N., AND ZHU, S.-C. 2013. Modeling 4D human-object interactions for event and object recognition. In *ICCV*.
- WEI, P., ZHENG, N., ZHAO, Y., AND ZHU, S.-C. 2013. Concurrent action detection with structural prediction. In *ICCV*.
- XU, K., ZHANG, H., COHEN-OR, D., AND CHEN, B. 2012. Fit and diverse: set evolution for inspiring 3d shape galleries. *ACM Transactions on Graphics (TOG)* 31, 4, 57.
- 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 Transactions on Graphics (TOG)* 32, 4, 123.
- XU, K., MA, R., ZHANG, H., ZHU, C., SHAMIR, A., COHEN-OR, D., AND HUANG, H. 2014. Organizing heterogeneous scene collections through contextual focal points. *ACM Transactions on Graphics (TOG)* 33, 4, 35.
- YEH, Y.-T., YANG, L., WATSON, M., GOODMAN, N. D., AND HANRAHAN, P. 2012. Synthesizing open worlds with constraints using locally annealed reversible jump mcmc. *ACM Transactions on Graphics (TOG)* 31, 4, 56.